RASCAR: Recovery-Aware Switch-Controller Assignment and Routing in SDN

S. Sedef Savas, Massimo Tornatore, *Senior Member, IEEE*, Ferhat Dikbiyik, Aysegul Yayimli, Charles U. Martel, and Biswanath Mukherjee, *Fellow, IEEE*

***Abstract*—Decoupling control and data planes in a software-defined network (SDN) has its advantages along with its challenges. Especially, resilient communication between elements in the data plane (switches) and in the control plane (controllers) is key to SDN's success as disruption of this communication after a failure can severely affect data-plane functions. After a failure, simultaneous recovery of all switch-controller communication paths (control paths) may not be possible, and multiple recovery stages may be required. Since restoration of disrupted data paths depends on the recovery of disrupted control paths feeding control information to switches, the performance of control-path recovery seriously affects data-path recovery performance. The assignment of controller to switches and the routing of controller-switch control paths are what determines the control-plane recovery performance, and hence should be performed in conjunction with a recovery plan after failures. This study proposes an algorithm for recovery-aware switch-controller assignment and routing (RASCAR), which enables fast data-path recovery after a set of failures (e.g., single point of failures and disasters). We formulate the problem as an integer linear program and propose an efficient heuristic algorithm to solve large problem instances. Our illustrative numerical studies show that RASCAR significantly reduces the data-path restoration times after any failure with a minor increase in resource consumption of control paths.**

***Index Terms*—SDN, disaster-resilient control plane, controller, control-path routing.**

Manuscript received March 28, 2018; revised August 12, 2018; accepted September 20, 2018. Date of publication November 6, 2018; date of current version December 10, 2018. This work was supported in parts by the Defense Threat Reduction Agency (DTRA) Grant No. HDTRA1-14-1-0047 and the National Science Foundation Grant No. 1818972. M. Tornatore acknowledges the research support from COST Action CA15127. The associate editor coordinating the review of this paper and approving it for publication was R. Pasquini. *(Corresponding author: S. Sedef Savas.)*

S. S. Savas is with the Computer Science Department, University of California at Davis, Davis, CA 95616 USA, and also with the Engineering Department, Big Switch Networks, Santa Clara, CA 95054 USA (e-mail: ssavas@ucdavis.edu).

M. Tornatore is with the Department of Electronics and Information, Politecnico di Milano, 20133 Milan, Italy, and also with the Computer Science Department, University of California at Davis, Davis, CA 95616 USA (e-mail: massimo.tornatore@polimi.it).

F. Dikbiyik is with the Engineering Department, NormShield Inc., McLean, VA 22102 USA (e-mail: fdikbiyik@normshield.com).

A. Yayimli is with the Computing and Information Sciences Department, Valparaiso University, Valparaiso, IN 46383 USA (e-mail: aysegul.yayimli@valpo.edu).

C. U. Martel and B. Mukherjee are with the Computer Science Department, University of California at Davis, Davis, CA 95616 USA (e-mail: cmartel@ucdavis.edu; bmukherjee@ucdavis.edu).

Digital Object Identifier 10.1109/TNSM.2018.2879865

I. INTRODUCTION

IN A SOFTWARE-DEFINED Network (SDN), data and control planes are separated to facilitate network programmability and management. This means that control functions are typically run on servers placed in different physical locations from the forwarding elements. Such separation increases network agility and flexibility, and allows operators to promptly adapt the network to changing service requirements. However, despite its advantages, separation of control and data planes in SDN creates its own challenges [1]. One of the main challenges is the resilient communication between data and control planes [1], [2], since communication networks are exposed to failures at different scales (varying from link failures to large-scale disaster failures). SDN switches (which we assume in this study to be dumb packet-forwarding devices [3]) rely on switch-to-controller connections to exchange control messages with controllers where the control logic is located. After a failure, several switch-to-controller connections (referred to as control paths) may be interrupted. While SDN switches disconnected from controllers can still function, i.e., traffic can still be processed based on old flow entries installed in the switches' flow tables, they cannot exchange control messages (e.g., flow setup request, flow installation, etc. [4]) with the controller; hence, switches without an active control path (i.e., uncontrolled switches) cannot be used to establish paths for new data traffic or reroute the disrupted data traffic. Moreover, as some SDN applications rely on an up-to-date view of the network to optimize their specific objectives (e.g., load balancing), failures that disconnect the control and data planes would prevent them from acquiring real-time states of the underlying network, and may result in performance degradation. Therefore, control-path health plays an important role in the performance of SDN networks.

