Controller Placements for Improving Flow Set-Up Reliability of Software-Defined Networks

Yuqi Fan, Tao Ouyang, and Xiaohui Yuan

Abstract Software-defined networking (SDN) is a new networking paradigm that decouples control plane from the data plane. A switch in the data plane device sends a flow set-up request to the controller, a device in the control plane, upon the arrival of an unknown flow. The controller responds the request with a flow entry to be installed in the flow table of the switch. Link failures can cause disconnections between switches and controllers. Most existing research on controller placement in SDNs investigated controller placements without considering single-link-failure impact on the number of dropped flow set-up requests in SDNs. In this paper, we formulate a novel SDN controller placement problem with the aim to minimize the average number of dropped flow set-up requests due to the single-link-failure. We propose two efficient algorithms for multiple-controller placements. The simulation results demonstrate that the proposed algorithms achieve competitive performance in terms of average number of dropped flow set-up requests under single-link-failure and average latency of flow set-up requests.

Keywords Software-defined network · Reliability · Single-link-failure · Network controller

1 Introduction

Software-defined networking (SDN) is a new networking paradigm that decouples control plane from data plane [1, 2]. Multiple-controller architectures have been introduced in SDNs and raised a new problem, the controller placement problem,

Y. Fan (✉) · T. Ouyang
School of Computer and Information, Hefei University of Technology, Hefei, Anhui, China
e-mail: yuqi.fan@hfut.edu.cn; ouyangtao@mail.hfut.edu.cn

X. Yuan
Department of Computer Science and Engineering, University of North Texas, Denton, TX, USA
e-mail: Xiaohui.Yuan@unt.edu

X. Yuan, M. Elhoseny (eds.), *Urban Intelligence and Applications*, Studies in Distributed Intelligence, https://doi.org/10.1007/978-3-030-45099-1_1

which needs to decide the controllers positions and how to associate switches with the controllers, since random placement is far from optimal [3, 4].

Some research has been conducted on the controller placement problem with the objective of minimizing the node-to-controller latency. The controller placement problem was first proposed by Heller et al. in [3] to minimize the communication latency between the switches and the controllers. A latency metric to minimize the total cost of flow set-up request from switches to controllers was introduced to deal with the mapping between the switches and the controllers under dynamic flow variations, and the metric considered the weight of switches and the delay from the switches to the controllers simultaneously, where the weight of a switch was related to the node degree of the switch and the maximum node degree in the network [5]. A network partition based scheme was designed, where the network was portioned into multiple subnetworks with revised k-means algorithm and a controller was placed in each subnetwork to minimize the maximum latency between the controller and the associated switches in the subnetwork [6]. A framework for deploying multiple controllers within a WAN was proposed to dynamically adjust the number of active controllers and delegate each controller with a subset of switches according to network dynamics [7].

The reliability is also an important performance metric for networks. A metric called expected percentage of control path loss due to failed network component was introduced to characterize the reliability of SDN networks, and a heuristic algorithm l-w-greedy was proposed to analyze the trade-off between reliability and latency; the expected percentage of control path loss was related to the number of control paths going through a component and the failure probability of the component [8, 9]. A controller placement strategy, Survivor, was proposed to explore the path diversity to optimize the survivability of networks with the aim to maximize the number of node-disjoint paths between the switches and the controllers; the strategy enhanced connectivity by explicitly considering path diversity, avoided controller overload by adding capacity-awareness in the controller placement, and improved failover mechanisms by means of a methodology for composing the list of backup paths [10]. The latency-aware reliable controller placement problem was investigated by jointly taking into account both the communication reliability and the communication latency between the controllers and the switches if any link in the network fails [11].

Link failures incur the breakdown of part of the network, during which some flow set-up requests from the switches are unable to reach the corresponding controllers and hence get dropped. To the best of our knowledge, very little attention in literature has ever been paid on the single-link-failure impact on the number of dropped flow set-up requests in SDNs. In this paper, we tackle the multiple-controller placement problem with the aim to improve the reliability in terms of the number of dropped flow set-up requests under single-link-failure.

