IEICE TRANS. COMMUN., VOL.E102–B, NO.3 MARCH 2019

PAPER Special Section on Network Virtualization and Network Softwarization for Diverse 5G Services

Designing Distributed SDN C-Plane Considering Large-Scale Disruption and Restoration

Takahiro HIRAYAMA[†a)], **Masahiro JIBIKI**[†b)], ***and*** **Hiroaki HARAI**[†c)], ***Members***

SUMMARY Software-defined networking (SDN) technology enables us to flexibly configure switches in a network. Previously, distributed SDN control methods have been discussed to improve their scalability and robustness. Distributed placement of controllers and backing up each other enhance robustness. However, these techniques do not include an emergency measure against large-scale failures such as network separation induced by disasters. In this study, we first propose a network partitioning method to create a robust control plane (C-Plane) against large-scale failures. In our approach, networks are partitioned into multiple sub-networks based on robust topology coefficient (RTC). RTC denotes the probability that nodes in a sub-network isolate from controllers when a large-scale failure occurs. By placing a local controller onto each sub-network, 6%–10% of larger controller-switch connections will be retained after failure as compared to other approaches. Furthermore, we discuss reactive emergency reconstruction of a distributed SDN C-plane. Each node detects a disconnection to its controller. Then, C-plane will be reconstructed by isolated switches and managed by the other substitute controller. Meanwhile, our approach reconstructs C-plane when network connectivity recovers. The main and substitute controllers detect network restoration and merge their C-planes without conflict. Simulation results reveal that our proposed method recovers C-plane logical connectivity with a probability of approximately 90% when failure occurs in 100 node networks. Furthermore, we demonstrate that the convergence time of our reconstruction mechanism is proportional to the network size.

key words: *software-defined networking (SDN), distributed SDN control, failure recovery, control plane*

1. Introduction

Software-defined networking (SDN) technology such as OpenFlow [1] provides flexible configuration of network equipment via logically centralized controllers. Therefore, SDN technology is expected to offer dynamic and fine-grained traffic engineering and introduction of various epoch-making services. However, SDN control plane (C-Plane) architecture is vulnerable to some circumstances owing to centralization. For example, connectivity between controllers and switches will be damaged when control paths are disconnected. Besides, the controllers' response will degrade when flow requests get concentrated in a short term. Many protection or restoration methods to maintain controller-switch connectivity have been proposed [2], [3]. However, these approaches are mainly focused on covering a single failure on control paths. They cannot cope with network disruption induced by catastrophic failure such as seismic events.

Recently, distributed SDN control architecture has been discussed for scalability and robustness [4], [5]. Redundant placement of controllers prevents serious disconnection between controllers and switches due to mutual backup. Additionally, deployment of multiple controllers reduces concentration of flow requests to each controller using a load distribution technique. The coordinating functionality between multiple controllers is the critical part while realizing the smooth working of distributed control. Hereinafter, we term a controller having co-ordinative authority as "root" controller, and the other distributed controllers as "branch" controllers. The root controller's functionality includes collecting information, advertising commands from network operators to equipment, among others.

If network components are isolated from a management range of the root controller owing to large-scale failures, in the worst case, the isolated components cannot accommodate any more flows. Especially during catastrophic disasters, each separated domain is expected to cope with flow requests for urgent use such as exchanging safety and rescue information. Additionally, discarding unreachable flows at ingress nodes is essential for dealing with the increased traffic post disaster.

In this study, we first propose a network partitioning method to maintain controller-switch connectivity in distributed SDN C-plane against large-scale failures. We use graph-theoretic approach for network partitioning. We define robust topology coefficient (RTC) which denotes the probability of a sub-network isolating from its controller after a large-scale failure. We partition a network into sub-networks based on RTC and place branch controllers on each of them. As a result of simulations, network partitioning based on RTC exhibits higher robustness than the other network partitioning methods.

Nevertheless, predefined backup plans with partitioning based on RTC cannot maintain entire connectivity during possible failure patterns. To tackle such cases, we further introduce a reactive C-plane reconstruction scheme between distributed controllers and switches against catastrophic disasters, in this study [6]. For example, in case of seismic events, there is a risk that all primary and backup controller-switch connections are broken. In our reactive approach, the nodes disconnected from any controllers seek and re-

Manuscript received April 12, 2018.
Manuscript revised July 26, 2018.
Manuscript publicized September 20, 2018.

[†]The authors are with National Institute of Information and Communications Technology (NICT), Koganei-shi, 184-8795 Japan.
a) E-mail: hirayama@nict.go.jp
b) E-mail: jibiki@nict.go.jp
c) E-mail: harai@nict.go.jp

DOI: 10.1587/transcom.2018NVP0005

connect to a survived controller. Additionally, one of the strayed branch controllers from the root rises for playing the substitute root controller. In this proposal, each node has predetermined priority rank. The node having the highest rank of the uncontrolled ones plays the root role.

We also see to it that the separated areas are reconnected. Due to recovery from disconnection, two or more root controllers may exist simultaneously. To avoid roots' conflict, the C-plane architecture must be immediately reconstructed immediately after network recovery. In our proposal, the substitute coordinator nodes detect network recovery and merge their C-planes via negotiation with each other.

In this study, we evaluate a reactive C-plane reconstruction method against large-scale disruption and restoration, i.e., regional and simultaneous failure of nodes, and recovery from that, respectively. Simulation results reveal that our reconstruction mechanism recovers C-plane connectivity to the theoretical upper limit with a probability of approximately 90% in 100 node networks. The reconstruction scheme completes within approximately 2 min from when regional failure occurs. Furthermore, the convergence time of our reconstruction mechanism is proportional to the number of nodes. Therefore, our mechanism is scalable against network size.

This paper is organized as follows. In Sect. 2, we present an overview of distributed SDN control and C-plane recovery methods. We propose a network partitioning method based on RTC in Sect. 3 and evaluate it Sect. 4. We discuss reactive failover mechanism against network disruption and restoration in Sect. 5 and investigate it in Sect. 6. Finally, we conclude this study in Sect. 7.

2. Distributed SDN Architecture and Failover Mechanism

2.1 Distributed SDN Control

One of the purposes of deployment of multiple controllers is to distribute the load of each controller appropriately [4], [5], [7]. In this research, the distributed C-plane architecture is categorized into three groups namely, decentralized architecture with global or local view and hierarchical architecture [8]. For example, in Hyperflow [7], controllers exchange network information with each other and grasp information of the entire network. This is one of the decentralized architecture with global view. Therefore, optimal control is possible because all controllers share the entire network information. However, in this case, computational complexity is high. In Onix [4], each controller grasps information of its neighboring area only and exchanges abstract information with other controllers. This is one of the decentralized architecture with local view. Computational complexity is reduced by abstraction by sacrificing optimality.

Kandoo [5] has hierarchical architecture consisting of root controller and local controllers. Local controllers manage flows that traverse within their control domain, and root controller manages flows that travel between multiple local domains. This obvious role separation leads to load distribution. Scalability of this architecture is discussed in [8]. Therefore, decentralized architecture with local view and hierarchical architecture can adapt to large-scale networks from the view point of complexity.

2.2 Distributed Controller Placement

It is known that the controller placement problem is NP-hard [9]. Consequently, many researchers have been discussing controller placement methods because they are significant from various viewpoints such as response time and robustness. To shorten the response time to new flow requests, controllers are placed on the basis of latencies between controllers and switches in [9]. In [10], [11], they propose controller placement methods that focus on making the control path recovery easier.

Recently, some researchers have focused on the method of constructing a survivable SDN C-plane [12], [13]. For example, Savas et al. proposed C-plane design by considering large-scale failures [13]. They formalized and solved the optimization problem for minimizing the risk of control-path loss by using forecast of node or link failure probability in some disaster scenarios. However, their method is very complex for solving with large networks due to its complexity $O(m \cdot N^4)$, where m denotes the number of failure scenarios and N represents the number of nodes.