The problem becomes more severe if connections between switches and controllers are realized in band, i.e., control and data paths share the same physical infrastructure. Now, along with data paths, a single physical failure may affect multiple control paths, which may be needed for data-path recovery. A late data-path recovery may cause significant penalty to the network operator depending on Service-Level Agreements (SLAs) stipulated between customers and the network operator. Maximum allowed restoration times (MART) for data paths after failures are stated in SLAs. The restoration deadline may vary based on the nature of the failure, but is usually set to a short interval on the order of milliseconds ($<$50 ms) [5], [6].

As, in an in-band scenario, control paths fail along with data paths, uncontrolled switches (switches out of control due to control-path disruption) cannot be used to restore data paths. So, disrupted data paths cannot be restored on a path including uncontrolled switches until control-path recovery. To meet strict data-path restoration times, all disrupted control paths must be recovered after any failure of interest by a deadline. As long as all switches become controlled soon enough to allow time for data-path restoration to meet its deadline, control-plane recovery performance is assumed to meet its recovery deadline. Not only recovery of data paths depends on recovery of control paths, but also to recover some disrupted control paths, other control-path recoveries may be essential; and during control-path recovery, only controlled switches can be used. So, not all control paths can be recovered simultaneously but some have to wait for others to recover first (multi-stage recovery), leading to unpredictable, prolonged recovery times. Thus, switch-controller assignment and control-path routing should be performed intelligently rather than using simplistic shortest-path strategies to minimize multi-stage recovery time of control paths. Note that switches may be opted-in to switch to legacy protocols (e.g., shortest-path routing) after their control paths are disrupted, but the intelligence to minimize multi-stage recovery time will be lost. The situation may be more tangled if data-path recovery is started before restoring all disrupted control paths due to limited view of the network.

To the best of our knowledge, no prior work has incorporated switch-controller assignment and control-path routing with minimizing multi-stage control-path recovery. In this study, we investigate such a recovery-aware switch-controller assignment and routing solution to minimize the time required for recovery. Therefore, significant economic loss can be reduced due to performance degradation and late data-path recovery.

A. Related Work

Resilience of control plane is crucial to an SDN's performance and has been well studied in the literature. Prior works mainly deal with the controller-placement problem to increase survivability, and they do not focus on switch-controller assignment and routing, relying on simplistic assumptions, e.g., assigning switches to the closest controllers and routing control paths over shortest paths [7]–[9]. One of our previous work also focus on resilient control-plane design in SDNs to minimize disaster-failure risk without focusing on restoration [10].

Restoration in control plane, specifically how to route control traffic and how to assign controller to switches to guarantee a given restoration-time target, is a largely unexplored issue. Among the few works that focus on fast control-traffic recovery after failures, [11] proposes two control-path protection approaches: switches have to be connected i) to a controller over two disjoint paths or ii) to two different controllers over two disjoint paths. Both approaches find working and backup control paths of minimum length to enable fast and efficient control-path switching between primary and backup paths. But large failures, which may affect both primary and backup paths, are not considered as this is a protection strategy and does not provide a solution for fast restoration or determine the paths considering failures and multi-stage recovery.

To achieve fast restoration of control traffic after failures, local-rerouting protection methods are addressed in [1], [2], and [12]. These schemes enable switches to locally react to failures and to redirect control traffic to controllers by using a pre-set backup outgoing link. In OpenFlow, a group table capability named "fast-failover" specifies alternate ports to be used in case of failures, so it allows local rerouting [13]. Reference [14] also discusses local fast failover mechanisms to immediately restore "connectivity" in data plane without interaction with controller. A similar approach is introduced in [15] to ensure connectivity via data-plane failover mechanisms. Local rerouting has low-overhead communications between the controller and the switches, and enables fast failure recovery. However, the following scenarios cannot be handled by local rerouting.

- If there is no local detour that can be used as a backup path, these strategies do not guarantee fast recovery.
- Even if there is an alternate path from a switch to a controller, which can be selected, local rerouting gives myopic decisions (only the switch that detects the failure does the switchover) and cannot ensure that the rerouted traffic will be forwarded correctly by the other switches (which may also be uncontrolled and may not have the proper entry in their flow tables for newly-routed traffic).
- Rerouting decisions not coming from the controller may be suboptimal [13].
- This mechanism may not be viable as it consumes high memory resources because the protection forwarding table may be installed on Ternary Content-Addressable Memory (TCAM) (which has very limited storage [16]) at each switch. As the number of failure scenarios increase, the cost increases as well. TCAM-aware local rerouting techniques for fast recovery in SDNs are discussed in [17].
- If the network policy is changed, then the pre-installed protection forwarding table should also be updated, which can produce additional communication and operation overhead between the controller and the switches [5].

Restoration introduces signaling overhead and a recovery delay, but it may be necessary when protection strategies are not enough for survivability after failures.

