Received July 8, 2020, accepted July 22, 2020, date of publication August 3, 2020, date of current version August 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3013323

On Designing a Resilient SDN C/M-Plane for Multi-Controller Failure in Disaster Situations

LUIS GUILLEN[1], (Member, IEEE), HIROYUKI TAKAHIRA[2], SATORU IZUMI[2], TORU ABE[2,3], AND TAKUO SUGANUMA[2,3], (Member, IEEE)

[1]Research Institute of Electrical Communication, Tohoku University, Sendai 980-8577, Japan
[2]Graduate School of Information Sciences, Tohoku University, Sendai 980-8577, Japan
[3]Cyberscience Center, Tohoku University, Sendai 980-8577, Japan

Corresponding author: Luis Guillen (lguillen@ci.cc.tohoku.ac.jp)

ABSTRACT Network survivability is the ability to maintain service continuity in the presence of failures. This ability might be critical in times where large-scale failures occur, as in the case of disasters. In the past years, Software Defined Networking (SDN) has shown a great potential to allow network programmability by segregating the Control/Management Plane (C/M-Plane) from the forwarding or Data Plane (D-Plane). The controller, a centralized entity, has an overview of the entire network under its domain, which allows it to make informed routing decisions. However, the controller becomes a single-point-of-failure as network devices will have limited knowledge if not connected to a controller. Moreover, in disaster situations, if the affected area is considerably large, there is a high probability that more than a single controller will fail in a short period. Various studies, either following a protection or restoration techniques, have been proposed to address resiliency on SDN, but most of them only consider link or device failure; however, the failure of various controllers due to a large-scale disaster is less explored. In this paper, we consider multi-controller failure and propose a mechanism to reduce the non-operational network devices in disaster situations. Preliminary results show that, by applying the proposed approach, it is possible to achieve substantial improvements in network survivability, with considerably less cost of implementation than existing methods. In particular, using simulation, we achieved a 20% decrease of non-operational devices at the C/M-Plane; and an increase of 30% of success rate at the D-Plane, even if half of the controllers in the topology failed.

INDEX TERMS Disaster resilient networks, multi-controller failure, network management, SDN.

I. INTRODUCTION

Software Defined Networking (SDN), which segregates the Control/Management Plane (C/M-Plane) from the forwarding or Data Plane (D-Plane) [1], has proven to be useful in creating innovative and flexible solutions for managing network resources. However, there are two major issues with SDN-based solutions, the first of which being scalability [2], and the second one being reliability of both the C/M- and D-Plane. In terms of reliability at the C/M-Plane, the SDN controller becomes a single-point-failure [3]. Although there have been significant advances in tackling scalability, such as the creation of distributed topologies with support to multiple controllers, there are still some open issues with the proper handling of Controller reliability [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Mamoun Alazab.

Soon after the release of OpenFlow (OF) [5], which was the enabler of SDN, the community started to raise their concerns about network failure recovery. However, initial designs were implemented for single-link failures at the D-Plane; wherein once the connection between two network nodes is interrupted, the end-to-end path is recalculated so that the transmission can resume; this process is called *fast restoration* [6], which is the most straightforward mechanism to recover from a failure in SDN. Of course, fast restoration assumes that the network device is operational, and for it to be connected to a controller that will compute an alternative path. However, when the device is operational but not connected to the controller, then the SDN-enabled network device, by default, does not have enough knowledge to recover from path failure. Thus, network devices must be connected to a controller so that they can benefit from programmability. As a consequence, when the controller fails, the options are not as straightforward.

Therefore, this paper addresses the cases when one or more controllers fail, and present a three-step mechanism to protect the C/M-plane in SDN. Through simulation, we show that, by applying the proposed approach, it is possible to reduce the controller-device disruption and enable service continuity. Preliminary results show an increase in the transmission success rate at the D-plane, even in the presence of large-scale failures.

In the remainder of this Section, we present a brief background and overview of network survivability on SDN. Then in Section II, we present the related work and formulate the target issues. In Section III, which is the main contribution of this paper, we design a mechanism to maximize service continuity in scenarios where multiple controllers fail in SDN. The proposal was evaluated by simulation, whose results are presented in Section IV, and we discuss its implications/limitations in Section V. Finally, in Section VI, we conclude this paper and devise some future directions for this study.

A. SDN CONTROLLER CONNECTION AND RECOVERY MECHANISMS

The control line, in the context of SDN, connects the controller to the underlying network devices and can be of two types: *out-of-band* and *in-band*. As depicted in Fig. 1a, in an out-of-band connection, the traffic is sent via a dedicated line connected through a management port in the device. On the other hand, as shown in Fig. 1b, in an in-band connection, the traffic is sent through a shared line with the data traffic; therefore, the connection also uses intermediate network devices to reach the controller.

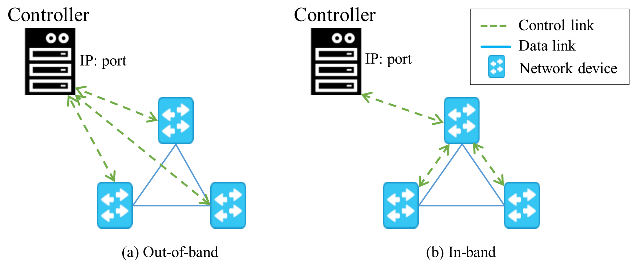


FIGURE 1. Out-of-band and In-band C/M-Plane connection.

Table 1 shows a brief description of these two mechanisms and their characteristics. As observed, in terms of scalability, since out-of-band connections require a module in the device's bare-metal connected to the SDN controller, the maintenance cost is high if the topology requires redundant links. By contrast, since in-band connections use the D-Plane infrastructure, it has high scalability at no additional cost, except for the additional traffic. Moreover, if one of those links fails, it is not easy to restore out-of-band lines; by contrast, in the case of in-band connections, it can be reconfigured to traverse different paths in the available infrastructure. However, in-band connections have some drawbacks regarding the availability and the security aspects, since those will depend on the way they are configured. Therefore, it is vital to find the appropriate type of connection, depending on the application requirements.

TABLE 1. Comparison between Out-of-band and In-band SDN connections.

Feature	Out-of-band	In-band
Scalability	Limited	High
Cost	Expensive	No additional cost
Congestion overhead	Network size dependent	Configuration dependent
Resiliency to failure	Restricted	Configuration dependent
Bootstrapping mechanism	Immediate	Configuration dependent
Security	Less vulnerable	Vulnerable

Regardless of the nature of the connection (i.e., out-of-band or in-band), when a controller fails, SDN-enabled devices will contain the necessary information on how to attempt reconnection. This process will depend on how the protocol is implemented; for instance, OF specification [7] states that the connection recovery process works as follows:

- Contact backup controllers: At this step, the network device will try to contact a backup controller (or controllers) based on a pre-planned configuration. However, the protocol itself does not provide the mechanisms to perform the hand-off between controllers; this process should be handled by the network devices themselves, which has proven to be a complex task [3].
- Enter Fail mode: If the device cannot reconnect to any backup controller, then the device will enter into either *Fail secure mode* or *Fail standalone mode*. In the first case, packets and messages destined to the failed controller will be dropped, and the D-Plane rules installed on the device will continue to apply until they expire as set up in their configuration. In *standalone mode*, the device will act as a legacy device, in which case it loses all the SDN programmability; moreover, this mode is only enabled in devices that support hybrid mode (i.e., OF and Legacy).

Additionally, it is also possible to use *Auxiliary connections*, which are created to improve the device processing performance by using parallel connections to the controller. However, if the main control line fails, then the auxiliary connections also fail.

B. PROTECTION AND RESTORATION MECHANISMS FOR NETWORK SURVIVABILITY

Network survivability, which refers to the capability of a network to maintain service continuity in the presence of failures [8], is an important feature for Quality of Service (QoS). However, despite the significant advances in terms of network technologies and techniques, it is still a challenging task. There are different mechanisms to ensure survivability based on the technologies used and in which layers they are applied. However, there is an additional complication in SDN as it needs to maintain the survivability of both the C/M- and D-Plane [9]. In the D-Plane, traditional mechanisms can handle, with relative ease, flexible solutions [10], [11];

However, most of them assume a constant C/M-plane connection. These mechanisms fall into one, or a combination, of the following categories:

- Restoration Mechanisms: This type of approach is, for the most part, *reactive*. The recovery mechanisms comprised into this category (e.g., path re-routing) can be either pre-planned or dynamically calculated after the failure occurs; therefore, additional time and computation are needed for the implementation.
- Protection Mechanisms: This type of approach is *proactive*; the recovery mechanisms under this category are always pre-planned. Therefore, the contingency steps can be applied immediately after the failure is detected. However, this will presuppose additionally reserved space in the device memory that might never be used.

Fig. 2 describes the time-line of failure recovery in protection and restoration mechanisms; as observed, the recovery process can be structured as follows: In the first step, *detection*, the mechanism should provide the means to detect a failure, which can take a considerable amount of time. Next, in the *restoration* step, a set of actions are calculated (in reactive approaches) or selected (in proactive approaches). Finally, in the *restitution* step, one of the available solutions must be implemented to devices so that the network can continue its operation. The service outage time will vary depending on the effectiveness of these steps.