2.3 Failover Mechanisms of C-Plane Connectivity

Control path recovery techniques can be classified into two namely, proactive and reactive. The proactive approach is further categorized into two types: path and controller redundancy. In path redundancy, k backup paths are set at the same time when the primary path is established as shown in Fig. 1. Controllers periodically send probe packets to all switches. If a reply packet is not received from the primary path, the controller starts to use a backup path [3].

OpenFlow already supports multiple redundant controllers. Each switch can connect to several controllers. Only one controller can be the MASTER controller (CNT 1 in Fig. 1) among several controllers connecting to the switch. Once the MASTER is fixed, the other controllers (CNT 2 to

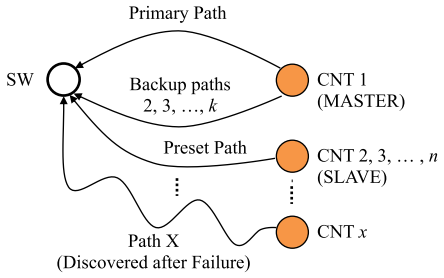


Fig. 1 Control path failover mechanisms. Backup paths: Path redundancy, CNT 2 to CNT n: Controller redundancy, Path X: Established after failure by reactive approach.

CNT n in Fig. 1) become SLAVE controllers. If MASTER controller goes down, one of SLAVE controllers becomes the new MASTER. In OpenFlow ver. 1.5, each switch sends a role change request to one of SLAVE controllers when its MASTER controller fails. If the version is older than 1.5, additional functionalities to enable a SLAVE become the MASTER are required.

Meanwhile, in reactive approaches, controllers and/or switches start to discover routes to the disconnected ones and reconnect when failure similar to Path X in Fig. 1 occurs.

3. Network Partitioning with Consideration of Isolation Probability of Sub-Networks

Many researchers have been concerned about connectivity repairing method or robustness evaluation against small-scale failures such as random deletion of nodes or links. However, impacts of large-scale failure should also be discussed for the actual network management. Network equipment placed in a certain area could be simultaneously broken by disasters such as earthquakes and hurricanes. In [14], the authors mention that we should consider the cases when the nodes in a certain area are simultaneously deactivated by assuming seismic events. According to the report about adverse effects of the Great East Japan earthquake on communication networks [15], network failure had widely occurred owing to building collapses and link disconnection.

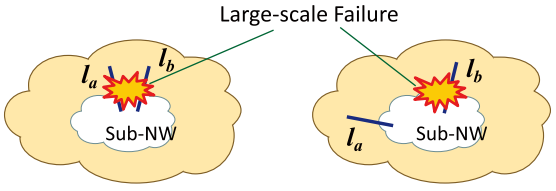
In this section, we consider the network partitioning and location of controllers that can cope with a large-scale failure efficiently. First, we assume the term **"Large-scale Failure"** to be the failure that extends to two or more adjacent links/nodes successively. Second, the term **"Sub-network (Sub-NW)"** denotes a domain that is managed by root or branch controllers.In the following section, we consider the Sub-NW partition method corresponding to various given network topologies. This method aims to minimize the influence of a large-scale failure (i.e. nodes are isolated from their controller).

3.1 The Index of Indicating a State of Isolated Topology

To prevent isolation of a node owing to the failure, we first partition the network topology ($G = (V, L)$) into subgraphs (i.e. Sub-NWs) for maintaining connectivity between the nodes and at least one controller. Second, we place a controller on each Sub-NW. The following index can be used for quantitatively indicating the connectivity between a subgraph and its exterior to find the isolated topology:

- The number of boundary links between a subgraph and the exterior.
- The number of nodes in a subgraph.

Generally, there is a multi-path or a protection of a cutting point in a graph as the method of improving these indices. However, these methods are not sufficient to minimize the influence when large-scale failures spread to adjacent nodes/links. Figure 2 shows simple examples of damages



(a) Boundary links (l_a and l_b) are close to each other. Sub-NW will be isolated due to large-scale failure.

(b) Boundary links are far from each other. Sub-NW will not be isolated after large-scale failure.

Fig. 2 The Sub-NW that is easy to isolate (a), and the one that is hard to isolate (b), against large-scale failures.

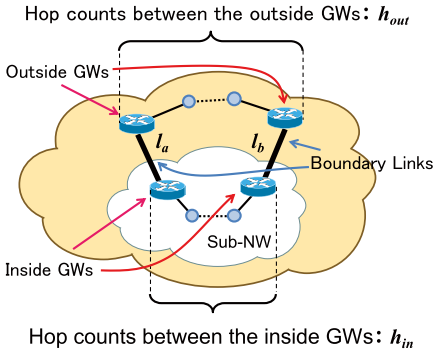


Fig. 3 Sub-NW and its boundary link pair, GWs.

to boundary links against large-scale failures. In this figure, two boundary links, l_a and l_b, connect the Sub-NW and its exterior nodes. If two boundary links are close to each other as shown in Fig. 2(a), the Sub-NW is less robust because it can be easily isolated in the case when a large-scale failure occurs. Meanwhile, if two boundary links are far from each other as shown in Fig. 2(b), the Sub-NW is hard to be separated by large-scale failures. Therefore, for each boundary link, it is necessary to formulate an index representing its robustness (Robust Link Coefficient: rlc) according to the shortest path length to other boundary links (for example, how many links do a single boundary link correspond to?).

Figure 3 shows the relation between a Sub-NW and its exterior. This Sub-NW is isolated when both links l_a and l_b are simultaneously disconnected. For example, the minimum case in which a large-scale failure spreading to adjacent nodes/links separates Sub-NW is the case when all nodes of the path between the inside gateways (GWs) or between the outside GWs are broken. Here, the probability that one of nodes between the inside GWs and between the outside GWs fails, is as follows:

$$\frac{1}{(h_{in}+1)+(h_{out}+1)} = \frac{1}{(h_{in}+h_{out}+2)}, \tag{1}$$

where $h_{in}(h_{out})$ denotes the hop count between the inside (outside) gateways of boundary links, l_a and l_b in Fig. 3. Hereinafter, we only use the hop count as the parameter that denotes how far a pair of nodes exist for simplicity. We regard that the following discussion is extensible in the future, even if any coefficients that reflect parameters such as the Euclidean distance or link bandwidth are introduced.

When one of the four GWs fails, the number of boundary links reduces from two to one. Therefore, the expected value of the number of survived boundary links is $1 \cdot 1/(h_{in}+h_{out}+2) \cdot 4$. Meanwhile, when one of the $(h_{in} + h_{out} - 2)$ non-GW nodes is broken, both the boundary links still survive. Therefore, the expected value of the number of the surviving boundary links is $2 \cdot 1/(h_{in} + h_{out} + 2) \cdot (h_{in} + h_{out} - 2)$. More specifically, the survival expected value (**SEV**) of l_a or l_b against the failure that occurs in these paths is considered as the index indicating that the boundary link can survive according to the scale of failure, i.e.,

$$SEV = \frac{4 + 2(h_{in} + h_{out} - 2)}{(h_{in} + h_{out} + 2)} = \frac{2(h_{in} + h_{out})}{(h_{in} + h_{out} + 2)} \quad (2)$$

For a certain boundary link l_i, if SEV of boundary link l_j is the maximum among the boundary links in its Sub-NW, then the SEV value is defined as E_i $(0 \leq E_i \leq 2, E_i = E_j)$. Here, consider the case of $h_{in} = h_{out} = 0$ (the case when boundary links overlap with each other). If one failure occurs on the inside or the outside GW in the abovementioned case, two links are cut simultaneously. Therefore, the link pair has only one link's worth of boundary links (i.e. one link has only 1/2 link's worth). Meanwhile, E_i exhibits the largest value when the hop count between two boundary links (h_{in} or h_{out}) becomes infinite (in that case, rlc should equal one). Therefore, so that $1/2 \leq rlc(l_i) < 1$ corresponding to the value of E_i, $rlc(l_i)$ is defined in the following:

$$rlc(l_i) = -\frac{2}{E_i + 2} + \frac{3}{2}. \quad (3)$$

We assume that the impact of the increase in the rlc value becomes smaller as E_i increases. Therefore, we introduce a simple upward convex function, which is the inverse proportional function of E_i. The constants 2 and 3/2 are introduced for adjust the range of rlc $(1/2 \leq rlc < 1)$. Additionally, the other upward convex functions are also suitable as long as their value converges one, as E_i reaches to two.

