Reducing Redundant Control Messaging for Fast Response in SDN

Gang Wu, Peng Yang, Jingjing Luo, and Li Yu
School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China
Email: {hust_wugang, yangpeng, luojingjing, hustlyu}@hust.edu.cn

***Abstract*—The control mechanism of SDN tends to be distributed when the network scale expands. As the number of controllers grows, how to efficiently respond to network events in a timely fashion becomes challenging. Specifically, distributed controllers should reach a consensus on network states before making a network update decision in legacy consensus algorithm, which causes unexpected latency. While eventual correctness algorithms relies on Gossip schemes for fast recovery from network failures, leading to highly redundant controller responses when handling one single network event. In this paper, we aim at developing efficient control algorithm that eliminates unnecessary controller processing, facilitating rapid response to network events. To this end, we design a joint controller selection and message forwarding scheme, which identifies a subset of controllers for control message dissemination. An optimization problem is formulated with the objective of minimizing the total control overhead, which is found to be reducible to the K-center problem. Simulation results show that our method reduces up to 61% of the control overhead compared with legacy benchmarks.**

I. Introduction

With the soaring amount of data traffic and growing scale of network, conventional network architecture faces high pressure in handling the large amount and highly diversified network events. In contrast, Software Defined Networking (SDN) simplifies network control by decoupling the control and data plane, resulting in fast computing of forwarding states and rapid installation of control policy [1][2][3]. Network update is usually triggered by an internal or external network event. The former includes network dynamics caused by the controller via policy enforcement, while the latter usually means data plane changes (e.g., link and node failure), which call for rapid response from the network controllers.

When responding to network events, the architecture of SDN control plane can be divided into two categories according to the network scale: single-controller and multi-controller [4]. Single-controller SDN uses only one controller to ensure centralized control in small scale networks, while multiple controllers are usually deployed for large scale networks. In multi-controller networks, it is crucial to reach consensus among different controllers before making network update decisions, i.e., their views on network state should converge.

In the case of distributed control plane, most existing literature employs consensus-based mechanisms to ensure multiple controllers manage the network like a single controller logically, such as Paxos [5] and RAFT consensus [6][7]. However, those consensus mechanisms require constant information exchange among different controllers, which incurs large latency. Katta *et al.* proposed another consensus algorithm called Ravana [8], which requires even stronger consistency among controllers. Consistency is essential for centralized control, yet it is costly in terms of both control overhead and response delay. Recently, Panda *et al.* proposed an approach called simple coordination layer (SCL) to handle network events in distributed SDN control plane, which simplifies the coordination and accelerates the recovery from network failures [9]. SCL relies on eventual correctness algorithm via Gossip to ensure convergence among multiple controllers. Instead of pursuing controller consensus before making update decisions, SCL focuses on the eventual correctness, i.e., network information among all controllers will converge after event response. By reducing the controlling overhead for consensus, SCL can accelerate network response to events. However, new challenges arise as SCL requires all controllers to process events upon notification, which results in resources wastage and redundant network updates.

Although consensus and eventual correctness algorithms are two ends for convergence, in this paper, we are committed to develop network update algorithms that are of both low latency and minimal redundant responses when responding to network events. To this end, we propose a two-step network update scheme. Firstly, a subset of controllers are identified based on the control plane topology to process the network events. Secondly, the forwarding principle for network response is designed based on the selected subset. Similar to SCL, each SDN controller responds to the network events independently, i.e., there is no coordination among controllers before processing network events, which contributes to faster response. Note that control plane will reach eventual correctness after several Gossip periods, it is unnecessary for all controllers to respond to each event. If some of the controllers' network state information is correct, Gossip mechanism can ensure the dissemination of event information from the correct controllers to the others. In this way, the control plane can be divided into two types, one receives event and responds, while the other receives event and update information through Gossip without responding. As a result, the total response frequency from control plane can be reduced. The problem boils down to find a optimal division of network controllers. Based on this controller subset, an algorithm that reduces redundant response frequency and response latency is proposed. Experimental results show that repeated control response frequency

is reduced by 61% on average compared with SCL. The main contribution of this paper is summarized as follows:

- We focus on multi-controller scenario and propose a new control scheme that responds network events by only a subset of controllers, while the other ones contribute to network convergence.
- We derive the necessary condition of this controller subset, and find the optimal solution by reducing the controller selection problem to the *K-center* problem.
- Based on controller subset selection, we devise a novel control plane forwarding scheme, which is proved to be effective in reducing the network response frequency, as demonstrated by extensive simulation comparisons.