Other than protection strategies mentioned above, control-switch assignment and routing control paths considering restoration of control paths have not been well studied in the literature, to the best of our knowledge. Prior works have not considered multi-stage recovery effect of control paths on data paths, and control-path recovery is not a part of control-plane design optimization. They assume that control-path recovery takes place simultaneously, and all control paths can be recovered at once after failures, which we contradict and show counter examples in Section I-B. When control-path recovery takes longer, achieving fast data-path failure recovery, e.g., within 50 ms [5], [6], becomes a more challenging task.

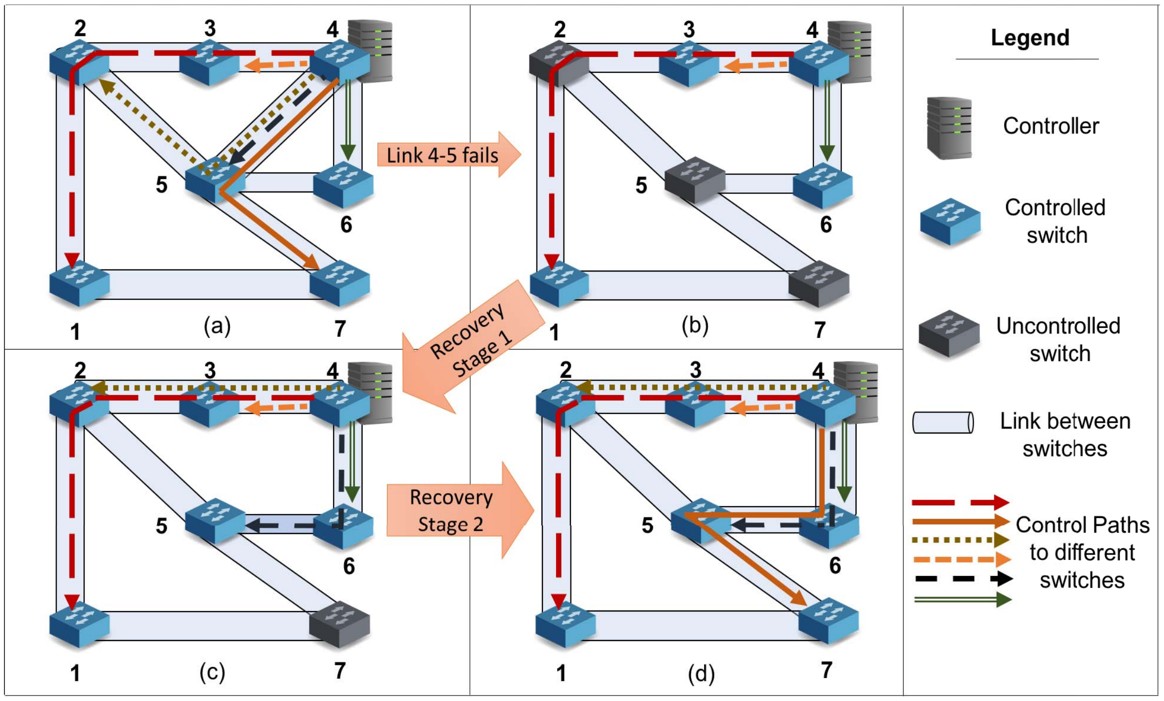


Fig. 1. An example of multi-stage control-path recovery.

B. Our Contribution

In the literature, control paths are generally routed along the shortest path to reduce propagation latency during normal operation. For similar reasons, in a multi-controller scenario, switches are assigned to the closest controller. This approach ignores failure mode of operation and the effects of multi-stage control-path recovery on data-path recovery.

In this work, we show that more effective switch-controller assignment and control-path routing strategies can reduce the control-path restoration delay, which significantly reduces the restoration time of affected data paths. We propose a novel Recovery-Aware Switch-Controller Assignment and Routing (RASCAR) approach that determines controller assignment and control-path routing by minimizing the number of control-path recovery stages after a failure out of a set of failures of interest.

Our main contributions are three-fold:

i) In a single-controller scenario, given a set of failures (each of which can consist of multiple network equipment), we propose a control-path routing strategy that minimizes the number of control-path recovery stages after any given failure.

ii) In a multi-controller scenario, we propose a strategy to jointly optimize switch-controller assignments and control-path routing to minimize the number of recovery stages.

iii) We analyze the impact of number of controllers on the number of stages needed for control-path restoration and data-path restoration.