The main contributions of this paper are as follows. We address the controller placement problem to maximize the reliability of the flow set-up requests under single-link-failure. We define a novel controller placement metric, the average number of dropped flow set-up requests, and propose two efficient algorithms

for multiple-controller placements based on the proposed placement metric. We also evaluate the performance of the proposed algorithms through simulations. Experimental results demonstrate the proposed algorithms are very promising.

The rest of the paper is organized as follows. Section 2 presents the problem of this work. Section 3 discusses the proposed two algorithms: Reliability Aware Controller placement (RAC) and Fast-RAC (FRAC). Section 4 discusses our experimental evaluation. Section 5 concludes our work with a summary.

2 Problem Formulation

We model an SDN network topology as graph $G = (V, E)$, where V is the set of switches (or nodes) and E is the set of links. Each controller is co-located with a switch, and each switch is mapped to one controller. We assume that there is at most one link failure in the network [12]. The notations used in the paper are listed in Table 1.

When link e on the control path $p_{i,k}$ fails, we calculate the number of dropped flow set-up requests as follows:

$$\mathcal{D}(e) = \sum_{s_i \in V} \sum_{c_k \in C} r_{i,k} \cdot p^e_{i,k} \cdot x_{i,k}. \tag{1}$$

Our objective is to minimize the average number of dropped flow set-up requests by

Table 1 Symbols and notations used in our description

Notation	Description
s_i	Node/switch i
c_k	Controller c_k
C	Controller set
K	The number of controllers
N	The number of nodes/switches
L	The set of the links in all the control paths
u_k	The processing capacity of controller c_k
$r_{i,k}$	The number of requests from switch s_i to the mapped controller c_k
$x_{i,k}$	Indicate whether switch s_i is mapped to controller c_k (= 1) or not (= 0)
$y_{i,k}$	Denote whether controller c_k is co-located switch s_i (= 1) or not (= 0)
$p_{i,k}$	The link set on the control path between switch s_i and controller c_k
$p^e_{i,k}$	Denote whether link e is a link on control path $p_{i,k}$ (= 1) or not (= 0)

minimizing

$$\bar{\mathcal{D}} = \frac{\sum_{e \in L} \mathcal{D}(e)}{|L|} \tag{2}$$

subject to

$$\sum_{k=1}^{K} x_{i,k} = 1, \quad \forall s_i \in V. \tag{3}$$

$$\sum_{i=1}^{N} y_{i,k} = 1, \quad \forall c_k \in C. \tag{4}$$

$$y_{i,k} \leq x_{i,k}, \quad \forall s_i \in V, \forall c_k \in C. \tag{5}$$

$$\sum_{i=1}^{N} x_{i,k} \cdot r_{i,k} \leq u_k, \quad \forall c_k \in C. \tag{6}$$

where $|L|$ denotes the number of links in L, and the average number of dropped flow set-up requests due to single-link-failure in the control paths are defined with Eq. (2). Equation (3) ensures that each switch is mapped to one and only one controller. Equation (4) mandates that each controller is placed onto exactly one switch. Equation (5) dictates that switch s_i is mapped to controller c_k if controller c_k is co-located with switch s_i. Equation (6) signifies that the number of requests to the controller cannot exceed the processing capacity of the controller.

3 RAC and FRAC Controller Placement Algorithms

In this section, we propose two controller placement algorithms, flow set-up request Reliability Aware Controller placement (RAC) and Fast-RAC (FRAC).

3.1 Reliability Aware Controller Placement Algorithm

Initially, algorithm RAC assumes that there are N controllers and each controller is co-located with a switch. The algorithm removes the redundant controllers iteratively until the number of controllers is K. For each controller, the algorithm evaluates the cost of removing it (steps 2–10). Assume the set of switches needing to be re-mapped after removing controller c_k is S_k. For each switch $s_i \in S_k$, algorithm RAC chooses the controller which incurs the least cost. During the re-mapping, Eq. (6) should be satisfied. After re-mapping all the switches in S_k, the algorithm can obtain the cost of removing c_k. Algorithm RAC evaluates the removal cost for all the controllers and removes the one which incurs the least cost (steps 11–12).

