Controller Placement for Resilient Network State Synchronization in Multi-controller SDN

Tamal Das and Mohan Gurusamy

***Abstract*—In a multi-controller SDN, the network state is replicated or distributed between the controllers requiring repeated synchronization. This synchronization cost depends on controller locations, as well as needs to be low and resilient to network failures. Leveraging a Steiner tree-based inter-controller latency model, a multi-objective integer linear program is presented to deduce the controller placement optimizing (a) synchronization cost in failure-free scenarios, and, (b) resilience against single-link failures. The proposed approach incurs low computational complexity and post-failure network reconfiguration cost, while achieving near-optimal post-failure synchronization cost. We extensively evaluate the proposed model over real networks, and demonstrate its effectiveness.**

***Index Terms*—Software Defined Networks, Inter-controller communication, Network state synchronization, Controller placement, Steiner tree model.**

I. Introduction

Based on the seemingly simple principle of separation of data and control planes, Software Defined Networking (SDN) has evolved from campus networks to being a core component of several next generation network architectures such as VANET, IoT, 5G, etc. The SDN control plane is logically centralized in SDN controller(s). While a single controller may suffice for small networks, often multiple controllers are required for the purpose of scalability (i.e. single controller being incapable of serving large networks), security (i.e. single controller being most vulnerable for attacks) and resilience (i.e. backup controllers to recover from network congestion/failures). In such a multi-controller SDN, the overall network state information is either replicated or distributed across all controllers. The controllers are synchronized either periodically or based on events, to ensure network state consistency between them using consensus algorithms such as RAFT, PAXOS, etc. [1].

The performance of the network state synchronization depends on the controller locations and their communication protocol. The controller placement problem investigates the number of controllers required in a network, their locations, and the associated switch-controller mappings [2]. The inter-controller communication has been modelled based on unicast (i.e. pairwise communications between each controller pair) or multicast (i.e. tree-based communications between all controllers). The network state synchronization between the controllers is directly affected by the inter-controller latency, and must be resilient to network failures.

Studies in the area of controller placement in software defined networks have primarily focused on placements optimizing the switch-controller communication, be it in terms of improving its latency, resilience, etc. The inter-controller communication, on the other hand, has not received as much attention, which is the focus of this letter. Several inter-controller latency models have been proposed in the literature, which can broadly be classified in two categories – *unicast*-based or *multicast*-based. The unicast-based models assume that the controllers synchronize the network state between themselves via pair-wise communications between each controller pair, while the multicast-based models assume the controllers do so over a tree connecting all of them. The unicast-based models comprise sum [3], average [4], maximum [5] of all controller pair latencies, whereas a multicast model based on spanning tree between the controllers [6] have been proposed. Zhang et al model the inter-controller latency based on the exact message exchange in the RAFT consensus algorithm for network state consistency [7].

The unicast-based models are computationally efficient (only involves shortest path computations), while the multicast-based models are more resource-efficient (though involving complex spanning tree computations). Among multicast-based inter-controller latency models, the only one proposed in the literature is based on the minimum spanning tree between the controllers. By definition, however, a minimum spanning tree is not best suited to capture the inter-controller communications, and a Steiner tree offers a better abstraction of the same. In addition, resilience of the inter-controller network state synchronization against network failures has not been investigated in the literature.

Different from existing studies, this letter presents a Steiner tree-based inter-controller latency model to analyze the network state synchronization cost between the controllers in failure-free scenarios, as well as resilient network state synchronization in post-failure scenarios. Although single/multiple switch/link/controller failures comprehensively span the spectrum of potential SDN network failure scenarios, in practice, *single-link failures* are the dominant cause of network disruptions [8], which is the particular focus of this letter. Instead of recomputing the post-failure inter-controller Steiner tree (which requires controller intervention, is computationally prohibitive, and requires network-wide reconfiguration), the proposed approach connects the endpoints of the failed link over the next available shortest path[1] (which is precomputed and preconfigured in advance) thereby offering fast restoration, avoiding controller intervention, and only requiring local rerouting/reconfiguration. Based on the multi-commodity flow problem, a multi-objective integer linear program is formulated

[1]Note that such a path exists for a 2-connected graph.

to deduce the controller placement to optimize both the pre- and post-failure network synchronization cost. We extensively evaluate the proposed model over a large number of real networks to analyze the trade-off involved between pre- and post-failure network synchronization cost, as well as benchmark its performance against optimal post-failure resilience. Performance metrics include network synchronization cost (based on inter-controller Steiner tree), and network reconfiguration cost (based on the number of link reconfigurations).