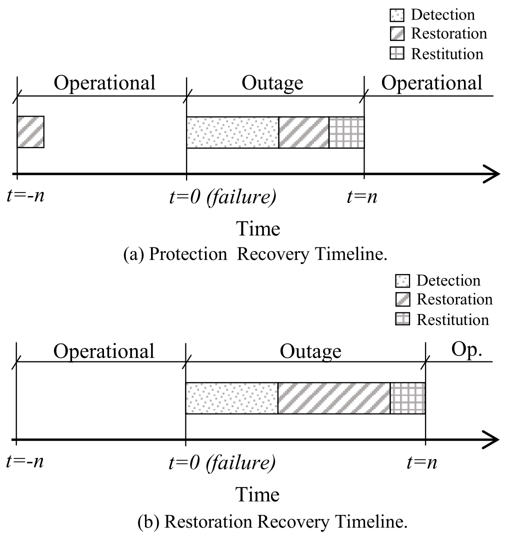


FIGURE 2. Recovery process timeline in protection and restoration mechanisms.

However, in case of a large-scale failure (e.g., a disaster), a few seconds can make a big difference. As a reference, the Great East Japanese Earthquake in 2011 showed unprecedented losses not only in infrastructure but also in human lives. In terms of infrastructure, there were tens of thousands of base stations suspended/damaged, millions of telecommunication lines interrupted, and tens of exchange buildings unusable [12]–[14], which were restored several months after the disaster. Thus, it is crucial to take into account these events when building a comprehensive solution for service resilience in SDN.

Therefore, being the connection to the controller vital for adequately provide service resilience and continuity in SDN-based applications, this paper focuses on how to ensure reliable communication at the C/M-Plane so that the service at the D-Plane can continue in as much as possible, despite the failure of one or multiple controllers. The main hypothesis is that by ensuring the controller connection, the overall network survivability can improve.

II. RELATED WORK AND TARGET ISSUES

A. RELATED WORK ON SDN RESILIENCY

To the best of our knowledge, multi-controller failure in SDN has not been addressed yet; therefore, in this section, we present a summary of representative studies on SDN resiliency.

Sharma *et al.* [11] is considered a seminal work on failure recovery for the C/M- and D-Plane. To achieve the highly demanding carrier-grade network quality [15], [16], the authors presented a fast failure recovery of both restoration and protection mechanisms; however, only protection mechanisms (proactive), for both the control and data traffic, can achieve those requirements. Following the same re-routing approach, Hu *et al.* [17] presented a mechanism that is handled directly by network devices called: *local re-routing* and *constrained reverse forwarding* where the device searches for the closest controller based on the defined number of hops to reach the controller. Although these studies shed light on developing the field, they only take into account single link failures, and controller failure was not discussed.

On the other hand, other authors proposed hybrid approaches that embed additional modules for resilience to SDN-enabled devices. For instance, Omizo *et al.* presented *ResilientFlow* [18] that uses protection and restoration for multi-link failures; additionally, a self-healing mechanism for devices that have lost connection to the controller in an OSPF-like module embedded in each device. Osman *et al.* presented *Hybrid SDN* [19], which also embeds additional modules in each device, by adding a distributed light controller triggered by the loss-percentage on each link. These approaches have various advantages, but the convergence time and control traffic overhead are significant. Additionally, deploying modifications on devices at a large scale would be challenging.

Asadujjaman *et al.* [20], proposed an innovative mechanism entitled Fast Control Channel Recovery for Resilient In-band OF Networks; their approach consists of protecting the Switch-to-controller (S2C) and controller-to-switch (C2S) communication. To do so, they use a logical ring topology with source-routed forwarding. Görkemli, *et al.* [21] presented a Dynamic Control Plane for SDN at Scale that includes on-the-fly dynamic load balancing of control traffic using a hybrid out-of-band and in-band connection. These proposals are visionary, but still challenging to implement and deploy, as they will require functions that are not widely available in current controllers or devices.

It is also worth mentioning that there are authors, as in [22]–[28], that propose resiliency by design. These approaches imply, for instance, strategically position the controller within the network so that the fail-over mechanisms are more robust to failures. However, in the case of disasters, it is challenging to foresee all possible failure cases, rendering these solutions impractical. For instance, Astaneh and Heydari [25] presented an analytical approach to minimize the number of operations in multi-link failure scenarios, but only for a reduced number of links. Hirayama *et al.* [26] presented a Distributed SDN C-Plane for Large-Scale Disruption and Restoration mechanism by partitioning the network devices into groups managed by different controllers so that in case of a failure, the control plane is merged to the surviving controller. Although the results were promising, the failure detection mechanism takes a long time to converge, causing a long delay in service restitution. Moazzeni *et al.* [27] presented a detection and recovery mechanism for clustered controllers in distributed environments, which take over control of *orphan* devices when these fail. However, their selection of the coordinator controller might not be effective when various controllers fail in a short period.

Finally, Xie *et al.* [28] proposed an approach that uses Multiple Routing Configurations (MRC), which prepares multiple backup topologies and selects the proper backup topology based on the current network failure state. However, even if the idea is interesting, calculating all possible outcomes will not only constitute a high computation cost, but these calculations also will not be practical if the network topology changes in a fast rate or if the disrupted area is of considerable size.

Table 2 shows a comparison summary of the related work compared to this study.

TABLE 2. Comparison of related work features.

Work	Scope†	Type‡	Impl.◇	Conn.⋆	Scale○	Failure Coverage§		
						Link	Device	Controller
[11]	C,D	P,R	C	I		√		
[17]	D	R	D	I		√		
[18]	C	R	C,D	I,O	√	√		
[19]	C	R	C,D	I,O		√		
[20]	C	P	D	I		√	√	
[21]	C	R	C	I	√	√		
[23]	D	P	C	I		√		
[22]	C	R	C	O		√		√
[24]	D	P	C	O	√	√	√	
[25]	D	R	C	I	√	√	√	
[26]	C	R	C,D	I,O	√	√	√	√
[28]	D	R	D	I,O		√	√	
[27]	C	R	C	I,O	√			√
Our	C,D	P	C	I,O	√	√	√	√

† Scope of application of the approach: C=C/M-Plane, D=D-Plane.
‡ Type of Recovery approach: P=Proactive, R=Reactive.
◇ Scope of Implementation: C=Controller, D=Device.
⋆ Type of connection to the controller: I=In-band, O=Out-of-band.
○ Check mark if the approach tackles large-scale failures.
§ Check mark if the approach covers link, device, and controller respectively.

B. TARGET ISSUES

From the related work presented in the preceding section, the target issues are summarized as follows:

- *(P1) Traditional approaches are not effective in recovering from failures*: Restoration mechanisms (which are mostly reactive) are more useful to maximize connectivity, but the processing and implementation cost is high. On the other hand, proactive protection mechanisms are adequate to avoid disconnection, but it takes unnecessary memory of non-involved devices.
- *(P2) Large-scale failures are still not adequately considered*: When several links and network devices fail at the same time, or when various controllers fail in a short period, current approaches cannot handle service continuity.

This study complements and extends the existing work by considering the above issues, and proposes a controller connection resilience mechanism to allow service continuity in case of disasters.

III. PROPOSED RESILIENT SDN C/M-PLANE COMMUNICATION

A. PRELIMINARY CONSIDERATIONS

Given that current approaches, which follow traditional protection and restoration mechanisms, might not be able to handle large-scale failure of networking elements in disaster situations, the proposed approach presents a pragmatic solution based on the following considerations:

- Prevent before suffering: Early warning systems might greatly benefit from preparing for failure. In normal conditions, it is difficult to predict a network failure; however, there have been significant improvements in early alert warnings in case of disasters. For instance, the impact of an earthquake can be predicted within few seconds after the first wave [31], while in other types of disasters can be done long in advance (e.g., minutes for tsunami or hours for hurricanes).
- Use the available connections: Take advantage of the available methods to connect a network device to a controller, namely out-of-band and in-band, and use them according to the availability. Since network topologies in SDN are constrained to one or few out-of-band lines per device, prioritize their usage first and then use in-band connections.
- Find the trade-off between restoration and protection mechanism: While restoration mechanisms may take a long time to calculate a solution when the failure is detected, protection mechanisms take more space on the devices' memory.

B. NETWORK MODEL

The network model consists of an acyclic weighted graph $G(V, E)$ where V is the set of nodes $\{v_i \mid i \in \mathbb{N} \land i > 0\}$ that represents the SDN-enabled network devices (i.e., switches and routers) with a known position $(x_i,\ y_i)$, and E is the set of edges $\{e_{i,j} = e_{j,i} \mid i, j \in \mathbb{N} \land i, j > 0 \land i \neq j\}$ between two nodes $v_i,\ v_j \in V$ which represents the physical links at the D-Plane. Moreover, each edge $e_{i,j} = e_{j,i} \in E$ has an associated bandwidth $b_{i,j} \geq 0$ and a cost $\varphi_{i,j} \geq 0$. In the case of $b_{i,j}$

each edge will have a maximum bandwidth defined by the capacity of the physical link. However, $\varphi_{i,j}$ can adopt various values according to the property used in the calculation, for instance, it may be related to the remaining bandwidth, the delay, or the failure risk.