II. Problem Statement and Assumptions

Since a physical link or a switch can support more than one control path, the failure of a link or of a switch may lead to the failure of multiple control paths simultaneously or near simultaneously [18]. After physical failures, new control paths should be restored to re-establish disrupted control paths. But, during restoration, flow entries that are necessary to establish the new paths can only be installed at switches with an active control path. Note that we refer to switches with an active control path as *controlled switches* and to switches that are not supported by an operational control path as *uncontrolled switches*. To establish a path, a controller needs to send a flow-installation message to all the switches on the path. Thus, disrupted control paths can only be restored by employing controlled switches, i.e., all the switches on a new control path have to be controlled. Therefore, in cases where a path with all controlled switches does not exist to reroute a control path, restoration of disrupted control paths may take multiple stages. We define a *stage* as a recovery step, where all control paths that can be recovered using the controlled switches available at that stage are recovered. After each stage, the number of controlled switches increases until the final stage, where the recovery is completed (i.e., all switches in the network become controlled).

To better illustrate multi-stage recovery, we demonstrate a 7- node (i.e., switch) network in Fig. 1, where the controller is located at Node 4. In Fig. 1(a), the control paths are routed from Node 4 to relevant nodes over the shortest paths. Note that the link between Nodes 4 and 5 supports three control paths, namely those to Nodes 2, 5, and 7. If this link fails, all these switches become uncontrolled as shown on Fig. 1(b), and new control paths should be established. The new control paths, to re-gain control of Nodes 2, 5, and 7, might be routed over the paths 4-3-2, 4-6-5, and 4-6-5-7, respectively. The former two paths can be routed immediately in the same stage as shown in Fig. 1(c), because all the switches on the paths are controlled switches and they are independent from each other.

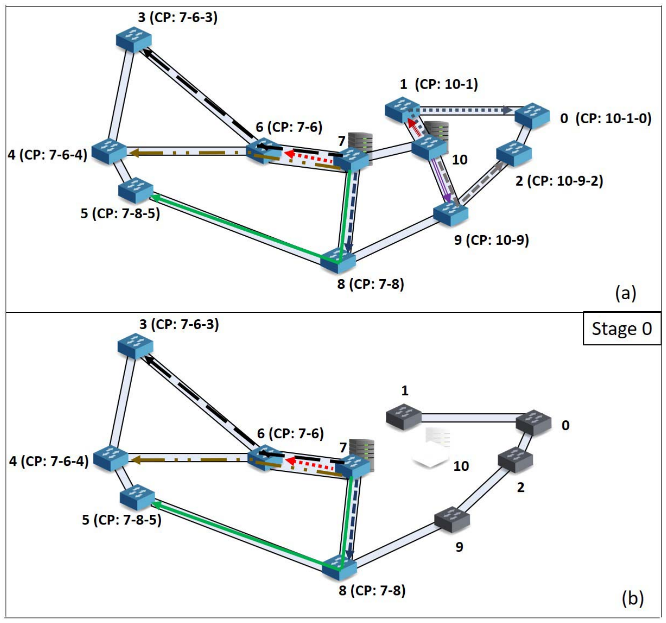


Fig. 2. A multiple-controller scenario.

However, the latter path (4-6-5-7) cannot be established before Node 5 regains control. So, it has to be routed in a later stage as shown in Fig. 1(d). Note that this is true for other alternate routings of Node 7's control path, because either Node 2 or 5 (or both) will be on the path. Hence, if the scenario is in-band (control and data traffic share the same infrastructure), the data paths originated from/destined to Node 7 have to wait for two recovery-stage durations.

To explain the necessary and sufficient condition for restoration, we define *a fully-controlled path* as a path where all nodes on the path are controlled, and *a fully-controlled node* as a controlled node that has *a fully-controlled path* to a controller. It is important to note that adjacency to at least one controlled node is not always sufficient for immediate restoration of an uncontrolled node. This is the case of Node 7 (one of its neighbors, Node 1, has always been controlled, but not fully-controlled). The necessary and sufficient condition for recovery of an uncontrolled node in the next stage is to be neighbor to at least one *fully-controlled node*. In our example, Node 1 is not a fully-controlled node, because there is no fully-controlled path between Node 1 and the controller.

Surely, *a recovery-aware routing* during the initial set up of control paths may reduce the number of recovery stages required. For instance, if the initial control-path setup in Fig. 1(a) is altered by only routing the control path of Node 7 over the path 4-6-5-7, then, after *any* single-link failure, the recovery would require only one stage.

In a multiple-controller scenario, we should also study the switch-controller assignment problem as a recovery-unaware switch-controller assignment and routing problem, which may result in an increased number of recovery stages for control path restoration, especially for large-scale failures.

In Fig. 2, we consider a scenario where there are two controllers placed on the 11-node US-wide Abilene Network [19]. The switches are assigned to the closest controller, and control paths are routed over shortest paths as shown in Fig. 2(a). On the labels at each node, the control path is written in parenthesis. Controllers are located at Nodes 7 and 10.

