Journal of AI and Data Mining
Vol 8, No 2, 2020, 269-277.
DOI: 10.22044/JADM.2019.6319.1745

A New Reliable Controller Placement Model for Software-Defined WANs

A. Jalili[1*] and M. Keshtgari [2]

1. Department of Computer Engineering, Gonbad Kavous University, Gonbad Kavous, Iran.
2. Department of Computer Engineering and Information Technology, Shiraz University of Technology, Shiraz, Iran.

Received 15 October 2017; Revised 21 July 2018; Accepted 30 November 2019
**Corresponding author: jalili@gonbad.ac.ir (A.Jalili).*

Abstract
Software-Defined Network (SDN) is a decoupled architecture that enables administrators to build a customizable and manageable network. Although the decoupled control plane provides a flexible management and facilitates the task of operating the network, it is the vulnerable point of failure in SDN. In order to achieve a reliable control plane, multiple controllers are often required so that each switch must be assigned to more than one controller. In this paper, the Reliable Controller Placement Problem Model (RCPPM) is proposed to solve such a problem so as to maximize the reliability of SDN. Unlike the previous works that only consider the latencies parameters, the new model takes into account the load of control traffic and reliability metrics as well. Furthermore, a near-optimal algorithm is proposed to solve the NP-hard RCPPM in a heuristic manner. Finally, through an extensive simulation, a comprehensive analysis of RCPPM is presented for various topologies extracted from Internet Topology Zoo. Our performance evaluations show the efficiency of the proposed framework.

Keywords: *Software-Defined Network, Reliable Controller Placement, Controller, Latency, Graph theory.*

1. Introduction

The Software-Defined Network (SDN) is a promising architecture that can overcome the challenges facing the traditional networks. Unlike the traditional networks, where both the control and the data planes are tightly coupled on the same boxes, it decouples the control and data planes [1]. Such a separation architecture enables the administrator to build a simpler, customizable, programmable, and manageable network. In SDN, the network owners can dynamically and efficiently configure their network by the external intelligent elements called the controllers [2].

Recently, a substantial attention has been paid to the SDN concepts extending into wide area networks (WANs) and carrier networks. Utilizing the advantages of the logically centralized control of this architecture, it is possible for the Carrier Network Infrastructure/WAN organizations to simplify and optimize the management of their network.

Today, the WAN/carrier technologies are facing a rapid growth that provides remarkable characteristics and benefits from high availability, high resiliency, high scalability, and high reliability. For the failure recovery, some networks offer a carrier-grade quality, meaning that a network should recover from failures within 50 ms. For instance, SONET/SDH has a specific protection strategy to provide a high availability of service and it can achieve the restoration time after failure in the order of 50 ms [3]. Achieving a highly resilient communication is one of the major goals of networking. As a replacement for other well-established technologies, SDNs per sec are expected to yield the same levels of resiliency as the legacy alternative technologies in WAN.

One of the resilience mechanisms used to smooth the failures effect is to incorporate the redundancy into the network design [4]. For example, B4, a large Google's project in SDN, uses the replica controllers for a switch to protect network in the events of destruction. Therefore, due to the resiliency issues, multiple controllers are required. One especially crucial task in SDN networks is controller placement, i.e. locating a restricted

number of controllers in a network so that several requirements are satisfied.

The previous approaches only consider the latency between the controllers and switches. They disregard either the inter-controller latency or the capacities of the controllers. Both of these factors are critical in the context of real networks. Although the latency between the controllers and switches constitutes a crucial metric in positioning the controllers properly, other objectives are required to be regarded as well. Therefore, the objectives like the latency between each switch and its assigned controller, inter-controller latency, load balancing and failure in nodes, links, and controllers should be taken into account. Hence, the controller placement problem can be formulated as a multi-objective combinatorial optimization (MOCO) problem [5, 6]. Since the objectives are supposed to be pairwise conflicting, applying multi-objective approaches allow for a more obvious demonstration of the trade-offs between the possibly competing criteria [7].

In this paper, a new Controller Placement Problem Model (RCPPM) is proposed to maximize the network reliability. Unlike the recent works that often consider the latency between the switch and controller, the proposed model takes other important metrics into account. These metrics include the control traffic of switches, controller's capacity, and latency between the controllers. Secondly, using the clique concept, a near-optimal and time-efficient solution scheme is proposed to solve the NP-hard RCPPM in a heuristic manner. The clique is an important concept in the graph theory. Also known as a complete graph, it is defined as a graph where every vertex is adjacent to every other. Finally, through an extensive simulation, an exhaustive analysis of RCPPM is presented for various topologies (ranging from small- to large-scale WAN graphs) under some parameter settings.