Algorithm 1 *RAC*

Input: Network topology $G = (V, E)$,
The number of requests from the switches,
The number of controllers K
Output: The set of locations placed with controllers C_p,
Mapping relationship between switches and controllers

1: Place a controller at the location of each switch, map each switch to the co-located controller, $C_p = V$, the number of placed controllers is $K' = N$;
2: **while** $K' \neq K$ **do**
3: $\bar{\mathcal{D}}_{min} = \infty$;
4: **for** each location $k \in C_p$ **do**
5: Assume c_k is removed, and denote the set of switches mapped to c_k as S_k;
6: Re-map each switch in S_k to the controller incurring the least average number of dropped flow set-up requests, and denote the average number of dropped flow set-up requests after re-mapping all the switches in S_k as $\bar{\mathcal{D}}_k$;
7: **if** $\bar{\mathcal{D}}_k < \bar{\mathcal{D}}_{min}$ **then**
8: $k_{min} = k$, $\bar{\mathcal{D}}_{min} = \bar{\mathcal{D}}_k$;
9: **end if**
10: **end for**
11: Remove the controller $c_{k_{min}}$ which incurs the least average number of dropped flow set-up requests $\bar{\mathcal{D}}_{min}$;
12: $C_p = C_p \setminus k_{min}$, $K' = K' - 1$;
13: **end while**

Time Complexity of Algorithm RAC The algorithm needs to remove $N - K$ controllers. In the worst case, a controller manages $N - K + 1$ switches. If the controller is removed, the algorithm performs re-mapping for the $O(N - K + 1)$ switches. The re-mapping for a switch checks at most $N - 1$ controllers. We can calculate all the shortest paths between all the node pairs in the network within $O(N^2 \cdot N) = O(N^3)$ time so that we can get the link set on the control path between each node pair before performing algorithm RAC, and hence the calculation of the average number of dropped flow set-up requests when the switch is mapped to the controller runs in time $O(N)$. Therefore, the time complexity of algorithm RAC is $O((N - K) \cdot N \cdot (N - K + 1) \cdot (N - 1) \cdot N) = O(N^3 \cdot (N - K)^2) = O(N^5)$, since $K \ll N$.

3.2 Fast-RAC

We propose algorithm Fast-RAC (FRAC) to reduce the time complexity of algorithm RAC. FRAC maintains a mapping controller priority list PL_i for each switch s_i, where each item in the list is a controller and the controllers in the list are sorted in the non-ascending order of the path lengths between the switch and the controllers. FRAC uses two arrays $Current$ and $Next$, each with length N. $Current[i] = k$ denotes that switch s_i is currently mapped to controller c_k. Initially, each switch is mapped to the co-located controller, that is, $Current[i] = i$. $Next[i] = k'$ indicates

that switch s_i will be mapped to controller $c_{k'}$ if controller c_k is removed, where $c_{k'}$ will cause the least number of dropped flow set-up requests caused by single-link-failure. We initialize arrays $Current$ and $Next$ before placing a controller at each switch (step 1), and update them when a controller is removed (step 11). During the update, if switch s_i is re-mapped from c_k to $c_{k'}$, $Current[i] = k$ is updated as $Current[i] = k'$ and $Next[i]$ should be updated by searching the first controller available in the list PL_i.

Time Complexity of Algorithm FRAC The construction of the mapping controller priority lists for all the switches can be performed in time $O(N^3)$ by calculating the shortest paths between all the node pairs. The initialization and update of the array $Current$ can be performed in time $O(N)$. Array $Next$ can also be constructed in time $O(N)$, while the update of the array $Next$ is conducted in time $O(N^2)$, since FRAC checks at most $N - 2$ controllers to find the controllers available for each of the N switches. The time consumed to re-map a switch is reduced from $O(N)$ with RAC to $O(1)$ with FRAC. Therefore, the time complexity of algorithm FRAC is $O(N^4)$.

4 Performance Evaluation

In this section, we evaluate the performance of the proposed controller placement algorithm. We also investigate the impact of important parameters on the performance of the proposed algorithms.