The remainder of this paper is organized as follows. Related work is introduced in Section II. In Section III we present the problem formulation and complexity analysis, and the proposed solution is given in Section IV. Section V shows the performance of our scheme compared with other consensus algorithm and SCL. We conclude this paper in Section VI.

II. Related Work

In SDN control plane, to maintain the stability during network update, consensus algorithms are employed. Consensus algorithms such as Paxos [5] and RAFT [6][7] usually select a master controller from control plane to carry out and enforce network update decisions. Every event from data plane will be forwarded to the master controller. When the master controller notifies an event, before responding, it forwards the event to all controllers and waits for a majority commit. Then, the master controller makes a network update decision, which is also been disseminated to the whole control plane for another majority commit. Before the majority commits the update, the control plane will not respond to the event. This consistency ensures the correct state information among all controllers, however it induces large latency in coordination. To address this issue, Sakic *et al.* propose an adaptive state consistency for different states to reduce coordination delay [10], Katta *et al.* focus on fault-tolerance while introduces even strong consistency [8]. Other works such as [11] researches for the verification safety problem of RAFT consensus protocol.

In particular, SCL focuses on fast recovery from network failures [9]. As the time duration from incorrectness to correctness is enlarged by control coordination, so it advocates all controllers, rather than the master controller alone, are responsible for processing network events and enforcing the corresponding updates. SCL argues for eventual correctness, which means state information in multiple controllers should converge eventually, rather than that before response. To this end, periodic Gossip mechanism [12] is used in SCL, which exchanges information between every neighboring pair. Research work on Gossip mainly discuss the convergence rate [13][14], while leaving the resource utilization unattended.

Since controllers are independent and do not coordinate with each other, if an event is not sent to some of the controllers, these controllers will never be aware of the event, and thus do not respond. Based on such rationale, we consider that if network events are only sent to a certain subset of controllers, the response time of the whole control plane is determined by this subset. Hence, this scheme contributes to reduced control overhead, and meanwhile guarantees eventual correctness.

TABLE I
Notations Table

Notation	Meaning
G	topology graph of control plane
N	node set of G
E	edge set of G
n	size of N
N_i	neighbors of node n_i
$\triangle$	the greatest degree of G
S	a subset in G
$\bar{S}$	the complementary set of S
T	the response times of a node or a set
L_{ec}	eventual correctness latency of whole control plane

III. Problem Formulation

The control plane in SDN can be represented as a connected undirected graph G including node set N and edge set E, i.e. $G = (N, E)$[1]. $|N|$ and $|E|$ denotes the size of N and E respectively and we assume $|N| = n$. For each node n_i in N, N_i denotes its neighbors and $|N_i|$ is its degree. N_{ci} represents the closed neighbors of n_i, which consists of n_i itself and nodes in N_i. The greatest degree in G is denoted by $\triangle$. Important notations are presented in TABLE I.

A path between two distinct nodes u and v is composed of a sequence of nodes $\{n_1, n_2, \cdots, n_k\}$ such that $(n_i, n_{i+1}) \in E$ $(1 \leq i \leq k)$. Thus, a path between u and v can be denoted as $p(u, v)$ and the length of a path is $|p(u, v)|$, equivalent to the hop count along the path. Among several paths between u and v, the shortest path is denoted as $p_s(u, v)$ and the distance between two distinct nodes u and v can be defined via the shortest path as $dist(u, v) = |p_s(u, v)|$.

To proceed, we introduce the two main functions used in our discussion. Response function is associated with each node in G and the latency function that depends on end-to-end distance. We consider the transmission latency along the path, with detailed definitions as follows.