The rest of this paper is stated as what follows. In Section 2, an overview of the existing works on RCPPM in software-defined networks is stated. Section 3 introduces the proposed model and the problem formulation. In Section 4, the proposed solution is provided. Section 5 evaluates the performance evaluation of the proposed solution. Finally, Section 6 draws the conclusion and provides some future research directions.

2. Related works

Please In the following, a survey of the main existing studies on the controller placement problem from the reliability perspective is presented.

Heller et al. [8] have motivated the controller placement problem and advocates its relevance. It examines the impacts of the controller placement on average and the worst-case propagation latencies for real-world topologies. Thus there is a guarantee for finding optima with respect to the latency. These optima are used to derive guidelines for dimensioning the control plane. For example, most of the investigated topologies require only one controller to comply with the realistic latency constraints.

Xiao et al. [9] have provided a challenge by focusing on two specific questions: how to partition a wide-area network topology into several small SDN domains and the way that the controller should go in each SDN domain. They aimed at maximizing the reliability of controller as well as minimizing the latency of Wide Area Network (WAN) using the spectral clustering placement algorithm. However, in their clustering approaches, the importance of the assignment process and the path between the control plane and data plane have not been considered.

A new reliable controller placement framework is designed in [1], which considers both the control plane architecture and relation between the control and data planes. The framework is considered as a multi-objective optimization model with two objective functions to minimize the flow setup time and inter-controller latency. In order to solve the framework, a multi-objective algorithm called Non-dominated Sorting-Moth Flame Controller Placement Optimizer (NS-MFCPO) is designed. The authors compare the proposed framework with other models using the expected path loss and link load balancing metrics. The results on the real Wide Area Network topologies show the efficiency of the proposed framework.

Guo in [10] focuses on the controller placement for network resilience improvement in SDN. The author first analyzed the impact of the controller placement on SDN resilience from the perspective of interdependent networks. Then a new resilience metric based on the cascading failure analysis on the interdependence graph is designed.

In [11], two strategies to address the Reliable Controller Placement (RCP) problem is provided. In the first method, each switch connects to a controller over two Disjoint Control Paths. In the second method, switches connect to two Different Controller Replicas over two disjoint paths. Both strategies have been compared in terms of control path length and expected control path loss. The results obtained show that the two methods significantly improve the resilience of the control

plane. In order to achieve a high south-bound reliability, the research work in [12] has introduced a framework for Pareto-based Optimal COntroller placement (POCO) that provides the operators with Pareto optimal placements with respect to different performance metrics. This framework considers some important metrics like scalability, resilience, and control plane communication delays. In this paper a new specialized heuristic algorithm is introduced, which takes into account a particular set of optimization objectives and return solutions representing the possible trade-offs between them.
Some papers formulate the controller placement problem as a Multi-Objective Combinatorial Optimization (MOCO) problem and some important objectives have been proposed [2, 13, 14]. In [13], other important objectives in network, which play key roles in deciding the location of controllers, are proposed. These objectives comprise the latency between each switch and its assigned controller, latency between each pair of controllers, and load balancing among the controllers. Jalili et al. have provided a multi-objectives genetic algorithm–based solution for CPP [2]. They have proposed the heuristics-based NSGA-II to solve the controller placement problem. In [14], the authors have developed a specialized heuristic to optimize the same objectives called Pareto Capacitated k-Medoids (PCKM). They investigated PCKM by considering a particular set of optimization objectives and returning solutions representing the possible trade-offs between them. New multi-objective algorithms and location of controllers have been evaluated based on these objectives. The main challenge of these multi-objective models is the lack of analysis for the assignment paths and its effective factors.
Generally, some papers have investigated RCPP based on only the load of control traffic and the controller's capacity [3, 12]. Moreover, some papers considered the dynamic traffic and re-assignment mechanisms but did not consider the cost of re-assignment and packet lost rate [2, 14]. Moreover, some papers have proposed algorithms that are not appropriate for large-scale SDNs due to the massive time required to search for the solution space [11]. Therefore, in this paper, a formulation of the reliability CPP is proposed, which considers all the important metrics while being easily adaptable.