II. Problem Statement and Proposed Approach

The specific problem statement investigated in this letter is as follows: Given a network and a fixed number of controllers, it is required to place the controllers and choose the communication pattern between the controllers, so as to minimize the inter-controller network state synchronization cost both before and after failures.

Figure 1 illustrates the proposed approach with a sample topology. For a given set of controllers and the associated Steiner tree, suppose that link 6-7 fails. While the best way to recover would be to optimally recompute the inter-controller Steiner tree requiring global reconfiguration and high computational complexity, the proposed approach joins the endpoints of the failed link via the next available shortest path, which is pre-computed for each link. The flow redirection required in the proposed approach can be implemented using `FAST FAILOVER` feature of OpenFlow `Group Tables`, i.e. without controller intervention, unlike the optimal approach. This lowers computational complexity, while achieving near-optimal resilience, as demonstrated in section IV.

The proposed Steiner tree formulation is different from the existing formulations [9]. In our case, the controller locations (which is generally considered as an input for Steiner tree) are not known *a priori*, since the controller locations and the associated inter-controller Steiner tree are jointly deduced. Secondly, the multi-commodity flow formulation for the Steiner tree requires marking a controller as a *root* controller, which again is not possible as the set of controllers is not known *a priori*. Both these issues further complicate formulating a suitable inter-controller latency model, which is addressed in this letter.

III. JCpStC: Joint Controller Placement and Steiner Tree Construction

In this section, a multi-objective problem formulation is presented to optimally place the SDN controllers, connect them to the switches, and with each other both in the failure-free and post-failures cases.

Network Model: A network $G(\mathcal{V},\mathcal{E})$ is considered, where $\mathcal{V}$ denotes the set of switches, and $\mathcal{E}$ denotes the set of links. Link $e \in \mathcal{E}$ has latency w_e. Let ϕ, μ and Δ denote the number of controllers in the network, controller capacity and switch-controller latency threshold, respectively. A binary parameter $\omega_e^{e'}$ attains 1 if and only if after failure of edge $e \in \mathcal{E}$, the next available shortest path between its endpoints traverses edge $e' \in \mathcal{E}$. In addition, binary parameter $\tau_{u,v}^{e}$ attains 1 if and only if the shortest path between $u, v \in \mathcal{V}$ traverses $e \in \mathcal{E}$.

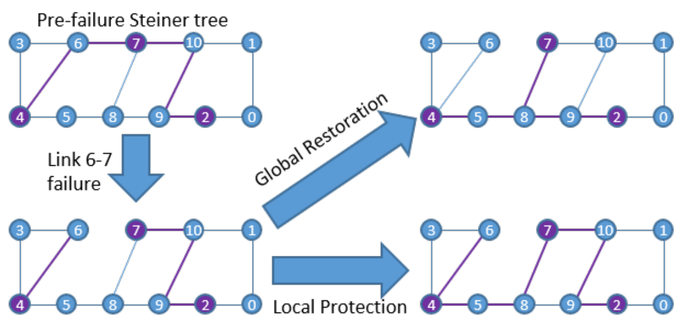

Fig. 1: Single-link failure recovery for network synchronization

Decision Variables: The primary decision variables include the controller locations ($y_v = 1$, if controller collocated with switch v; 0, otherwise), switch-controller mappings ($z_u^v = 1$, if switch u is mapped to controller collocated with switch v; 0, otherwise), and inter-controller Steiner tree in the failure-free case ($q_e = 1$, if link e is part of the inter-controller Steiner tree; 0, otherwise).

