Capacity-Aware and Delay-Guaranteed Resilient Controller Placement for Software-Defined WANs

Maryam Tanha, *Student Member, IEEE*, Dawood Sajjadi, *Student Member, IEEE*, Rukhsana Ruby, and Jianping Pan, *Senior Member, IEEE*

***Abstract*—Currently, one of the main enablers for network evolution is software-defined networking (SDN), where the control plane is decoupled from the data plane. A controller, as a (logically) centralized entity in the control plane, is the Achilles' heel of SDN resilience since its failure would affect the proper functioning of the entire network. The resilience of the control plane is strongly linked to the controller placement problem, which deals with the positioning and assignment of controllers to the forwarding devices (i.e., switches). A resilient controller placement problem needs to assign more than one controller to a switch while it satisfies certain quality of service requirements. In this paper, we propose a solution for such a problem that, unlike most of the former studies, takes both the switch-controller/inter-controller latency requirements and the capacity of the controllers into account to meet the traffic load of switches. The proposed algorithms, one of which has a polynomial-time complexity, adopt a clique-based approach in graph theory to find high-quality solutions heuristically. It is evaluated with real wide area network (WAN) topologies and the corresponding results are extensively analyzed. The resultant studies equip the service providers with helpful insights into the design of a resilient software-defined WAN.**

***Index Terms*—SDN, resilient controller placement, clique, WAN, controller, latency.**

I. Introduction

THE EMERGENCE of Software-Defined Networking (SDN), as a promising technology, brings substantial benefits such as network programmability, flexible and efficient network management, vendor-independent control interfaces, accelerated innovation, and cost-effective design and maintenance. By decoupling the routing decision making from packet forwarding, all the control functionalities are incorporated into a (logically) centralized entity called controller. Particularly, Software-Defined Wide Area Networks (SD-WANs) have made considerable headways in 2016, and Gartner envisioned that about one third of network operators will deploy the SD-WAN technology by 2020. Service providers such as CenturyLink, EarthLink, and AT&T have already unveiled SD-WAN services [1]. Moreover, B4 [2], a private WAN connecting Google's data centers, is a practical example for one of the first and largest SDN deployments.

However, the great reliance of SDN on the logically centralized control plane has heightened the concerns of research communities and industries about the resilience of the control plane. Although the controller provides flexible and fine-grained resilience management features that contribute to faster and more efficient failure detection and containment in the network, it is the Achilles' heel of SDN resilience. The malfunction of the control plane resulting from natural disasters, malicious attacks or accidental faults/human errors, may have adverse impacts on the correct functioning of the whole system and affect many applications and services. Thus, connecting the network devices to a single controller may lead to a single point of failure and a performance bottleneck [3].

The reliable design of the control plane is tightly interwoven with the *Controller Placement Problem* (*CPP*), which determines the number and location of the controllers in a given topology. Resilient controller placement influences almost all of the resilience disciplines, including survivability (as the superset of fault tolerance), dependability (as the superset of reliability), security, performability, traffic tolerance, and disruption tolerance [4]. A survey on the main research efforts addressing the resilience disciplines in SDN can be found in [5]. Note that we utilize the umbrella term *resilient controller placement* to refer to our proposed controller placement problem since it covers more than one resilience discipline. Usually, improving one resilience discipline overlaps with another one (e.g., redundancy improves both fault tolerance and reliability). Our focus is on incorporating redundancy into the network design, which is one of the most commonly used methods to mitigate the impacts of node/link failures [6]. For instance, one of the fault tolerance techniques adopted in B4 is using software replicas (placed on different physical servers) to protect servers and control processes in case of failures.

With regard to the aforementioned issues, each OpenFlow-enabled switch should be connected to multiple controllers [3], [7] to achieve a resilient control plane. To decrease the communication overhead between the switch and its assigned controllers, instead of having simultaneous connections of a switch to multiple controllers, we focus on a

Manuscript received October 27, 2017; revised February 11, 2018; accepted April 9, 2018. Date of publication April 23, 2018; date of current version September 7, 2018. This work is supported in part by NSERC, CFI and BCKDF. The associate editor coordinating the review of this paper and approving it for publication was S. Schmid. *(Corresponding author: Maryam Tanha.)*

M. Tanha, D. Sajjadi, and J. Pan are with the Department of Computer Science, Faculty of Engineering, University of Victoria, Victoria, BC V8P 5C2, Canada (e-mail: tanha@uvic.ca).

R. Ruby is with the College of Computer Science, Shenzhen University, Guangdong 518060, China.

Digital Object Identifier 10.1109/TNSM.2018.2829661

master/slave design. It requires each switch to be connected to one primary controller (master) and one or more slave (backup) controllers. The controller placement should satisfy performance requirements such as the maximum allowable latency between a switch and its assigned controllers as well as the inter-controller communication delay for synchronization purposes. Moreover, the capacity limitation of the controllers as well as the traffic load of switches should be taken into account. Therefore, designing a control plane which is resilient to controller node failures (in contrast with the less frequent controller site failures) while satisfying the Quality of Service (QoS) requirements is of great importance. Although assigning more controllers to a switch enhances the resilience of the control plane, it increases the incurred cost in terms of the number of required controllers (each controller incurs the cost of deployment, maintenance, etc). Hence, in order to have a cost-effective design, minimizing the number of controllers is crucial for service providers.

Summary of contributions: The contribution of this paper is threefold. First, the *Resilient Capacitated Controller Placement Problem* (*RCCPP*) in SD-WANs has been formulated, which is more inclusive and easily adjustable compared with the existing research works, and it is mainly focused on the resilience against controller node failures. The proposed formulation is among the few schemes that take factors, such as the capacities of controllers, the traffic load of switches and switch-controller and inter-controller propagation latencies into account simultaneously. It also offers more flexibility for the design of an SDN-based network since it is tailored for the satisfaction of the SLAs by the service providers as well as it provides a resilient controller placement scenario that is independent from the master controller selection process. Second, to the best of our knowledge, we are the first one to model the NP-hard *RCCPP* based on the clique concept in graph theory, and this approach is applicable to other similar variants of reliable facility location problems. While both proposed heuristic algorithms provide high-quality solutions (small gap with the optimal value), the second one gives a solution in polynomial time. Finally, a detailed analysis of the *RCCPP* is provided for different real topologies under various parameter settings.

The rest of the paper is structured as follows. Section II gives an overview of the important factors for the *CPP*, which is followed by a review of the existing works on *resilient CPP* in SDN. The system model and the problem formulation are presented in Section III while the proposed solution is provided in Section IV. Section V evaluates the performance of the proposed solution. Finally, Section VI draws the conclusion and provides some future research directions.

II. Related Work

A. Overview of the CPP in SDN

Given a topology, the *CPP* (first coined by Heller *et al.* [8]) finds the number and the locations of required controllers while minimizing the cost associated with the controller placement. This cost can be expressed in terms of the number of controllers or the switch-controller communication latency or the synchronization time of controllers, or a combination of more than one of these metrics (as a multi-objective optimization problem). Based on the existing research works on the *CPP*, we list the crucial factors for placing the controllers in an SDN-enabled network as follows.

- *Switch-controller latency:* This is the first and most significant factor for controller placement. Flow-setup latency for an unmatched flow in each of the switches is composed of transmission delay, processing delay and propagation delay [9] (as the main contributor to the switch-controller latency in SD-WANs [8], [10]). Long propagation delay between a switch and its assigned controller can adversely affect the capability of the controller to respond to network events in a timely manner as well as it decreases the communication reliability [11]. Thus, almost all of the research works on the *CPP* aim to minimize this latency [4], [8], [12]–[15] or to keep it below a certain threshold [11], [16]–[22].
- *Inter-controller latency:* This latency is also of great importance, particularly for synchronization purposes in case of having multiple controllers assigned to a switch or for inter-domain controller communications [23]. Particularly, large SDN-based networks function according to a global network view that is logically centralized. However, to achieve resiliency and scalability goals, the control state and logic must be physically distributed. Regardless of the methods employed to manage the state consistency of the network, the connectivity among the controllers indicates the maximum time required to update information among them [11]. Examples of considering this type of latency can be found in [11], [14], [15], and [18]. Similar to the switch-controller latency, it should be minimized or bounded.
- *Controller capacity:* Due to the resource constraints (i.e., CPU, memory, and access bandwidth), each controller can only handle a limited number of requests per second. An overloaded controller would have a higher probability of failure [10] and it causes the processing latency to increase, which subsequently affects the switch-controller latency. The capacity of a controller is usually defined as the number of flow setup requests (packets) per second that it can handle [24], [25]. The works in [4], [9], [10], [15], [20], and [25] are examples of the *capacitated CPP*. It should be noted that load balancing among the controllers does not necessarily correspond to a *capacitated CPP*. For instance, Lange *et al.* [14] minimized the load imbalance, i.e., the difference between the maximum and minimum number of switches connected to the controllers for improving the load balance among the controllers. Other works such as [26] and [27] defined the controller load based on the structural properties of their assigned switches (e.g., degree of the node) which ignores the non-uniform traffic load of switches. However, no assumption was made to consider the real capacity of the controllers in the three aforementioned works. A more exact way of defining the load on a controller is the number of requests per second incurred by its connected switches.
- *Traffic load of switches:* In a practical SDN controller placement design, the traffic load of switches should

be taken into account to avoid network congestion and the overload of the controllers. This load can be based on the worst/average-case load of switches as in [4], [9], [15], and [25] or time-varying traffic load of switches [16], [20].

- *Scalability:* While some of the solutions offered for the *CPP*, such as [9] and [11], deal with small to medium-scale networks, others are helpful for large-scale implementations (e.g., [14], [20], and [28]). In SD-WANs, with a large number of switches and high traffic volume, controller placement has a great impact on the performance of the system [20].
- *Resilience:* The resilience of the control plane plays a significant role in the sustainability of the entire SDN-based network. Disruptive events may decompose the network and isolate the switches from their assigned controllers [14]. *Resilient CPP* is a variant of the *CPP* in SDN, which emphasizes the optimization of different reliability aspects of the control plane (examples of existing works are [15], [29]–[31]). Enhancing the fault tolerance (by assigning more than one controller to a switch) while minimizing the number of required controllers or expected control path loss (i.e., the number of broken control paths resulted from network failures [29]) exemplifies such reliability goals.

It should be noted that there are trade-offs and interdependencies among the aforementioned factors. Therefore, no single best placement strategy exists and the decision makers need to seek a balanced trade-off for a certain use case [14]. Moreover, there exist some overlaps between the *CPP* and the research on middlebox deployment (such as [32]) and placement of Virtualized Network Function Managers (VNFMs) in virtualized and software-defined networks. However, the *CPP* differs from such problems in terms of both scalability and dynamics of the system [33]. In addition, [32] does not require reliability and inter-middlebox delay upper bound both of which are important in the *resilient CPP*. Hence, its offered solution is not applicable to the *resilient CPP* without relaxing the aforementioned key constraints. In the following, we provide an overview of the main existing research works on the *resilient CPP* and highlight their contributions and limitations considering the aforementioned factors.