3. Proposed model

The topology of an SD-WAN is determined by a graph $G(V = S \cup C; E)$, where V is the set of nodes (including the sets of OpenFlow-enabled switches, i.e. S, and potential controllers sites, i.e. C), and E states the set of weighted links. The weights of the links are the propagation delay between the nodes. Suppose that the controllers can locate at the same location with the switches (i.e. $C = S$). In the following, we introduce the notations used in the formulation of RCPPM. The delay boundary between a switch and its assigned controllers is denoted by SC_{max}, whereas the inter-controller latency threshold is indicated by CC_{max}. u_j, r, and l_i represent the capacity of the controller j, number of required controllers to handle a supposed switch (i.e. resiliency level), and the traffic load of switch i, respectively; d_{ij} denotes the minimum propagation latency between nodes i and j. The binary decision variable y_j equals 1 if node j is selected to locate a controller, and 0 otherwise. The assignment of a switch i to the controller located at site j is stated by a binary variable x_{ij}, while $x_{ij} = 1$ means that switch i is assigned to the controller at site j; $x_{ij} = 0$ implies the other case. RCPPM is represented as follows:
Minimize

$$\sum_{j \epsilon C} y_j \tag{1}$$

Subject to:

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

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

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

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

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

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

The constraint in (2) prohibits a switch from being assigned to a controller site that is not open, while the constraint in (3) ensures that each switch is connected to r controllers ($r > 1$). 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 variable $w_{j',j''}$ using the McCormick envelopes [15]), which is given by:

$$w_{j',j''} = y_{j'} y_{j''} \quad (8)$$

and subsequently, replacing it with the following constraints:

$$d_{j',j''} w_{j',j''} \leq CC_{max}, \quad \forall j', j'' \epsilon C \quad (9)$$

$$w_{j'j''} < y_{j''}, \quad \forall j', j'' \epsilon C \quad (10)$$

$$w_{j',j''} < y_{j''}, \quad \forall j', j'' \epsilon C \quad (11)$$

$$w_{j',j''} \geq y_{j'} + y_{j''} - 1, \quad \forall j', j'' \epsilon C \quad (12)$$

$$w_{j',j''} \epsilon \{0, 1\}, \quad \forall j', j'' \epsilon C \quad (13)$$

3. Proposed solution

In this section, we elaborate our idea to solve the formulated optimization problem in Section 3 based on the clique concept in the graph theory. Then a case study is provided to delineate the proposed algorithm, and it is followed by the time complexity analysis of the proposed solution.

We define a complete graph (denoted by 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 G_o by removing the links that do not satisfy the latency bound CC_{max}, and call the resultant graph 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 3, we observe that each switch and its assigned controllers is a clique of G_p. A clique [16, 17] is defined as a complete sub-graph of an undirected graph. Since a switch is required 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 G_p), the switch and its associated controllers construct a complete sub-graph, i.e. a clique of G_p. Moreover, the inter-controller latency in the constraint in (6) implies that the set of open controllers in a solution must be a subset of one of the maximal cliques 1 of G_p. This is due to the fact that the set of open controllers must be a complete sub-graph of G_p to comply with the inter-controller latency constraint (all the open controllers are required to be directly connected to each other, and hence, they must be a subset of a maximal clique). Furthermore, the possible controller-switch assignments are the r-cliques and $(r+1)$-cliques (if any) of G_p. The cliques of size r correspond to the case where one of the potential controllers of the switch is co-located with it, while the $(r+1)$-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 a near-optimal algorithm to solve the problem, and the description of it is provided in the following sub-section.

4.1. Description of algorithm

The steps of the proposed solution in the algorithmic form is presented in Algorithm 1. We denote the diameter of the given WAN topology G by D_G (the length of the longest shortest path). Thus the upper bound of CC_{max} is D_G and SC_{max} is lower bounded by the minimum shortest path length in G. Consequently, the values of CC_{max} or SC_{max} that are less than the aforementioned path lengths cause the infeasibility of RCPPM (step 3 in Algorithm 1). Moreover, if the required resilience level r is more than the clique number (i.e. size of the maximum clique2 of G_p), the problem becomes infeasible.

However, a practical upper bound for r is 3, which is far less than the clique number of a large mesh-like graph (more common in WAN topologies). It should be noted that if G_p is a disconnected graph, the problem becomes infeasible for the chosen value of CC_{max} (step 6 of Algorithm 1).

In addition, the maximal cliques that the total capacity of their nodes is less than the total traffic load of switches (indicated by the product of the total load of switches and r) should be excluded from the set of maximal cliques. Thus if there is no maximal clique in G_p that satisfies the aforementioned condition, the problem becomes infeasible $Feasibility - Check(G_p; r; M))$ in step 8 of Algorithm 1.