Finally, considering the survivability of the boundary link, we define a new index indicating the degree of influence for Sub-NW separation: Robust Topology Coefficient (**RTC**) of sub-NW $S(S = (V_S, L_S), S \subseteq G)$, which reflects rlc according to the position of other boundary links as follows:

$$RTC(S) = \frac{1}{|V_S|} \sum_{l \in L_S} rlc(l), \quad (4)$$

where $l \in L_S$ denotes a link belonging to Sub-NW S and $|V_S|$ represents the number of nodes in the Sub-NW S.

3.2 Network Partitioning Based on RTC

As shown in Appendix A, we validated that the first order difference of RTC (ΔRTC) changes in case of Sub-NW extension (i.e. addition of a new node to a Sub-NW). Figure 4 shows a state on the way of Sub-NW extension. The extension algorithm determines whether G_i and its neighbor G_j should include the nodes n_a and n_b or not. For example, if b' or b'' is the boundary, G_i has three boundary links. However, when b is the boundary, there is only one boundary link of G_i, so the possibility of separation between G_i and G_j is considered to be higher. The ideal borderline between G_i and G_j is candidate b. If n_a is included by G_i, the $RTC(G_i \cup n_a)$ value decreases from $RTC(G_i)$ value because the number of nodes in G_i increases and the number of boundary links decreases. Further, if n_b belongs to G_i, the $RTC(G_i \cup n_a \cup n_b)$ value becomes larger than $RTC(G_i \cup n_a)$ value because the number of boundary links increases. Therefore, when a network is to be partitioned, we decide whether or not a certain node v should be included in subgraph G_i according to the sign change of ΔRTC, (i.e., from a decrease tendency to increase tendency).

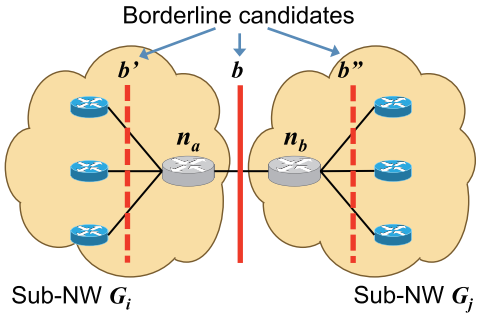


Fig. 4 Determining borderline between Sub-NWs. Candidate b is the ideal borderline between Sub-NW G_i and G_j.

Algorithm 1 RTC-based partitioning

Input: Network topology, G
Output: Set of k Sub-NWs, $G_i | i = 1, 2, ..., k$
1: **if** $neighbor(G_i) \neq \emptyset$ **then**
2: $v \leftarrow$ argmin $RTC(G_i \cup v)$
3: **if** $RTC(G_i \cup v) < RTC(G_i)$ **then**
4: $neighbor(G_i) \leftarrow neighbor(G_i \cup v)$
5: $G_i \leftarrow G_i \cup v$
6: go to line 1
7: **end if**
8: remove v from $neighbor(G_i)$
9: go to line 1
10: **end if**
11: $i \leftarrow i + 1$
12: $G_i \leftarrow v_f$ (choose a node from untreated nodes)
13: go to line 1

Algorithm 1 shows how to partition a network with RTC. In this pseudo code, $neighbor(G_i)$ denotes the set of external neighbor nodes of Sub-NW G_i. We start from the initial state that no node belongs to any Sub-NWs. First, we choose the farthest node v_f from the center node as the first member of Sub-NW G_1. The center node has the highest betweenness centrality value. Note that, betweenness centrality of a node is defined as the number of shortest paths that pass through the node. Second, we choose one of neighbor nodes of G_1, namely v, which has the minimum RTC value, if it is included in the G_1. Third, we add v to G_1 if the node satisfies $(RTC(G_1 \cup v) < RTC(G_1))$. This extension is

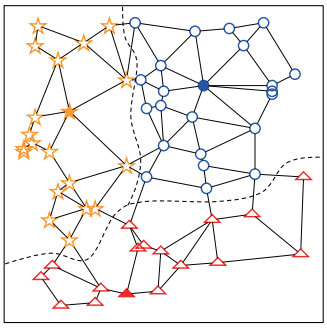


Fig. 5 An example of network separation via Algorithm 1 ($N = 60$). The nodes having the same shape (color) belong the same Sub-NW. The filled symbols denote the nodes that deploy controller functionality for their Sub-NWs. Star(orange), triangle(red), and circle(blue) Sub-NWs include 20, 17, and 23 nodes, respectively.

repeated until no neighbors can be included by G_1 (lines 1 to 9 in Algorithm 1). When extension process of G_1 converges, the farthest node from the center in the untreated nodes is chosen as the first member of G_2. This algorithm is repeated until all nodes belong to some Sub-NW. Figure 5 shows an example of network partitioning according to the algorithm. The network is partitioned such that the boundary links are geographically distributed (that contributes raising rlc) and their number is kept low.

We choose the first member of a Sub-NW according to the betweenness centrality. However, we already proved that a Sub-NW is not influenced by the starting point of the partition, as shown in Appendix B.

4. Evaluation of Network Partitioning

4.1 Network Topology and Controller Placement

In the simulations, we first generate network topologies by using geographic graph generating models. We regard these topologies as physical networks constructed by SDN switches. Second, we construct logical C-plane networks connecting controllers and switches over each physical network. In the physical network, N nodes are randomly placed in the square area having the length of one side as L. Further, the nodes are connected according to graph generation models. We use a Gabriel graph to generate physical networks as geometric graphs. Gabriel graph is one of planner graphs as shown in Fig. 5. In Gabriel graphs, nodes A and B construct a unidirectional edge when there are no nodes in the circle, whose center is the midpoint of A and B and whose diameter is the Euclidean distance between A and B. The authors of [16] compared some geographic graph models with an actual ISP physical network. As a result, Gabriel graph generates a network that is most similar to an actual one.

To construct C-plane, we aimed at partitioning a network into c Sub-NWs where each of the Sub-NWs has more than rN/c (r is tunable, $0 < r < 1$) nodes. As a result of partitioning according to Algorithm 1, the number of Sub-NWs may become larger than the target. In that case, we merge the minimum Sub-NWs and its minimum neighbor.

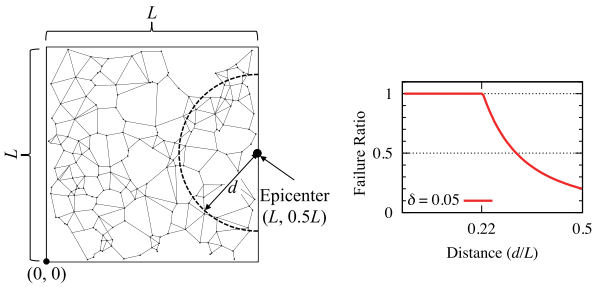


Fig. 6 Left: Seismic events. The epicenter is placed at eastside $(L, 0.5L)$ following the Great East Japan Earthquake. Right: Relationship between distance from epicenter and failure ratio.