Function $T(\cdot)$ is the indicator of responses induced by each controller. For each node $n_i \in N$, we assign x_i such that if n_i has responded to the network event, $x_i = 1$, otherwise $x_i = 0$. Formally,

$$T(n_i) = \begin{cases} 1, & x_i = 1 \\ 0, & x_i = 0 \end{cases} \tag{1}$$

Therefore, the frequency of network update execution of control plane is determined by the subset S of controllers which actually respond to network event, i.e.,

$$T(G) = \sum_{n_i \in G} x_i = \sum_{n_i \in S} x_i = T(S). \tag{2}$$

[1]Since we focus on the control plane, we use nodes to represent controllers, and G to represent control plane topology in the following discussion.

Our objective is to reduce response overhead of controllers, i.e.,

$$\min T(G) \Rightarrow \min \sum_{n_i \in G} x_i \tag{3}$$

As discussed above, the responsibility of controllers without coordination motivates us to find a subset $S \subset G$. Eq. (2) suggests that the response times of G is equivalent to that of S, so it is crucial to find a subset S with minimal size, so as to respond network events at minimum overhead.

Let function $L(\cdot)$ represent the latency on each path $p_s(u,v), \forall u, v \in N$, which is determined by the distance between two nodes. Thus, the latency between u and v is denoted as

$$L(u,v) = L(dist(u,v)). \tag{4}$$

Since the subset S is used to respond to the network events, the transmission latency of network event from arbitrary node in control plane topology to the subset S should be discussed firstly. Given a subset S of G, the event forwarding latency between an arbitrary node r and S can be defined as the maximal latency between r and the node of S, because not until the last node in S receives the network event all the nodes in subset S will reach convergence of the event. Thus, the latency between r and S is defined as the maximal latency between r and $n_i \in S$:

$$L(r,S) = \max_{n_i \in S} L(r, n_i). \tag{5}$$

Similarly, the convergence latency between the set S and the complementary set of S in G, denoted as $\bar{S}$, can be defined as the maximal latency between the nodes of the two sets

$$L(S,\bar{S}) = \max_{\substack{n_i \in S \\ n_j \in \bar{S}}} L(n_i, n_j). \tag{6}$$

The latency of eventual correctness of the whole control plane is determined by the time when an arbitrary controller firstly receives an event to the last notified controller through Gossip, which should be minimized for fast convergence. The latency consists of three parts: latency from an arbitrary controller to the target subset, latency from this subset to its complementary set and the delay induced by Gossip (denoted as t_g). Suppose T_g is the total duration of Gossip mechanisms, t_g is the delay caused by several gossips from one controller to another and thus $t_g = k \cdot T_g$, where k is the times of gossips. Therefore, the latency of eventual correctness of the whole control plane can be written as

$$L_{ec} = L(r,S) + L(S,\bar{S}) + t_g. \tag{7}$$

The delay t_g is associated with the second term $L(S,\bar{S})$ because the smaller distance between S and $\bar{S}$, the fewer gossips will be needed from S to $\bar{S}$. Thus, minimizing $L_{ec}(G)$ is boiled down as

$$\min L_{ec}(G) \Rightarrow \begin{cases} \min L(r,S) \\ \min L(S,\bar{S}) \end{cases} \tag{8}$$

Furthermore, since the times of controller responses and the latency of eventual correctness should be minimized simultaneously, the objective function can be represented as

$$\Phi = \begin{cases} \min T(G) \\ \min L_{ec}(G) \end{cases} \tag{9}$$

The selected subset S should meet the conditions of Φ in Eq. (9), which means the size of S, the distance between r and S, as well as the distance between S and $\bar{S}$ should be minimized concurrently. Thus, the optimal solution Φ can be formally written as

$$\Phi \Rightarrow \begin{cases} \min |S| \\ \min L(r,S) \\ \min L(S,\bar{S}) \end{cases} \tag{10}$$

Equation (10) indicates the conditions that the subset should satisfy. In practice, it is difficult to find such a subset directly in a general graph. Thus, we analyze these minimizing terms to find a feasible solution in next section.

IV. Intractability and Solution

In this section, we analyze the intractability of the optimization problem given by Eq. (9) and propose an algorithm by resorting to the solution of the *K-center* Problem.