From the set of nodes V, two nodes s and $t \in V$ represent the source and destination respectively. A path $p_{s,t}$ is the shortest path from s to t, and is represented by a set of edges $e_{i,j}$ such that the initial and last edges are $e_{s,x}$ and $e_{y,t}$ respectively. For instance, in the simple scheme shown in Fig. 3, the path $p_{6,3}$, represented by the black dotted double-headed arrow line in Fig. 3, is defined as $p_{6,3} = \{e_{6,7}, e_{7,3}\}$. The cost of a path $PC_{s,t}$ is defined, as shown in (1), by the summation of the cost of all the edges comprised in $p_{s,t}$.

$$PC_{s,t} = \sum_{e_{i,j} \in p_{s,t}} \varphi_{i,j} \tag{1}$$

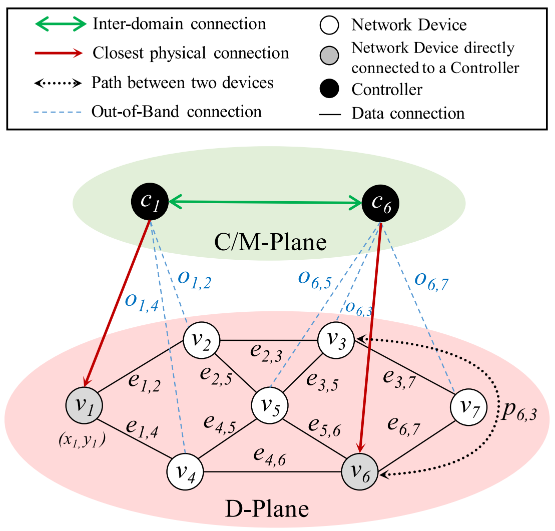


FIGURE 3. C/M- and D-Plane network model sample.

At the C/M-plane, a graph $G'(C, D_m, Z_m)$ where C is the set of SDN-controllers with nodes $\{c_m \,|\, m \in \mathbb{N} \wedge m = i \iff v_i \text{ encompasses a controller}\}$, so that if the node $v_1 \in V$ is directly/physically connected to a controller then $\exists\ c_1$ (i.e., hosted on-premises physical servers in the closest data center where the controller is located), this relationship is represented by the red arrow in Fig. 3. For instance, in the small sample, the set C is comprised by c_1 and c_6. Moreover, the set D_m which represents the Domain of c_m is comprised by all the nodes $v_i \in V$ that are controlled by the SDN-controller c_k. For instance, the domain D_1 of the controller c_1 in Fig. 3 is defined as $D_1 = \{v_1, v_2, v_4\}$.

In the case of the connection from the controller to network devices (C2D), which is equivalent to the network device to controller (D2C) connection, the super set Z represents the sets of out-of-band (O_m) and in-band (I_m) connections for each controller $c_m \in C$ to each node in its domain D_m. Each element $z_{c_m,i} = z_{i,c_m} \in Z_m$ is a defined as $z_{c_m,i} = O_{c_m,i} \cup I_{c_m,i}$. Note that this study assumes that there is only one out-of-band controller connection $|O_{c_m,i}| = 1$ and is represented by the blue dotted lines in Fig. 3; however, there might be a number (k) of in-band connections $|I_{c_m,i}| = k$ which is, in principle, the k different paths with the minimum cost from c_m to $v_i \in D_m$, as shown in (2).

$$I_{c_m,i} = \{\bigcup_{n=1}^{k} p_{c_m,i} \,|\, \min(PC_{c_m,i}) \wedge (p_{c_m,i})_n \neq (p_{c_m,i})_{n+1}\} \tag{2}$$

For the inter-domain communication, represented by the green double-headed arrow line in Fig. 3, the connection is the shortest path $p_{i,j}$ from $v_i(c_i)$ to $v_j(c_j)$ such that $PC_{i,j}$ is minimized.

C. PROBLEM FORMULATION

Assume that F is a set of failed controllers ($F \subseteq C$), such that F is not empty ($F \neq \emptyset$) due to an unexpected disaster within an affected area A. Consequently the network devices in the domain D_i of $c_i \in F$ lose out-of-band (O_i) and in-band (I_i) connectivity to c_i at the C/M-Plane. Moreover, at the D-Plane the network devices $v_i \in V$ and the associated links $e_{i,j}$ within the affected area A will also fail, rendering the data-flows ($P_{s,t}$) traversing paths $p_{i,j}$ through those devices unusable. Let that set of failed flows be $F_{s,t} \subseteq P_{s,t}$; since there are devices $v^*{}_i \in D_i$ which are not affected by A, the operation of flows $F^*{}_{s,t} = P_{s,t} \cap F_{s,t}$ should continue as much as possible. However, the devices $v^*{}_i$ cannot request an updated path to c_i due to connectivity loss; therefore, those devices need to try to reconnect to any alternative controller (k) and then recalculate the paths of the failed data-flows.

We formulate the problem as shown in (3):

$$\begin{aligned} \min_{i \in V^*} \min & \Bigg(\sum_{\forall i, c_k \in D^*{}_k} \sum_{i,j \in E^*} PC_{i,c_k} x_{ij}^{ic_k} \\ & + \sum_{\forall s,t \in F_{s,t}} \sum_{i,j \in E^*} PC_{s,t} y_{ij}^{st} \\ & + \sum_{\forall s,t \in F^*{}_{s,t}} \sum_{i,j \in E^*} PC_{s,t} z_{ij}^{st} \Bigg) \\ s.t. & \begin{cases} x_{ij}^{ic_k}, y_{ij}^{st}, z_{ij}^{st}, & \in \{0,1\} \\ x_{ij}^{ic_k}, y_{ij}^{st}, z_{ij}^{st}, & = \begin{cases} 1, & \text{if } x_{ic_k}, y_{st}, z_{st} \text{ traverses } e_{i,j} \\ 0, & \text{otherwise} \end{cases} \\ x_{ic_k}, y_{st}, z_{st}, & \text{are the safest path} \end{cases} \end{aligned} \tag{3}$$

where the first term refers to the C/M-Plane, the objective would be to maintain the connectivity to the closest $k \geq 1$ controllers, such that the cost of establishing those new connections using the safest paths in the surviving edges E^* should be minimized. The second and third terms, refer to the cost to maintain the connectivity of the D-Plane flows of the failed data-flows $F_{s,t}$ (in the first case) and the surviving flows that might fail in a near-future $F^*{}_{s,t}$ (in the second case) by redirecting the flows through the safest paths. Note that, for simplicity, we refer to the *safest path* as the one that traverses devices and edges outside A.

D. DISASTER MODEL

The impact of a disaster, and the failure of devices, will depend on several factors, which would make it challenging to model a realistic disaster scenario; thus, for simplicity, this study assumes a basic approach.

The disaster model will consist of a circular-shaped disaster with a given epicenter ϵ with a foreseen maximum damage radio r_{max} as similarly done in related work [25], [26], and whose initial design was presented in previous work [29]. The disaster area expands from ϵ homogeneously at speed s, so that a device v_i or the links attached to the device will fail if the Euclidean distance from the node d_{v_i} or a link $d_{e_{i,j}}$ is within the affected radio r_{t_n} as shown in Fig. 4.

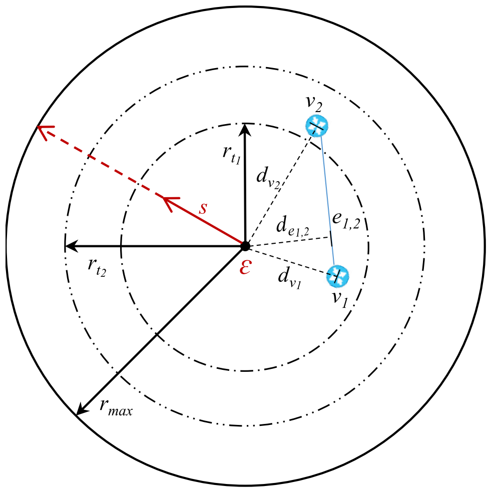


FIGURE 4. Disaster failure model.

Moreover, the expected time to failure (TTF) for the node and link denoted by T_{v_i} and $T_{e_{i,j}}$ for the node v_i and link $e_{i,j}$ respectively, depends on the speed at which the disaster expands from ϵ and is determined as shown in (4) below.

$$\begin{aligned} T_{v_i} &= \Delta t + d_{v_i}/s \\ T_{e_{i,j}} &= \Delta t + d_{e_{i,j}}/s \end{aligned} \tag{4}$$

where Δt represents the prediction time by the early alert for the expected failure, and the proposed approach assumes that the disaster will expands from time $t = \Delta t$ towards the area of the circle. So that, if the disaster area at a time t_i with a radius r_{t_i} reaches a node $v \in V$, that node will fail as well as all the connections attached to it (both at the D-, and C/M-Plane). Moreover, if the node v_i fails and encompasses a controller c_i, then all the control lines Z_i of c_i are also disconnected.