If a Sub-NW has less than rN/c nodes, it is merged in the same manner. Next, we place controllers onto Sub-NWs. They are placed at the node having the highest betweenness centrality in each of Sub-NWs. The root node is the nearest controller from the center of the whole network, and the others are branch controllers. Branch nodes connect to the root node and switch (non-controller) nodes connect to the controller of their Sub-NWs. We set the number of node $N = 200$ based on the assumption that each node accommodates approximately 10,000 users in Miyagi Prefecture (with a population of 2.3 million). Further, we assume $L = 85$ km because the area is 7,300 km^2. In the following evaluations, we set $c = 10$ and $r = 0.6$. These values are coordinated in order to that controllers manage 20 switches on average. The number is derived from the evaluation in [17]. In the paper, Darianian et al. shown that the appropriate number of switches allocated to the controller is more than 16 for the effective use an ONOS controller (24-core 2.30-GHz CPU with Hyper-Threading).

4.2 Disaster Scenario

In the earthquake scenario, failure probability of nodes depends on their distance from the epicenter. The failure probability of nodes/links pe is denoted by $\min(1, \delta/(d/L)^2)$ where $\delta(> 0)$ is a tunable parameter, d is the distance from the epicenter placed at $(L, 0.5L)$ as shown in the left part of Fig. 6, and L is the length of one side of the target area. The relationships between normalized distance (d/L) and failure probability are shown in the right part of Fig. 6. The damage caused by an earthquake is affected by various factors such as geological features. However, we use the simple model in which the failure probability is inversely proportional to the square of the distance. This failure model is similar to the one discussed in [14], in which all nodes and links placed within a certain distance from the epicenter are broken.

4.3 Comparison with Existing Partitioning Methods

We numerically evaluate the superiority of the network partitioning with RTC as compared to three approaches: random placement (RND), betweenness-based placement (BET), and partitioning with Louvain method [18] (LOU).

In random (or betweenness-based) placement, we choose c nodes as controllers randomly (or according to betweenness centrality). Switch nodes connect to the nearest controller node. Louvain method is one of the popular partitioning methods aimed to achieve highly modulated partitioning. Modularity is determined by the fraction of intra-module and inter-module links. The term "module" denotes the set of nodes densely connected by intra-module links. As the fraction of inter-module links decreases and that of intra-module links increases, the portioned network earns high modularity. Louvain method may show high robustness if we regard modules as Sub-NWs, because nodes in a module are densely connected. In the evaluation, the network is separated to achieve the best modularity. Further, the modules are merged by keeping the modularity value as high as possible when the number of separated modules is larger than c.

We also evaluate the effects of path and controller redundancy. For path redundancy, switch nodes connect to each of their controllers via k-shortest paths (based on hop counts), which are determined by Yen's algorithm [19]. In the same manner, branch nodes connect to the root node via k-shortest paths. For controller redundancy, each switch connects to the first, second, ..., (n-1)th nearest controllers as SLAVE controllers in RTC and LOU. Switch node connects the second, third, ..., nth nearest controllers in RND and BET. After the failure, a switch is regarded as controllable by the root if at least one preset route to the root (including indirect routes via branch nodes) is available. Otherwise, a switch is viewed as controllable by the substitute root node if at least one path to any of the branch nodes is available.

Figure 7 shows the simulation results. In this subsection, we created 100 topologies of 200 nodes ($N = 200$). We examined robustness of C-plane for each topology with the disaster scenario as described in Sect. 4.2. On average, approximately 25% of nodes failed in each trial (dashed line in Fig. 7). Filled and blank bar graphs represent the mean value of C_r and $C_r + C_s$, respectively. C_r and C_s denote the number of controllable nodes by the root and substitute root nodes, respectively. Error bars show 95% confidence interval.

RTC is aimed at partitioning networks for establishment of robust controller-switch connectivity against large-scale failures. In other words, the goal is that the controller-switch connectivity within a Sub-NW is hardly separated after disaster. As a result, partitioning with RTC shows the best $C_r + C_s$ value whether reference to path and controller redundancy are used ($n = 3, k = 3$) or not ($n = 1, k = 1$). The $C_r + C_s$ value is 6%, 10%, and 9% larger than RND, BET, and LOU, respectively ($n = 1, k = 1$). Betweenness-based placement shows the best C_r value and the worst C_s value. This is because the controllers are concentrated nearby the center, and switch nodes at the end of the squared area tend to be isolated. Louvain method shows larger $C_r + C_s$ value than betweenness-based placement. However, it does not reach to that of partitioning with RTC. This difference is caused by the policy of our partitioning algorithm with RTC. As described in Sect. 3.2 and Fig. 2, our algorithm prefers a case such as in Fig. 2(b) than Fig. 2(a), i.e., Sub-NWs are hardly isolated due to geographically partitioned boundary links. As a result, the C_r value is superior to the case of RND placement. Additionally, as described in Sect. 3.2, our algorithm determines the borderline between two Sub-NWs based on the change of sign ΔRTC. As a result, Sub-NWs are separated so that they have many links inside, as seen in Fig. 4. This contributes to improving the C_s value as compared with other policy-based placements such as BET and LOU. In summary, RTC shows the best $C_r + C_s$ value in this evaluation.

Path redundancy improves C_r and controller redundancy improves C_s, by comparing the results of $(n, k) = (1, 1), (3, 1)$, and $(1, 3)$ (the results are not shown in this paper). Combination of these two redundancy methods ($n = 3, k = 3$) further improves controllability (approximately 20% increase from ($n = 1, k = 1$). However, its performance does not reach that of ideal "Reactive" approaches as shown in Fig. 7. We also evaluate the case when n and k are unlimited ($n = \infty, k = \infty$) with the assumption that the ideal reactive approach is used. It recovers C-plane connectivity perfectly after the disaster. Specifically, controller-switch connectivity will be reconstructed if at least one path to any of the controllers exists. In the reactive approach, $C_r + C_s$ value earned by each placement is almost the same. This result is interpreted as follows. Figure 8 shows examples of placements based on RTC and RND. In this figure, two Sub-NWs are directly connected by two links l_1 and l_2. Additionally, the link l_∞ placed at ∞-hop away, connects the Sub-NWs. When n and k are small (ex. $k = 2$) in random placement (Fig. 8(b)), the connections between nodes in the right Sub-NW and their controllers are lost if l_1 and l_2 are simultaneously broken. Meanwhile, all controller-switch connectivity is kept in RTC-based placement (Fig. 8(a)) after the links fail. However, the above difference disappears if the link l_∞ is used for redundancy, i.e., $k = \infty$.

In Fig. 7, the proactive approaches cannot reach the reactive ones, especially about C_r. According to the above results, the reactive approaches are essential to keeping controllability of the root node against the regional large-scale failures. So, we discuss a reactive approach in the next sections, which allow C_r (C_s) reach to the ideal.

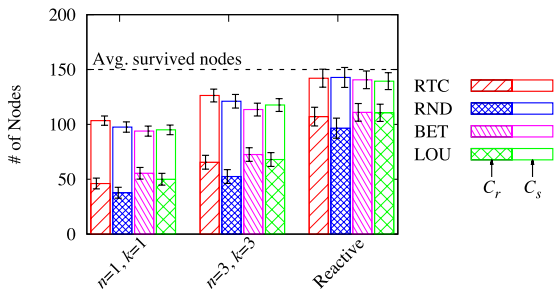
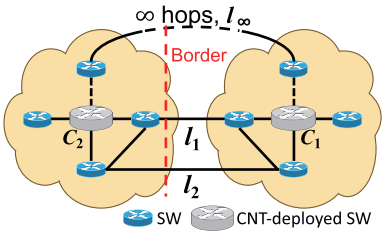
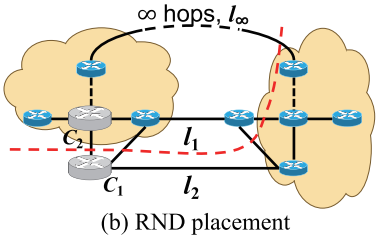


Fig. 7 Comparison between RTC and the other approaches.