When node 10 fails (together with its controller and the links connected to it), Nodes 0, 1, 2, and 9 become uncontrolled as shown in Fig. 2(b). We refer to this initial stage as Stage 0. The controller at Node 7 will connect these nodes in multiple stages as shown in Fig. 3. We observe that it takes 4 stages to recover all control paths (in Stage 1, only Node 9 become controlled; in Stage 2, only Node 2 become controlled; etc.). This will induce a long control-path recovery time for the data paths with disrupted connections. However, an intelligent recovery-aware switch-controller assignment and routing mechanism can significantly reduce the number of stages required to recover the control paths, as will be discussed in the next section.

III. RASCAR: Recovery-Aware Switch-Controller Assignment and Routing

Above, we showed how control-path routing and switch-controller assignment may affect the recovery time of control paths after a failure. Since the number of stages is a key parameter that affects the control-path recovery duration, we propose the Recovery-Aware Switch-Controller Assignment and Routing (RASCAR) algorithm to accelerate the restoration of control paths after any failure among a set of failures of interest. Given a network topology where some controllers are already placed, RASCAR assigns switches to controllers and routes control paths by considering post-failure scenarios. Figure 4 shows RASCAR's solution for the same scenario shown in Fig. 2. The assignment of switches to the controllers and routings of control paths are shown in Fig. 4(a).[1] After failure, Nodes 1, 4, 5, and 9 become uncontrolled in Fig. 4(b).

Figure 5 shows that, using RASCAR, the number of recovery stages is only two, while it was four for the recovery-unaware approach (Fig. 2). In the first stage, Nodes 4, 5, and 9 become controlled; and in Stage 2, Node 1 is connected to the controller (Fig. 5). Observe that, although the number of control paths affected by the failure is the same in both approaches, with RASCAR, the disconnected nodes are distributed over a smaller uncontrolled (isolated) island (a region consisting of only uncontrolled nodes), allowing more control paths to be recovered simultaneously. Since the maximum number of recovery stages equals the number of uncontrolled nodes, simultaneous recovery of control paths can reduce the number of stages.

Note also that in this work we have not considered controller placement (but only controller-to-switch assignment), hence the location of the controller is considered as given.

In this work, the recovery of control paths at each stage is a deterministic process, which we call *basic recovery strategy*. At each stage, all uncontrolled nodes that have a fully-controlled neighbor in the previous stage are connected. Hence, using this strategy, at each stage, we reconnect the

[1]In this work, we assume that a switch can be controlled by any controller. In practice, there might be restrictions on this generalization and these restrictions can be adapted to our approach.