A. Intractability of the optimization problem

We start our analysis from the form of Φ in Eq. (10). Firstly, since we aim at reducing redundant responses of responsible controllers, the size of selected controller subset should be limited by a predefined threshold, i.e. $|S| \leq M$, where M is an integer smaller than n. Meanwhile, the latency of eventual correctness should be also limited for the stability of network. When considering $L_{ec}(G)$, as discussed before, t_g has positive relation with $L(S,\bar{S})$. From the definitions of $L(r,S)$ and $L(S,\bar{S})$, it can be inferred that $L(r,S)$ is also positively related to $L(S,\bar{S})$ because once the distance between an arbitrary node and the subset S is minimized, the distance from S to its complementary set $\bar{S}$ is also minimized. Thus our objective can be further reduced as: finding a subset S with limited size such that the latency from an arbitrary node r to S is minimized, and it can be formulated as

$$\begin{aligned} &\min L(r,S) \\ &s.t. \quad |S| \leq M \end{aligned} \tag{11}$$

The above formulation is a cover problem in graph theory which can be reduced to the *K-center* problem [15].

B. K-center Problem

The *K-center* problem is about finding the best location of facility in topology and is defined as follows. For a graph $G = (N, E)$, find a subset $S \subset N$ of size at most K such that the distance from each node in N to S and the size of the subset is minimized. The objective function of *K-center* problem is defined as follows

$$\min \max_{n \in N} dist(n,S) \tag{12}$$

$$s.t. \quad |S| \leq K$$

The definition of *K-center* problem is quite similar with our objective target. Since the forwarding latency between an arbitrary node and a subset is mainly determined by the distance between them in this paper, our optimal problem can be reduced to *K-center* problem in control plane topology. Unfortunately, the *K-center* problem is known to be NP-hard, so it is unlikely to be solved by polynomial-time algorithms [16]. To proceed, we resort to the minimum dominating set to solve this problem, which was proposed by [17].

The dominating set is defined as follows. For a given graph $G = (N, E)$, there exists a subset D, such that $\forall n \in N$ is either in D or adjacent to a node in D, the subset D is called a dominating set (DS). Minimum dominating set (MDS) can be derived from DS if its cardinality is minimized. The description of MDS problem is that, in the graph G, D represents the dominating set and a_i, which is assigned to each node $n_i \in N$ such that if $n_i \in D, a_i = 1$; otherwise $a_i = 0$, find a DS with minimal sum of a_i [18]. The problem of finding MDS in a graph is also proved to be NP-hard [16] and it can be represented as

$$\min \sum_{i=1}^{n} a_i. \tag{13}$$

Though it may not be tractable directly, MDS can still be used to solve our optimization problem. On one hand, by inducing MDS, the upper bound of cardinality of objective subset S is up to $n/2$ according to [18], as a result the redundant response overhead can be reduced by 50% at least. On the other hand, each node in $\bar{S}$ is adjacent to S so that only one round of gossip is required for control plane to reach eventual correctness, which will reduce large delay in the convergence stage. Therefore, based on MDS, we design an algorithm of forwarding event in the responsible control plane to reduce repeated responses.

C. Algorithm design for event forwarding in control plane

We focus on the reduction of redundant updates induced by the responsibility of controllers and discuss this problem in the similar control plane scenario as that of SCL [9]. The controllers are responsible to address the network events once they are notified, and the Gossip mechanism is used for eventual correctness. Since controllers in SCL are replicas, it is unnecessary to notify every controller of network event. Gossip executes periodically to ensure eventual correctness. Then, the MDS of control plane is selected to respond to events and gossip with other controllers.

Our algorithm is shown in Algorithm 1, which reduces redundant response times with low eventual correctness latency. Lines 1-3 sets up a MDS in case of initialization or changes (e.g. controller be added or deleted) of control plane by referring to [19], which is briefly described in Algorithm 2. The network update times T is set to 0 and the $Tgcount$ initially equals to the gossip period. $Tgcount$ decreases over time. Lines 4-5 is the parameter initialization phase and lines 6-7 is the stage that events are forwarded to the control plane, where it is sent to a random nearest controller r for response and r can determine which controllers to send the event, and the event is then sent to MDS in our scheme. Controllers in MDS respond to the event upon reception as well, the latency from r to n in MDS is recorded as $L_f[n]$, the total response times as T, and the rest time of $Tgcount$ as tg in lines 8-12. In lines 13-16, when the Gossip period completes, each controller in MDS exchanges its event information to its neighbors and we record this latency. The maximal latency of L_f is extracted as L_1 in line 18 and the average value of tg is obtained in line 19. The latency of eventual correctness is the sum of L_1, L_2 and Tg in Line 20.