(a) RTC placement



(b) RND placement

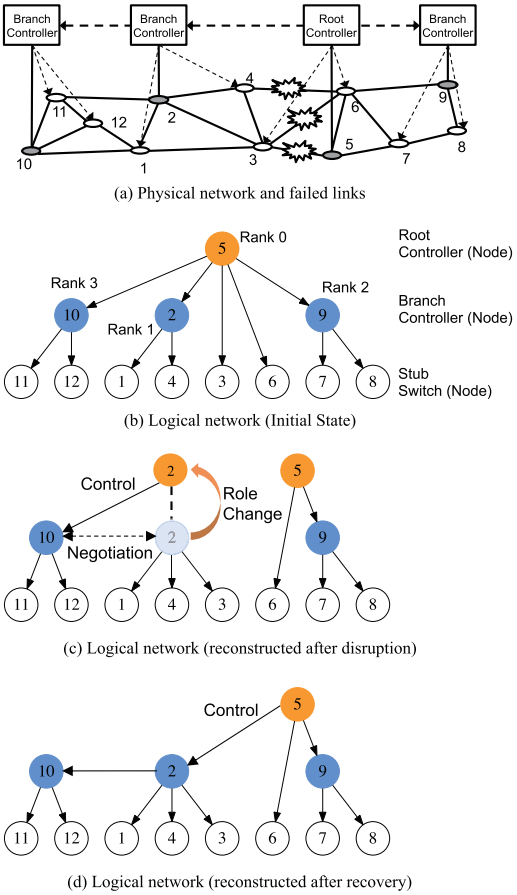
Fig. 8 Typical difference between RTC and RND placement.

5. C-Plane Reconstruction against Network Disruption and Restoration

In the previous section, we described how to separate a network into Sub-NWs. Simultaneously we show that proactive failover approach does not recover controller-switch connectivity perfectly. In this section, we introduce the reactive C-plane reconstruction mechanism for adapting to large-scale failures. We focus on reconstruction against network disruption and restoration. First, we discuss C-plane reconstruction procedure when a network is separated into some components due to large-scale failures. Backup branch controllers (nodes) rise as the substitute (sub) root nodes. Further, connections between controllers and isolated switches are reconfigured. Second, we introduce C-plane reconstruction procedure when separated components are reconnected. The root nodes of separated components communicate with each other and reconstruct the C-plane architecture.

5.1 Network Model

We focus on hierarchical architecture like Kandoo. A disaster may drastically change the network situation. In such a case, decentralized architecture just depending on local view may not adapt to the network change. Meanwhile, a root node in hierarchical architecture has a key role in network operation. First, multiple nodes are placed in the network, and controller-controller and controller-switch connections are also predetermined. Figure 9(a) shows an example of a network at the initial state. Figure 9(b) shows a logical C-plane structure of the network shown in Fig. 9(a). In this example, the root controller is deployed on node 5 and has the highest rank, i.e., 0. Branch nodes' ranks are determined as follows: node 2, 9 and 10 are set to ranks 1, 2, and 3, respectively. Non-controller nodes' rank is set to the lowest rank in the network (for example, we temporalily set the rank 1000). Non-controller nodes are called stub nodes. The initial state of the tree structure in Fig. 9(b) is built to equalize the number of nodes connecting to each controller. Branch nodes periodically get control information from their nearest higher node, to prepare for emergency.



(a) Physical network and failed links

(b) Logical network (Initial State)

(c) Logical network (reconstructed after disruption)

(d) Logical network (reconstructed after recovery)

Fig. 9 C-plane reconstruction against network disruption and recovery.

5.2 Reconstruction against Network Disruption

We assume the situation when links 3-5, 3-6, and 4-6 are simultaneously broken. In that case, the network is separated into two components. The node groups 1-4, 10-12, and 5-9 are physically disconnected; it is desired to accommodate flows between two nodes in the same group, especially in the case of disasters. Therefore, the branch node 2 becomes a sub root node in the left component. The C-plane reconstruction is completed when the node 2 connects to the isolated node 3. Figure 9(c) shows the C-plane tree structure when the reconstruction has finished.

Figure 10 shows C-plane reconstruction sequence of the left component in Fig. 9(a). The lower node starts to seek another controller and attempts to connect to a higher node when the control path is disconnected. To detect path disconnection, each directly connected pair of root (or branch) and stub node in Fig. 9(b) sends polling packets to each other periodically. If a certain node does not receive any polling packets within the time limit, the node recognizes a discon-

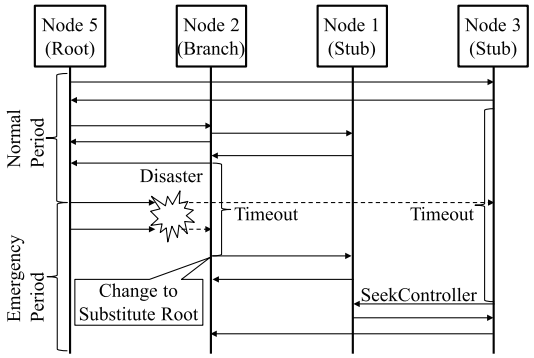
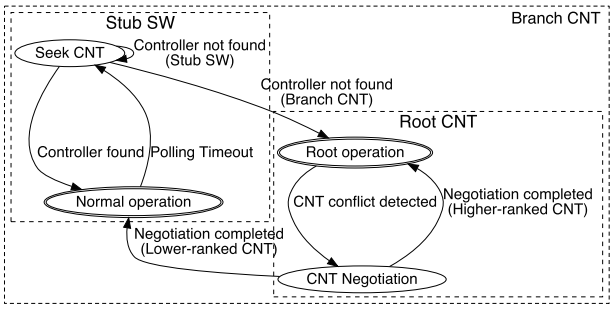
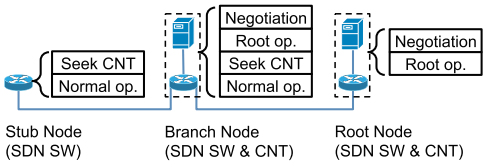


Fig. 10 Sequence of C-plane reconstruction when failure occurs.



(a) State transition diagram for C-plane Reconstruction.



(b) Node architecture.

Fig. 11 State transition diagram and node architecture for C-plane reconstruction.

nection and seeks a higher node by flooding queries. To avoid broadcast storm, each node avoids sending the same query back to the receiving port. The branch node becomes the sub root node if it does not receive the reply within a definite period. In some cases, two or more nodes having the same rank may exist in one cluster (node 2 and 10 in the left part). The node having the highest rank (node 2, in this example) plays the role of the substitute (sub) root node. Node 3 seeks a higher node because it does not communicate with node 5. Node 3 sends queries to its adjacent nodes, i.e., nodes 1, 2, and 4. The node can notice existence of node 2 via one of its adjacencies. Further, node 3 can join under the control of node 2. As described above, this reconstruction procedure is based on the bottom-up approach.

5.3 Reconstruction against Network Restoration

The network can manage flows within each separated component just after disasters because the branch node 2 compensates functionalities of the root node. For example, if link 4-6 is repaired, connectivity among all nodes is restored. At that time, duplicated root nodes exist in the same network simultaneously. That may cause control conflicts. To avoid the conflicts, the sub root node 2 should turn into the branch node and join under the control of the root node 5.