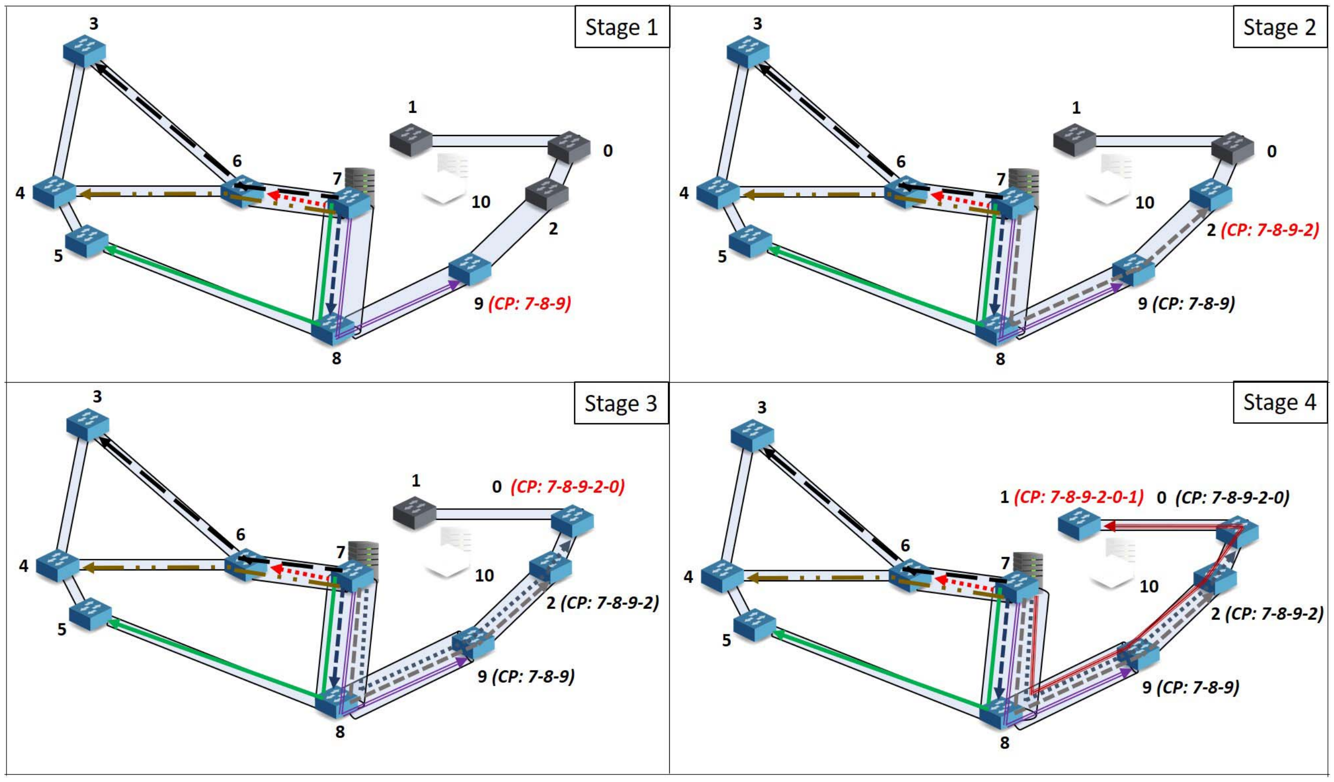


Fig. 3. Control-Path recovery after recovery-unaware controller assignment and routing (Only new paths are labeled).

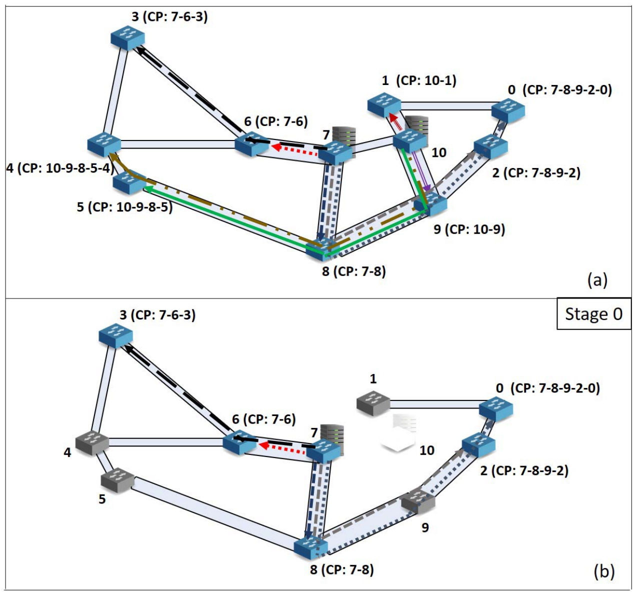


Fig. 4. A recovery-aware switch controller assignment and routing (RASCAR) example.

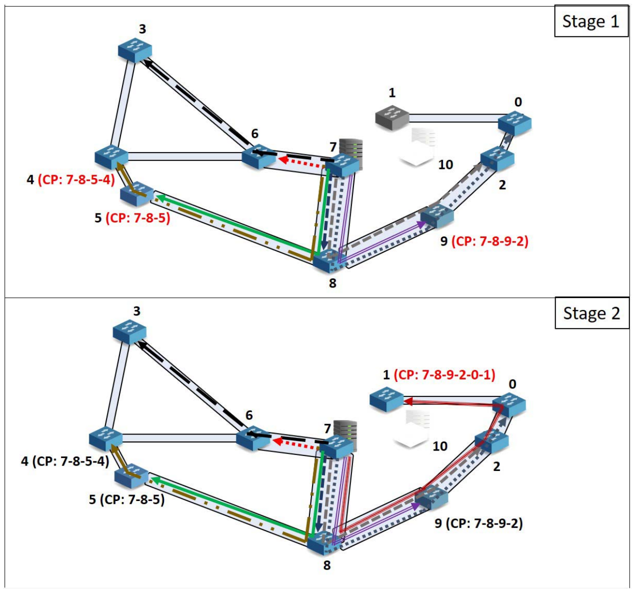


Fig. 5. Recovery stages after RASCAR.

maximum number of uncontrolled nodes possible. In Fig. 4(b), Nodes 4, 5, and 9 have fully- controlled neighbors in Stage 0 (right after the failure), so they are recovered in Stage 1. Node 1 has only one neighbor, namely Node 0; and Node 0 is a controlled but not a fully-controlled node. After the first stage (after Node 9's recovery), Node 0 becomes a fully-controlled node, and Node 1 can be recovered in the next stage (Stage 2).