E. CONTROLLER PLACEMENT AND CONTROLLER ASSIGNMENT

The location of the controller is a critical factor that will determine the survivability of the network devices under its domain; therefore, it should be carefully considered. The Controller Placement Problem has been identified as an NP-hard problem [32], and several mechanisms have been proposed over the years [33]. However, due to the disaster unpredictability (e.g., position, scale), rather than formulating the optimal controller placement and assignment (which is outside the scope of this study), for simplicity, we adopt a randomized approach. In this approach, we assume that all controllers are randomly associated with node $v_i \in V$, randomly located at (x_i, y_i). Moreover, the domain assignment is based on the distance to each controller, as described in Algorithm 1. As observed, initially, a k number of controllers is randomly selected from the set of nodes in V. Then, all nodes $v \in V$ are assigned to a controller $c' \in C$, obtained by the function *selectMinDistance* (at line 7), which compares the Euclidean distance from the device to each of the controllers. As a result of this initial assignment, each device belongs to a domain connected via an out-of-band link.

Algorithm 1 Controller Placement and Domain Assignment

```
1:  function SETINITIALCONTROLLERSTATE(K, G, G')
2:      $C \leftarrow \emptyset$
3:      for ($i = 0; i < k; i++$) do
4:          $c_i \leftarrow$ selectRandom($v \in V$)
5:      end for
6:      for each $v \in V$ do
7:          $c' \leftarrow$ selectMinDistance($v, C$)
8:          $D_{c'} \cup v$
9:          $O_{c'} \cup o_{c',v}$
10:     end for
11: end function
```

Note that, the only restriction on the assignment is the distance; however, this could be easily extended to limit other parameters, i.e., the maximum distance to the controller, or the maximum number of network devices to a single controller. Nevertheless, based on preliminary experiments, the average number of devices assigned to each controller is reasonably and homogeneously distributed.

F. DESIGN AND OVERVIEW OF THE PROPOSED APPROACH

Based on the elements presented in the preceding sections, the proposal consists of a three-stage mechanism to solve the target issues in Section II-B, namely:

- Controller Disconnection Avoidance (CDA)
- Data Communication Protection (DCP)
- Disaster Impact Monitoring (DIM)

1) CONTROLLER DISCONNECTION AVOIDANCE (CDA)

Fig. 5 depicts the proposed timeline, which starts at time Δt (early alert) by calculating the restoration procedure and applying it before the expected failure ($t = 0$). The primary goal of this stage is to avoid the disconnection of the controller; to this aim, a determined number of k alternative

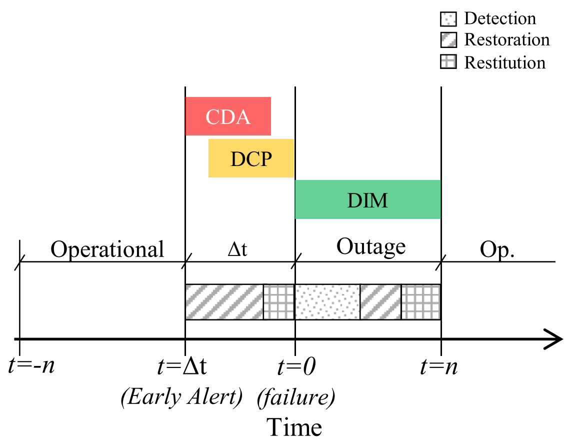


FIGURE 5. Recovery process timeline in the proposed approach.

control-lines are calculated around the disaster perimeter to the k-nearest controllers. Since it will be difficult to establish alternative unplanned out-of-band lines instantaneously, in-band connections are used as the preferred controller connection method. However, in case additional out-of-band lines are available per device, this stage would also consider those connections as the highest priority.

$$RFI_i = \frac{||d(c_i, \epsilon)||}{r_{max}} \tag{5}$$

The main function is described in Algorithm 2, and the overall process works as follows:

- **Step 1:** At line 3, the first task at this phase is to calculate which of the controllers is vulnerable, to do so, the *Risk Factor Index (RFI)* is calculated for all controllers $c \in C$ which are within the maximum expected disaster area (πr_{max}^2) from the epicenter ϵ as defined in (5), i.e., the Euclidean distance from the controller to the epicenter. This process, called *getVulnerableControllers*, is described in Algorithm 3, in case a controller has an $RIF < 1$, then it is within this area.
- **Step 2:** For each of those vulnerable controllers, the next step is to determine the k closest controllers by the function *getkClosestControllers* at line 5, where k is a given number of alternative controllers, and the function fetches the ones that are not vulnerable, avoiding, therefore, the affected area.
- **Step 3:** As the next step (at line 8 in Algorithm 2), alternative in-band paths are calculated around the disaster area to the k closest controllers (Region-Disjoint). Moreover, to decrease the probability of having various connections passing by the same links, it is also necessary to calculate maximum disjoint paths to the alternative controllers. However, since the region-disjoint paths problem is known to be NP-hard [35], and the maximum disjoint paths an NP-complete problem [36], [37], this study proposes a heuristic based on the well-known Floyd-Warshall [38] algorithm, which is a dynamic programming method used to find the shortest paths of all pairs of vertices $v \in V$. As observed in Algorithm 4, the adjacency matrix is calculated based on the D-Plane *RFI* of the links. Note that (at lines 15–23 of Algorithm 4) in case any of the nodes that comprise the edge is within the affected area, then the edge is vulnerable and therefore avoided. It is also worth mentioning that a constant α is added to $RFI_{e_{i,j}}$ (at line 26). The value of α varies according to the stage and the number of connections going through the edge; therefore, the more parallel connections, the higher the cost of that edge. Note that (at line 37 of Algorithm 4), this cost is assigned as the initial cost of the links, which will be updated in the next phase, but in case the Δt is very short, there will

Algorithm 2 Function to Select the Paths From All Vulnerable Nodes to Alternative Safe Controllers

```
1:  function SETALTERNATIVECONTROLLERSPATH(G, G', $k$, $\epsilon$, $r_{max}$)
2:      $paths \leftarrow \emptyset$
3:      $vulnerableControllers \leftarrow getVulnerableControllers(C, \epsilon, r_{max})$
4:      for each $c \in vulnerableControllers$ do
5:          $alternativeControllers \leftarrow getkClosestControllers(c, C, k, \epsilon, r_{max})$
6:          for each $altC \in alternativeControllers$ do
7:              for each $v \in D_c$ do
8:                  getAllDisjointPaths (G, $\epsilon$, $r_{max}$, $allPaths$, $adjMatrix$)
9:                  $path \leftarrow getBestPath(v, altC, allPaths, adjMatrix, E)$
10:                 if ($path = \emptyset$) then
11:                     $path \leftarrow getDijkstraShortestPath(v, altC, allPaths, adjMatrix, E)$
12:                 end if
13:                 $paths \cup path$
14:             end for
15:         end for
16:     end for
17:     return paths
18: end function
```

Algorithm 3 Function to Select the Vulnerable Controllers Within the Affected Area

```
1:  function GETVULNERABLECONTROLLERS(C, $\epsilon$, $r_{max}$)
2:      $vulnerableControllers \leftarrow \emptyset$
3:      for each $c \in C$ do
4:          $RFI_c \leftarrow \frac{||d(c,\epsilon)||}{r_{max}}$
5:          if $RFI_c < 1$ then
6:              $vulnerableControllers \cup c$
7:          end if
8:      end for
9:      return $vulnerableControllers$
10: end function
```

Algorithm 4 Function to Calculate the Region Disjoint and Max-Disjoint Paths

```
1:  function GETALLDISJOINTPATHS( G, $\epsilon$, $r_{max}$, $allPaths$, $adjMatrix$)
2:      $n \leftarrow |V|$
3:      $adjMatrix$ , $allPaths \leftarrow \emptyset$
4:      for ($i = 0; i < n; i++$) do
5:          for (($j = 0; j < n; j++$) do
6:              if $i = j$ then
7:                  $allPaths_{i,j}$ , $adjMatrix_{i,j} \leftarrow 0$
8:              else
9:                  $allPaths_{i,j} \leftarrow -1$
10:                 $adjMatrix_{i,j} \leftarrow \infty$
11:             end if
12:         end for
13:     end for
14:     for each $e \in E$ do
15:         $i \leftarrow e.getSourceVertex()$
16:         $j \leftarrow e.getDestinationVertex()$
17:         $RFI_i \leftarrow \frac{||d(v_i,\epsilon)||}{r_{max}}$
18:         $RFI_j \leftarrow \frac{||d(v_j,\epsilon)||}{r_{max}}$
19:         $vulnerableVertex \leftarrow false$
20:         if ($RFI_i < 1 \vee RFI_j < 1$) then
21:             $vulnerableVertex \leftarrow true$
22:         end if
23:         $RFI_{e_{i,j}} \leftarrow \infty$
24:         $numberOfConnections \leftarrow e.getConnectionNumber()$
25:         if !($vulnerableVertex$) then
                                ▷ The RFI calculation changes in each stage
26:             $RFI_{e_{i,j}} \leftarrow \alpha + numberOfConnections$
27:         end if
28:         $adjMatrix_{i,j}$, $adjMatrix_{i,j} \leftarrow RFI_{e_{i,j}}$
29:         $allPaths_{i,j} \leftarrow j$
30:         $allPaths_{j,i} \leftarrow i$
31:     end for
32:     for ($k = 0; k < n; k++$) do
33:         for ($i = 0; i < n; i++$) do
34:             for ($j = 0; j < n; j++$) do
35:                 if ($adjMatrix_{i,k}$, $adjMatrix_{k,j} \neq \infty \wedge adjMatrix_{i,j} > adjMatrix_{i,k} + adjMatrix_{k,j} \wedge i \neq j$) then
36:                     $adjMatrix_{i,j} \leftarrow adjMatrix_{i,k} + adjMatrix_{k,j}$
37:                     $\varphi_{i,j} \leftarrow adjMatrix_{i,j}$
38:                     $allPaths_{i,j} \leftarrow allPaths_{i,k}$
39:                 end if
40:             end for
41:         end for
42:     end for
43: end function
```

at least have an initial value. As a result of this step, this function will return all the paths in the matrix *allPaths*.