Algorithm 1 Event Forwarding Algorithm among Multiple Controllers

Input: graph $G = (N, E)$
Input: an event from data plane
Output: small update times and eventual correctness latency
1: **while** network control plane is initialized or changed **do**
2: find an MDS as D in G (Algorithm 2)[19]
3: **end while**
4: $T \leftarrow 0$
5: $Tgcount \leftarrow$ period of Gossip
6: event is sent to the nearest controller r
7: r responds to the event and forwards it to D
8: **for** each $n \in D$ **do**
9: $L_f.append(L(r, n))$
10: n responds to the event
11: $T \leftarrow T + 1$
12: $tg[n] \leftarrow Tgcount$
13: **while** $Tgcount == 0$ **do**
14: n gossips with its neighbors, $N[n]$
15: $L_2[n] \leftarrow L(n, N[n])$
16: **end while**
17: **end for**
18: $L_1 \leftarrow$Extract-Max(L_f)
19: $Tg \leftarrow E[tg]$
20: $L_{ec} \leftarrow L_1 + L_2 + Tg$
21: **return** T, L_{ec}

Algorithm 2 transforms the integer problem of MDS to liner program using LP relaxation. It finds solutions of the linear problem where each node calculates the corresponding a_i, which is similar to the classic greedy dominating set algorithm that extracts the a value of each node distributedly [19].

D. Complexity analysis

Algorithm 2 sets up a DS in $O(k^2)$ rounds with an expected size of $k\triangle^{2/k}\log(\triangle)|DS_{OPT}|$ where $|DS_{OPT}|$ is the cardinality of MDS, and k is an arbitrary constant parameter [19]. Since our algorithm needs rounds of the size of DS derived from Algorithm 2, it takes $O(k\triangle^{2/k}\log(\triangle) + k^2)$ rounds in case of control plane initialization and changes, or $O(k\triangle^{2/k}\log(\triangle))$ rounds in other situations for event process and convergence in our algorithm.

Algorithm 2 Constant-time Distributed Dominating Set Approximation [19] (the greatest degree $\triangle$ known)

Input: a_i of each node
Output: MDS solution x_{DS}

derive linear program of MDS using LP relaxation
for each node n_i **do**
  $a_i \leftarrow 0$
end for
solve LP of MDS to determine the value of a_i
sen a_i to all neighbors
if $\sum_{j \in N_i} a_j \geq 1$, N_j is the neighbors of node n_i **then**
  include v_i into DS
end if
for each solution a_i and degree parameter $\delta_i^{(2)}$ **do**
  $p_i \leftarrow \min\{1, a_i \cdot \ln(\delta_i^{(2)} + 1)\}$
  if $p_i > 0$ **then**
    $a_{DS,i} \leftarrow 1$
  else
    $a_{DS,i} \leftarrow 0$
  end if
  if $a_{DS,j} = 0$ for all $n_j \in N_i$ **then**
    $a_{DS,i} \leftarrow 1$
  end if
end for

V. Numerical Emulation

A. Emulation setup

To evaluate the performance of the proposed method, we implement our algorithm on NetworkX[2], which is a Python package for emulating the structure and function of network graph. We create a responsible controller class to run our algorithm. We also create switches that are responsible of forwarding events to controllers. Our experiment runs on Pycharm 2017.2.3[3] Platform with Python 3.6[4].

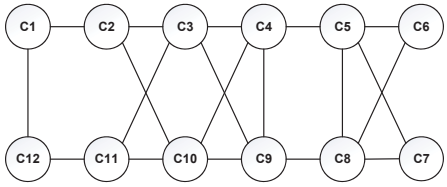
We use the topology of Google's inter-data center Software Defined WAN called B4 [20], which consists of 12 nodes and 19 links. Figure 1 illustrates the global deployment of Google B4 WAN and its topology graph. In our evaluation, the response times on each node are investigated. In addition, for simplicity, we record the number of hops to denote the latency, where the response latency is denoted by the hops count from data plane to controller that firstly responds to the event, and the convergence or eventual correctness latency is represented by the hops count from the first notified controller to the last one. Note that, from Eq. (7), t_g also affects the latency of eventual correctness. To account for the time duration of Gossip, we assume that the Gossip period is equivalent to the latency from event occurrence to reception on MDS controllers such that controllers can respond to events steadily.