C-plane reconstruction sequence is executed when the link 4-6 is reconnected. First, border nodes 4 and 6 exchange information of their root nodes. Second, border nodes notify information about the opposite component's root node of their own root node, i.e., node 4 notifies node 5's information of node 2, and vice versa. After notification, nodes 2 and 5 negotiate and merge their C-plane. In this case, node 2 is controlled by node 5. The C-plane structure is built as illustrated in Fig. 9(d) when disappearance of the duplicated root nodes is preferred. Load of each controller is not homogeneous; however, the structure can be modified if any load distribution mechanisms are installed.

5.4 Node Behavior for C-Plane Reconstruction

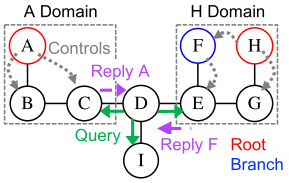
Figure 11(a) shows the general state transition diagram for C-plane reconstruction. The root node stays at *Root operation* and plays the role of an SDN controller. When the separated networks are merged, the root node transits *controller (CNT) negotiation* state. Stub nodes transit among *Normal operation* and *Seek CNT* state. Branch nodes go through all states in this figure according to different situations. If a branch node cannot find the root node in the isolated network, the branch transits to *Root operation*. When the branch rediscovers the root after recovery, it turns back to *Normal operation* via *Negotiation*. Figure 11(b) shows root, branch, and stub node architecture. Root and branch nodes consist of a pair of SDN controller and switch components. Meanwhile, stub node consists of SDN switch only. At each state in Fig. 11(a), nodes take actions and change state as follows:

Root Operation: Root and branch nodes can transit to this state. Branch nodes transit into this state when they become substitute root nodes after failures. If network recovery is detected or noticed, the node changes to *CNT Negotiation* state.

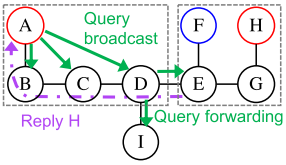
Normal operation: This is the normal state of branch nodes and stub nodes. When connection to the CNT is broken (detected by *Polling Timeout*), a node changes its state to *Seek CNT*. When a node is asked CNT information from neighbors, it replies with information of its CNT. If a node detects network recovery, it notifies CNT.

Seek CNT: Branch and stub nodes enter into this state when connections to their CNTs are down. In this state, the nodes seek a survived higher root or branch node. Branch nodes change into *Root operation* if a higher controller is not detected within a definite period of time. Besides, Stub nodes stay in this state until an upper controller is found.

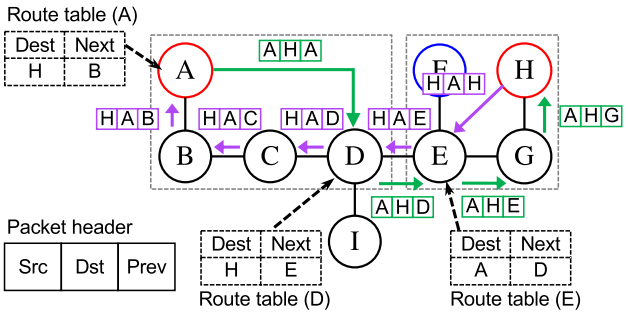
CNT Negotiation: The root node and substitute root nodes transit into this state when the separated networks are connected again. The root nodes attempt to establish connection to each other when the link recovers. Ac-



(a) Isolated node D discovers node A's domain. Node D sends queries to its adjacent nodes and it recognizes info of node A(F) from node C(E).



(b) Node A finds adjacent domain dominated by node H. Node A broadcasts query to its children. Node D forwards the query to its neighbors. Node E replies info of Node H.



(c) Negotiation packet is sent via hop-by-hop forwarding for routing. The packet includes the previous node's information. Intermediate node E recognizes a route to source of the negotiation packet.

Fig. 12 Illustration of other root finding mechanism and rerouting process for negotiation.

cording to rank comparison, the lower-ranked node's network joins the controlled group of the higher-rank node. The higher-ranked node transits to *Root operation* and the lower one turns into *Normal operation*.

C-plane reconstruction is performed by the following procedures. Figure 12 illustrates one of examples. In this figure, nodes A and H act as root (or sub root) node. Node F is a branch node and the others are stub nodes. At the initial state in Fig. 12(a), node A controls its peripheral nodes, i.e., B and C. Meanwhile, node H controls nodes F and G directly and node E indirectly. Node D floods queries to seek higher (root, sub root, or branch) nodes. Node D can get information of nodes A and F from its neighbor nodes C and E. In this example, we assume that node D knows existence of node A earlier than node F. Thus, node D joins under the control of node A as shown in Fig. 12(b).

If the root and sub root nodes are apart from each other, it is hard to recognize the existence of other root nodes by sending queries to their neighbors only. Nodes A and H should negotiate with each other for C-plane reconstruction because root and sub root nodes exist simultaneously in Fig. 12(b). However, these two nodes do not know the existence of other nodes if they send queries for seeking controllers to their neighbors (node B or G) only. This is because the neighbor node B(G) notifies information of its parent node A(H), similar to node 1 in Fig. 10. To recognize other distant root nodes, root and sub root nodes broadcast queries to their children. Additionally, children nodes forward the queries to their neighbors. In the case of Fig. 12(b), node A sends queries to nodes B, C, and D. Further, each child forwards queries to its neighbors. After that, node A's queries reach node E, controlled by the other root node H. Node E notifies its parent node F's information of node A.

When node A receives reply from node E, node A starts negotiating with node F. Node F is not the root node of node H group. However, node A tentatively attempts to negotiate with node F because node A only recognizes the existence of branch node F. After that, node F forwards a negotiation message to node H. Therefore, node H recognizes the existence of the other root node, i.e., A. Route tables should be updated in case the past route is not available because of failures. As one of solutions, we implemented a simple rerouting mechanism like AODV, as shown in Fig. 12(c). For rerouting, each packet includes the identifier of previous node in the forwarding path. Each node rewrites its route table when it receives reply packets of query for seeking controllers or packets to negotiate with the other controllers. In this case, node H can communicate with node A by using reverse path of the negotiation packet from node A. This reactive approach takes a long time to reroute. However, this approach adapts to many patterns of failures.

6. Validation of C-Plane Reconstruction

We investigate performances of our reconstruction protocol by using ns-3.22. Bitrate of all links is 1 Gbps in physical networks, which is enough to exchange control packets, and therefore packet dropping induced by buffer overflow does not occur. Initially, all non-root nodes are connected to the nearest root or branch node. To evaluate our reactive method, we did not use proactive recovery method in these simulations, i.e., in short, $(k, n) = (1, 1)$.

At the time $t = 0$ [s], regional failure as described in Sect. 4.2 occurs. Polling interval between each pair of nodes is 20 s. Nodes detect disconnections of control path if they do not receive polling packets within 30 s. Stub nodes repeat sending queries to seek higher nodes at every $T(= 45)$ s until they find the parent nodes. Branch nodes change their roles to sub root if they do not find root or sub root nodes within 45 s. The root or sub root nodes broadcast queries (like Fig. 12(b)) at every $T(= 45)$ s for searching other root nodes.

At the time $t = 600$, the first node recovery occurs. After that, randomly selected failed nodes are recovered until all failed nodes are recovered. Node recovery event occurs at random time and repeats until all nodes recover. When a node detects restoration of its neighbor, it first notifies the information of its parent. The notification is relayed to the root node of the component. Then, the root node starts the negotiation process. By repeating the negotiation and merging process, C-plane of all nodes is completely reconstructed after network recovery.