- **Step 4:** From the previous step, the paths with the lowest cost will be assigned from all the nodes v in the domain of vulnerable controllers D_c to the k nearest safe controllers (at line 9 in Algorithm 2). The path selection is performed by the function *getBestPath*, which is described in Algorithm 5; however, in case there are no available paths with those characteristics, then the shortest paths are used instead, which are calculated by the well-known Dijkstra's algorithm (as shown at line 10–12 in Algorithm 2). Finally, the resulting paths will be added to the set of in-band connections $I_{c_m,i}$ of the network device v_i, and the alternative controller c_m.

Algorithm 5 Function to Select Best Path From a Source to a Destination Vertex

```
1:  function GETBESTPATH($s$, $t$, $allPaths$, $adjMatrix$, $E$)
2:      $bestEdges \leftarrow \emptyset$
3:      if ($adjMatrix_{s,t} < 0.1$) then
4:          return $\emptyset$
5:      else
6:          $nextV \leftarrow \emptyset$
7:          for ($nextVertex = s$; $nextVertex \neq t$; $nextVertex = allPaths_{nextVertex,t}$) do
8:              if ($allPaths_{nextVertex,t} > 0$) then
9:                  $bestEdges \cup e_{nextVertex,t}$
10:                 $e_{nextVertex,t}.addInBandController(t)$
11:             end if
12:         end for
13:     end if
14:     if ($bestEdges = \emptyset$) then
15:         return $\emptyset$
16:     else
17:         return $bestEdges$
18:     end if
19: end function
```

For instance, consider the scenario depicted in Fig. 6, which shows the Science Information NETwork (SINET5) [34] of Japan, at time Δt the controller c_1 is within the maximum expected affected area. Therefore, alternative in-band lines need to be established in all the devices which are in the domain of c_1. Assume the node v_1 (depicted in dark blue in Fig. 6) is part of the domain of c_1, moreover, the number of alternative controllers is $k = 2$. Therefore, it is necessary to calculate two alternative in-band connections to controllers c_2 and c_3, which are the closest to c_1 outside the maximum expected disaster area. However, if the shortest path is selected as the traditional methods would do (Shown as the cyan arrows in Fig. 6), when the disaster area reaches the nodes and edges that traverse those paths (i.e., v_2) they will not be usable anymore; leaving v_1 disconnected to any controller and therefore non-operational. On the other hand, the proposed approach uses the paths that will not be affected by the disaster (Region-disjoint) by using the paths through v_3 (shown as the magenta arrows in Fig. 6). Finally, note that

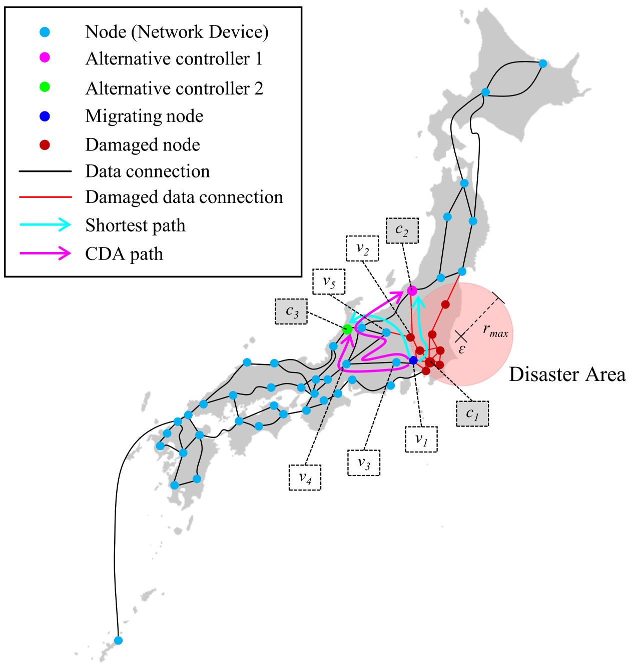


FIGURE 6. **CDA phase sample on SINET5 [34].**

on v_4, one of the paths goes through v_5 instead of using the same paths (max-disjoint paths).

2) DATA COMMUNICATION PROTECTION (DCP)

Once the C/M-Plane has been ensured in the CDA phase, there will be at least one alternative connection to the controller, either via an out-of-band or through in-band connections to alternative controllers. The primary goal of this phase is to protect the Data Communication by updating the Risk Factor Index (RFI) of all the edges in the expected affected area so that the traffic traversing the links can be aware of the risk of continuing the service through the paths involving these edges. To this aim, we use Algorithm 4 to calculate the safest paths; however, at this phase, the $RFI_{e_{i,j}}$ is calculated as shown in (6).

$$RFI_{e_{i,j}} = RFI^0_{e_{i,j}} = \frac{T_{e_{i,j}}}{T_{max}} \tag{6}$$

where $T_{e_{i,j}}$ is the expected time of failure of $e_{i,j}$ after the early alert at Δt, and T_{max} is the time of the disaster to expand from ϵ to r_{max}. Thus, this value can alert the upper layers or routing algorithms of the risk. It is also worth noting that this process is only performed once in this phase.

3) DISASTER IMPACT MONITORING (DIM)

Finally, at the expected failure ($t = 0$), this phase monitors the impact of the disaster by periodically updating the RFI based on the state of the network and the progress of the disaster. T_{update} gives the update interval, and it is performed a given maximum check-points $n > 1$, which might depend on the type of disaster (i.e., an earthquake could have a shorter interval but fewer check-points). The check-point time t_n,

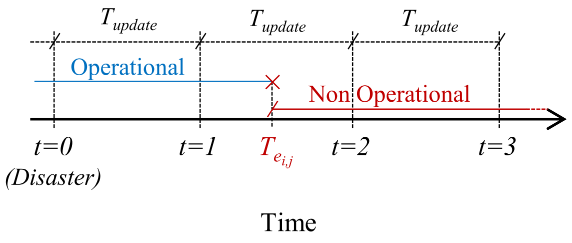


FIGURE 7. **Timeline of the expected time to failure for an edge in the DIM phase.**

as shown in Fig. 7, will determine the monitoring area from the epicenter ϵ to the affected radius r_{t_n} and update the $RFI^n_{e_{i,j}}$ (in Algorithm 4) as described in (7).

$$RFI^n_{e_{i,j}} = \frac{T_{e_{i,j}} - n \times T_{update}}{n \times T_{update}} \tag{7}$$

When $RFI^n_{e_{i,j}} < 1$, then the edge $e_{i,j}$ has become non-operational, and therefore removed from set E of available edges, so that at time $(n+1)$ of the RFI of this edge will be $RFI^{n+1}_{e_{i,j}} = \varphi_{i,j} = \infty$.

At this stage, if the node is still active, then from the set of paths $P^n_{s,t}$ at time n, the set of failed paths $F^n_{s,t} \subseteq P^n_{s,t}$ defined as in (8), contains all the paths that need to be re-routed, since the data-flows must continue to transmit as much as possible.

$$F^n_{s,t} = \{p_{s,t} \mid e_{i,j} = e_{j,i} \in p_{s,t} \wedge RFI^n_{e_{i,j}} < 1\} \tag{8}$$

Of course, in case the node is not active, and there is no path to replace the failed ones, then the data transmissions will fail.

Note that, the heaviest procedure on the proposed solution is the search of paths between all pairs of Algorithm 4, which is based Floyd-Warshall [38], the time and space complexity would be of order $O(k \cdot |V|^3)$ and $O(k \cdot |V|^2)$ respectively as the upper limit. Although it might not seem asymptotically optimal or better than traditional Dijkstra-based approaches, it runs better in practice [39].

IV. EVALUATION

Due to known limitations in emulated and simulated environments concerning the C/M-Plane properties, such as the support for multiple-controllers, support for multiple out-of-band/in-band connections, among others. This study uses a custom-made simulation in Java to show the feasibility and effectiveness of the proposed approach. The nodes and links were modeled to implement the most primitive properties and functions; therefore, we leave the exploration of a more comprehensive implementation in a more standard setting as future work.

A. EXPERIMENT SETUP

The environment simulates an earthquake with a simple circular pattern as the disaster area with an affected radius and speed of propagation, as described in Table 3, which also describes the other parameters. The experiments were conducted in a virtual machine using Ubuntu 16.04 LTS,

TABLE 3. Experimental parameters.