Controller count constraint: The total number of controllers placed in the network must be equal to ϕ, i.e.,

$$\sum_{v \in \mathcal{V}} y_v = \phi. \tag{1}$$

Controller capacity constraint: The total control load arriving at a controller from the switches mapped to it must not exceed μ, i.e.,

$$\forall v \in \mathcal{V} : \sum_{u \in \mathcal{V}} \lambda_u z_u^v < \mu. \tag{2}$$

Switch-controller mapping constraint: Each switch is mapped to one and only one controller, i.e.,

$$\forall u \in \mathcal{V} : \sum_{v \in \mathcal{V}} z_u^v = 1. \tag{3}$$

A controller exists if and only if one or more switches are mapped to it, i.e.,

$$\forall v \in \mathcal{V} : y_v \le \sum_{u \in \mathcal{V}} z_u^v. \tag{4}$$

For a switch to be mapped to a controller, the controller must exist in the first place, i.e.,

$$\forall u, v \in \mathcal{V} : z_u^v \le y_v. \tag{5}$$

Switch-controller latency constraint: Each switch-controller communication latency must not exceed Δ, i.e.,

$$\forall u, v \in \mathcal{V} : \sum_{e \in \mathcal{E}} w_e \tau_{u,v}^e z_u^v \le \Delta. \tag{6}$$

Bounds constraint: All primary decision variables (y_v, z_u^v and q_e) and associated secondary decision variables (r_v and f_{uv}^k) assume binary values, i.e.,

$$\begin{aligned} &\forall e \in \mathcal{E} : q_e \in \{0,1\} \quad \forall u, v, k \in \mathcal{V} : f_{uv}^k \in \{0,1\} \\ &\forall v \in \mathcal{V} : r_v \in \{0,1\} \quad \forall u, v \in \mathcal{V} : y_v, z_u^v \in \{0,1\}. \end{aligned} \tag{7}$$

Steiner tree constraints: We adopt the multi-commodity flow formulation for the Steiner tree construction, wherein for a given set of controllers to be connected, a unique commodity (one for each non-root controller) is routed from a root controller to every other controller [9]. In our case, however, as the controller locations are not known before the Steiner tree construction, the root controller cannot be selected *a priori*. To resolve this issue, a binary variable r_v is introduced to mark the root controller, which attains 1 if and only if the controller is a root controller; 0 otherwise. Thus, there must exist a unique root controller in the network, i.e.

$$\sum_{v \in \mathcal{V}} r_v = 1. \tag{8}$$

In addition, for a switch to be able to host a root controller, it must host a controller in the first place, i.e.,

$$\forall v \in \mathcal{V} : r_v \leq y_v. \tag{9}$$

Let f_{uv}^k denote the flow of commodity k on edge $(u, v) \in \mathcal{E}$. A flow with commodity k traverses edge $e = (u, v) \in \mathcal{E}$ if and only if e is part of the Steiner tree (i.e. $q_e = 1$) and the commodity corresponds to a non-root controller (i.e. $y_k = 1$ and $r_k = 0$). To capture the latter condition, a binary variable $h_k = y_k \oplus r_k$ is introduced using the XOR operator, which can be linearized using standard transformations. The overall relationship between f_{uv}^k, h_k and q_e can then be stated as:

$$\forall e = (u, v) \in \mathcal{E}, k \in \mathcal{V} : f_{uv}^k + f_{vu}^k \leq \frac{h_k + q_e}{2}. \tag{10}$$