B. Existing Works on Resilient CPP

The unavailability of a controller to its connected switches may result from single/multiple switch/link failures on the south-bound connection or the failure of the controller node itself. While having a main connection from the switch to its assigned controller along with multiple auxiliary connections [7] is beneficial for the former case, controller replication is helpful regarding the latter case. Vizarreta *et al.* [29] proposed two resilient controller placement strategies for tolerating single link and node failures although they did not take the capacity of the controllers into account. While the first strategy involved connecting each switch to its assigned controller through two disjoint paths, the second one required that each switch is connected to two controllers via two disjoint paths. The performance of their solution was evaluated using the expected control path loss and the average control path availability.

To achieve a high south-bound reliability, a *resilient CPP* was introduced in [34]. In such a design, each switch is required to satisfy a reliability constraint in a way that the probability of having at least an operational path to its assigned controller(s) is higher than a given threshold. Zhong *et al.* [19] defined two reliability metrics for the control network based on the average number of disconnected switches resulting from a single physical link failure. Moreover, a heuristic algorithm was proposed to find the min-cover solution with most reliability. Beheshti and Zhang [35] presented two placement algorithms to maximize the controller-switch connection resilience. Using a similar resilience objective, [36] provided a solution that leverages a min-cut-based graph partitioning algorithm to select the subset of switches for connecting to specific controllers while focusing on switch and link failures. A cause-based reliability analysis model was proposed in [37] to minimize the expected percentage of control path loss whereas different heuristic algorithms (greedy, simulated annealing and random placements) were evaluated for the same objective in [30]. Guo and Bhattacharya [38] investigated the *resilient CPP* using interdependent network analysis. They solved the problem using a greedy optimization method and partitioning scheme for different types of network topologies. To maximize the switch-controller connectivity while satisfying controller capacity constraints in SD-WANs, a solution for the *CPP* was proposed in [25] along with two failover mechanisms. However, no assumption was made about the switch-controller and inter-controller latencies.

The Pareto-Optimal Controller Placement (POCO) framework in [39] and [40] was extended in [14] (using the Pareto simulated annealing heuristic) to include large-scale networks. Given the number of controllers, their solution gives Pareto-optimal placements to minimize different objectives, including switch-controller latency, inter-controller latency, load imbalance and the maximum number of disconnected switches when a node/link failure happens (without considering the capacities of the controllers). The research works in [4] and [18] have similar objective functions while considering the controller failure probabilities and minimizing the total cost including the cost of deployment and expected failure cost.

One of the most recent works is *CNCP* [15] (*Capacitated Next Controller Placement*), which proposed a *resilient* and capacitated controller placement strategy while considering the controller failures. Given a budget in terms of the number of controllers and the inter-controller latency threshold, a switch is assigned to a number of reference controllers and the objective is to minimize the maximum worst-case latency in case of controller failures. The authors also proposed a simulated annealing heuristic to solve the problem. Although *CNCP* has the most similarity to our problem (Q parameter corresponds to our parameter r), the main differences in terms of formulation and the offered solution are as follows. First, *CNCP* assumes a given number of controllers as the input of the problem formulation whereas the number of controllers is our objective function. Second, *CNCP* requires each switch to be connected to its nearest controller as its primary (master) controller, however, this is not necessary in our formulation, which gives more freedom to the selection of a master controller (not only based on propagation latency). We believe such a selection criteria should be independent from the *CPP* to have more flexibility in design. Furthermore, connecting the switches to their nearest controller(s) may result in more load imbalance among the controllers. We will show later in Section V that our formulation achieves similar results, usually with a lower load imbalance, while it is simpler, more inclusive and easily extendable (especially to include single link failures). Finally, the last difference is that we have proposed a polynomial-time algorithm, which takes into account the structure of the problem using cliques, to solve this NP-hard problem for large-scale topologies. However, the efficiency of the simulated annealing algorithm for *CNCP* in terms of both solution quality and time complexity should be investigated for large-scale topologies as it does not guarantee any polynomial-time solution (only a limited evaluation for one medium-size topology was provided in this work and the main focus was on using a solver to acquire a solution).

Among all of the aforementioned research works on the *resilient CPP*, the capacity of controllers and the load of switches were only considered in [4], [15], and [25]. Also, not all of the works took the inter-controller latency into account and no comparison was made with the optimal solution when heuristics were proposed. Moreover, since the *CPP* is NP-hard [8], most of the proposed solutions for *resilient CPP* are not appropriate for large-scale networks due to the enormous time required to search the solution space, especially when multiple factors are considered simultaneously. Therefore, a formulation of the *resilient CPP* which incorporates all of the important factors while being easily adaptable is of great significance.

III. System Model and Problem Formulation

A. Preliminaries

The *CPP* and its *resilient* form, are variants of the Facility Location Problem (FLP) [14], [41], which is an NP-hard problem [8]. In the discrete form of such a problem, we have a finite set of users with their associated demands for service and a finite set of potential locations to place the facilities. In an SD-WAN, the controllers play the role of the facilities and switches are the customers/clients. While many of the works on the *resilient CPP* (e.g., [18] and [31]) have investigated the uncapacitated version, we focus on the capacitated version which is a better representation of many real-life applications, including the *resilient CPP* in SDN. Generally, to seek the solution for such problems, two types of decisions should be made. Location decisions determine where to place the controllers while the assignment decisions involve how to allocate the established controllers to the switches.

B. Our Assumptions

1) Single and Multi-Controller Node Failures: It should be noted that we consider controller node failures in contrast with controller site failures. The latter applies to a more severe impact resulting from a disaster/attack that completely destroys the data center where the controller is located rather than the controller itself, and it is less frequent compared with the former one. As we mentioned in Section I, to improve the resilience of control plane, multiple controllers should be assigned to a switch. We focus on a master/slave model in which we have a primary (master) controller (with full control over the switch) assigned to a switch along with one or more backup controllers (with read-only access to the switch). While having one backup controller results in tolerance to single controller failures, having more than one backup controllers (as a design parameter indicated by the network designer) leads to handling multiple controller failures. This planning ahead for controller failures (rather than manual and administrative intervention) is in line with some of the existing works on *resilient CPP* such as [4], [15], and [25]. Although a simple fail-over mechanism which involves defining a list of controllers for a switch was proposed in OpenFlow v0.9, the controller role change mechanism to support multiple controllers for fail-over and load balancing purposes is included in OpenFlow v1.2 and the later versions (more information can be found in [7]). The assigned controllers to the switch organize the management of the switch among themselves and they make the decision to choose the master controller. Such a coordination mechanism is also needed for distributed controllers (as in [23]) and it is out of the scope of this paper.

2) Objective of the Optimization Problem: Due to the incurred cost of having multiple controllers assigned to a switch for increasing reliability, a preferable solution to the *resilient CPP* for service providers is the one that is more cost-effective (i.e., minimizes the total number of controllers). However, one can minimize the average or the worst-case switch-controller latency in a multi-objective optimization problem or when a budget is given in terms of the number of controllers (e.g., [15]) similar to *p*-median and *p*-center FLPs. We believe that it is more practical to give the network designer the freedom to choose and tune different QoS parameters (switch-controller latency and inter-controller latency) to minimize the cost of resilient controller placement before the final deployment according to the requirements of a certain network. Thus, by changing the aforementioned parameters for a given topology, the required budget is changed accordingly.

3) Bounds for Switch-Controller and Inter-Controller Latencies: The interplay between switch-controller and inter-controller latencies has been studied in some of the existing works such as [14]. A group of controllers which are close to each other leads to low inter-controller latencies and high switch-controller latencies whereas spatially distributed controllers result in the opposite case. Connecting the switches to their nearest controller(s) may result in load imbalance among the controllers, and subsequently increase the delay due to the queuing time at some of the controllers [14]. In contrast, considering a threshold for switch-controller and inter-controller latencies can guarantee the latency not to exceed the delay upper bound. This case can be converted to the former case by reducing the threshold as much as possible for a given topology (i.e., more than or equal to the minimum propagation delay between two nodes). Moreover, the threshold-based approach is critical for delay-sensitive applications or for the satisfaction of the Service Level Agreements (SLAs) by service providers. Considering the aforementioned issues, in this paper, we follow the delay bound-based approach, which is more inclusive, particularly when the number of required controllers needs to be minimized. Since the switch-controller communication is more frequent than the inter-controller interactions, we assume that the former threshold is less than or equal to the latter one. In addition, both switch-controller and inter-controller delays can be approximated by the propagation delay in SD-WANs (due to the fact that it is the main part of the total latency). Furthermore, since each controller is in charge of managing a subset of all switches or due to having a distributed control plane, to maintain a consistent global view of the network and subsequently to ensure the proper functioning of the network, not only the primary and backup controllers of a switch should be synchronized with each other, but also all the controllers need to communicate with each other [15]. Therefore, the inter-controller latency should be embedded into problem formulation.

4) Single Link Failures: Although our key focus is on controller node failures similar to [15], we show the extension of our problem formulation to include link-disjoint paths between a switch and its assigned controllers.

C. Problem Formulation

The topology of an SD-WAN is represented by a connected graph $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V} = \boldsymbol{S} \cup \boldsymbol{C}$, $\boldsymbol{S}$ is the set of OpenFlow-enabled switches, and $\boldsymbol{C}$ is the set of potential controller locations while $\boldsymbol{E}$ denotes the set of weighted links. The weights of the links are the propagation latencies (shortest path lengths) between the nodes based on their geographical locations. Assuming that the controllers can share the same location with the switches, the potential locations for the controllers are equal to the set of switches (i.e., $\boldsymbol{C} = \boldsymbol{S}$). We define two binary variables, namely y_j and x_{ij} to determine the controller location decisions and the assignments of controllers to the switches, respectively. The *RCCPP* is defined as follows.

$$\min \sum_{j \in C} y_j, \tag{1}$$

subject to,

$$y_j \geq x_{ij}, \quad \forall i \in S,\ j \in C \tag{2}$$

$$\sum_{j \in C} x_{ij} = r, \quad \forall i \in S \tag{3}$$

$$\sum_{i \in S} l_i\ x_{ij} \leq u_j, \quad \forall j \in C \tag{4}$$

$$d_{ij}\ x_{ij} \leq sc_{\max}, \quad \forall i \in S,\ \forall j \in C \tag{5}$$

$$d_{j'j''}\ y_{j'}\ y_{j''} \leq cc_{\max}, \quad \forall j', j'' \in C \tag{6}$$

$$x_{ij}, y_j \in \{0, 1\}, \quad \forall i \in S,\ \forall j \in C. \tag{7}$$

The constraint in (2) prohibits a switch from being assigned to a controller site which is not open while the constraint in (3) ensures that each switch is connected to $r > 1$ controllers (if $r = 1$, the formulation corresponds to the *capacitated CPP*). Note that having $r = 2$ emphasizes resilience against single controller node failures while $r > 2$ corresponds to resilience against the multi-controller failure case. The constraint in (4) prevents the total incurred load by the switches on a controller from exceeding its capacity. The constraint in (5) expresses that the propagation latency between a switch and its assigned controllers satisfies the delay bound $sc_{\max}$. Satisfying the maximum allowed delay among the open controllers is enforced by the constraint in (6). Finally, (7) provides the integrality constraints. Since the constraint in (6) is non-linear, we linearize it by defining a new binary variable $w_{j'j''}$ using the McCormick envelopes [42]), which is given by

$$w_{j'j''} = y_{j'} y_{j''}, \tag{8}$$

and subsequently replacing it with the following constraints

$$d_{j'j''}\ w_{j'j''} \leq cc_{\max}, \quad \forall j', j'' \in C \tag{9}$$

$$w_{j'j''} \leq y_{j'}, \quad \forall j', j'' \in C \tag{10}$$

$$w_{j'j''} \leq y_{j''}, \quad \forall j', j'' \in C \tag{11}$$

$$w_{j'j''} \geq y_{j'} + y_{j''} - 1, \quad \forall j', j'' \in C \tag{12}$$

$$w_{j'j''} \in \{0, 1\}, \quad \forall j',\ j'' \in C. \tag{13}$$

The above problem formulation can be extended by adding the following constraint to include the protection against single link failures.