Parameter	Value
Topology	Gabriel graphs [40] (neighboring distance 10 Km)
Bandwidth ($b_{i,j}$)	1000 Mbps
Test Surface	$30 \times 20\, km^2$
# of Nodes	200
# of Flows (transmissions)	25 per test
Amount of data per Flows	Randomly set from 1 to 4 GB per flow
# Controllers	10
Early alert (Δt)	100 s
Checkpoint time (T_{update})	30 s
Maximum affected radius (r_{max})	5 km
Disaster propagation speed (s)	50, 100, 150, and 500 m/s
Number of Alternative Controllers (k)	2

with six CPUs Intel Xeon(R) E5-2650 v4 @ 2.20 GHz and 16 GB of memory. Synthetic Gabriel graphs [40] were used to create the network topology since, as shown in [41], it creates the closest structure geographic models compared with physical networks, such as the ones available in public datasets [42]. For every run in the experiment, a new topology is created, and each node is assigned to a single controller, as described in Section III-E.

B. ASSUMPTIONS

Note that this study assumes the following points:

- The detection time of failure is instantaneous, which in real devices might take some considerable time depending on the detection mechanism.
- The controller-line handover is instantaneous so that if a controller line fails, the next one will be selected. If none are available, then the network device has no connection to any controller (either using out-of-band or in-band), then the device is non-operable.
- The inter-domain hand-off, which refers to the change from a controller to another one, is also assumed to occur instantaneously.
- The out-of-band connection will only fail if one of the devices fails, which will simplify the possible failure for reasons other than the disaster.
- The implementation of all the operation takes no time, the time that takes to implement a solution will vary according to the device specifications, the type, or the number of operations. However, to simplify the model, this variable was not considered in the current simulation.

C. COMPARISON METRICS AND APPROACHES

Regarding the benchmark approaches, the experiments were performed using the following ones:

1) **Proactive:** In this approach, which represents the basic protection mechanism, all the alternative controllers and their paths are pre-planned and installed in all devices. In case of failure of one of the controllers, all its dependent network devices will try to use the alternative connections to the alternative. In principle, the best alternative will be the k closest controllers using the shortest paths.
2) **Reactive:** This approach represents the basic restoration mechanism, which calculates the possible alternative controllers and their paths immediately after a failure is detected. As it was in the case of the Proactive approach, the criteria to select the alternatives will be the closest controllers using the shortest paths.
3) **Proposed:** This approach is the proposed scheme, which implements the functions described in Section III.

Moreover, the following parameters were measured as the metrics to test the feasibility and effectiveness of the proposed approach:

- **Percentage of non-operational nodes** (Ψ), which refers to the number of network devices that did not fail, but they are not connected to any controller, as defined in (9):

$$\Psi = \frac{(\text{Total} - \text{Non-operational} - \text{failed})}{\text{Total}} \cdot 100 \quad (9)$$

- **Implementation cost** (λ), although this metric can involve various other forms, for simplicity, λ will be computed based on the number of paths calculated as defined in (10):

$$\lambda = (\text{\# involved devices} + \text{\# recalculated paths}) \cdot k \quad (10)$$

  However, this metric can be used to build up more elaborated ones, which may imply, e.g., the implementation time, path lengths, among others.
- **Completion rate** ($\Gamma_{success}$), which refers to the number of successful transmissions a the D-Plane over all the initiated flows from the early alert (Δt), as defined in (11):

$$\Gamma_{success} = \frac{\text{\# of successful transmissions}}{\text{\# of total transmissions}} \quad (11)$$

  Note that, since this metric evaluates the D-Plane, there are many possible variations on the parameters described in Table 3. However, for these preliminary experiments, we are only considering the disaster propagation speed (s).

D. EXPERIMENT PROCEDURE

The procedure of the experiment was as follows:

- At time $t = \Delta t$: To simulate the background traffic, all the edges are assigned a randomly available bandwidth ($b_{i,j}$) from 500 to 1000 Mbps. Then, 25 transmissions are initiated from different sources (s) to various destinations (t) regardless of their position, creating, therefore, the set of current transmissions ($P_{s,t}$).

- At time $t = 0$: At this instant of time, the disaster starts to spread using the model described in Section III-D.
- At time $t = 1, 2, 3, \ldots, n$: The status of the nodes and edges are updated every check-point time defined by T_{update}.
- At time t_{max}: When the disaster has reached its maximum area, the experiment concludes and all the variables are collected.

This process was repeated hundreds of times; for each run, a new topology was created, new devices were designated as controllers with their respective domain, and the epicenter of the disaster changed the position as defined in the process of Algorithm 1. However, since the main focus in this paper is to investigate the effect of multi-controller failure, we only collect the data of those cases when there was at least one failed controller within the disaster area.

E. RESULTS

1) NON-OPERATIONAL NODES (Ψ)

Regarding the amount of non-operational nodes, active nodes but not connected to any controller, Fig. 8 shows a comparison of the results obtained by each of the approaches. As observed, the proposed method got the lowest percentage of non-operational nodes, which will allow the surviving nodes to continue the transmission even if its initially assigned controller failed. Note that the obtained percentage was from the total of nodes in the topology. In the case of the Reactive approach, when five controllers failed, as high as 20% of the nodes were non-operational; additionally, 20 to 25% nodes failed due to the disaster, therefore, around half of the devices were unusable. This can significantly affect the whole network and the services running on top of it.

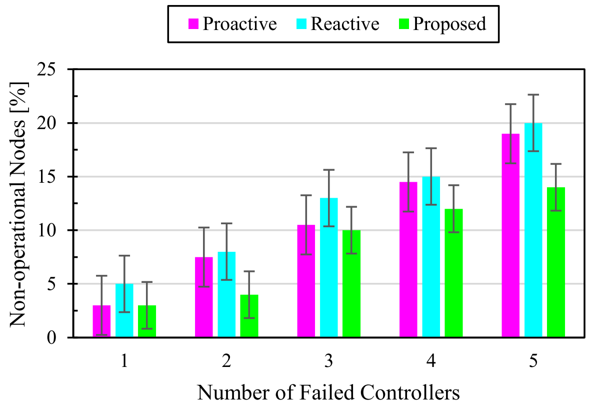


FIGURE 8. Percentage of non-operational nodes per approach.

2) IMPLEMENTATION COST (λ)

Concerning the cost of implementation (λ), which is currently based on the number of paths calculated to implement the solution. Fig. 9 shows the results obtained, as observed, although the Reactive approach was the one that has the least cost due to its intrinsic mechanisms of only performing the calculations in the related devices. Nevertheless, the proposed approach had a comparable cost to the Reactive one, but with better performance.

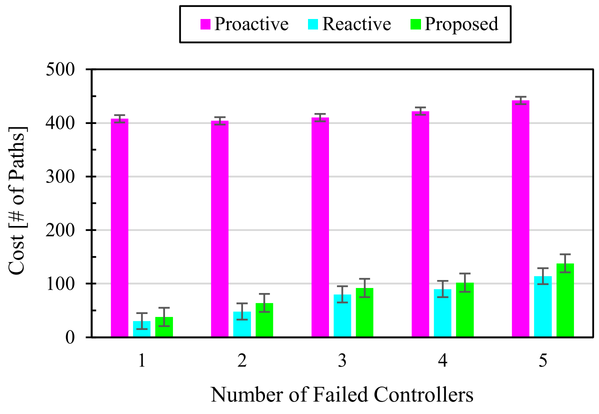


FIGURE 9. Cost of recovery implementation per approach.

Note that the Proactive approach presents a λ significantly higher than that the other approaches since the pre-planned calculations need to be performed in advance for all the devices. Nonetheless, despite having all the pre-installed paths, it still needed to perform additional calculations as the nodes and edges failed during the disaster for both the D- and C/M-Plane, although, at this time, only the C/M-Plane were considered for computing λ.

3) COMPLETION RATE ($\Gamma_{success}$)

The last measured parameter was the completion ratio ($\Gamma_{success}$) of D-Plane Transmissions. Fig. 10 summarizes the results obtained. The experiments were conducted using four different speeds of disaster propagation (i.e., 50, 100, 150, and 500 m/s). As observed, from the 25 transmissions that were active from the early alert time (Δt), the proposed approach achieved the highest successful completion rate regardless of the number of failed controllers, or the disaster propagation speed. Moreover, note that as the propagation increased, only the proposed approach could finish up to 40% of the initiated data flows. In the case where the propagation speed was relatively slow ($s = 50$ m/s), the difference in the percentage of successful transmissions compared to the other approaches is as large as 50%, even if half of the controllers in the topology stopped working.

Note that (as shown in Fig. 11), even if the completion rate of the other approaches was low, the number of paths re-routes was higher in each of the variants. The additionally computed re-routes were due to the failure of nodes or edges in the transmissions. The reduction of the number of re-routes was up to 40% compared to the other approaches, which confirms that by maintaining connectivity to the controller and selecting safer paths, more transmissions can finish successfully.