Algorithm 1 RCPP-CLIQUE

1. **Input:** G, CC_{max}, SC_{max}, r , switch loads, controller's capacity u_c , shortest path matrix.
2. **Output:** controller locations and controller-switch assignments or infeasible.
3. Feasibility-Check (G, CC_{max}, SC_{max}).
4. G_0 = OverlayGraph (G).
5. G_p = Prune (G_0 , CC_{max}).
6. Feasibility-Check (G_p).
7. M = Maximal-Cliques (G_p).
8. Feasibility-Check ($G_p, r,$ M).
9. Find ($Q^r + Q^{r+1}$).
10. S'= Sort ($S', |Q_i|$).
11. S_{init}=S'.
12. $M = M \setminus \{m\}$
13. $C_0 = m$.
14. **while** $S' != \emptyset$ **do**
15. $S' = S' \setminus \{i\}$.
16. A = Find-Assignment (Q_i , C_0).
17. **if** A == $\emptyset$ then
18. **if** M == $\emptyset$ then
19. Infeasible.
20. **else**
21. $S' = S_{init}$.
22. $M = M \setminus \{m\}$.
23. $C_0 = m$.
24. **end if**
25. **end if**
26. **end while**
27. $C_0 = C_0 \setminus \{c \in C_0 \; if \; u_c^{rem} = u_c\}$

As shown in Algorithm 1, in order to identify the possible controller switch assignments, we find the sets of all r-cliques and $(r+1)$-cliques (if any) of G_p and denote them by Q^r and Q^{r+1}, respectively. We define Q_i as the set of all cliques that include switch i (i.e. $Q_i = Q_i^r \cup Q_i^{r+1}$) according to the following two cases:

1) $SC_{max} = CC_{max}$: Considering Q^r, the switch can be any of the r nodes in a clique. If $r <$ CliqueNumber (G_p), then $(r+1)$-cliques include the switch.

2) $SC_{max} < CC_{max}$: We consider the subsets of Q^r and Q^{r+1}, which include switch i as long as the weight of all incident links to switch i in the clique is less than or equal to the value of SC_{max}.

We sort the switches according to the size of Q_i (i.e. the number of possible controller assignments for that switch) in an increasing order (a greedy approach).

This means that the switches with fewer possible sets of assignments are handled first. While there are switches with no assigned controller, Algorithm 1 finds the assignments (step 16). To choose among the cliques of a switch in Q_i (step 4 of Algorithm 2), 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 C_o. In addition, all the cliques that have at least a controller node $c(c \in C_o)$ whose remaining capacity (denoted by u_c^{rem} in Algorithm 2) is less than the traffic load of switch i are excluded from Q_i (the resultant set is denoted by Q_i'). Then if there is any clique in Q_i' that all of its 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 (steps 5–7 in Algorithm 2).

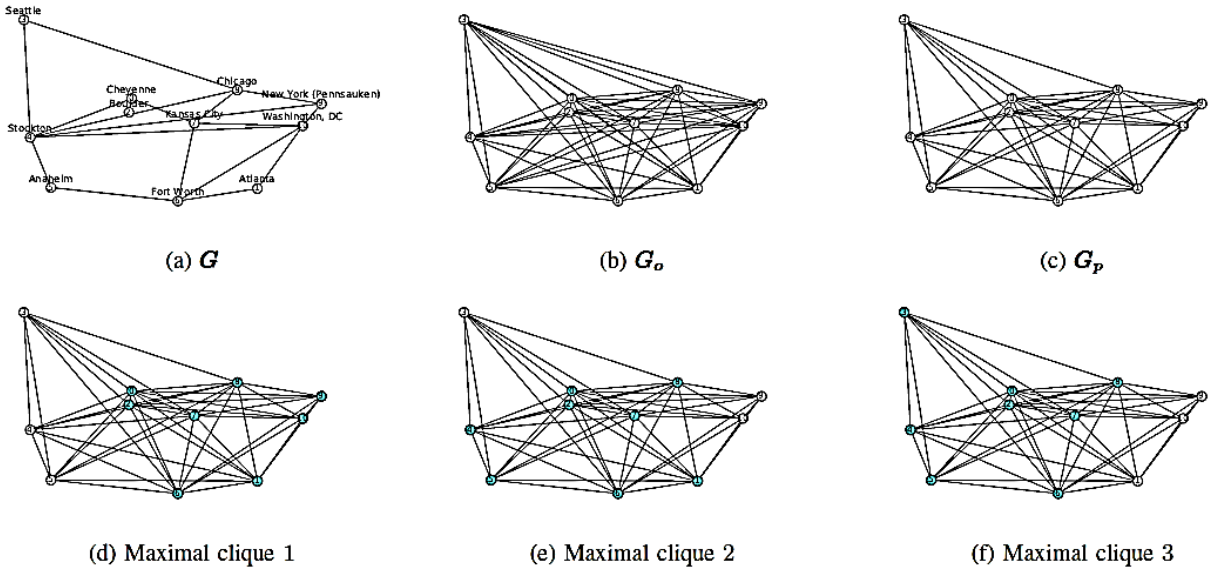
Algorithm 2 Find-Assignment

1. **Input:** Q_i , C_0.
2. **Output:** controller-switch assignments for switch i (a_i).
3. $a_i = \emptyset$.
4. Q_i= select- Cliques (Q_i , C_0).
5. $B = \{c \in C_0 \; if \; u_c^{rem} < u_c\}$.
6. **if** Find-Cliques (Q_i , B) **then**
7. Choose a clique q from Q_i
8. **else**.
9. Rank ().
10. Choose a clique q with the highest rank from the Q_i
11. **end if**
12. **if** q ! = $\emptyset$ **then**
13. a_i = Assignment (q).
14. **end if**