Figure 13 shows the simulation results in Gabriel graph. In this subsection, we created 10 patterns of N-node topologies ($N = 50, 100, 200$). The controller nodes are randomly placed. We examined with 10 patterns of node failures on each topology. Specifically, we evaluated the results of 100

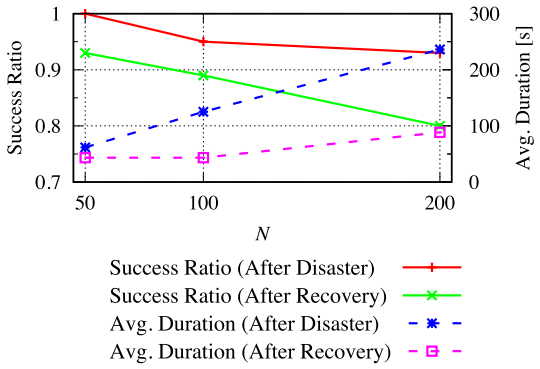


Fig. 13 Simulation results of C-plane reconstruction.

trials with every N value. Approximately 25% of nodes failed in each trial. In this figure, success or failure is determined by whether the number of controllable nodes of reconstructed C-plane reaches the ideal value within 600 s (after disaster) or within 300 s from when the last node recovers (after recovery). Duration after disaster is defined as the time from when regional failure occurs to when reconstruction finishes in success cases. Moreover, duration after recovery is defined as the time from when the last node recovers to when reconstruction finishes in success cases. In each graph, success ratio ranges from 0.88 to 0.98 when $N = 100$. The C-plane logical connectivity is recovered with probability of approximately 90%.

Average duration after disaster is proportional to the number of nodes N and query interval T (approximately $1.25T$, $2.5T$, and $5T$ in the case $N = 50$, 100, and 200 respectively). This means that the duration time is controllable by tuning the query interval T according to requirement of use cases. This result is interpreted as follows. Reconstruction finishes if (a) switches reconnect to their nearest controllers, and at the same time, (b) the root and all branch controllers reconnect each other. The time to find and reconnect to parent controllers is proportional to the distance between two nodes. In the simulations, the number of controllers is proportional to number of nodes. Therefore, convergence time of (a) does not change because the distance from switches to their nearest controllers does not stretch. Meanwhile, convergence time of (b) changes because each distance between the only root controller and branch controllers is proportional to the network diameter (the longest of all the shortest paths in a network). Network diameter is $N - 1 (\in O(N))$ in the worst case (in linear networks). That is, the convergence time of our method is proportional to the number of nodes. However, our method does not reach the ideal value, especially in large-scale networks ($N = 200$). This is because our routing method causes control packet loop when root and branch nodes concentrate in a narrow area.

7. Conclusion

Distributed SDN control improves network scalability and robustness. In this study, we have first proposed network partitioning method based on RTC to establish robust C-plane connectivity against large-scale failures. RTC reflects the risk of separation between controller and nodes in Sub-NWs. Simulation results show that partitioning with RTC improves robustness of connectivity between switch nodes and especially local (branch) controllers. To recover from the damages that proactive failover mechanism cannot adapt to, we have also proposed the reactive C-plane reconstruction procedure against network disruption and restoration. We have already implemented our reconstruction mechanism in the existing OpenFlow framework (not described in this paper due to page limitation). Further, convergence time of our reconstruction mechanism is proportional to network size in Gabriel graph. C-plane connectivity is not fully recovered by only our approach. However, we have confidence that a hybrid approach between proactive method and our reactive method will reconstruct C-plane completely. In future, we consider establishing the hybrid approach.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," ACM SIGCOMM Computer Communication Review, vol.38, no.2, pp.69–74, April 2008.
[2] S.S.W. Lee, K.-Y. Li, K.-Y. Chan, G.-H. Lai, and Y.-C. Chung, "Path layout planning and software based fast failure detection in survivable OpenFlow networks," Proc. International Conference on the Design of Reliable Communication Networks (DRCN), pp.1–8, April 2014.
[3] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Fast failure recovery for in-band OpenFlow networks," Proc. International Conference on the Design of Reliable Communication Networks (DRCN), April 2013.
[4] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: A distributed control platform for large-scale production networks," Proc. USENIX Symposium on Operating Systems Design and Implementation (OSDI), Oct. 2010.
[5] S.H. Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," Proc. first ACM SIGCOMM workshop on Hot topics in software defined networking (HotSDN), pp.19–24, Aug. 2012.
[6] T. Hirayama, T. Miyazawa, H. Furukawa, and H. Harai, "Reconstruction of control plane of distributed SDN against large-scale disruption and restoration," Proc. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp.253–258, May 2017.
[7] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," Proc. internet network management conference on Research on enterprise networking, April 2010.
[8] J. Hu, C. Lin, X. Li, and J. Huang, "Scalability of control planes for software defined networks: Modeling and evaluation," IEEE International Symposium of Quality of Service (IWQoS), pp.147–152, May 2014.
[9] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," Proc. First ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN), pp.67–72, Aug. 2012.
[10] Y. Hu, W. Wendong, X. Gong, X. Que, and S. Shiduan, "On reliability-optimized controller placement for software-defined networks," China Commun., vol.11, no.2, pp.38–54, Feb. 2014.
[11] Y. Jimnez, C. Cervell-Pastor, and A.J. Gara, "On the controller placement for designing a distributed SDN control layer," Proc. IFIP Networking Conference, pp.1–9, June 2014.
[12] L.F. Mller, R.R. Oliveira, M.C. Luizelli, L.P. Gaspary, and M.P. Barcellos, "Survivor: An enhanced controller placement strategy

for improving SDN survivability," Proc. Globecom, pp.1909–1915, Dec. 2014.
[13] S.S. Savas, M. Tornatore, M.F. Habib, P. Chowdhury, and B Mukherjee, "Disaster-resilient control plane design and mapping in software-defined networks," Proc. IEEE International Conference on High Performance Switching and Routing (HPSR), July 2015.
[14] S.A. Astanch and S.S. Heydari, "Optimization of SDN flow operations in multi-failure restoration scenarios," IEEE Trans. Netw. Serv. Manag., vol.13, no.3, pp.421–432, June 2016.
[15] Reported by Ministry of Internal Affairs and Communications (in Japanese). (2011, Dec.) [Online]. http://www.soumu.go.jp/menu_news/s-news/01kiban02_02000043.html
[16] E.K. Çetinkaya, M.J.F. Alenazi, Y. Cheng, A.M. Peck, and J.P.G. Sterbenz, "On the fitness of geographic graph generators for modelling physical level topologies," 5th International Workshop on Reliable Networks Design and Modeling (RNDM 2013), pp.38–45, Sept. 2013.
[17] M. Darianian, C. Williamson, and I. Haque, "Experimental evaluation of two OpenFlow controllers," Proc. IEEE International Conference on Network Protocols (ICNP), Oct. 2017.
[18] V.D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," J. Statistical Mechanics: Theory and Experiment, vol.2008, Oct. 2008.
[19] J.Y. Yen, "An algorithm for finding shortest routes from all source nodes to a given destination in general networks," Quart. Appl. Math., vol.27, pp.526–530, 1970.

Appendix A: Validity Verification of RTC

In case of Sub-NW expansion/reduction, if the change of RTC is steady, it is difficult to construct a Sub-NW using RTC as the index. Therefore, we prove that the first order difference of RTC (ΔRTC) changes in case of Sub-NW extension (i.e. addition of the new node to Sub-NW).

In the following example, consider the case where a node is added to the Sub-NW. When the boundary links of the Sub-NW increase by addition of one node (ex. addition of one node with plural links), its increment (ΔRTC) is expected to be a positive number from the definition of RTC. Consequently, we consider the case where boundary links do not increase according to addition of one node. Since the number of links in NW is limited, the extension of the Sub-NW without increasing boundary link surely occurs. Therefore, it is enough that the sign change of RTC is shown in this case.