As for the stage that the event is sent from data plane to control plane, we suppose that each controller is directly connected with a data plane switch which generates and forwards network event to the controller. Network event is generated by a random switch and then forwarded to the nearest controller. We compare our algorithm with the following benchmarks:

- RAFT: A legacy consensus algorithm proposed in [6] for the consistency among distributed controllers.
- SCL: A scheme using Gossip algorithm for fast recovery from network failures, reaching correctness eventually [9].

When implementing the RAFT algorithm, a leader controller should be elected firstly. Based on the leader election conditions in [7], $C4$ is elected as leader because it has a smaller average distance to other nodes in B4 topology. For our algorithm, we select a MDS during initialization, which includes $C1$, $C3$, $C5$ and $C10$. We suppose each event from data plane is sent to a random controller firstly and then forwarded to the whole control plane. We run each algorithm 20 times and show the average results.

(a) Global deployment of Google B4 WAN



(b) Topology graph of Google B4 WAN

Fig. 1. Google's inter-data center Software Defined WAN

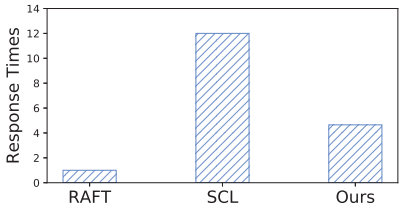


Fig. 2. Average Response Times of 3 Schemes

B. Results analysis

In terms of response times of controllers, our algorithm outperforms SCL by introducing MDS into to control plane, where the response overhead of whole control plane has been reduced by 61% compared with SCL in Figure 2. RAFT algorithm reaches the best result of once response to network

[2]http://networkx.github.io/
[3]http://www.jetbrains.com/pycharm/
[4]https://www.python.org/

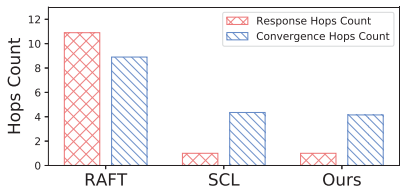


Fig. 3. Average Response and Convergence Hops Count of 3 Schemes

TABLE II
AVERAGE PROGRAM RUNTIME OF 3 SCHEMES

Scheme	Program Run Time(ms)
RAFT	1.000
SCL	1.002
Ours	0.999

event because after coordination among controllers just one exact response is required for network update. Figure 3 shows that SCL and our scheme both perform much better than RAFT in terms of response rate. Because of responsibility, controllers respond to event on reception, it takes only one hop latency for the nearest controller to handle network event in our scheme and SCL, while the master controller of RAFT needs coordination for a majority of commits to the event for network update. Note that since we aim at fast response, the response rate is achieved at the cost that there are repeated responses in our algorithm when compared with RAFT. To conclude, our algorithm reduces redundant responses effectively and meanwhile ensures fast event response of control plane.

As for the convergence latency, it takes twice coordination for all controllers to be aware of the event and then update in RAFT, thus, the convergence of RAFT needs more hops, as shown in Figure 3. While for SCL and our design, controllers are actually aware of the event after they receive the forwarded event from other controllers or through Gossip. Figure 3 shows their hops counts are lower than RAFT and our algorithm needs a fewer hops to reach eventual correctness because MDS accelerates the Gossip stage in our design. Figure 3 also shows another difference between legacy consensus algorithm and eventual correctness scheme, where control plane responds to network event after convergence in RAFT while for SCL and our method, convergence happens after response, causing faster event response than RAFT. In addition, we record the average program runtime of the three schemes for single event response in TABLE II and the results show that our algorithm achieves same complexity with better performance.