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 the used controllers as much as possible. If a clique q is found, the controllers in this clique are assigned to switch i (step 13 of Algorithm 2). Once we are done with the assignments for all switches, if there is any controller in C_o that is not involved in any of the controller-switch assignments, it is removed from the set of open controllers C_o.

4.2. Case study

In order to illustrate the effectiveness of the proposed algorithm, we studied an example for the Sprint topology. We set the input parameters as follows: CC_{max} = 0.8 D_G, SC_{max} = 0.4 D_G, $r = 2$, $u_c = 2000 \; kreq/s$ (controller capacity), and $l_s = 200 \; kreq/s$ (uniform switch traffic load). The original Sprint topology G, G_o, G_p and the set of all three maximal cliques of G_p are shown in figure 1. The set of open controllers in the solution is {1,4,5,6,7}, which is a subset of maximal clique 2. Figure 2 depicts the controller-switch assignments in the solution (acquired by *Algorithm 1* and it is optimal). More specifically, these assignments are the subsets of the 2-cliques and 3-cliques of G_p. The switch nodes are marked in 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.



(a) G (b) G_o (c) G_p

(d) Maximal clique 1 (e) Maximal clique 2 (f) Maximal clique 3

Figure 1. Sprint topology and its illustrations in different phases of Algorithm 1.

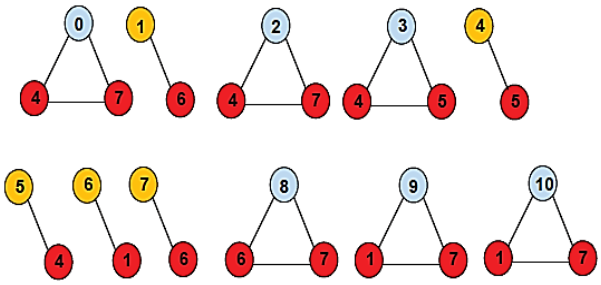


Figure 2. Controller-switch assignments for the Sprint topology.

5. Performance evaluation

In this section, we first provide a detailed description of our experiment setup, and then assess the performance of our proposed solution with respect to different metrics and parameters.

5.1. Experimental setup

As the input of the optimization problem in Section III, we utilize a wide spectrum of WAN topologies including the mesh, linear, ring, and hub-and-spoke-like topologies of various sizes. We conducted our experiments on about 40 WAN topologies from Internet Topology Zoo (ITZ) [18], which is a publicly available dataset and it has been used in many research works on SDN controller placement problems such as [10, 13]. Such network maps are of great importance in network design optimization, 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). The following shows a summary of the steps taken to conduct the experiments:

1) Pre-processing: For this step, we applied a similar approach as in [5, 10]. Multi-graphs were converted to simple graphs (the parallel edges did 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 with regard to the topology size (e.g. for UUNET and TATA, 7 out of 49 nodes and 2 out of 145 nodes were removed, respectively). 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 great geodesic distance). Also the shortest path lengths between nodes were calculated using the Dijkstra algorithm.

2) Parameter settings: Uniform capacities were associated to the potential controllers. Three values were used from $\{2000, 5000, 10000\}$ $kreq/s$ for each topology. We considered both homogeneous and heterogeneous switch traffic load. While for the homogeneous case the traffic load of switches was assumed to be $200\,kreq/s$, for the heterogeneous case, the traffic load of switches (as integer numbers) was uniformly distributed in $[1,400]$ $kreq/s$. The applied values were based on the prior studies on the capacitated CPP [9] [14] as well as the research works conducted on the performance of SDN controllers [4, 12]. The values for CC_{max} and SC_{max} were chosen as a percentage of the graph diameter since it was the largest possible propagation latency for a given topology. Note that assigning lower values than $0.4\ D_G$ to the