The flow conservation constraints are defined as follows. At node v, the *net* outgoing flow of commodity k is equal to 0, if commodity k is destined for node v, and 1 otherwise. Conversely, if a commodity k is not destined for node v, then the net outgoing flow of commodity k at node v is equal to 1 if and only if k corresponds to a non-root controller (i.e. $h_k = 1$) and v is a root controller (i.e. $r_v = 1$). To capture the latter condition, a binary variable $d_v^k = h_k r_v$ is introduced, which can be linearized using standard transformations. The overall flow conservation constraint can then be stated as:

$$\forall k, v \in \mathcal{V} : \quad \sum_{w \in \mathcal{N}(v)} f_{vw}^k - \sum_{u \in \mathcal{N}(v)} f_{uv}^k = \begin{cases} -h_k, & \text{if } v = k \\ d_v^k, & \text{if } v \neq k \end{cases}. \tag{11}$$

Finally, an edge $(u, v) \in \mathcal{E}$ may belong to the Steiner tree if and only if a flow of commodity k (i.e. $h_k = 1$) traverses it (i.e. $f_{uv}^k = 1$). For this purpose, a binary variable $l_{uv}^k = h_k f_{uv}^k$ is introduced, which can be linearized using standard transformations. The relationship between q_e and l_{uv}^k can then be stated as:

$$\forall e = (u, v) \in \mathcal{E} : q_e \leq \sum_{k \in \mathcal{V}} (l_{uv}^k + l_{vu}^k). \tag{12}$$

Objectives: The overall objective is to place controllers to minimize both the pre- and post-failure network synchronization cost. The inter-controller latency in failure-free scenario is formulated as the total weight of the Steiner tree connecting all controllers, as:

$$\sum_{e \in \mathcal{E}} w_e q_e. \tag{13}$$

To maximize resilience against all single-link failures, the maximum post-failure Steiner tree weight considering all single-link failures is minimized. After a link failure, the affected traffic is assumed to be locally re-routed over the shortest path between endpoints of the failed link. A link can be part of the post-failure Steiner tree if (a) it is part of the pre-failure Steiner tree (and is not the failed link), or (b) it is not part of the pre-failure Steiner tree, but the shortest path between endpoints of the failed link traverses through it. The maximum post-failure Steiner tree weight is thus given by:

$$\max_{e \in \mathcal{E}} \sum_{e'(\neq e) \in \mathcal{E}} w_{e'} \Big(q_{e'} + (1 - q_{e'}) \omega_e^{e'} q_e \Big). \tag{14}$$

The overall objective is modelled as a weighted sum of the single objectives, as:

$$\min \left\{ (1-\alpha) \sum_{e \in \mathcal{E}} w_e q_e + \alpha \max_{e \in \mathcal{E}} \sum_{e'(\neq e) \in \mathcal{E}} w_{e'} \Big(q_{e'} + (1-q_{e'}) \omega_e^{e'} q_e \Big) \right\}, \tag{15}$$

where, $\alpha \in [0, 1]$ is a weight parameter. The product of binary variables ($q_e q_{e'}$) and the max term in the above objective function can be linearized using standard transformations.

IV. Performance Evaluation

In this section, the proposed approach is extensively evaluated over real networks from the Internet topology zoo [10].

A. Evaluation Setup

The proposed approach is evaluated over all topologies from the Internet Topology Zoo with complete latitude/longitude information (to compute real link latencies based on geodesic distances and 5μs/km as the propagation speed) and are at least 2-connected (for resilient inter-controller communication against single-link failures). Each switch generates control traffic from a uniform distribution in $[100, 300]$ kilo-requests/sec [11], and 10 such random network-wide traffic profiles are considered for statistical stability. 95% confidence intervals (for line plots) or confidence bands (for distribution functions) are plotted. Controller processing capacity of 1.1 million requests/sec is assumed [12]. Up to five values of switch-controller latency thresholds are considered, viz., {2,4,6,8,10} *ms*. α ranges from 0 to 1 in steps of 0.1. The minimum number of controllers required to serve each network instance is derived, based on which up to five different values for number of controllers are considered. Thus, each simulation run is uniquely characterized by a 5-tuple parameter combination comprising the topology, traffic profile, number of controllers, switch-controller latency threshold and α.