$$x_{ij'} x_{ij''} \leq DP\big(i, j', j''\big), \quad \forall i \in S\ \forall j', j'' \in C. \tag{14}$$

The constraint in (14) can be linearized by introducing a three-indexed binary variable $z_{ij'j''}$ using the McCormick envelopes similar to the constraint in (6). DP denotes a function that determines whether or not all the control paths for a switch are link-disjoint. The input of this function is switch i and two potential controller locations (j' and j''). Therefore, its output is equal to 1 if the control paths of any two controllers (deployed at nodes j' and j'') of assigned controllers to switch i are link-disjoint. Note that the shortest paths between a node and all other nodes of $\boldsymbol{G}$ are the input of the optimization problem. It should be noted that the values of both $cc_{\max}$ and $sc_{\max}$ are indicated by a fraction of the graph diameter for consistency across various topologies (similar to some of the existing works such as [14] and [15]). We denote the diameter of the given WAN topology $\boldsymbol{G}$ by D_G (the length of the longest shortest path) and we indicate the minimum shortest path length in $\boldsymbol{G}$ by $D_{\min}$. We assume the following holds.

$$D_{\min} \leq sc_{\max} \leq cc_{\max} \leq D_G. \tag{15}$$

Table I summarizes the notations used in the formulation of *RCCPP* and the proposed algorithms in Section IV.

IV. Proposed Solution

In this section, we elaborate our idea to solve the formulated optimization problem in Section III based on the clique concept in graph theory by introducing two heuristic algorithms. Then, a case study is provided to delineate the proposed algorithm.

TABLE I
NOTATIONS USED IN THE PROBLEM FORMULATION AND PROPOSED ALGORITHMS

Symbol	Definition
$\boldsymbol{S}$	Set of OpenFlow-enabled switches
$\boldsymbol{C}$	Set of potential controller locations
$sc_{\max}$	Latency bound between a switch and its assigned controllers
$cc_{\max}$	Inter-controller latency bound
u_j	Capacity of controller j
l_i	Traffic load of switch i
r	Number of controllers to serve a given switch (resilience parameter)
d_{ij}	Minimum propagation latency between node i and node j
y_j	1 if node j is selected to deploy a controller, and 0 otherwise
x_{ij}	1 if switch i is connected to the controller at node j, and 0 otherwise
$w_{j'j''}$	1 if two controllers are at nodes j' and j'', respectively, and 0 otherwise
$z_{ij'j''}$	1 if switch i is connected to the controllers j' and j'', and 0 otherwise
DP	1 if the all the control paths of a switch are link-disjoint, and 0 otherwise
$\boldsymbol{N}$	Network size, i.e., $\lvert \boldsymbol{S} \rvert$
$\boldsymbol{G}$	Topology of an SD-WAN
$\boldsymbol{G_o}$	Overlay graph
$\boldsymbol{G_p}$	Pruned overlay graph
$\boldsymbol{M}$	Set of all maximal cliques $\boldsymbol{G_p}$
$\boldsymbol{A}$	Set of all r-cliques and $(r+1)$-cliques of $\boldsymbol{G_p}$
$\boldsymbol{A_i}$	A subset of $\boldsymbol{A}$ that includes switch i
$\omega(\boldsymbol{G_p})$	Clique number of $\boldsymbol{G_p}$
$\boldsymbol{F}$	Set of all found feasible solutions for RCCPP-AMC
$\boldsymbol{LB}$	Lower bound of the number of controllers

A. Clique Graphs and Their Relevance to Our Problem

We define a complete graph (denoted by $\boldsymbol{G_o}$) of the physical network topology as an overlay, in which the nodes correspond to the switches and/or controllers and the weights of the links correspond to the shortest path lengths between each pair of nodes. Then, we prune $\boldsymbol{G_o}$ by removing the links which do not satisfy the latency bound $cc_{\max}$ and we call the resultant graph $\boldsymbol{G_p}$. In this graph, the existence of a link between each pair of nodes means that these nodes can be in the set of controllers in a potential solution. By studying the structure of the optimal solution to the formulated problem in Section III, we observe that each switch and its assigned controllers is a clique of $\boldsymbol{G_p}$. A clique [43] is defined as a complete subgraph of an undirected graph. Since a switch needs to be directly connected to all of its assigned r controllers and such controllers themselves require to interact with each other (and thus, each pair of the r controllers must be adjacent in $\boldsymbol{G_p}$), the switch and its associated controllers construct a complete subgraph, i.e., a clique of $\boldsymbol{G_p}$. Moreover, the inter-controller latency in the constraint (6) implies that the set of controllers in a solution must be a subset of one of the maximal cliques[1] of $\boldsymbol{G_p}$. All of the controllers need to be directly connected to each other, and hence they must be a clique which is a subset of a maximal clique of $\boldsymbol{G_p}$. Furthermore, possible controller-switch assignments are the r-cliques and $(r+1)$-cliques (if any) of $\boldsymbol{G_p}$. Cliques of size r correspond to the case where one of the potential controllers of the switch is co-located with it while $(r+1)$-cliques indicate that none of the assigned controllers to a switch is co-located with it. Based on all these observations and insights of the optimal solution, we have developed two heuristic algorithms to solve the problem, the description of which is provided in Section IV-B.

Lower bound and upper bound of the objective function value: Due to the fact that the controllers in a feasible/optimal solution construct a clique which is a subset of one of the maximal cliques, the upper bound of the number of controllers (i.e., the value of the objective function) is equal to the clique number (i.e., size of the maximum clique)[2] of $\boldsymbol{G_p}$ denoted by $\omega(\boldsymbol{G_p})$. Moreover, the number of controllers in an optimal solution must be at least equal to the total traffic load of switches divided by the capacity of a controller (assuming uniform capacity u for all the controllers) as well as it must be greater than or equal to the value of r. Hence, the following must hold

$$\max\left(r, \frac{r \times \sum_{i \in S} l_i}{u}\right) \leq y^* \leq \omega(\boldsymbol{G_p}), \tag{16}$$

where y^* denotes the optimal (minimum) number of controllers in a solution.

Algorithm 1 General Algorithmic Framework

```
1:  Input: $\boldsymbol{G}$, $cc_{\max}$, $sc_{\max}$, $r$, switch loads, controller's capacity ($u_c$), shortest paths matrix.
2:  Output: controller locations and controller-switch assignments or infeasible state.
3:  Feasibility-Check ($\boldsymbol{G}$, $cc_{\max}$, $sc_{\max}$).
4:  $\boldsymbol{G_o}$ = OverlayGraph ($\boldsymbol{G}$).
5:  $\boldsymbol{G_p}$ = Prune ($\boldsymbol{G_o}$, $cc_{\max}$).
6:  Feasibility-Check ($\boldsymbol{G_p}$).
7:  $\boldsymbol{A}$ = all $r$-cliques and $(r+1)$-cliques of $\boldsymbol{G_p}$.
8:  For each switch $i$, find a subset of $\boldsymbol{A}$ ($\boldsymbol{A_i}$) that includes that switch w.r.t. the values of $sc_{\max}$ and $cc_{\max}$.
9:  Sort the switches w.r.t. the total number of their associated cliques ascendingly.
10: Feasibility-Check ($\boldsymbol{S}$, $\boldsymbol{A_i}$).        ▷ $\forall i \in \boldsymbol{S}$
11: RCCPP-AMC ()
12: RCCPP-SMC ()
```

B. Descriptions of the Proposed Algorithms

Algorithm 1 serves as a general framework that includes the common steps of our proposed algorithms, namely *RCCPP-AMC* (*RCCPP* with All Maximal Cliques) and *RCCPP-SMC* (*RCCPP* with a Single Maximal Clique). Table I shows the notations used in the proposed algorithms. The first step is the feasibility check w.r.t. (15) (which requires constant time). Building the overlay graph $\boldsymbol{G_o}$ by *OverlayGraph (G)* has the time complexity of $O(N^2)$ (where $N = \lvert \boldsymbol{S} \rvert$). The process of pruning the complete graph $\boldsymbol{G_o}$ is of $O(N^2)$ complexity since it involves checking all the edges. Then, a feasibility check for $\boldsymbol{G_p}$ w.r.t. the chosen value of $cc_{\max}$ is performed in step 6. If $\boldsymbol{G_p}$ is a disconnected graph, the problem is infeasible (checking the connectivity of $\boldsymbol{G_p}$ takes $O(N^2)$ time using BFS or DFS). As shown in step 7 of *Algorithm* 1, to identify the possible controller-switch assignments, we find the sets of all r-cliques and $(r+1)$-cliques (if any) of $\boldsymbol{G_p}$. Given the fixed value of r in *RCCPP*, finding the cliques of size r and $r+1$ takes $O(N^r)$ and $O(N^{r+1})$ time, respectively. In particular, for

[1]A clique is maximal if it cannot be extended (turned into a larger clique) by adding more adjacent vertices to it [43].

[2]A clique is maximum, if there is no other clique of larger size in the graph [43].

finding the r-cliques, $\binom{N}{r}$ subgraphs (with r vertices) should be checked and since the subgraphs of interest must be complete, the presence of at most $r(r-1)/2$ edges in $\boldsymbol{G_p}$ must be checked. Thus, the worst-case time complexity of this operation is $O(r^2N^r)$, which is converted to the polynomial form $O(N^r)$, thanks to the fixed value of r in our problem. Similar explanation applies to finding cliques of size $r+1$. However, more efficient algorithms for finding the cliques of fixed size have been found in [44]. Then, for each switch, we define the set of all cliques that include switch i ($\boldsymbol{A_i}$) according to the following two cases:

1) $sc_{\max} = cc_{\max}$: A switch can be any of the r nodes in an r-clique that includes the switch. The same applies for the $(r+1)$-cliques that contain this switch if $r < \omega(\boldsymbol{G_p})$.
2) $sc_{\max} < cc_{\max}$: $\boldsymbol{A_i}$ includes a number of r-cliques and/or $(r+1)$-cliques such that the weight of all incident links to switch i in each of such cliques is less than or equal to the value of $sc_{\max}$.

We sort the switches according to the size of their associated $\boldsymbol{A_i}$ (i.e., the number of possible controller assignments for each switch i) in an increasing order ($O(N \log N)$ time). This means that the switches with fewer possible sets of assignments are handled first. If there is at least one switch i with $|\boldsymbol{A_i}| = 0$, the problem becomes infeasible (step 10).