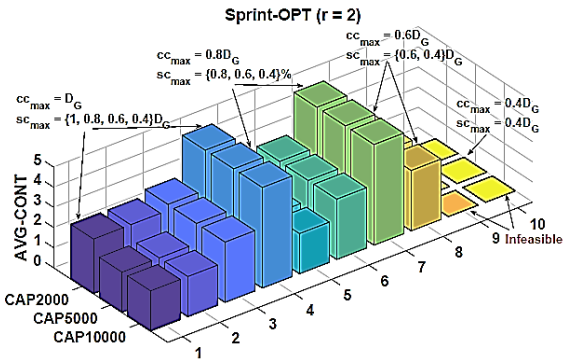
aforementioned parameters leads to infeasible solutions for many of the topologies. The resilience level r was set to 1, which indicated CPP (i.e. no resilience), and 2 and 3 to specify RCPPM. Considering the heterogeneous load of switches, 50 independent experiments were conducted to obtain both the optimal and near-optimal (using the proposed algorithm) solutions.

3) Obtaining the results: The Python interface of the GUROBI optimization software (version 6.5.2) [19] was used to obtain the optimal solutions. Furthermore, a python code was developed to solve RCPPM based on the proposed algorithm. All the experiments were carried out on an Intel(R) Core(TM) i7-3770 CPU@3.40GHz and 32GB RAM with Windows 10 Pro (64-bit) installed.

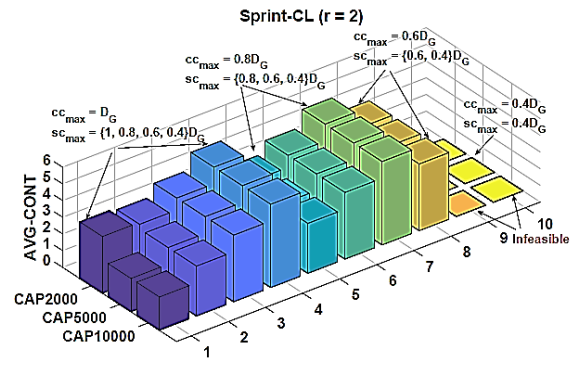
5.2. Results and discussion

The acquired results were analyzed in terms of a set of metrics including the average number of assigned controllers (i.e. the value of the objective function), average controller utilization, execution time, and reliability of the control plane. For each topology, the results obtained 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. For the ease of analysis and presentation, the chosen topologies were classified according to their network sizes (i.e. number of nodes N). Four groups were defined and labeled as what follows. Groups 1 (“small-size”), 2 (“mediumsize”), 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 the representatives for each group, we chose multiple graphs that covered different types of topologies (i.e. mesh, linear, ring, and hub-and-spoke). In the following, we demonstrate the impacts of different parameters on the aforementioned metrics for some representatives of the topologies on which we conducted our experiments. Similar observations and explanations apply to other topologies as well. All the experiments were assumed to have a heterogeneous traffic load for switches, unless otherwise stated.

1) Number of required controllers: Figure 3 illustrates the impacts of different parameters including controller capacities, CC_{max}, and SC_{max} on the average number of required controllers (denoted by AVG-CONT in the figure) for both optimal (OPT) and near-optimal (CL, acquired by *Algorithm 1*) solutions for the Sprint topology. It should be noted that the numbers on the x-axis correspond to different sets of scenarios. For instance, number 1 shows the set of 3 scenarios in which $CC_{max} = SC_{max} = D_G$, and u_c changes from $2000\ kreq/s$ to $10000\ kreq/s$ (i.e. fixed delay bounds while changing the capacity of controllers). While increasing u_c leads to a lower number of controllers for some scenarios (with respect to the delay requirements), it does not necessarily cause a decreasing trend for the others. For instance, as shown in figure 3a, 5 controllers are required in the set of scenarios denoted by the scenario set number 4 ($CC_{max} = D_G$, $SC_{max} = 0.4\,D_G$) on the x-axis regardless of the capacity of the controllers.
However, in all the three scenarios corresponding to different capacities on the y-axis, the maximum total traffic load of all switches for $r = 2$ is lower than the total capacity provided by only 3 controllers. This mainly results from the reduced value of SC_{max}, i.e. $0.4\,D_G$ compared with the scenario set number 3 ($CC_{max} = D_G$, $SC_{max} = 0.6D_G$), in which the value of CC_{max} has remained unchanged. Therefore, the number and set of the controller nodes that satisfy the switch-controller latency are mostly different from each other in these two scenario sets.



(a)



(b)

Figure 3. Impacts of controllers capacity, ccmax, and scmax on the average number of required controllers (the numbers on the x-axis correspond to different scenarios).