Baseline Approach: For each such generated network scenario, the proposed approach (labelled JCpStC) returns the controller placement and inter-controller Steiner tree in failure-free case, based on post-failure inter-controller Steiner trees corresponding to each link failure. However, Steiner tree constructions are computationally intensive, and computing one corresponding to each link failure quickly turns out to be infeasible for medium to large networks. To keep the problem formulation tractable, JCpStC approximates the post-failure

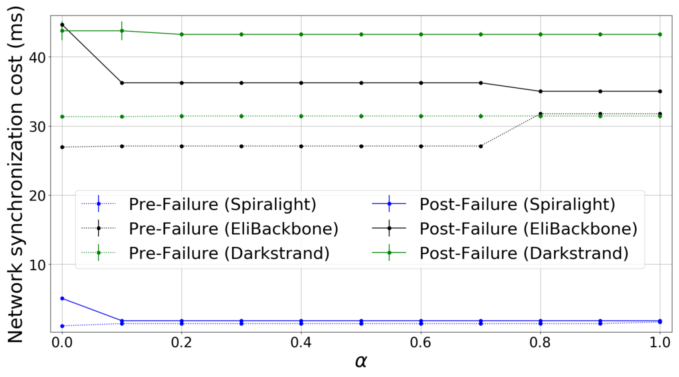

Fig. 2: Effect of α on network synchronization cost.

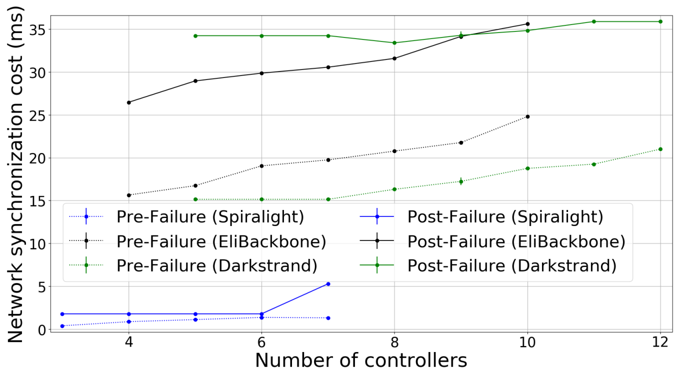

Fig. 3: Effect of number of controllers on network synchronization cost.

Steiner tree corresponding to a link failure by replacing the failed link in the pre-failure Steiner tree by the shortest path connecting its endpoints. Alternatively, the *optimal* post-failure Steiner tree for a given link failure (labelled `OptPfStC` for Optimal Post-failure Steiner tree Construction) can be computed for a *given* controller placement and pre-failure inter-controller Steiner tree (both resulting from `JCpStC`). Thus, `OptPfStC` is used as the baseline to benchmark the performance of `JCpStC` in terms of post-failure network state synchronization cost.

Performance Metrics: Performance metrics such as the pre/post-failure network synchronization cost (measured as sum of all link latencies of the inter-controller Steiner tree), and post-failure network reconfiguration cost (measured upon a failure in terms of the number of link additions/deletions with respect to the pre-failure Steiner tree) are considered.

B. Sensitivity Analysis

Effect of α on network synchronization cost: Figure 2 plots the effect of α on the pre/post-failure network synchronization cost for three networks, namely, Spiralight (15 nodes, 16 links), EliBackbone (20 nodes, 30 links) and Darkstrand (28 nodes, 31 links). This plot corresponds to 6 controllers in each network and switch-controller latency threshold of 6 *ms*. Each topology is plotted with a unique color, and for each topology, the pre-failure Steiner tree weight is plotted with a dotted line, while a solid line plots the post-failure scenario.

As each failed link is replaced by the next available shortest path between its endpoints, the post-failure network synchronization cost is always higher than the pre-failure network synchronization cost. Both the pre/post-failure synchronization cost increase with topology size. Increasing the value of α shifts weightage from pre-failure to post-failure objective, leading to a rise in pre-failure network synchronization cost and fall in post-failure network synchronization cost. Thus, the difference between the pre- and post-failure network synchronization cost decreases with α (for each topology), and increases with topology size (for each α). The latter is because, for the same controller count and switch-controller latency threshold, controllers tend to move further apart from each other in a larger network. Variation in α results in a set of non-dominated solutions, one of which can be chosen by the network operator based on its importance of the pre-failure vs. post-failure synchronization cost.