RCCPP-AMC: As shown in *Algorithm* 2, the set of all maximal cliques of $\boldsymbol{G_p}$ (denoted by $\boldsymbol{M}$) is computed and the maximal cliques whose number of nodes is less than the lower bound calculated in (16) are excluded from $\boldsymbol{M}$. Thus, if $\boldsymbol{M}$ becomes empty after checking the aforementioned condition, the problem becomes infeasible (step 2). Then, for each of the remaining maximal cliques, it is assumed that all the nodes in that maximal clique are open and the algorithm proceeds with finding the assignments for each switch, i.e., a feasible solution is found if there is any (step 5). Particularly, to choose among the cliques of a switch in $\boldsymbol{A_i}$, we first leave out all the r-cliques and $r+1$-cliques whose potential controller nodes are not a subset of the currently chosen maximal clique m. In addition, all the cliques that have at least a controller node such that its remaining capacity is less than the traffic load of switch i, are excluded from $\boldsymbol{A_i}$. Afterward, if there is any clique whose controllers have been used already (i.e., their remaining capacity is less than the initial capacity), that clique is chosen as the assignment for switch i. Otherwise, we rank the cliques based on the number of existing used controllers in them, and then we choose the clique with the highest rank as the assignment for switch i. This results in the reuse of used controllers as much as possible. If a clique is found, the controllers in this clique are assigned to switch i. Once we are done with the assignments for all switches, if there is any controller in the chosen maximal clique m that is not involved in any of the controller-switch assignments, it is removed from the set of open controllers in the found solution. This solution is added to the list of found solutions. Finally, if more than one feasible solution is found, the best one (with the least number of controllers) is selected (if there exists only one feasible solution, it will be chosen as the best solution), otherwise the problem is infeasible (steps 7–11). *RCCPP-AMC* has a high chance of escaping the local optima by finding all the maximal cliques to produce multiple feasible solutions of good quality and then choosing the solution with the minimum number of controllers.

Algorithm 2 *RCCPP-AMC*

```
1:  $\boldsymbol{M}$ = All-Maximal-Cliques ($\boldsymbol{G_p}$).
2:  Feasibility-Check ($\boldsymbol{M}$, $LB$).
3:  $\boldsymbol{F} = \varnothing$.
4:  for $m \in \boldsymbol{M}$ do
5:      Find a feasible solution (if any) and add it to $\boldsymbol{F}$.
6:  end for
7:  if $\boldsymbol{F}! = \varnothing$ then
8:      Output the best solution.
9:  else
10:     The problem is infeasible.
11: end if
```

Algorithm 3 *RCCPP-SMC*

```
1:  $\boldsymbol{F} = \varnothing$.
2:  for $a \in \boldsymbol{A}$ do
3:      Construct a single maximal clique $m$ using clique $a$.
4:      if visited ($m$) or Infeasible ($m$, $LB$) then
5:          Goto 2.
6:      end if
7:      Find a solution w.r.t. the sorted list of switches.
8:      if a feasible solution is found then
9:          Add it to $\boldsymbol{F}$.
10:     end if
11: end for
12: if $\boldsymbol{F}! = \varnothing$ then
13:     Output the best solution.
14: else
15:     The problem is infeasible.
16: end if
```

Time complexity of RCCPP-AMC: Finding all maximal cliques has $O(3^{N/3})$ worst-case running time, since any arbitrary graph with N vertices has at most $3^{N/3}$ maximal cliques [45], [46]. However, it is possible to list all of the maximal cliques in polynomial or even in linear time for special families of graphs [47], [48]. For instance, in our problem, if $cc_{\max} = D_G$, $\boldsymbol{G_p}$ is a complete graph which is its own maximal clique. Similar observations are true for the cases where $\boldsymbol{G_p}$ is a planar graph [47] or a sparse graph [49]. The running time of *Feasibility-Check ($\boldsymbol{G_p}, r, \boldsymbol{M}$)* is dominated by $O(3^{N/3})$ to exclude the maximal cliques that do not satisfy the total traffic load of switches. Note that finding assignments for a switch has running time of at most $O(N^{r+1})$ (the maximum number of $r+1$ cliques). Finally, the overall worst-case time complexity of the algorithm is $O(3^{N/3})$.

RCCPP-SMC (a polynomial-time algorithm): RCCPP-AMC (shown in *Algorithm* 3) finds all maximal cliques of $\boldsymbol{G_p}$ and chooses the solution with the best quality among all the found feasible solutions (if any). However, it has an exponential time complexity which is not desirable in practical settings and for large-scale networks. Therefore, we propose a polynomial-time algorithm, *RCCPP-SMC*, that also gives us high-quality solutions (while not affecting the quality of many solutions found by *RCCPP-AMC* to a great degree as shown in Section V). The operational difference of this algorithm comparing with *RCCPP-AMC* is as follows. As shown in *Algorithm* 3, we compute a single maximal clique based on an element (a clique) of $\boldsymbol{A}$ using the algorithm in [50]. That is, we begin with that clique in $\boldsymbol{G_p}$ and try to add the other nodes of the graph to this clique one by one. A node is added if it is a neighbor of all the nodes inside the clique. The aforementioned algorithm for finding a single maximal clique has time complexity $O(N^2)$. If the created maximal clique has not been used before (not visited) and the number of nodes in this maximal clique is greater than or equal to the lower bound calculated in (16), the algorithm proceeds to the next step. Otherwise, another element of $\boldsymbol{A}$ (if any) is chosen. The constructed maximal clique serves as the set of potential locations for controllers. Then, we assign the controllers from this set to other switches following the same approach in *RCCPP-AMC*. If a feasible solution is found, it is added to the set $\boldsymbol{F}$, otherwise we choose another element of $\boldsymbol{A}$ (if any) and repeat steps 3–10. Finally, if $\boldsymbol{F}$ is not empty, the best feasible solution (with the least number of controllers) is selected, otherwise the problem is infeasible. The worst case time complexity of *RCCPP-SMC* is $O(N^{2r+3})$. This is due to the fact that the maximum number of iterations of *for loop* is $O(N^{r+1})$ (the number of elements in $\boldsymbol{A}$) and finding the assignments for all the switches in step 7 has the highest time complexity ($O(N) \times O(N^{r+1})$) among other steps inside the loop.

C. Case Study

To illustrate the effectiveness of the proposed algorithm, we study an example for the Sprint topology. We set the input parameters as follows: $cc_{\max} = 0.8D_G$, $sc_{\max} = 0.4D_G$, $r = 2$, $u_c = 2000$ $k\text{req}/s$ (controller capacity) and $l_s = 200$ $k\text{req}/s$ (uniform switch traffic load). The original Sprint topology $\boldsymbol{G}$, $\boldsymbol{G_o}$, $\boldsymbol{G_p}$, and the set of all three maximal cliques of $\boldsymbol{G_p}$ are shown in Fig. 1. The lower bound of the number of controllers in the optimal solution is 3 whereas the upper bound is 8 (i.e., the clique number of $\boldsymbol{G_p}$). In this example, the obtained solutions by both *RCCPP-AMC* and *RCCPP-SMC* are optimal. The set of open controllers in both solutions is $\{1, 4, 5, 6, 7\}$, which is a subset of maximal clique 2. The difference between *RCCPP-AMC* and *RCCPP-SMC* is that the former finds all of the 3 maximal cliques and finds the best solution among the ones produced by each of these maximal cliques while the latter only finds a single maximal clique from the 2-clique $\{3, 4\}$ for switch 3 (which is the first switch in the sorted list of the switches). Fig. 2 depicts the controller-switch assignments in the solution. More specifically, these assignments are the subsets of the 2-cliques and 3-cliques of $\boldsymbol{G_p}$. The switch nodes are marked by blue color. The controller nodes not co-located with the switch they serve are marked with red color while the ones co-located with the switch they serve are highlighted by orange color.

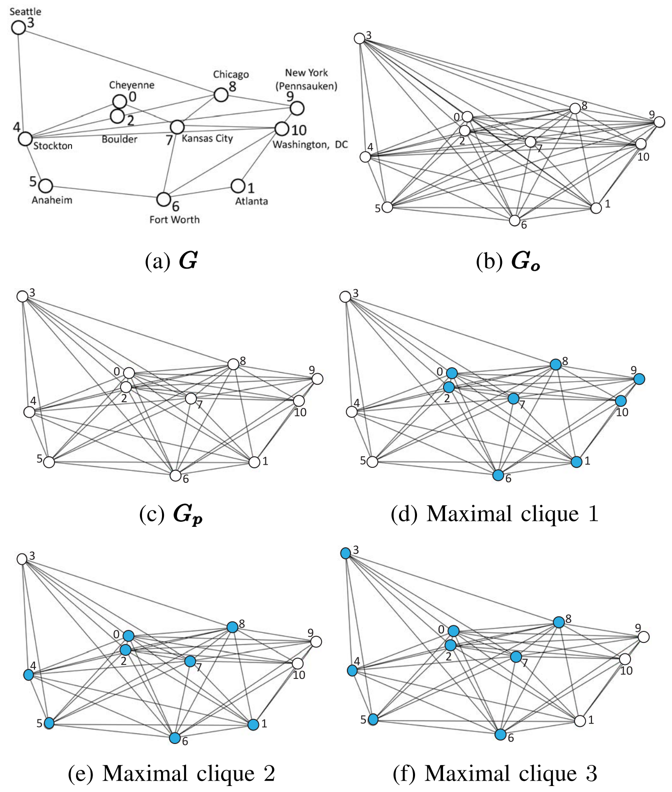


Fig. 1. Sprint topology and its corresponding constructed graphs as well as the maximal cliques of $\boldsymbol{G_p}$.

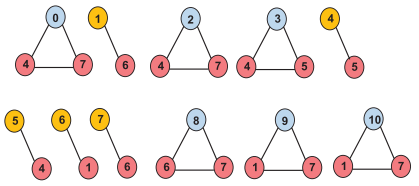


Fig. 2. Controller-switch assignments for the Sprint topology.

V. Performance Evaluation

In this section, we first provide a detailed description of our experiment setup and then we assess the performance of our proposed solutions w.r.t. different metrics and parameters.

A. Experiment Setup

We conducted our experiments on about 40 WAN topologies from Internet Topology Zoo (ITZ) [51], which is a publicly available data set and it has been used by many of the research works on SDN controller placement problems such as [8], [14], and [31]. Such network maps are of great importance in the optimization of network design and they represent the level at which the resilience and redundancy are highly likely to be considered. Moreover, the aforementioned dataset contains a broad range of topologies spanning over different geographical areas (ranging from regional/state networks to the continental ones). For the ease of analysis and presentation, the chosen topologies were classified according to their sizes (i.e., the number of nodes N). Four groups were defined and labeled as follows. Groups 1 ("small-size"), 2 ("medium-size"), 3 ("large-size"), and 4 ("very large-size") include the topologies with $N < 20$, $20 \leq N < 50$, $50 \leq N < 100$, and $N \geq 100$, respectively. As representatives for each group, we chose multiple graphs that cover different types of topologies (i.e., mesh, linear, ring and hub-and-spoke). Note that we chose the most recent version of a topology if there were more than one versions. The following shows the summary of the steps taken to conduct the experiments:

1) Pre-Processing: For this step, we applied a similar approach as in [8] and [14]. Multi-graphs were converted to simple graphs (the parallel edges do not affect the propagation latencies) and nodes with missing location information (i.e., latitude and longitude) were removed from the graph. The number of node removals was negligible w.r.t. the topology size (e.g., for TATA, 2 out of 145 nodes were removed). If the graph was disconnected, the largest connected component was taken into account. We assigned weights to the edges from the calculated propagation latencies (based on the geodesic distance). Also, the shortest path lengths between nodes were calculated using the Dijkstra algorithm.