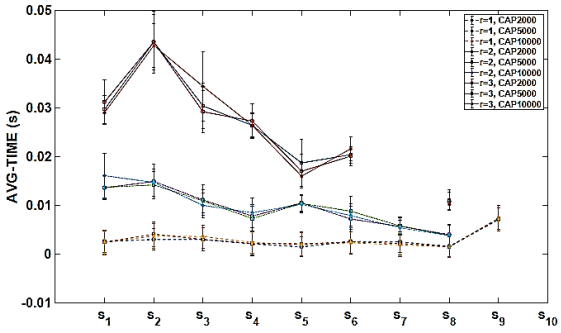


Figure 4. Average execution time of the proposed algorithm for the Sprint topology.

More specifically, considering the optimal solution, for the scenario set 3, in most of the experiments, the controllers are at nodes {0,6,8} (the index of the controllers), while for the scenario set 4, the controllers are at nodes {1,4,5,8,9}. Considering the aforementioned issues, in order to guarantee a resilience level, the trade-off between the number of controllers and the capacity of the controllers should be investigated for different topologies, and tailored decisions should be made based on the preference of the decision-makers.

2) Execution time: Figure 4 illustrates the execution time (AVG-TIME) of feasible solutions with respect to different scenarios for the proposed algorithm. The main part of the execution time is associated with obtaining the maximal cliques and cliques of size r and $r+1$ of G_p (as discussed in Section 4.3). Therefore, the average execution time goes up when the resilience level r is increased. In addition, regardless of the capacity of the controllers, the average execution time of the feasible solutions remains the same. The maximum average execution time (about 600 s) was observed for the largest topology, i.e. Cogent by setting $r = 2$, $CC_{max} = D_G$, and $SC_{max} = \{0.8,0.6,0.4\}D_G$ (in which G_p has the highest density with one large maximal clique).

Moreover, as shown in Fig. 4, the lowest average execution time for all resilience levels corresponds to the sets of scenarios in which the value of CC_{max} changes from D_G to $0.6D_G$ and then to $0.4D_G$ (i.e. s_1, s_5, and s_8). Obviously, reducing the values of CC_{max} and SC_{max} cuts the average execution time due to the resultant reduced density of G_p, which subsequently lowers the search space. In particular, for a fixed CC_{max}, by decreasing the value of SC_{max} (e.g. the change in AVG-TIME for $s_2 - s_4$), more cliques are excluded from the set of cliques that include each switch (steps 9–10 in *Algorithm 1*). In addition, when there is no delay requirement for the inter-controller latency for all of the topologies (i.e. $CC_{max} = D_G$, $s_1 - s_4$ on the x-axis of figure 6), $G_p = G_o$ since G_p is not pruned. Thus G_p is a complete graph, and finding all the maximal cliques is polynomially bounded. Particularly, a complete graph is its only maximal clique, and hence, the running time complexity goes down. If the original graph G is a complete graph and $CC_{max} = D_G$, it is possible that $G_p = G_o = G$. Among the graphs of the ITZ that we tested, we found that GlobalCenter was a complete graph that fulfilled the aforementioned criteria.

3) Reliability of control network with respect to controller node failures: It is assumed that all controller nodes have the same failure probability and fail independently from each other. We define the resilience of the control network, denoted by R_c, as the average number of disconnected switches when one or more controller nodes fail. Therefore, the control plane is protected against $r-2$ controller node failures (the failure of all the controller nodes would result in $R_c = N$). The possible range for multiple controller node failures that affect the value of R_c is $\{r, r+1, \dots, k+1\}$, where k is the number of controllers.

Table 1 shows the values of R_c (as a percentage of N) for $r = 2$, $l_s = 200\ kreq/s$ (switch load), $CC_{max} = 0.8D_G$, and $SC_{max} = 0.6D_G$ with respect to different numbers of controller node failures ($k = 9$) for the UUNET topology. Due to its larger network size compared with topologies such as Sprint, UUNET can better present how R_c changes by considering a wider range of controller node failures. It can be seen that a by-product of increasing the controller capacity (in addition to decreasing the average controller utilization, as discussed earlier) is the rise of R_c for the same number of controller node failures (due to the decreased value of k). Similar trends apply to other topologies as well (e.g. GARR, as a hub-and-spoke topology with almost the same N and k). Another observation is for smaller topologies such as Sprint.

Table 1. R_c for UUNET (OPT).

No. of failures	CAP2000	CAP5000	CAP10000
1	0	0	0
2	2.76%	16.66%	33.33%
13	8.33%	50%	100%
4	16.66%	100%	
5	27.76%		
6	41.66%		
7	58.33%		
8	77.76%		
9	100%		

6. Conclusions and future works

In this paper, a solution scheme for RCCPM, which provides near-optimal solution, has been

introduced. The efficiency of the new solution was analyzed with regard to various parameter settings for 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. Future research directions involve extending the proposed algorithm to handle the controller site or link failures. These possible extensions substantiate the fact that the proposed solution can be easily amended to cover node or link failures even with different objective functions (e.g. minimizing the expected control path loss). Another possible direction is to look into the dynamic RCPPM, which changes the controller-switch assignments according to the time-varying traffic load of the switches.