Effect of number of controllers on network synchronization cost: Figure 3 plots the effect of number of controllers on the pre- and post-failure network synchronization cost for the same three topologies as in Figure 2. This plot corresponds to switch-controller latency threshold of 10 *ms* and $\alpha = 0.5$.

An increase in the number of controllers, in turn requires more edges to inter-connect them, resulting in higher network synchronization cost. Thus, both the pre- and post-failure network synchronization cost increase with number of controllers. Secondly, it is also observed that larger the network, larger the problem search space, greater the difference between the pre- and post-failure network synchronization cost. In other words, the difference between the pre- and post-failure network synchronization cost increases with network size. Finally, the minimum number of controllers required for a network is also found to increase with network size.

C. Benchmarking against optimal post-failure synchronization

Computational complexity vs. optimality: In Figure 4, ~11k simulation runs are considered, each characterized by a unique 5-tuple parameter combination. For each simulation run, the post-failure network synchronization cost is deduced by both approaches – `JCpStC` and `OptPfStC`. The % deviation of the maximum post-failure network synchronization cost (across all single-link failures) resulting from `JCpStC` with reference to that resulting from `OptPfStC` is computed. Figure 4 plots the cumulative distribution function (CDF) of this % deviation.

`JCpStC` trivially deduces the post-failure Steiner tree by connecting the end-points of the failed link along the next available shortest path, instead of deducing the optimal post-failure Steiner tree as done by `OptPfStC`. Nevertheless, `JCpStC` performs as good as `OptPfStC` in ~70% of all scenarios, and its maximum deviation is bounded by 55%. While the proposed algorithm is based on shortest paths to reconnect the failed link, which requires $O(\mathcal{V}^2)$ computational time, the optimal algorithm uses Steiner tree computation requiring $O(3^{\phi})$ time, where ϕ denotes the number of controllers. Thus,

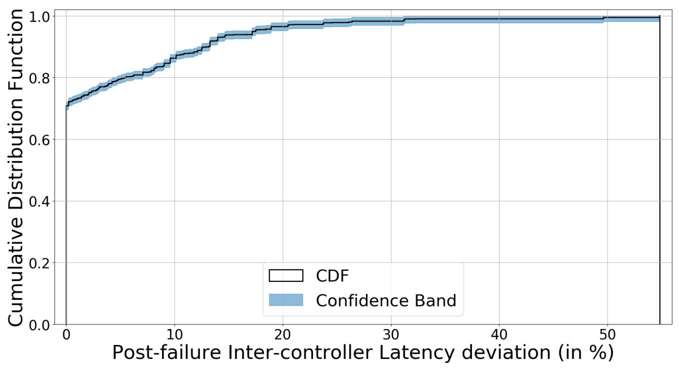

Fig. 4: Post-failure Steiner tree weights: `JCpStC` vs. `OptPfStC`

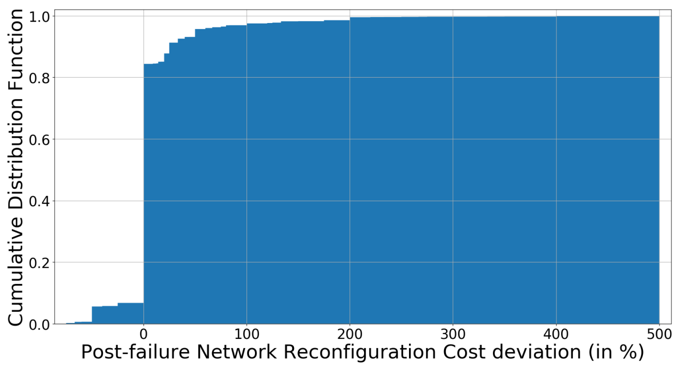

Fig. 5: Post-failure Reconfiguration Cost: `JCpStC` vs. `OptPfStC`.