2) Parameter Settings: Table II shows the assigned values to different parameters of the problem in our experiments. Uniform capacities were associated to the potential controllers while both homogeneous and heterogeneous traffic loads for the switches were considered. For the heterogeneous case, the traffic loads of switches (as integer numbers) were uniformly distributed in $(0, 400]$ $k\text{req}/s$. All the applied values were based on prior studies on the capacitated *CPP* [10], [20], [25] as well as the research conducted on the performance of SDN controllers [52], [53]. Considering the heterogeneous load for the switches, 50 independent experiments were conducted to obtain the results. The resilience level r was set to 1, which indicates the *capacitated CPP* (i.e., no resilience), and 2 and 3 to specify *RCCPP*. Note that $r = 2$ and $r = 3$ indicate the resilience against single controller node failures and dual-controller node failures, respectively. The values for $cc_{\max}$ and $sc_{\max}$ were chosen as a percentage of D_G (the largest possible propagation latency for a given topology) for keeping the consistency across various topologies. However, network designers may choose specific values according to the QoS requirements of a specific network.

3) Obtaining the Results: The Python interface of the GUROBI optimization software (version 6.5.2) [54] was used to obtain the optimal solutions. Furthermore, a Python code was developed to solve *RCCPP* based on the proposed algorithms. All the experiments were carried out on an Intel Core i7-3770 CPU @3.40GHz and 32GB RAM with Windows 10 Pro (64-bit) installed. For each topology, the results shed light on the feasibility of using certain switch-controller and inter-controller latency values to satisfy a resilience level while minimizing the number of controllers.

In the following, we first compare our scheme with *CNCP* [15], which is the most relevant existing work on the *resilient CPP* to ours. Then, we make a comparison between our proposed algorithms w.r.t. their solution quality. We also provide a sensitivity analysis by considering the impact of different factors (including topology, switch-controller/inter-controller latency thresholds, and the controller capacity) on the number of controllers (our objective function). The change in the number of controllers may subsequently affect the utilization of the controllers. Furthermore, we discuss the dominant role of $sc_{\max}$ on the infeasibility of the problem instances. Finally, we briefly present the viability of incorporating the protection against single link failures that affect the connectivity between a switch and its assigned controllers.

TABLE II
PARAMETERS AND THEIR ASSOCIATED VALUES

Parameter	**Value**
Controller capacities (u_c)	$\{2000, 5000, 10000\}$ $k\text{req/s}$
Homogeneous traffic load for the switches (l_s)	200 $k\text{req/s}$
Heterogeneous traffic load for the switches (l_s)	$(0, 400]$ $k\text{req/s}$
Number of experiments	50
Resilience level (r)	$\{1, 2, 3\}$
$cc_{\max}$	$\{1, 0.8, 0.6\}D_G$
$sc_{\max}$	$\{1, 0.8, 0.6, 0.4\}D_G$

B. Comparison With CNCP

We delineated the differences between our problem formulation and *CNCP* in Section II-B. To compare *RCCPP* with *CNCP*, the optimal solutions to *RCCPP* and *CNCP* were acquired for small to large topologies. Particularly, we set $cc_{\max} = 0.6D_G$ (corresponding to parameter γ in *CNCP*), $r = 2$ (corresponding to parameter Q in *CNCP*), and $u_c = 2000$ $k\text{req}/s$, and we considered a homogeneous switch traffic load. Then, we solved *RCCPP* using the formulation in Section III by setting $sc_{\max} = 0.6D_G$. Next, we decreased the value of $sc_{\max}$ (usually by steps of 0.05 or 0.1) such that *RCCPP* was still feasible with the same number of controllers (as the case for $sc_{\max} = 0.6D_G$) and recorded the solutions. Afterwards, *CNCP* was solved by limiting the number of controllers to the value obtained by *RCCPP*.

The key differences in the output of the two schemes lie in the location and assignment decisions, which subsequently affect the load imbalance (i.e., the difference between the load of the controller with the lowest remaining capacity and that of the controller with the highest remaining capacity). For most of the topologies, *RCCPP* provides a load imbalance which is equal to or lower than that of *CNCP* with almost similar maximum switch-controller latency. A lower load imbalance is desirable due to having a better load distribution among the controllers as well as the decline in the queueing delay of the controllers. Table III summarizes the results of the topologies for which *RCCPP* achieves a lower load imbalance than *CNCP*. This stems from the fact that the switches are not necessarily connected to their nearest controllers in *RCCPP*. It should be noted that the fifth column of Table III shows the values of $sc_{\max}$ for acquiring the corresponding solution by *RCCPP*. For topologies such as Oxford, AT&T, Hibernia, NIIF, and CESNET, both schemes achieve a zero load imbalance with the average controller utilization of 100%.

For few topologies, the load imbalance of *RCCPP* is more than that of *CNCP*, which mainly results from the trade-off between satisfying the constraint related to $sc_{\max}$ for *RCCPP* and the load on the controllers. As an example, we consider

TABLE III
COMPARISON BETWEEN *RCCPP* AND *CNCP* IN TERMS OF LOAD IMBALANCE

Topology	Size	Number of Controllers	Average Controller Utilization	$sc_{\max}$	Load Imbalance (*RCCPP*)	Load Imbalance (*CNCP*)
GlobalCenter	9	3	60%	$0.59D_G$	600 $k$req/s	800 $k$req/s
GridNet	9	3	60%	$0.56D_G$	1000 $k$req/s	1200 $k$req/s
NAVIGATA	13	3	86.67%	$0.52D_G$	600 $k$req/s	800 $k$req/s
GoodNet	17	4	85%	$0.6D_G$	400 $k$req/s	800 $k$req/s
BELNET	19	16	95%	$0.538D_G$	1000 $k$req/s	1200 $k$req/s
KAREN	23	5	92%	$0.5D_G$	400 $k$req/s	600 $k$req/s
PSINet	24	5	96%	$0.45D_G$	200 $k$req/s	400 $k$req/s
UUNET	42	9	93.33%	$0.4D_G$	400 $k$req/s	600 $k$req/s
GARR	48	10	96%	$0.5D_G$	200 $k$req/s	400 $k$req/s
DFN	51	11	92.73%	$0.5D_G$	800 $k$req/s	1000 $k$req/s

the Abilene topology with 11 nodes. The set of controllers in a solution obtained by *RCCPP* is {6, 7, 8} with the load imbalance of 1400 $k$req/s whereas it is {6, 7, 10} with the load imbalance of 600 $k$req/s for *CNCP*. The value of $sc_{\max}$ for *RCCPP* was set to $0.53D_G$, which corresponds to the maximum allowed switch-controller latency of 12.97 ms. Regarding *RCCPP*, the assignments of the first and second controllers to each switch must satisfy the aforementioned latency, and thus, the maximum switch-controller latency in its solution is 12.86 ms (delay between the switch at node 3 and one of its assigned controllers at node 7). However, the maximum switch-controller latency in the solution obtained by *CNCP* is 14.71 ms, which results from the assignment of the controller at node 7 to the switch at node 5 as its secondary controller (not a valid assignment in *RCCPP*). The primary controller of the switch at node 5 is placed at node 6, which is the nearest to node 7 (4.52 ms). This is due to the fact that the objective of *CNCP* is to minimize the maximum (for all switches) of the sum of the delay from the switch to its first reference controller (which is the nearest controller to the switch with enough capacity) and the delay from the first reference controller to the second reference controller (which is chosen as the closest node to the first reference controller with enough capacity). Thus, using *CNCP*, the assignment of the controllers to the switches does not necessarily lead to having the lowest possible switch-controller latency which has a higher priority compared with the load imbalance for delay sensitive applications. Finally, we conclude that *RCCPP* is more flexible for achieving a good trade-off between the distribution of the load among the controllers and the maximum switch-controller latency. As we mentioned earlier, in contrast to *CNCP*, *RCCPP* is independent from the master controller selection process, simpler, and easily extendable, as well as it is solvable by a near-optimal and polynomial-time algorithm.

C. Solution Quality of Our Proposed Algorithms

Fig. 3 shows the (sorted) gap between the results obtained by our proposed algorithms and the optimal solution (OPT) considering a homogeneous traffic load for the switches, $u_c = 2000$ $k$req/s, $cc_{\max} = 0.8D_G$, and $sc_{\max} = 0.6D_G$. We sorted the topologies based on this gap in an increasing order (in each subfigure, the order is different accordingly). The presented results in this figure demonstrate the quality of the solutions acquired by both of the proposed algorithms due to their small gap with OPT (at most 1.5×OPT for *RCCPP-AMC* and 2×OPT for *RCCPP-SMC*). For instance, considering the Oxford topology in Fig. 3b, the gap shown on the y-axis is 0.5, which corresponds to 1.5×OPT. Particularly, the value on the y-axis is calculated by subtracting the optimal value from the acquired number of controllers with *RCCPP-SMC*, divided by the optimal value.

RCCPP-AMC achieves an optimal solution for around 60% of the topologies w.r.t. resilience against single controller node failures and dual-controller node failures, respectively (Fig. 3c and Fig. 3e). However, *RCCPP-SMC* obtains an optimal solution for around 30% of the topologies for the same failure cases (Fig. 3d and Fig. 3f). Obviously, when we have no resilience (Fig. 3a and Fig. 3b), both algorithms lead to solutions of higher quality. Note that for very large topologies such as TATA and Cogent, the solver was unable to obtain the optimal results in a reasonable amount of time. Therefore, we used the equality of the objective value obtained by either of the proposed algorithms and our calculated lower bound (in Section IV) as an indicator of acquiring an optimal solution. Table IV shows the number of required controllers and the execution time for such topologies.

D. Sensitivity Analysis

1) Impact of Topology: We analyze the impact of topology on the number of controllers by considering three aspects, including the size of the topology, its shape and the shortest path lengths.

Size of the topology: As shown in Fig. 4, the average number of controllers increases linearly w.r.t. the network size (N) for both the optimal solution and the solution provided by *RCCP-SMC* for a heterogeneous switch traffic load, $r = 2$, $u_c = 2000$ $k$req/s, $cc_{\max} = 0.8D_G$, and $sc_{\max} = 0.6D_G$ (a similar trend applies to other parameter settings). The average number of controllers for all of the analyzed topologies ranges from around 30% (for $u_c = 2000$ $k$req/s) to 15% (for $u_c = 10,000$ $k$req/s) of their size. It should be noted that for topologies with a very large network size, such as TATA, Colt Telecom, and Cogent, only the results from *RCCPP-SMC* are shown since the solver was unable to obtain the optimal results in a reasonable amount of time (the blue line is discontinued for $N > 80$ in Fig. 4).