VI. CONCLUSION

Agile controller response is crucial to fast recovery of network failures. Observing that existing control schemes lead to unnecessary overhead when dealing with network events, in this paper, we propose that the redundant response can be reduced by identifying a suitable controller subset. We revealed the conditions of redundant response reduction, and proper latency threshold of eventual convergence that the elected controller subset should meet. Based on these conditions, we formulate our optimal problem and solve it by resorting to the *K-center* problem. We design an algorithm of event forwarding in control plane, which significantly reduces the redundant responses. Emulation results show that our scheme outperforms other benchmarks with reduced response redundancy and latency.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (NSFC) (No. 61871437).

REFERENCES

[1] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.
[2] Y. Li and M. Chen, "Software-Defined Network Function Virtualization: A Survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.
[3] P. Yang, N. Zhang, Y. Bi, L. Yu, and X. S. Shen, "Catalyzing Cloud-Fog Interoperation in 5G Wireless Networks: An SDN Approach," *IEEE Network*, vol. 31, no. 5, pp. 14–20, 2017.
[4] F. Bannour, S. Souihi, and A. Mellouk, "Distributed SDN Control: Survey, Taxonomy, and Challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 333–354, 2017.
[5] T. D. Chandra, R. Griesemer, and J. Redstone, "Paxos Made Live: An Engineering Perspective," in *Proc. of ACM*. ACM, Portland, Oregon, USA, 2007, pp. 398–407.
[6] H. Howard, "ARC: Analysis of Raft Consensus," University of Cambridge, Computer Laboratory, Tech. Rep., 2014.
[7] E. Sakic and W. Kellerer, "Response Time and Availability Study of RAFT Consensus in Distributed SDN Control Plane," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 304–318, 2018.
[8] N. Katta, H. Zhang, M. Freedman, and J. Rexford, "Ravana: Controller Fault-Tolerance in Software-Defined Networking," in *Proc. of ACM SIGCOMM*. ACM, Santa Clara, California, USA, 2015, pp. 4:1–4:12.
[9] A. Panda, W. Zheng, X. Hu, A. Krishnamurthy, and S. Shenker, "SCL: Simplifying Distributed SDN Control Planes," in *Proc. of 14th USENIX Symposium on NSDI*, Boston, USA, 2017, pp. 329–345.
[10] E. Sakic, F. Sardis, J. W. Guck, and W. Kellerer, "Towards adaptive state consistency in distributed SDN control plane," in *IEEE Proc. of ICC*, Paris, France, 2017, pp. 1–7.
[11] D. Woos, J. R. Wilcox, S. Anton, Z. Tatlock, M. D. Ernst, and T. Anderson, "Planning for Change in a Formal Verification of the Raft Consensus Protocol," in *Proc. of ACM SIGPLAN*, St. Petersburg, FL, USA, 2016, pp. 154–165.
[12] F. He, A. S. Morse, J. Liu, and S. Mou, "Periodic Gossiping," in *Proc. of IFAC*, Milano, Italy, 2011, pp. 8718–8723.
[13] G. Shi, B. Li, M. Johansson, and K. H. Johansson, "Finite-time convergent gossiping," *IEEE Transactions on Networking*, vol. 24, no. 5, pp. 2782–2794, 2016.
[14] F. He, S. Mou, J. Liu, and A. Morse, "Convergence rate on periodic gossiping," *Information Sciences*, vol. 364-365, pp. 111 – 125, 2016.
[15] S. Khuller and Y. J. Sussmann, "The Capacitated K-center Problem," *SIAM Journal on Discrete Mathematics*, vol. 13, no. 3, pp. 403–418, 2000.
[16] M. R. Garey, "Computers and Intractability: A Guide to the Theory of NP-completeness, Freeman," *Fundamental*, 1997.
[17] B. Robič and J. Mihelič, "Solving the k-center Problem Efficiently with a Dominating Set Algorithm," *Journal of computing and information technology*, vol. 13, no. 3, pp. 225–234, 2005.
[18] G. Mahalingam, "Connected Domination in Graphs," University of South Florida, Tech. Rep., 2005.
[19] F. Kuhn and R. Wattenhofer, "Constant-time distributed dominating set approximation," *Distributed Computing*, vol. 17, no. 4, pp. 303–310, 2005.
[20] e. a. Jain Sushant, "B4: Experience with a Globally-Deployed Software Defined WAN," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, 2013.