`JCpStC` strikes a balanced trade-off between computational complexity and optimality.

Post-failure reconfiguration cost: For the optimal post-failure network synchronization cost as computed by both approaches (`JCpStC` and `OptPfStC`), the corresponding amount of network reconfigurations required to recover from single-link failures is studied in this evaluation. Similar to Figure 4, Figure 5 plots the CDF of the % deviation of the post-failure reconfiguration cost of `JCpStC` with respect to `OptPfStC` across ~11k simulation runs, each characterized by a unique 5-tuple parameter combination.

Both approaches require equal amount of post-failure reconfiguration efforts in majority (~77%) of the cases, as indicated by data points with zero deviation. To recover from a single-link failure, switching to an alternative path generally involves fewer link reconfigurations, than re-evaluating the entire post-failure Steiner tree, as observed in ~16% cases (those with positive deviation). There are, however, few exceptions to this as observed in ~7% case of outliers (those with negative deviation). But even when `JCpStC` underperforms, the extent of performance degradation (up to 75%) is significantly lower than the extent of performance improvement (up to 500%) when it outperforms.

The reason why `JCpStC` under-performs with respect to `OptPfStC` is as follows. Under certain scenarios, the two partitions of the pre-failure Steiner tree resulting from a link failure can be better connected instead of connecting its endpoints via the next available shortest path. In some cases, connecting endpoints of a failed link via the next available shortest path even results in a loop in the resulting Steiner tree, resulting in higher tree weight (which `JCpStC` is not designed to eliminate).

V. Conclusion and Future Work

Although the SDN control plane comprises both switch-controller and inter-controller communications, the former has been predominantly investigated. In this letter, we extend the state-of-the-art in terms of novel Steiner tree-based inter-controller latency model. We studied the proposed model to optimize the network state synchronization cost both before and after single-link failures. A multi-objective ILP formulation was presented to deduce the associated controller placement. The proposed approach was extensively evaluated in terms of sensitivity analysis, as well as benchmarked against optimal post-failure network synchronization. Despite being of significantly lower computation complexity, the proposed `JCpStC` achieved the optimal post-failure network synchronization cost in ~70% cases. Further, it requires significantly lower post-failure network reconfiguration cost than the optimal approach, `OptPfStC`. Controller placement heuristics to jointly optimize both pre- and post-failure network synchronization cost remain a potential future work, in addition to considering controller utilization, load balancing, etc.

Acknowledgment

This work was supported by Singapore MoE AcRF Tier-2 Grant MOE2015-T2-2-116, NUS WBS R-263-000-C17-112.

References

[1] A. Panda, W. Zheng, X. Hu, A. Krishnamurthy, and S. Shenker, "{SCL}: Simplifying distributed {SDN} control planes," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2017.

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

[3] M. F. Bari, *et al.*, "Dynamic controller provisioning in software defined networks," in *Network and Service Management (CNSM), 2013 9th International Conference on*. IEEE, 2013, pp. 18–25.

[4] S. Lange *et al.*, "Heuristic approaches to the controller placement problem in large scale SDN networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 4–17, 2015.

[5] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-optimal resilient controller placement in SDN-based core networks," in *International Teletraffic Congress (ITC)*. IEEE, 2013.

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

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

[8] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot, "Characterization of failures in an operational ip backbone network," *IEEE/ACM Transactions on Networking (TON)*, vol. 16, no. 4, pp. 749–762, 2008.

[9] S. Chopra and C.-Y. Tsai, "Polyhedral approaches for the steiner tree problem on graphs," in *Steiner trees in industry*. Springer, 2001, pp. 175–201.

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

[11] Í. Cunha, F. Silveira, R. Oliveira, R. Teixeira, and C. Diot, "Uncovering artifacts of flow measurement tools," in *International Conference on Passive and Active Network Measurement*. Springer, 2009.

[12] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks." *Hot-ICE*, vol. 12, pp. 1–6, 2012.