Shape of the topology: By examining the results for the topologies of the same size, we found that hub-and-spoke topologies (star-like) usually require more controllers or even the problem tends to become infeasible more easily. Examples of the former case are KREONET with $N = 13$ (the same size as Navigata and GRENA, which are linear topologies) and

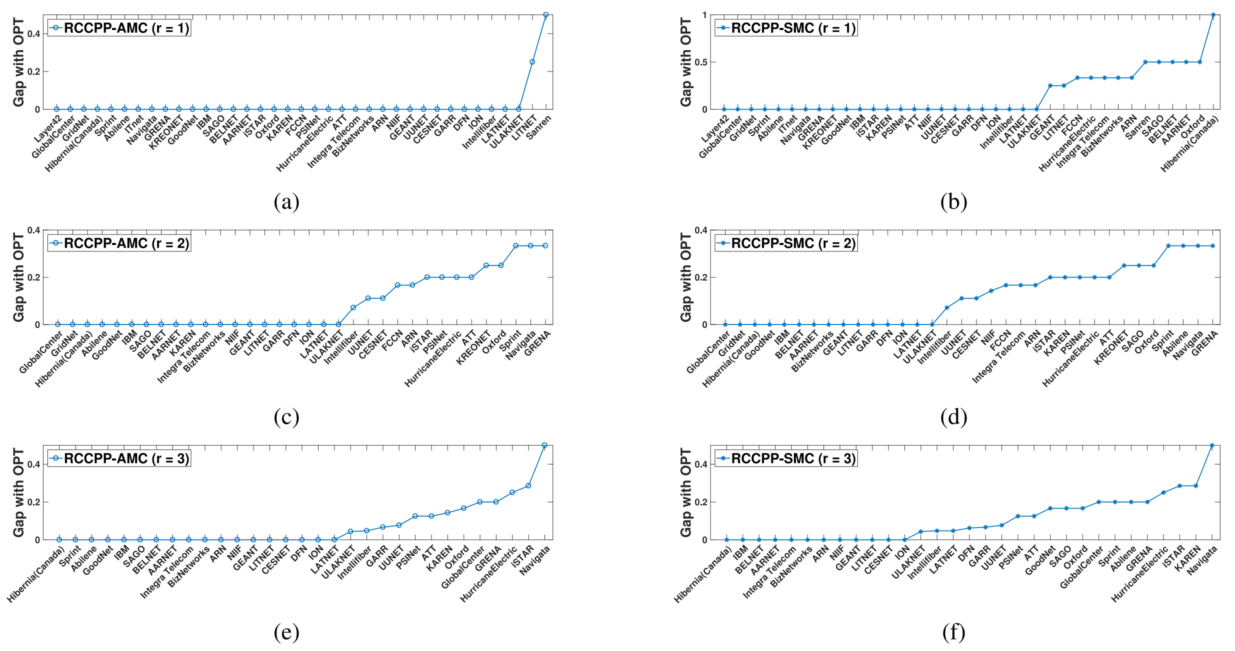


Fig. 3. Gap between the proposed algorithms and the optimal solution.

TABLE IV
THE VALUE OF THE OBJECTIVE FUNCTION AND EXECUTION TIME OF THE PROPOSED ALGORITHMS FOR LARGE TOPOLOGIES

Topology	Size	Lower Bound	Algorithm	Number of Controllers	Execution Time (s)
TATA	143	29	RCCPP-AMC	29	491.07
			RCCPP-SMC	30	196.26
Colt Telecom	146	30	RCCPP-AMC	30	247.39
			RCCPP-SMC	30	194.85
Cogent	180	36	RCCPP-AMC	36	779.46
			RCCPP-SMC	37	513.99

ARN with $N = 28$ (the same size as BizNetworks which is a linear topology). ITnet with 11 nodes (the same size as Sprint and Abilene which are mesh-like topologies) exemplifies the latter case (i.e., the problem is infeasible) which causes the graph to be discontinuous at this point. The reason is mainly due to the higher number of spokes (nodes with degree one). Particularly, since more than one controller is assigned to a switch and all the capacity and delay constraints must be satisfied, it is likely to place a controller at a spoke (e.g., the controllers of KREONET are placed at nodes $\{0, 1, 2, 9\}$ out of which 0, 1, and 9 are spokes). If such spokes are far away from the nodes with the highest degrees (such as node 1 in KREONET), they serve a limited number of switches, and thus they increase the number of required controllers. Moreover, most of the hub-and-spoke-like topologies have a low diameter (less than 10 ms) compared with mesh and linear topologies. Hence, they satisfy lower delay bounds but at the expense of having more controllers. It should be noted that although some topologies have large diameters, there is not much room to decrease the values of $cc_{\max}$ and $sc_{\max}$. For example, HurricaneElectric (a linear topology) has a diameter of 147.75 ms; however, no feasible solution is found for $cc_{\max} = 0.6D_G$ (88.64 ms) and $sc_{\max} = 0.4D_G$ (59.09 ms), which necessitates the use of graph augmentation to satisfy lower delay bounds with the same parameter settings for the traffic load of switches and capacity of controllers.

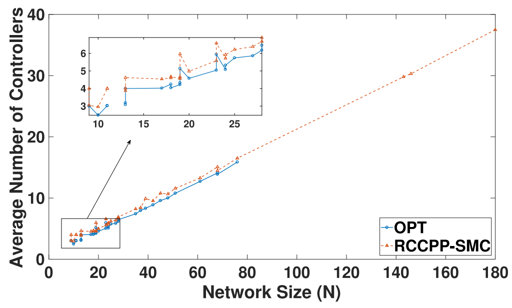
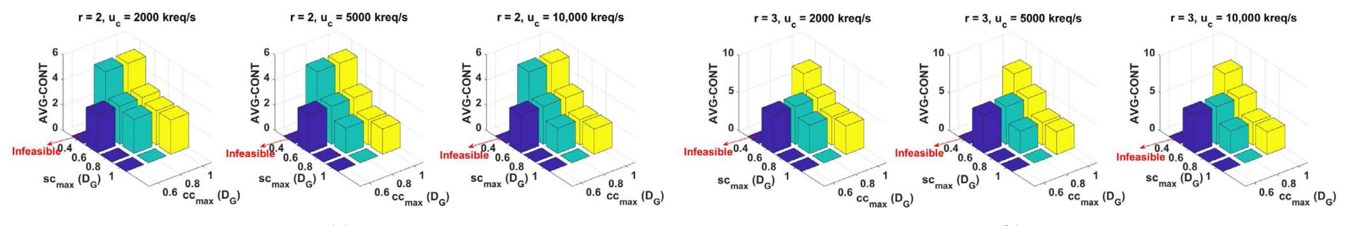


Fig. 4. Increase in the average number of the controllers w.r.t. network size for heterogeneous switch traffic load, $r = 2$, $u_c = 2000$ kreq/s, $cc_{\max} = 0.8D_G$, and $sc_{\max} = 0.6D_G$.

Shortest path lengths: From the experiments carried out on various topologies w.r.t. different values of cc_{max} and $sc_{\max}$, we can see that the shortest path lengths between the pairs of nodes have a great impact on the value of the objective function in contrast to the graph density and path redundancies. That is, if the topology has a lot of path redundancies and high density, we cannot conclude that it requires fewer controllers compared with its peers (i.e., topologies of the same size). For instance, GlobalCenter is a complete graph with the highest possible density and highest average node degree for a network size of 9. However, it needs almost the same number of controllers (for almost all the values of cc_{max} and $sc_{\max}$) as Gridnet, a topology with the same network size but less path redundancy. Another example is Integra Telecom with 27 nodes, that almost has the same size and average node degree as BizNetwoks, but with higher density.

2) Impact of $\boldsymbol{sc_{max}}$ *and* $\boldsymbol{cc_{max}}$*:* Changing the values of $cc_{\max}$ and $sc_{\max}$ helps the network operators with insights into the impact of such propagation latency bounds on the number of controllers, their respective locations, and the average controller utilization. More importantly, it sheds light on the



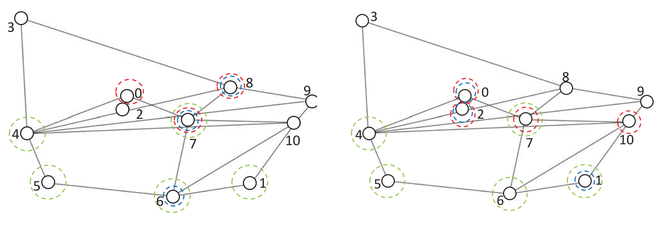
(a) (b)

Fig. 5. Impact of $cc_{\max}$, $sc_{\max}$, and u_c on the average number of required controllers in OPT for the Sprint topology.

situation when the problem becomes infeasible as well as the minimum possible propagation latency requirement that can be met by a specific topology without using any graph augmentation techniques or open search [20] (i.e., the controllers can be placed anywhere in the geographical area rather than being co-located with the switches). Also, it is possible to set both $cc_{\max}$ and $sc_{\max}$ with values of interest rather than using D_G. For instance, if 50 ms round-trip propagation latency between switches and each of their assigned controllers is needed, then the value of $sc_{\max}$ should be set to 25 ms.

The average number of controllers: The results in Fig. 5 indicate the dominant impact of $sc_{\max}$ compared with $cc_{\max}$ on the average number of controllers for the Sprint topology as a representative of a group of topologies. Particularly, for each topology, we kept the value of $cc_{\max}$ fixed and changed the value of $sc_{\max}$. In all of these cases, $\boldsymbol{G_p}$ remains the same. Thus, its density and the number of maximal cliques (in case of using *RCCPP-AMC*) are unchanged. Lower values of $sc_{\max}$ increase the average number of controllers or even result in the infeasibility of the problem. This is due to the fact that limiting the switch-controller latency would lead to fewer potential controllers (to be assigned to a switch), and subsequently it would diminish the possibility of assigning an existing controller in a partial solution to a new switch. However, for some of the topologies such as SAGO, BizNetworks, UUNET and DFN, if we change the value of $cc_{\max}$, the average number of needed controllers remains unchanged regardless of the value of $sc_{\max}$. One reason is that while decreasing the value of $sc_{\max}$, the total number of cliques of $\boldsymbol{G_p}$ (including 2-cliques and 3-cliques) does not vary much for such topologies. These topologies have denser $\boldsymbol{G_p}$ graphs and subsequently have more cliques and potential controller-switch assignments. It should be noted that when $cc_{\max} = sc_{\max} = D_G$, there is no delay requirement for the switch-controller latency and inter-controller latency. Furthermore, $cc_{\max} = D_G$ implies no delay requirement for the inter-controller latency (constraint (6) is relaxed), and hence $\boldsymbol{G_p} = \boldsymbol{G_o}$ is satisfied since $\boldsymbol{G_o}$ is not pruned. This results in having $\boldsymbol{G_p}$ as a complete graph and finding all maximal cliques is polynomially bounded, and thus *RCCPP-AMC* obtains a solution in polynomial time. If the original graph $\boldsymbol{G}$ is a complete graph (e.g., GlobalCenter) and $cc_{\max} = D_G$, it is possible that $\boldsymbol{G_p} = \boldsymbol{G_o} = \boldsymbol{G}$. Note that the value of $sc_{\max}$ is upper bounded by $cc_{\max}$, and hence no value is shown for the number of controllers in such cases in Fig. 5.