4.1 Simulation Set-up

We compare the proposed algorithms RAC, FRAC against the state of the arts: l-w-greedy [9], SVVR [10], and CPP [3]. The two coefficients l and w are set as $l = 2$ and $w = 1$ to enable l-w-greedy to achieve the best performance as described in [9]. The network topologies used in the simulation are ATT (ATT North America) and Internet2 (Internet2 OS3E) [13, 14]. All the controllers have the same processing capacity of 1800 kilorequests/s [10]. The requests from the switches are generated with uniform distribution pattern. 30 set of requests are generated for each request distribution pattern randomly, while the average number of flow set-up requests of the switches are 200 kilorequests/s. We use geographical distance as an approximation for latency [3].

4.2 Evaluation Metrics

The performance metrics evaluated are as follows:

1. The average number of dropped flow set-up requests under single-link-failure as calculated via Eq. (2).
2. The average latency of flow set-up requests $\bar{l}$, defined via Eq. (7), where $l_{i,k}$ denotes the communication latency of control path $p_{i,k}$.

$$\bar{l} = \frac{\sum_{s_i \in V} \sum_{c_k \in C} r_{i,k} \cdot x_{i,k} \cdot l_{i,k}}{\sum_{s_i \in V} \sum_{c_k \in C} r_{i,k} \cdot x_{i,k}}. \tag{7}$$

4.3 Simulation Results

4.3.1 Average Number of Dropped Flow Set-Up Requests

In this section, we evaluate the average number of dropped flow set-up requests of each algorithm by varying the number of controllers from 5 to 10.

Figure 1 shows that both algorithm RAC and algorithm FRAC obtain better performance than the benchmark algorithms because the proposed algorithms minimize the average number of dropped flow set-up requests when placing the controllers in the network. Algorithm RAC performs slightly better than FRAC since RAC removes the controller which incurs the least average number of dropped flow set-up requests, while FRAC finds the controller leading to the least number of dropped flow set-up requests.

Algorithm CPP achieves the best performance among three benchmark algorithms. Algorithm SVVR aims to maximize the number of disjoint paths between switches and controllers, l-w-greedy tries to minimize the expected percentage of control path loss, and CPP deploys the controllers with the objective of optimizing the average communication latency of switches and controllers. Algorithm CPP potentially aggregates the requests on a subset of the network, which reduces the number of links on the control paths and the total number of requests caused by the single-link-failure.

4.3.2 Average Latency of Flow Set-Up Requests

In this section, we evaluate the average latency of flow set-up requests of each algorithm, assuming the number of controllers varies from 5 to 10.

Figure 2 depicts the average latency of flow set-up requests of RAC, FRAC, SVVR, l-w-greedy, and CPP. Algorithms RAC and FRAC achieve a similar performance. Algorithm l-w-greedy results in the worst performance, because algorithm l-w-greedy aims to optimize the reliability of control path between switches and controllers, which potentially leads to long control paths between the switches and controllers. Algorithms RAC and FRAC place the controllers considering