Another observation is that switch-controller assignment and routing approach have a direct impact on the recovery duration of the control paths as affected control paths after a failure depend on it. The number of disrupted control paths does not give a clear picture of the number of stages required to restore all disrupted control paths. Different sets of control-path failures with same number of elements do not have the same effect in terms of the number of required recovery stages (e.g., the control loss of close nodes vs. distant nodes requires different numbers of recovery stages).

A. Restoration Time Calculation for Control and Data Paths

Given a maximum allowed restoration time (MART), we create a case study that shows the number of maximum stages that can be used to recover control paths to meet MART of data paths. To this end, we demonstrate how delay calculations are done in this section.

Control-path restoration delay is a major contributor to the data-path restoration time, MART. At each stage, at least one uncontrolled node becomes controlled until all nodes are controlled. Note that consecutive stages have to wait until previous ones are completed; hence, the sum of recovery times for all stages will be the total control-path restoration time.

Restoration time of node *n*'s control path *p* is denoted as:

$$t_n = prop_p + flow_n + proc_c \tag{1}$$

where $prop_p$ is control path *p*'s propagation delay, $flow_n$ is the flow-installation delay at node *n*, and $proc_c$ is the route-calculation delay by the controller *c*.

Since multiple control paths may be simultaneously recovered in a single stage,[2] total time consumed by stage *i*, where $i \in \{1, 2, \ldots, \text{max \# of stages}\}$, is calculated as follows:

$$D_i = \max_{N_i} t_n \tag{2}$$

where N_i is set of uncontrolled nodes recovered in stage *i*.

Total control-path restoration time of all uncontrolled nodes after a failure is:

$$D = \sum_i D_i \tag{3}$$

Restoration time for a data path that can be recovered after stage *k* is:

$$T_p = \sum_k D_k + d_p \tag{4}$$

where 0 < k < max # of stages, and d_p is the restoration time of data path *p*.

Numerical Example: Let us assume that MART for data paths T_p (usually relies on SLAs) is 100 ms after any node failure. Flow-installation delay varies in the range 5-30 ms [4], [20]. In this example, we assume flow-installation delay to be 10 ms and route calculation in the controller to be 10 ms [21]. We also assume that control paths are three hops on average: if a hop is 1000 km, the propagation delay is 15 ms. Thus,

$$\begin{aligned} \text{avg } D_i = \ & (15 \text{ ms propagation delay}) \\ & + (10 \text{ ms flow} - \text{installation delay}) \\ & + (10 \text{ ms route calculation in controller}) \\ = \ & 35 \text{ ms}. \end{aligned}$$

Assume that data-path recovery d_p takes 30 ms [22], [23]. Hence, from Eq. (4), $\sum_k D_k$ can at most be 70 ms to meet MART. Since each stage takes 35 ms on average, only two-stage recovery should be allowed for control-path recovery. If there exists a routing that ensures two-stage recovery after any node failure (as in Fig. 4(a)), then data paths can be recovered in 100 ms in the worst case after any node failure.

[2]The assumption of simultaneous recovery of multiple control paths may not be realistic if paths are too complex and involve additional switches. Our calculation here draws a baseline for the total control-path restoration time.

B. Problem Statement

To restore the control path of an uncontrolled node, a controller requires a fully-controlled path to the node. Below, we prove that the number of stages required to reconnect an uncontrolled node *n* is equal to the minimum number of uncontrolled nodes on any path that goes from *n* to any controller. We will use this information to solve RASCAR.

Lemma 1: With basic recovery strategy at each stage, all uncontrolled nodes can be reconnected by the minimum number of recovery stages.

Proof: By definition, basic recovery strategy recovers all uncontrolled nodes in stage *i*, if they have a fully-controlled neighbor in stage *i* – 1. This strategy, at each stage, connects the maximum number of uncontrolled nodes, hence it takes minimum number of recovery stages possible, and no other algorithm can do better.

Lemma 2: If path *p* is the shortest path between a node s_p and a controller d_p in terms of number of uncontrolled nodes, then the number of recovery stages required to make path *p* fully-controlled is equal to the number of uncontrolled nodes on path *p*, given that basic recovery strategy is used at each stage.

We denote the number of uncontrolled nodes on path *p* as B_p and the minimum number of recovery stages needed to make *p* fully connected as R_p.

Proof: We prove Lemma 2 by contradiction.

(1) If there exists a case where $B_p < R_p$, then recovery of some uncontrolled nodes on path *p* is not supported by path *p* to be recovered. Nodes on path *p* are denoted by $V_p = \{v_1, v_2, \ldots, v_m\}$, where *m* is the number of nodes on path *p*. Here, v_1 is the controller and v_n is the switch, where $n = 2, 3, \ldots, m$. The first uncontrolled switch on the path to controller v_1 can be connected to the controller v_1 in one stage as it has a fully-controlled path to v_1. The second uncontrolled switch waits at most one stage because at least one uncontrolled node has to become controlled. Hence, the last uncontrolled node $v_{|Bp|}$ has to wait at most $B_p - 1$ stages. v_{B_p} becomes connected at stage B_p. After v_{Bp} is controlled, all nodes on path *p* become controlled, and the path becomes a fully-controlled path.