Controller locations: Fig. 6 illustrates the controller locations for the Sprint topology as an example of the set of experiments carried out for $r = 2$, homogeneous traffic load



(a) Sprint (OPT) (b) Sprint (RCCPP-SMC)

Fig. 6. Controller locations for the Sprint topology. Blue, red and green dashed circles indicate the controller locations for [$cc_{\max} = sc_{\max} = 0.8D_G$], [$cc_{\max} = 0.8D_G, sc_{\max} = 0.6D_G$], and [$cc_{\max} = 0.8D_G, sc_{\max} = 0.4D_G$], respectively.

for the switches and $u_c = 2000$ $k$req/s. Since we assume that the controllers have the same capacity, their number is only affected by the traffic load of switches. On the other hand, both the number and the location of controllers can be affected by the delay bounds. As shown in Fig. 6a, 3 controllers (in the optimal solution for $cc_{\max} = sc_{\max} = 0.8D_G$) are placed at nodes 6, 7, and 8 (indicated by blue circles). These controllers satisfy the total traffic load of 11 switches (with $r = 2$, the total load is 4400 $k$req/s). Since the capacity of each controller is 2000 $k$req/s, at least 3 controllers are required. In this case the values of the delay bounds do not affect the number of controllers but only their placements. For instance, the controller located at node 7 (Kansas City), serves the switches at node 4 (Stockton) and node 9 (New York), and the propagation latency between node 7 and the aforementioned switches (11.91 ms and 9.18 ms, respectively) is less than 19.28 ms ($<= 0.8D_G$). Another set of controllers ({0, 1, 2}), which is also optimal, is acquired by *RCCP-SMC* and depicted in Fig. 6b (denoted by blue circles). However, as an example, if we choose nodes 4 and 10, these two nodes cannot be a member of the set of controllers at the same time or one serves as the controller for the other one since the latency between them (i.e., 19.38 ms) does not satisfy the delay bounds. Decreasing the value of $sc_{\max}$ to $0.6D_G$ does not change the number of controllers (but it changes their placement as shown in Fig. 6a with red circles) for the optimal solution while *RCCPP-SMC* provides a solution with 4 controllers (red circles in Fig. 6b). The green circles in Fig. 6a and Fig. 6b show the controller locations when the value of $sc_{\max}$ is reduced to $0.4D_G$, which subsequently affects the number of controllers. Actually, Sprint is one of the few topologies for which reducing $sc_{\max}$ to a small portion of D_G results in costly solutions in terms of the number of the required controllers (almost 50% of the network size). This case usually happens for small topologies.

Controller utilization: Table V shows the average controller utilization for the Sprint topology in the optimal solution w.r.t. different values of $cc_{\max}$ and $sc_{\max}$ and controller capacities (for $r = 2$). Note that the dashes in the table indicate the infeasibility of the problem. The 2-tuples in the first column show the values of $cc_{\max}$ and $sc_{\max}$ as a ratio of D_G. If the values of $cc_{\max}$ and $sc_{\max}$ are reduced, both the number of controllers and their assignments to the switches change, and therefore they affect the average controller utilization. Similarly, for a fixed value of $cc_{\max}$, lowering the value of $sc_{\max}$ reduces the average controller utilization. Other topologies such as DFN, BizNetworks, ULAKNET, and TATA have the same average controller utilization regardless of the values of $cc_{\max}$ and $sc_{\max}$, which is consistent with having the same average number of controllers. For such topologies, the average controller utilization is around $80-90\%$ for resilience against single and dual-controller node failures.

Infeasibility of the problem: For most of the topologies, infeasibility is mainly caused by lower values of $sc_{\max}$ (no controller-switch assignment could be found to satisfy all of the constraints in the problem). Nevertheless, for few topologies such as Integra Telecom, the problem becomes infeasible when $cc_{\max} = 0.6D_G$ regardless of the value of $sc_{\max}$. As we mentioned in Section IV, one reason for the infeasibility of the problem is when $\boldsymbol{G_p}$ becomes disconnected. This happens for topologies such as iSTAR, BizNetworks, Sanren, Gridnet, Geant2012, ITnet, ARN, FCCN, LATNET, and ULAKNET, if we set $cc_{\max} = sc_{\max} = 0.4D_G$. However, as shown in Table II, we assume that the value of $cc_{\max}$ is at most $0.6D_G$ (in line with the existing works such as [15]).

3) Impact of Controller Capacity ($\boldsymbol{u}_c$): While increasing u_c leads to a lower number of controllers (more than 50% decrease) for some of the topologies such as ULAKNET, TATA and Cogent, it does not necessarily cause a decreasing trend for the others such as Sanren and Sprint (topologies with a small size). For instance, as illustrated in Fig. 5a, regardless of the capacity of the controllers, 5 controllers are needed when $cc_{\max} = D_G$ and $sc_{\max} = 0.4D_G$. However, the maximum total traffic load of all switches is lower than the total capacity provided by only 3 controllers. This mainly results from the reduced value of $sc_{\max}$, i.e., $0.4D_G$ compared with the scenario in which $cc_{\max} = D_G$ and $sc_{\max} = 0.6D_G$. Therefore, the number and set of the controller nodes that satisfy the switch-controller latency are mostly different from each other in the aforementioned two scenarios. The controllers are at nodes {1, 4, 5, 8, 9} in most of the experiments for the former case while the controllers are at nodes {0, 6, 8} for the latter case. A similar trend is observed for the results obtained by *RCCPP-SMC* with 6 and 4 controllers on average for the former and the latter cases, respectively. Moreover, the capacity of controllers has a reciprocal relationship with the average controller utilization. The reason is that by increasing the controller capacities for the same amount of the total load, the controllers are under-utilized (as shown in Table V). It should be noted that we have used practical values for the capacity of controllers (as in Table II) and the value of capacity is usually much higher than the traffic load of switches. Therefore, $\boldsymbol{u_c}$ is not a restrictive factor that affects the feasibility of the solutions obtained by the proposed algorithms whereas $cc_{\max}$ and $sc_{\max}$ have a more dominant role in this case as we discussed in Section V-D2.

TABLE V
AVERAGE CONTROLLER UTILIZATION (SPRINT)

$(cc_{\max}, sc_{\max})$	Resilience Level	$u_c = 2000$ $k$req/s	$u_c = 5000$ $k$req/s	$u_c = 10,000$ $k$req/s
(D_G, D_G)	r=2	80.6%	43.5%	21.74%
	r=3	86.35%	43.5%	21.74%
$(D_G, 0.8D_G)$	r=2	80.6%	43.5%	21.74%
	r=3	86.35%	43.5%	21.74%
$(D_G, 0.6D_G)$	r=2	71.9%	29%	14.5%
	r=3	64.9%	26.09%	13.04%
$(D_G, 0.4D_G)$	r=2	43.5%	17.4%	8.7%
	r=3	46.41%	18.64%	9.32%
$(0.8D_G, 0.8D_G)$	r=2	80.6%	43.5%	21.74%
	r=3	86.35%	43.5%	21.74%
$(0.8D_G, 0.6D_G)$	r=2	71.9%	29%	14.5%
	r=3	64.9%	26.09%	13.04%
$(0.8D_G, 0.4D_G)$	r=2	43.5%	17.4%	8.7%
	r=3	-	-	-
$(0.6D_G, 0.6D_G)$	r=2	71.9%	29%	14.5%
	r=3	64.9%	26.09%	13.04%
$(0.6D_G, 0.4D_G)$	r=2	-	-	-
	r=3	-	-	-

E. Single Link Failures

As we discussed in Section III-C, our problem formulation can be extended to include the resilience against single link failures in addition to the controller node failures. Although having link-disjoint paths between a switch and all of its assigned controllers (whether the primary controller or the backup ones) may result in infeasibility for some of the problem instances, it is needed to assure the tolerance of single link failures. This infeasibility issue can be alleviated by using graph augmentation mechanisms or at least maximizing the number of such link-disjoint paths. It should be noted that the aforementioned constraint can be easily incorporated into both of the proposed algorithms (by choosing the r or $(r+1)$-cliques that contain link-disjoint paths between a switch and its assigned controllers after obtaining $\boldsymbol{G_p}$). Since this is not our key focus in this paper, here we just report some of our observations when applying this additional constraint to our problem formulation. Mesh topologies tend to be less affected by the infeasibility issue due to their higher density and path redundancy (the number of controllers increases for some values of the delay bounds and few of the previously feasible problem instances become infeasible) whereas hub-and-spoke and linear topologies are affected, especially when the value of $cc_{\max}$ is decreased. For instance, considering the DFN topology (a mesh topology with $N = 51$), applying the link-disjoint path constraint results in the infeasibility of the problem for $r = 2$, $cc_{\max} = 0.8D_G$ and $sc_{\max} = 0.4D_G$ as well as it causes no increase in the maximum number of assigned controllers for other scenarios. On the other hand, for the CESNET topology (a hub-and-spoke topology with $N = 45$), the problem becomes infeasible for $r = 2$ and $cc_{\max} < D_G$ as well as the maximum number of assigned controllers is increased by 3 times for other scenarios.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed two algorithms for *RCCPP*, which provide high-quality and cost-saving solutions. By finding the maximal cliques, we narrow down the search space and the optimal solution (if any) is always a subset of one of the maximal cliques. The effectiveness of the proposed solutions was extensively analyzed w.r.t. various parameter settings for a wide range of real WAN topologies. Such an analysis can assist the network operators with helpful insights into the design/modification and management of their SDN-based networks to meet different SLAs. The proposed solutions can be easily amended to cover node or link failures even with different objective functions (e.g., minimizing the expected control path loss). Another direction is to look into the dynamic *RCCPP* which changes the controller-switch assignments based on the time-varying traffic load of switches.

REFERENCES

[1] Jennifer English. (2016). *2016 SDN Trends: The Year of the Software-Defined WAN*. [Online]. Available: https://goo.gl/a2i9CY

[2] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," in *Proc. ACM SIGCOMM*, Hong Kong, 2013, pp. 3–14.

[3] R. Ahmed and R. Boutaba, "Design considerations for managing wide area software defined networks," *IEEE Commun. Mag.*, vol. 52, no. 7, pp. 116–123, Jul. 2014.

[4] M. Tanha, D. Sajjadi, and J. Pan, "Enduring node failures through resilient controller placement for software defined networks," in *Proc. IEEE GLOBECOM*, Washington, DC, USA, 2016, pp. 1–7.

[5] A. S. da Silva, P. Smith, A. Mauthe, and A. Schaeffer-Filho, "Resilience support in software-defined networking: A survey," *Comput. Netw.*, vol. 92, pp. 189–207, Dec. 2015.

[6] D. Tipper, "Resilient network design: Challenges and future directions," *Telecommun. Syst.*, vol. 56, no. 1, pp. 5–16, 2014.