Figure A·1 shows the extension of Sub-NW without increasing boundary link. Let $S(k)$ be a Sub-NW that has $|V_{S(k))}|$ nodes, and let l be the new boundary link in Sub-NW $S(k+1)$ that is extended by adding one node to $S(k)$. Here, we assume that i boundary links with which the path length of l changed, exist in $S(k+1)$.

- l_i Boundary links with which the path length of l changed
- $R_{(-i)}$ Total rlc of the boundary links except l_i
- $\widehat{E}_i$ The SEV of l_i before the path length from l changes
- $\widehat{E}_i{}'$ The SEV of l_i after the path length from l changes

In the following, let $T(k)$ be RTC of $S(k)$, and evaluation of the first order difference is done by adding one node.

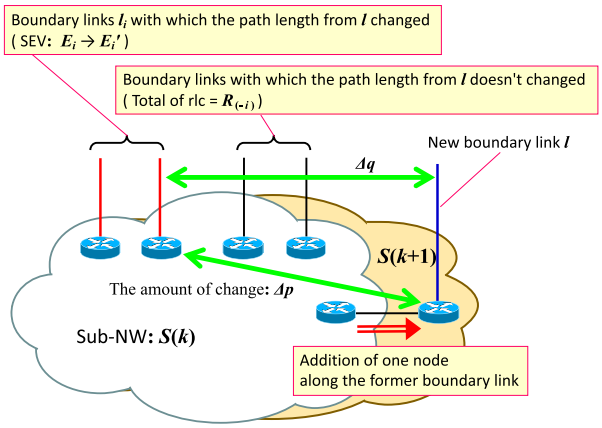


Fig. A·1 The relationship between boundary link pair in Sub-NW extension without increasing boundary link.

$$T(k) = \frac{1}{|V_{S(k)}|}\left[R_{(-i)} + \sum\left\{\frac{-2}{\widehat{E}_i + 2} + \frac{3}{2}\right\}\right] \quad (A\cdot 1)$$

$$\begin{aligned}\{T(k) - T(k+1)\}|V_{S(k)}||V_{S(k+1)}| &= |V_{S(k)}|\sum\left\{\frac{2}{\widehat{E}_i{}' + 2} - \frac{2}{\widehat{E}_i + 2}\right\} \\ &+ R_{(-i)} + \sum\left\{\frac{-2}{\widehat{E}_i + 2} + \frac{3}{2}\right\}\end{aligned} \quad (A\cdot 2)$$

Here, we consider the sign of $\Delta_i = 2\left\{\frac{1}{\widehat{E}_i{}' + 2} - \frac{1}{\widehat{E}_i + 2}\right\}$, which is the coefficient of $|V_{S(k)}|$. Let $h_{in} = p$ and $h_{out} = q$, where p and q is path length of the inside and outside GW between l and l_i, and let Δp and Δq be the amount of change by adding one node, respectively.

$$\Delta_i(\widehat{E}_i{}' + 2)(\widehat{E}_i + 2) = 2(\widehat{E}_i - \widehat{E}_i{}') \quad (A\cdot 3)$$

$$\widehat{E}_i - \widehat{E}_i{}' = -\left\{\frac{\Delta p}{(p+1)(p+\Delta p+1)} + \frac{\Delta q}{(q+1)(q+\Delta q+1)}\right\} \quad (A\cdot 4)$$

Since the number of additional nodes is one and the increase in path length between the inside GW is only at most one hop, $\Delta p \leq 1$ holds. Moreover, $\Delta q \geq -q$ holds from definition of path length. Consequently, in case of $\Delta p = 0, 1$ and $\Delta q > 0$, $|V_{S(k)}|\sum\Delta_i < 0$ holds from $\widehat{E}_i - \widehat{E}_i{}' < 0$. Therefore, since $T(k) - T(k+1)|V_{S(k)}||V_{S(k+1)}|$ is monotonically decreasing, the sign of the first order difference of RTC (ΔRTC) changes according to the increase in $|V_S|$.

From the above, in case of Sub-NW extension, since RTC increases or decreases according to change in the number of nodes or boundary links, it is possible to construct a Sub-NW on condition of the RTC value.

Appendix B: Unbiasedness with Regard to Initial Value

In the case where the Sub-NW is extended one by one using

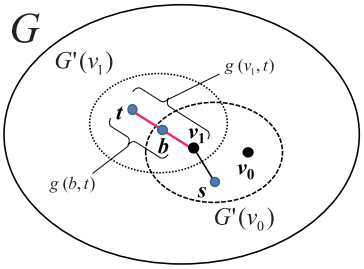


Fig. A · 2 The relationship between G' and g.

RTC as the index, if the topology of the Sub-NW is influenced by the starting point of the construction, the influence of large-scale failure cannot be minimized (i.e. a scope for optimization still exists). Therefore, in this section, we prove that the same Sub-NW set is created, even if the starting point of Sub-NW construction is any node in the NW. First, function F and subgraph G' of graph are defined as the following.

F	The Boolean function which takes node connection of G as argument
$G'_{(v)}$	The node connection including node v, where the output from $F(G'_{(v)})$ is the same value.
$\overline{V(G')}$	The node set of G' except the boundary node (i.e. $v \in \overline{V(G')} \Rightarrow \forall$adjacencies of $v \in G'$).
$\mathbb{G}$	$\{ G'(v) \mid v \in G \}$.

Proposition:

The same $\mathbb{G}$ is constructed, even if the starting point of calculation for F is any node v in G, where $v \in \overline{V(G')}$.

Proof:

Let $G'(v_0)$ be the subgraph given by calculation of F which is started from v_0 in G. Similarly, let $G'(v_1)$ be the subgraph given from v_1 in $G'(v_0)$, where $v_1 \in \overline{V(G'(v_0))}$.

Here, we assume that the following nodes s and t exist.

- $s \notin G'(v_1),\ s \in G'(v_0)$
- $t \in G'(v_1),\ t \notin G'(v_0)$

Next, let $g(v_1, t)$ be the graph which connects v_1 and t, and let b be the boundary node of $G'(v_0)$ on $g(v_1, t)$ (Fig. A· 2). Since t is not an element of $G'(v_0)$, $F(g(v_1, b)) \neq F(g(b, t))$ holds from the definition of G'. On the other hand, since v_1, b, and t are elements of $G'(v_1)$, $F(g(v_1, b)) = F(g(b, t))$ holds. Both are contradictory.

Therefore, such a node t does not exist. Node s also does not exist as per the same argument. Consequently, since the same set of subgraphs (i.e. $\mathbb{G}$) is obtained regardless of a starting point of calculation for F, the proposition is proved.

From this proposition, by defining F as a function showing whether or not the sign of first order difference of RTC (ΔRTC) has changed, even if the starting point of Sub-NW construction using RTC as the index is any node in NW, the same Sub-NW set is created.

Takahiro Hirayama received a M.S. Degree and Ph.D. Degree from Osaka University in 2010, 2013, respectively. In April 2013, he joined National Institute of Information and Communications Technology (NICT) and is now a researcher of Network System Institute in NICT. His research interests are in complex networks, optical networks, and software defined networking (SDN). He is a member of IEICE.

Masahiro Jibiki received the Ph.D. degree in systems management from the University of Tsukuba, Japan, in 2003. In 1992, he joined NEC Corporation and was a researcher in the Central Research Laboratories until 2011. From 2006 to 2009, he was also a visiting professor at the University of Wakayama, Japan. Currently, he is an expert researcher in the National Institute of Information and Communications Technology. His research interests include networking, software science, and mathematical model.

Hiroaki Harai received the M.E. and Ph.D. degrees in information and computer sciences from Osaka University, Osaka, Japan, in 1995 and 1998, respectively. He is currently a Director at National Institute of Information and Communications Technology, Tokyo, Japan. His research interests are Future Networks and Optical Networks. He is a member of IEEE and IEICE.