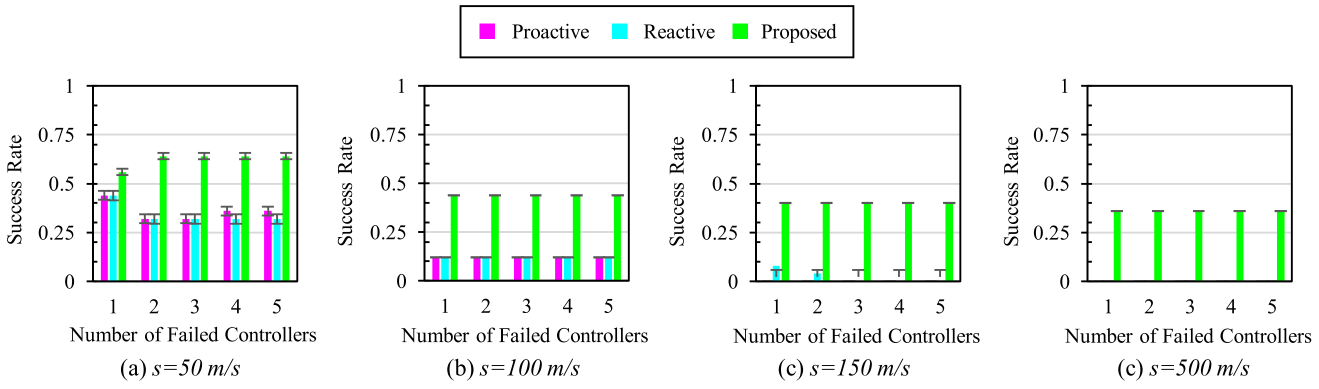


FIGURE 10. **Success rate of data transmission per approach.**

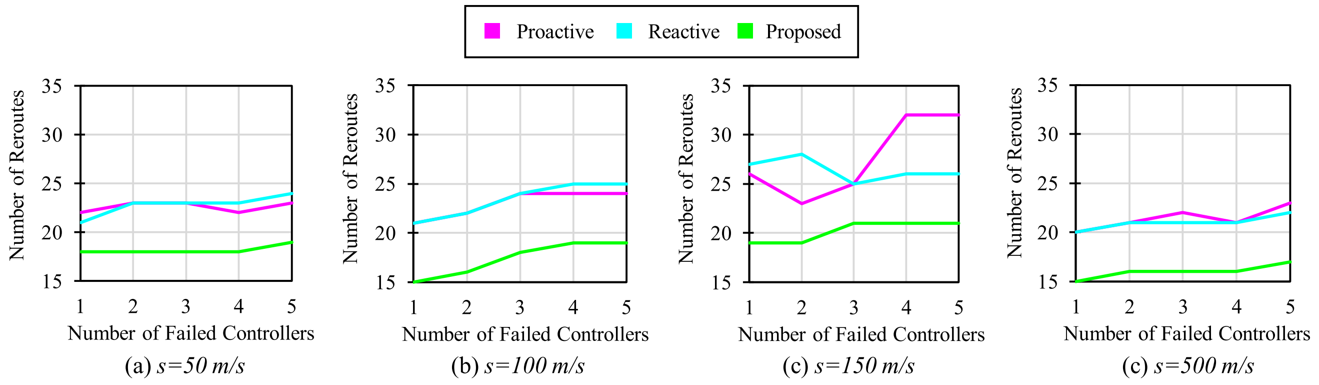


FIGURE 11. **Number of paths re-route for data transmission per approach.**

V. DISCUSSION

Based on the preliminary results obtained in the preceding section, compared to the traditional protection mechanism, the proposed approach does not pre-calculate a solution in advance so that there are no unnecessary stored rules in the network device; consequently, the cost of implementation is much lower. Moreover, since the proposal does not wait for the individual network devices' detection to start the procedures, it can be more effective than traditional Restoration mechanisms, with more effective use of the resources.

Concerning the implications of excluding some variables for simplicity (i.e., the time to perform/implement any operation or the inter-domain handover), although they were not included in the present simulation, the proposed approach might also have a positive influence even if those variables are included. For example, as shown in the results obtained in the implementation cost (in Section IV-E2), since the number of calculations will directly influence the implementation time, by reducing the former, the latter can also be reduced. Nevertheless, these variables will be considered in future implementations of the experiments to confirm their influence.

Regarding the disaster model, compared to the on-going research in that particular area [43], the current implementation is simplistic. However, as long as the model receives an approximate disaster area and type of disaster, other standard parameters such as the speed of propagation might be inferred based on available records from previous events.

A non-negligible improvement to the proposed approach would be to use advanced controller placement and assignment mechanisms, like the ones shown in related work [33]. However, the adopted mechanism might shed some light on preliminary resilience testing for further refinement by indicating the most vulnerable regions and the possible impact in the overall topology.

Finally, although the proposed approach does not consider self-healing mechanisms for each device, as done by other authors in related work [18], [19], it does reduce the amount of non-operational nodes. Therefore, the devices that could not be assigned to any controller might implement a self-healing mechanism after the proposed phases have finished.

VI. CONCLUSION AND FUTURE WORK

SDN-enabled network devices intrinsically need to be connected to a controller to offer service continuity, especially in cases where those services are of paramount importance (as is the case of disasters); this paper presented a three-stage protection mechanism to improve the resilience of the services.

The proposed approach considers an early alert of the disaster at the CDA phase. This early alert will allow keeping the network device connected to an SDN controller. So that, at the D-Plane, the services would still benefit from the programmability. Next, at the DCP phase, the D-Plane upper layers are aware of the risk and adapt accordingly. Based on this information, the transmissions mechanisms can make more informed decisions on how to route their traffic. Finally, the risk is periodically updated in the DIM phase.

Preliminary results show that, by applying the proposed approach, a trade-off can be achieved between the restoration and protection mechanism. The implementation cost is comparable (or lower) to reactive approaches, but much more effective, and the implementation effectiveness is higher compared to proactive approaches since it does not need to store and apply the solution to a large number of uninvolved devices. Based on these results, this paper showed that it is possible to maximize network survivability, and consequently, the service continuity by ensuring the connection to a controller, which minimizes the number of non-operable devices.

For future work, we plan to improve the proposal by extending the current implementation, which only considers limited variables to formulate the model. For instance, we assumed only wired technologies at the physical level; however, it would be interesting to create collaborative mechanisms with others such as Software-defined Radio or Software-defined Optical Networks. Also, using real topologies can fine-tune the proposed method to create more resilient and fail-tolerant services. In the field of Simulation for Natural Hazards Engineering, it would be useful to test the impact of the combination of multiple disaster events (i.e., earthquake and tsunami) and its effect on the adaption scheme. Finally, we are currently working on implementing the proposal in a more standard simulation tool (e.g., NS3, OMNeT++) so that we can simulate more realistic environments.

ACKNOWLEDGMENT

The authors of this article would like to thank the anonymous reviewers and Timothy Jim for their insightful suggestions which helped to improve the final manuscript.

REFERENCES

[1] Open Networking Foundation (ONF). *Software-Defined Networking (SDN) Definition*. Accessed: Oct. 14, 2019. [Online]. Available: https://www.opennetworking.org/sdn-definition/

[2] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 136–141, Feb. 2013, doi: 10.1109/MCOM.2013.6461198.

[3] L. Sidki, Y. Ben-Shimol, and A. Sadovski, "Fault tolerant mechanisms for SDN controllers," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2016, pp. 173–178.

[4] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (SDN)," *Comput. Netw.*, vol. 112, pp. 279–293, Jan. 2017, doi: 10.1016/j.comnet.2016.11.017.

[5] 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 Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008, doi: 10.1145/1355734.1355746.

[6] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Enabling fast failure recovery in OpenFlow networks," in *Proc. 8th Int. Workshop Design Reliable Commun. Netw. (DRCN)*, Oct. 2011, pp. 164–171.

[7] Open Networking Fundation (ONF). *OpenFlow Specifications*. Accessed: Nov. 11, 2019. [Online]. Available: https://www.opennetworking.org/software-defined-standards/specifications/

[8] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, *Overview and Principles of Internet Traffic Engineering*, document IETF RFC-3272. Accessed: Nov. 11, 2019. [Online]. Available: https://tools.ietf.org/html/rfc3272

[9] P. C. D. R. Fonseca and E. S. Mota, "A survey on fault management in software-defined networks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2284–2321, Jun. 2017, doi: 10.1109/COMST.2017.2719862.

[10] J. P. Vasseur, M. Pickavet, and P. Demeester, *Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP and MPLS*, 1st ed. San Mateo, CA, USA: Morgan Kaufmann, 2004.

[11] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Fast failure recovery for in-band OpenFlow networks," in *Proc. 9th Int. Conf. Design Reliable Commun. Netw. (DRCN)*, Budapest, Hungary, Mar. 2013, pp. 52–59.

[12] Nippon Telegraph and Telephone NTT East Corporation. *Recovering from the Great East Japan Earthquake: NTT East's Endeavors*. Accessed: Nov. 11, 2019. [Online]. Available: https://www.ntt-east.co.jp/info/detail/pdf/shinsai_fukkyu_e.pdf

[13] Y. Shibata, N. Uchida, and N. Shiratori, "Analysis of and proposal for a disaster information network from experience of the great east japan earthquake," *IEEE Commun. Mag.*, vol. 52, no. 3, pp. 44–50, Mar. 2014, doi: 10.1109/MCOM.2014.6766083.

[14] G. Sato, N. Uchida, N. Shiratori, and Y. Shibata, "Research on never die network for disaster prevention based on OpenFlow and cognitive wireless technology," in *Proc. IEEE 30th Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Mar. 2016, pp. 370–375.