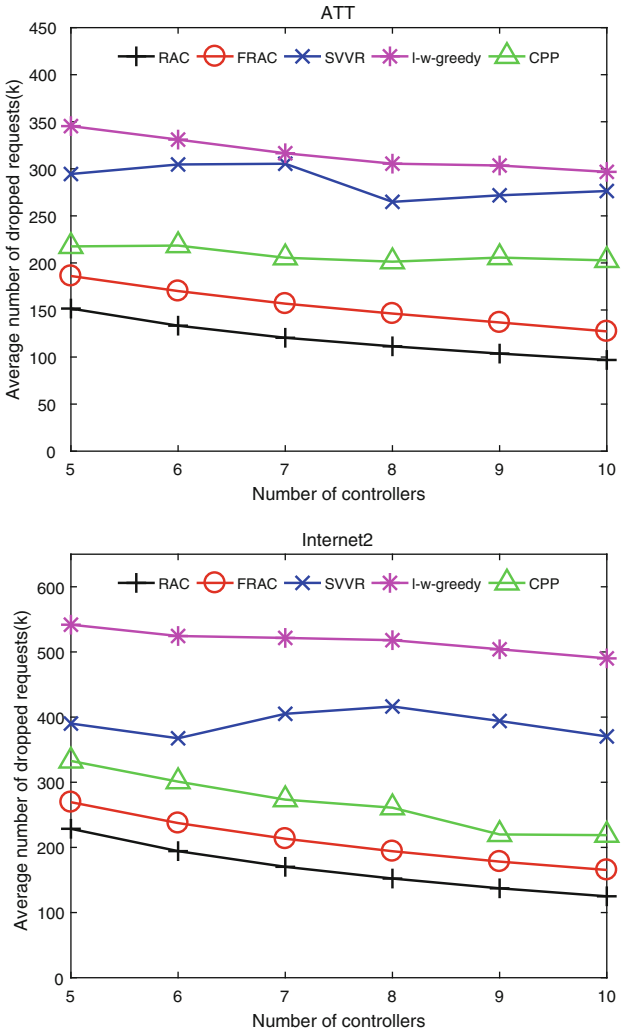

Fig. 1 The average number of dropped requests under different number of controllers

the number of dropped flow set-up requests under single-link-failure, so the two algorithms put the controllers close to the switches which generate a large number of flow set-up requests. Algorithm CPP minimizes the latency between the switches and the controllers without considering the number of flow set-up requests generated by the switches. Therefore, algorithm CPP obtains better performance than the other algorithms, where the distance between the switches and the controllers has an important impact on the controller placement.

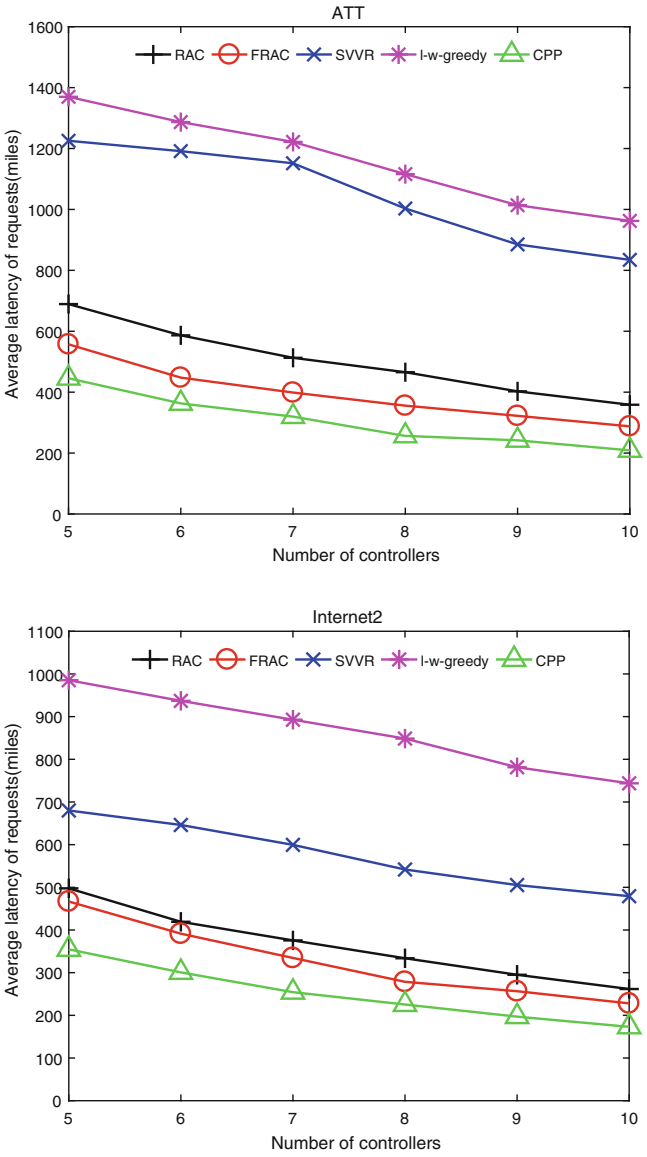

Fig. 2 The average latency of flow set-up requests under different number of controllers