[7] ONF. (2015). *OpenFlow Switch Specification-Version 1.5.1*. [Online]. Available: https://goo.gl/jE2JTW

[8] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 473–478, Sep. 2012.

[9] A. Sallahi and M. St-Hilaire, "Optimal model for the controller placement problem in software defined networks," *IEEE Commun. Lett.*, vol. 19, no. 1, pp. 30–33, Jan. 2015.

[10] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," *IEEE Commun. Lett.*, vol. 18, no. 8, pp. 1339–1342, Aug. 2014.

[11] Y. Jiménez, C. Cervelló-Pastor, and A. J. García, "On the controller placement for designing a distributed SDN control layer," in *Proc. IFIP Netw.*, Trondheim, Norway, 2014, pp. 1–9.

[12] A. Ksentini, M. Bagaa, T. Taleb, and I. Balasingham, "On using bargaining game for optimal placement of SDN controllers," in *Proc. IEEE ICC*, Kuala Lumpur, Malaysia, 2016, pp. 1–6.

[13] 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 *Proc. IEEE ICC*, Kuala Lumpur, Malaysia, 2016, pp. 1–6.

[14] S. Lange *et al.*, "Heuristic approaches to the controller placement problem in large scale SDN networks," *IEEE Trans. Netw. Service Manag.*, vol. 12, no. 1, pp. 4–17, Mar. 2015.

[15] B. P. R. Killi and S. V. Rao, "Capacitated next controller placement in software defined networks," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 3, pp. 514–527, Sep. 2017.

[16] M. F. Bari *et al.*, "Dynamic controller provisioning in software defined networks," in *Proc. CNSM*, Zürich, Switzerland, 2013, pp. 18–25.

[17] H. K. Rath, V. Revoori, S. M. Nadaf, and A. Simha, "Optimal controller placement in software defined networks (SDN) using a non-zero-sum game," in *Proc. IEEE WoWMoM*, Sydney, NSW, Australia, 2014, pp. 1–6.

[18] N. Perrot and T. Reynaud, "Optimal placement of controllers in a resilient SDN architecture," in *Proc. DRCN*, Paris, France, 2016, pp. 145–151.

[19] Q. Zhong, Y. Wang, W. Li, and X. Qiu, "A min-cover based controller placement approach to build reliable control network in SDN," in *Proc. IEEE/IFIP NOMS*, Istanbul, Turkey, 2016, pp. 481–487.

[20] M. T. I. ul Huque, W. Si, G. Jourjon, and V. Gramoli, "Large-scale dynamic controller placement," *IEEE TNSM*, vol. 14, no. 1, pp. 63–76, Mar. 2017.

[21] T. Y. Cheng, M. Wang, and X. Jia, "QoS-guaranteed controller placement in SDN," in *Proc. IEEE GLOBECOM*, San Diego, CA, USA, 2015, pp. 1–6.

[22] A. Sallahi and M. St-Hilaire, "Expansion model for the controller placement problem in software defined networks," *IEEE Commun. Lett.*, vol. 21, no. 2, pp. 274–277, Feb. 2017.

[23] T. Zhang, P. Giaccone, A. Bianco, and S. De Domenico, "The role of the inter-controller consensus in the placement of distributed SDN controllers," *Comput. Commun.*, vol. 113, pp. 1–13, Nov. 2017.

[24] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Proc. 2nd USENIX Hot-ICE*, 2012, p. 10.

[25] 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 *Proc. IEEE GLOBECOM*, Austin, TX, USA, 2014, pp. 1909–1915.

[26] H. Aoki, J. Nagano, and N. Shinomiya, "Network partitioning problem to reduce shared information in OpenFlow networks with multiple controllers," in *Proc. ICN*, 2015, pp. 250–255.

[27] H. Aoki and N. Shinomiya, "Controller placement problem to enhance performance in multi-domain SDN networks," in *Proc. ICN*, Lisbon, Portugal, 2016, p. 120.

[28] J. Liao *et al.*, "Density cluster based approach for controller placement problem in large-scale software defined networkings," *Comput. Netw.*, vol. 112, pp. 24–35, Jan. 2017.

[29] P. Vizarreta, C. M. Machuca, and W. Kellerer, "Controller placement strategies for a resilient SDN control plane," in *Proc. RNDM*, Halmstad, Sweden, 2016, pp. 253–259.

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

[31] F. J. Ros and P. M. Ruiz, "On reliable controller placements in software-defined networks," *Comput. Commun.*, vol. 77, pp. 41–51, Mar. 2016.

[32] T. Lukovszki, M. Rost, and S. Schmid, "It's a match! Near-optimal and incremental middlebox deployment," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 46, no. 1, pp. 30–36, 2016.

[33] M. Abu-Lebdeh, D. Naboulsi, R. Glitho, and C. W. Tchouati, "On the placement of VNF managers in large-scale and distributed NFV systems," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 4, pp. 875–889, Dec. 2017.

[34] F. J. Ros and P. M. Ruiz, "On reliable controller placements in software-defined networks," *Comput. Commun.*, vol. 77, pp. 41–51, Mar. 2016.

[35] N. Beheshti and Y. Zhang, "Fast failover for control traffic in software-defined networks," in *Proc. IEEE GLOBECOM*, Anaheim, CA, USA, 2012, pp. 2665–2670.

[36] Y. Zhang, N. Beheshti, and M. Tatipamula, "On resilience of split-architecture networks," in *Proc. IEEE GLOBECOM*, Kathmandu, Nepal, 2011, pp. 1–6.

[37] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability-aware controller placement for software-defined networks," in *Proc. IFIP/IEEE IM*, Ghent, Belgium, 2013, pp. 672–675.

[38] M. Guo and P. Bhattacharya, "Controller placement for improving resilience of software-defined networks," in *Proc. ICNDC*, Los Angeles, CA, USA, 2013, pp. 23–27.

[39] D. Hock *et al.*, "Pareto-optimal resilient controller placement in SDN-based core networks," in *Proc. ITC*, Shanghai, China, 2013, pp. 1–9.

[40] D. Hock, S. Gebert, M. Hartmann, T. Zinner, and P. Tran-Gia, "POCO-framework for Pareto-optimal resilient controller placement in SDN-based core networks," in *Proc. IEEE/IFIP NOMS*, Kraków, Poland, 2014, pp. 1–2.

[41] B. Behsaz, M. R. Salavatipour, and Z. Svitkina, "New approximation algorithms for the unsplittable capacitated facility location problem," *Algorithmica*, vol. 75, no. 1, pp. 53–83, 2016.

[42] G. P. McCormick, "Computability of global solutions to factorable non-convex programs: Part I—Convex underestimating problems," *Math. Program.*, vol. 10, no. 1, pp. 147–175, 1976.

[43] M. C. Golumbic and I. B.-A. Hartman, *Graph Theory, Combinatorics and Algorithms: Interdisciplinary Applications*, New York, NY, USA: Springer, 2005.

[44] V. Vassilevska, "Efficient algorithms for clique problems," *Inf. Process. Lett.*, vol. 109, no. 4, pp. 254–257, 2009.

[45] J. W. Moon and L. Moser, "On cliques in graphs," *Israel J. Math.*, vol. 3, no. 1, pp. 23–28, 1965.

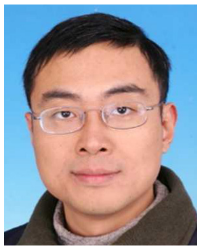
[46] C. Bron and J. Kerbosch, "Algorithm 457: Finding all cliques of an undirected graph," *Commun. ACM*, vol. 16, no. 9, pp. 575–577, 1973.

[47] B. Rosgen and L. Stewart, "Complexity results on graphs with few cliques," *Discr. Math. Theoret. Comput. Sci.*, vol. 9, no. 1, pp. 127–136, 2007.
[48] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa, "A new algorithm for generating all the maximal independent sets," *SIAM J. Comput.*, vol. 6, no. 3, pp. 505–517, 1977.
[49] D. Eppstein, M. Löffler, and D. Strash, "Listing all maximal cliques in large sparse real-world graphs," *ACM J. Exp. Algorithmics*, vol. 18, p. 3, Nov. 2013.
[50] S. S. Skiena, *The Algorithm Design Manual*. London, U.K.: Springer, 2008.
[51] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.
[52] Y. Zhao, L. Iannone, and M. Riguidel, "On the performance of SDN controllers: A reality check," in *Proc. IEEE NFV-SDN*, San Francisco, CA, USA, 2015, pp. 79–85.
[53] S. Mallon, V. Gramoli, and G. Jourjon, "Are today's SDN controllers ready for primetime?" in *Proc. IEEE LCN*, 2016, pp. 325–332.
[54] *GUROBI Optimizer*. Accessed: Mar. 25, 2016. [Online]. Available: http://www.gurobi.com/

Maryam Tanha (S'12) received the B.Sc. degree in computer software engineering from Yazd University, Iran, in 2005 and the M.Sc. degree in communication and network engineering from the University of Putra Malaysia, in 2013. She is currently pursuing the Ph.D. degree with the Department of Computer Science, University of Victoria, Canada. Her research interests are software-defined networking, resiliency for communication networks, and wireless mesh networks. She is a Student Member of the ACM.

Dawood Sajjadi (S'12) received the B.Sc. degree in computer engineering from the University of Bahonar Kerman, Iran, in 2004 and the M.Sc. degree in communication and network engineering from the University of Putra Malaysia in 2013. He is currently pursuing the Ph.D. degree with the University of Victoria, Canada. As a Post-Graduate student and a Research Assistant, he conducted the M.Sc. Project in Wireless Communication Cluster with MIMOS Berhad (National ICT Research Center of Malaysia) for about two years. He was a recipient of the Fellowship Award to start the Ph.D. Program in Computer Science with the University of Victoria in 2014. Since then, he is pursuing his research on wireless mesh networks, WLANs, and software-defined networking. He is a Student Member of the ACM.

Rukhsana Ruby (S'15) received the master's degree from the University of Victoria, Canada, in 2009 and the Ph.D. degree from the University of British Columbia, Canada, in 2015. From the broader aspect, her research interests include the management and optimization of next generation wireless networks. She has authored nearly 40 papers in well-recognized journals and conferences. She has served as the Lead Guest Editor for the special issue on NOMA techniques under *EURASIP JWCN* in 2017. She has also served as a technical program committee member for various conferences.



Jianping Pan (SM'08) is currently a Professor of computer science with the University of Victoria, Canada. He did his Post-Doctoral Research with the University of Waterloo, Canada. He was with Fujitsu Labs and NTT Labs. His area of specialization is computer networks and distributed systems, and his current research interests include protocols for advanced networking, performance analysis of networked systems, and applied network security. He was a recipient of the IEICE Best Paper Award in 2009, the Telecommunications Advancement Foundation's Telesys Award in 2010, the JSPS Invitation Fellowship in 2012, and the Best Paper Awards for WCSP'11, IEEE Globecom'11, and IEEE ICC'13. He has been serving on the technical program committees of major computer communications and networking conferences, including IEEE INFOCOM, ICC, Globecom, WCNC, and CCNC. He is the Ad Hoc and Sensor Networking Symposium Co-Chair of IEEE Globecom'12. He is a Senior Member of the ACM.