[15] FP7-ICT: Specific Programme Cooperation: Information and communication technologies. *SPARC: Split Architecture Carrier Grade Networks*. Accessed: Nov. 10, 2019. [Online]. Available: https://cordis.europa.eu/project/rcn/95200/factsheet/

[16] B. Jenkins, D. Brungard, M. Betts, N. Sprecher, and S. Ueno, *Requirements of an MPLS Transport Profile*, document IETF RFC-5654. Accessed: Nov. 10, 2019. [Online]. Available: https://tools.ietf.org/html/rfc5654

[17] Y. Hu, W. Wendong, G. Xiangyang, C. H. Liu, X. Que, and S. Cheng, "Control traffic protection in software-defined networks," in *Proc. IEEE Global Commun. Conf.*, Dec. 2014, pp. 1878–1883.

[18] T. Omizo, T. Watanabe, T. Akiyama, and K. Iida, "ResilientFlow: Deployments of distributed control channel maintenance modules to recover SDN from unexpected failures," *IEICE Trans. Commun.*, vol. E99.B, no. 5, pp. 1041–1053, May. 2016, doi: 10.1587/transcom.2015amp0007.

[19] M. Osman, J. Nunez-Martinez, and J. Mangues-Bafalluy, "Hybrid SDN: Evaluation of the impact of an unreliable control channel," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2017, pp. 242–246.

[20] A. S. M. Asadujjaman, E. Rojas, M. S. Alam, and S. Majumdar, "Fast control channel recovery for resilient in-band OpenFlow networks," in *Proc. 4th IEEE Conf. Netw. Softwarization Workshops (NetSoft)*, Jun. 2018, pp. 19–27.

[21] B. Gorkemli, S. Tatlicioglu, A. M. Tekalp, S. Civanlar, and E. Lokman, "Dynamic control plane for SDN at scale," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 12, pp. 2688–2701, Dec. 2018, doi: 10.1109/jsac.2018.2871308.

[22] L. F. Muller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, and M. P. Barcellos, "Survivor: An enhanced controller placement strategy for improving SDN survivability," in *Proc. IEEE Global Commun. Conf.*, Dec. 2014, pp. 1909–1915.

[23] N. Beheshti and Y. Zhang, "Fast failover for control traffic in software-defined networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2012, pp. 2665–2670.

[24] S. S. Savas, M. Tornatore, M. F. Habib, P. Chowdhury, and B. Mukherjee, "Disaster-resilient control plane design and mapping in software-defined networks," in *Proc. IEEE 16th Int. Conf. High Perform. Switching Routing (HPSR)*, Jul. 2015, pp. 1–6.

[25] S. A. Astaneh and S. Shah Heydari, "Optimization of SDN flow operations in multi-failure restoration scenarios," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 421–432, Sep. 2016, doi: 10.1109/tnsm.2016.2580590.

[26] T. Hirayama, M. Jibiki, and H. Harai, "Designing distributed SDN C-Plane considering large-scale disruption and restoration," *IEICE Trans. Commun.*, vol. E102.B, no. 3, pp. 452–463, Mar. 2019, doi: 10.1587/transcom.2018nvp0005.

[27] S. Moazzeni, M. R. Khayyambashi, N. Movahhedinia, and F. Callegati, "Improving the reliability of Byzantine fault-tolerant distributed software-defined networks," *Int. J. Commun. Syst.*, vol. 33, Jan. 2020, Art. no. e4372, doi: 10.1002/dac.4372.

[28] A. Xie, X. Wang, W. Wang, and S. Lu, "Designing a disaster-resilient network with software defined networking," in *Proc. IEEE 22nd Int. Symp. Qual. Service (IWQoS)*, May 2014, pp. 135–140.

[29] H. Takahira, M. Hata, L. Guillen, S. Izumi, T. Abe, and T. Suganuma, "A proposal and evaluation of network control method considering disaster risk and data amount immediately after disaster," (in Japanese), *IPSJ J.*, vol. 60, no. 3, pp. 738–749, Mar. 2019.

[30] A. Malik, B. Aziz, M. Adda, and C.-H. Ke, "Smart routing: Towards proactive fault-handling in software-defined networks," 2019, *arXiv:1904.00717*. [Online]. Available: http://arxiv.org/abs/1904.00717

[31] H. Nakamura, S. Horiuchi, C. Wu, S. Yamamoto, and P. A. Rydelek, "Evaluation of the real-time earthquake information system in japan," *Geophys. Res. Lett.*, vol. 36, no. 2, pp. 1–4, Jan. 2009, doi: 10.1029/2008gl036470.

[32] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw. HotSDN*, Aug. 2012, pp. 67–72.

[33] J. Lu, Z. Zhang, T. Hu, P. Yi, and J. Lan, "A survey of controller placement problem in software-defined networking," *IEEE Access*, vol. 7, pp. 24290–24307, Jan. 2019, doi: 10.1109/access.2019.2893283.

[34] Science Information NETwork 5. *SINET5 Ultra High Speed/Low Latency Network Throughput Japan*. Accessed: Dec. 18, 2019. [Online]. Available: https://www.sinet.ad.jp/

[35] S. Trajanovski, F. A. Kuipers, P. V. Mieghem, A. Ilić, and J. Crowcroft, "Critical regions and region-disjoint paths in a network," *Proc. IFIP Netw.*, Brooklyn, NY, USA, May 2013, pp. 1–9.

[36] A. Itai, Y. Perl, and Y. Shiloach, "The complexity of finding maximum disjoint paths with length constraints," *Networks*, vol. 12, no. 3, pp. 277–286, 1982, doi: 10.1002/net.3230120306.

[37] C.-L. Li, S. T. McCormick, and D. Simchi-Levi, "The complexity of finding two disjoint paths with min-max objective function," *Discrete Appl. Math.*, vol. 26, no. 1, pp. 105–115, Jan. 1990, doi: 10.1016/0166-218x(90)90024-7.

[38] R. W. Floyd, "Algorithm 97: Shortest path," *Commun. ACM*, vol. 5, no. 6, p. 345, Jun. 1962, doi: 10.1145/367766.368168.

[39] S. S. Skiena, *The Algorithm Design Manual*, 2nd ed. London, U.K.: Springer-Verlag, 2008.

[40] K. R. Gabriel and R. R. Sokal, "A new statistical approach to geographic variation analysis," *Syst. Biol.*, vol. 18, no. 3, pp. 259–278, 1969, doi: 10.2307/2412323.

[41] E. K. Cetinkaya, 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," in *Proc. 5th Int. Congr. Ultra Modern Telecommun. Control Syst. Workshops (ICUMT)*, Sep. 2013, pp. 38–45.

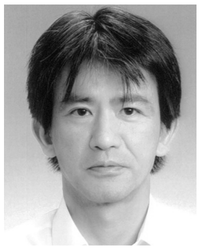
[42] J. P. G. Sterbenz, J. P. Rohrer, E. K. Çetinkaya, M. Alenazi, A. Cosner, and J. Rolfe, *ResiliNets: KU-Topview Network Topology Tool*. Lawrence, Kansas: Univ. Kansas, 2010.

[43] G. G. Deierlein and A. Zsarnóczay Eds., "State-of-art in computational simulation for natural hazards engineering," in Report 2019–01 in *Proc. Center Comput. Modeling Simulation (SimCenter)*, Feb. 2019, pp. 106–111, doi: 10.5281/zenodo.2579582.

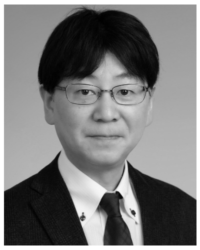
LUIS GUILLEN (Member, IEEE) received the M.S. degree in computer science from Vrije Universiteit Brussel, Belgium, in 2012, and the Ph.D. degree in information science from Tohoku University, Japan, in 2020. He is currently a Specially Appointed Assistant Professor with the Research Institute of Electrical Communication, Tohoku University. His research interests include HCI, the IoT, network management, resilient networks, software-defined networking, green ICT, DASH, and QUIC. He is a member of the IPSJ, Japan, and SIB, Bolivia.

HIROYUKI TAKAHIRA received the M.S. degree from the Graduate School of Information Sciences, Tohoku University, in 2018. He is currently working Hitachi, Ltd. His research interest includes software defined networking.

SATORU IZUMI received the M.S. and Ph.D. degrees from Tohoku University, Japan, in 2009 and 2012, respectively. He is currently an Associate Professor with the Graduate School of Information Sciences, Tohoku University. He received the IEEE GCCE 2013 Excellent Paper Award. His main research interests include semantic Web, ontology engineering, green ICT, and software defined networking. He is a member of the IEICE and the IPSJ.



TORU ABE received the M.E. and Ph.D. degrees in information engineering from Tohoku University, in 1987 and 1990, respectively. From 1990 to 1993, he was a Research Associate with the Education Center for Information Processing, Tohoku University. From 1993 to 2001, he was an Associate Professor with the Graduate School of Information Science, Japan Advanced Institute of Science and Technology. He is currently an Associate Professor with the Cyberscience Center, Tohoku University. His research interests include image processing and knowledge engineering.



TAKUO SUGANUMA (Member, IEEE) received the M.S. and Ph.D. degrees in engineering from the Chiba Institute of Technology, Japan, in 1994 and 1997, respectively. He is currently a Professor and the Director with the Cyberscience Center, Tohoku University, Japan. His main research interests include agent-based computing, flexible network middleware, network management, symbiotic computing, green ICT, and Disaster-resistant communications. He is a member of the IEICE and the IPSJ. He received the UIC-07 Outstanding Paper Award in 2007.