5 Conclusions

Reliability is an important concern during controller placements in SDNs since link failures can cause disconnections between switches and controllers, and even incur cascading failures of other controllers. In this paper, we take the transmission

reliability of flow set-up requests into consideration during controller placements. We formulated a novel SDN controller placement problem with the aim to minimize the average number of dropped flow set-up requests due to the single-link-failure. We propose flow set-up request Reliability Aware Controller placement (RAC) algorithm and Fast-RAC (FRAC) algorithm for multiple-controller placements. Experiments are conducted through simulations. Our experimental results demonstrate that the proposed algorithms achieve competitive performance in terms of average number of dropped flow set-up requests under single-link-failure and average latency of flow set-up requests.

Acknowledgments This work is partly supported by the National Natural Science Foundation of China (61701162, U1836102), the Anhui Provincial Natural Science Foundation (1608085MF142), and the open project of State Key Laboratory of Complex Electromagnetic Environment Effects on Electronics and Information System (CEMEE2018Z0102B).

References

1. B.A.A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, T. Turletti, A survey of software-defined networking: past, present, and future of programmable networks. IEEE Commun. Surv. Tutor. **16**(3), 1617–1634 (2014)
2. M. Elhoseny, H. Elminir, A.M. Riad, X. Yuan, Recent advances of secure clustering protocols in wireless sensor networks. Int. J. Comput. Netw. Commun. Secur. **2**(11), 400–413 (2014)
3. B. Heller, R. Sherwood, N. McKeown, The controller placement problem, in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks* (ACM, New York, 2012), pp. 7–12
4. X. Yuan, M. Elhoseny, H.K. El-Minir, A.M. Riad, A genetic algorithm-based, dynamic clustering method towards improved WSN longevity. J. Netw. Syst. Manage. **25**(1), 21–46 (2017)
5. L. Yao, P. Hong, W. Zhang, J. Li, D. Ni, Controller placement and flow based dynamic management problem towards SDN, in *2015 IEEE International Conference on Communication Workshop (ICCW)* (IEEE, New York, 2015), pp. 363–368
6. G. Wang, Y. Zhao, J. Huang, Q. Duan, J. Li, A k-means-based network partition algorithm for controller placement in software defined network, in *2016 IEEE International Conference on Communications (ICC)* (IEEE, New York, 2016), pp. 1–6
7. M.F. Bari, A.R. Roy, S.R. Chowdhury, Q. Zhang, M.F. Zhani, R. Ahmed, R. Boutaba, Dynamic controller provisioning in software defined networks, in *2013 9th International Conference on Network and Service Management (CNSM)* (IEEE, New York, 2013), pp. 18–25
8. Y. Hu, W. Wang, X. Gong, X. Que, S. Cheng, Reliability-aware controller placement for software-defined networks, in *Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)* (2013), pp. 672–675
9. Y. Hu, W. Wang, X. Gong, X. Que, S. Cheng, On reliability-optimized controller placement for software-defined networks. China Commun. **11**(2), 38–54 (2014)
10. L.F. Müller, R.R. Oliveira, M.C. Luizelli, L.P. Gaspary, M.P. Barcellos, Survivor: an enhanced controller placement strategy for improving SDN survivability, in *Global Communications Conference (GLOBECOM)* (IEEE, New York, 2014), pp. 1909–1915
11. Y. Fan, Y. Xia, W. Liang, X. Zhang, Latency-aware reliable controller placements in SDNs, in *International Conference on Communications and Networking in China* (Springer, Basel, 2016), pp. 152–162

12. A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, C. Diot, Characterization of failures in an IP backbone, in *INFOCOM 2004* (IEEE, New York, 2004), pp. 2307–2317
13. S. Knight, H.X. Nguyen, N. Falkner, R. Bowden, M. Roughan, The internet topology zoo. IEEE J. Sel. Areas Commun. **29**(9), 1765–1775 (2011)
14. Internet2 Open science, scholarship and services exchange. [Online]. Available: http://www.internet2.edu/network/ose/