References

[1] Jalili, A., Keshtgari, M., & Akbari, R. (2019) A new framework for reliable control placement in software-defined networks based on multi-criteria clustering approach. Soft Computing, pp. 1-20.

[2] Jalili, A., Keshtgari, M., Akbari, R., & Javidan, R. (2019). Multi criteria analysis of controller placement problem in software defined networks. Computer Communications, vol. 133, pp. 115-128.

[3] Jalili, A., Keshtgari, M., & Akbari, R. (2018). A New Set Covering Controller Placement Problem Model for Large Scale SDNs. Information Systems & Telecommunication, vol. 25.

[4] Smith, P., Schaeffer-Filho, A., Hutchison, D., & Mauthe, A. (2014). Management patterns: SDN-enabled network resilience management. In 2014 IEEE Network Operations and Management Symposium (NOMS) (pp. 1-9). IEEE.

[5] Hock, D., Hartmann, M., Gebert, S., Jarschel, M., Zinner, T., & Tran-Gia, P. (2013). Pareto-optimal resilient controller placement in SDN-based core networks. In Proceedings of the 2013 25th International Teletraffic Congress (ITC) (pp. 1-9). IEEE.

[6] Hock, D., Gebert, S., Hartmann, M., Zinner, T., & Tran-Gia, P. (2014). POCO-framework for Pareto-optimal resilient controller placement in SDN-based core networks. In 2014 IEEE Network Operations and Management Symposium (NOMS) (pp. 1-2). IEEE.

[7] Xiao, P., Qu, W., Qi, H., Li, Z., & Xu, Y. (2014). The SDN controller placement problem for WAN. In 2014 IEEE/CIC International Conference on Communications in China (ICCC) (pp. 220-224). IEEE.

[8] Heller, B., Sherwood, R. & McKeown, N. (2012). The controller placement problem. In Proceedings of the first workshop on hot topics in software defined networks, ACM, pp. 7-12.

[9] Xiao, P., Li, Z., Guo, S., Qi, H., Qu, W., & Yu, H. (2016). AK self-adaptive SDN controller placement for wide area networks, Frontiers of Information Technology & Electronic Engineering, no. 17, pp. 620-633.

[10] Guo, M. & Bhattacharya, P. (2013). Controller placement for improving resilience of software-defined networks, In 2013 Fourth International Conference on Networking and Distributed Computing, pp. 23-27.

[11] Vizarreta, P., Machuca, C., & Kellerer, W. (2016). Controller placement strategies for a resilient SDN control plane, In 2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM), pp. 253-259.

[12] Lange, S., Gebert, S., Spoerhase, J., Rygielski, P., Zinner, T., Kounev, S., & Tran-Gia, P. (2015). Specialized heuristics for the controller placement problem in large scale SDN networks. In 2015 27th International Teletraffic Congress, pp. 210-218.

[13] Bannour, F., Souihi, S., & Mellouk, A. (2017). Scalability and reliability aware SDN controller placement strategies. In 2017 13th International Conference on Network and Service Management (CNSM), pp. 1-4.

[14] Perrot, N., & Reynaud, T. (2016). Optimal placement of controllers in a resilient SDN architecture. In 2016 12th International Conference on the Design of Reliable Communication Networks (DRCN), pp. 145-151.

[15] McCormick, G. P. (1976). Computability of global solutions to factorable nonconvex programs: Part iconvex underestimating problems. Mathematical programming, vol. 10, no. 1, pp. 147–175.

[16] Golumbic, M. C., & Hartman, I. B. A. (2006). Graph theory, combinatorics and algorithms: Interdisciplinary applications (Vol. 34). Springer Science & Business Media. pp. 1-8.

[17] Tatari, F., & Naghibi-Sistani, M. B. (2015). Optimal adaptive leader-follower consensus of linear multi-agent systems: Known and unknown dynamics. Journal of AI and Data mining, vol 3, no. 1, pp. 101-111.

[18] Knight, S., Nguyen, H. X., Falkner, N., Bowden, R., & Roughan, M. (2011). The internet topology zoo. IEEE Journal on Selected Areas in Communications, vol. 29, no. 9, pp. 1765-1775.

[19] "GUROBI Optimizer," (2019), http://www.gurobi.com/, accessed: 2016-03-25.