(2) If there exists a case where $B_p > R_p$, then *p* is not the path that has the minimum number of uncontrolled nodes from source s_p to controller d_p, because this requires that at least one uncontrolled node is connected to the controller before it can be recovered on path *p*. This is only possible if that uncontrolled node has a path to the controller with fewer uncontrolled nodes, so that it could be restored sooner following that path. For example, assume that path *p* has the following nodes: $V_p = \{v_1, \ldots, v_m\}$, where v_1 is the controller, v_m is the source node, and v_i and v_j (1<*i*, *j*<*m* and $i \neq j$) are uncontrolled. According to Lemma 2, the path

between v_1 and v_m should be fully-controlled in exactly two stages. Since this path has uncontrolled nodes on it, the minimum number of stages required for it to be fully controlled is one, if all uncontrolled nodes can become connected in a single stage. Since $B_p > R_p$ in this case, $1 \leq R_p < 2$, then $\mathrm{R_p} = 1$. Therefore v_j does not wait for the recovery of v_i's control path, because it has another path to the controller where it is the only uncontrolled node. Then, starting from v_m, there exists another path stemming from v_j destined to v_1 which has only one uncontrolled node, v_j. Hence, p is not the best path if $\mathrm{B_p} > \mathrm{R_p}$.

Therefore, $\mathrm{B_p}$ has to be exactly equal to $\mathrm{R_p}$.

Remark 1: Since we need a fully-controlled path to restore a disrupted control path and making a path fully controlled requires the same number of stages as the number of uncontrolled nodes on that path, then the number of stages required to reconnect an uncontrolled node to a controller is equal to the number of uncontrolled nodes on a path with minimum uncontrolled nodes.

We define a recovery stage as a stage that consists of restoring control paths of uncontrolled nodes that are reachable from the controller by fully-controlled paths. After each stage, more fully-controlled paths become available, until all control paths are restored.

As we try to minimize the maximum number of recovery stages required to reconnect an uncontrolled node, it is essential to define the set of failures that needs to be considered. We denote failures as $Y = \{y_1, \ldots, y_m\}$, where Y is the set of failure sets, where each failure set y_i may include one or more failed network elements. Most common failures of interest to network operators are single point of failures (SPOF) as they occur as part of daily operation. But multiple failures will also be considered. Although large-scale failures caused by disasters are rare, they affect multiple network elements due to their span, hence they are very detrimental to network operations.

Remark 2: Minimum number of recovery stages required to reconnect an uncontrolled node *n* with a controller after failure *y* is denoted as π_y^n. *K* is defined as the maximum number of stages required to recover any disrupted control path after any failure $y \in Y$, and it is calculated as follows:

$$K = \max_y \left[\max_n \pi_y^n \right] \tag{5}$$

K is an important metric, which gives a close estimate on how much time is needed to restore a data path in the worst-case scenario after any failure considered.

Given the network topology, controllers' locations, and set of failures, RASCAR finds a control-path routing that minimizes the number *K* (the maximum number of stages required to recover all control paths after any given failure set).

C. Problem Construction

To achieve RASCAR's objective, we constructed the problem as follows:

- Given the network topology, controllers' locations, and set of failures, find a switch-controller assignment and a control-path routing with the objective to minimize resource consumption such that, after any given failure set, any control path can be recovered within *K* stages, where *N* is the number of nodes in the network and $K \in \{1, 2, \ldots N-1\}$.
- Starting from $K = 1$ and increasing it by one, solve the previous problem until a feasible solution is found.
- Find smallest *K* that has a feasible solution. Call this *K**.
- As a result, find a routing that requires minimum number of stages after any failure and consumes minimum bandwidth resources for recovering within *K** stages.
- Finding a solution *K** indicates that, after a failure in the given set, there exists a restoration path for all disrupted control paths which has at most *K** uncontrolled nodes.

Algorithm 1: Neighbor-Disjoint Path Routing (DR)

```
Input: Sort all nodes $v \in V$, where $v \neq v_c \forall c \in C$, based on
the node's node connectivity in descending order, and generate
set of nodes $W$ out of the sorted nodes. $\varepsilon$ is an arbitrarily small
number.
1. for each link $e$ where $e \in E$
     for each node $v$ where $v \in V$
         $\partial_e^v = \varepsilon$;

2. for each node $w$ where $w \in W$
     define solutionSet; // which contains shortest
     paths from node $w$ to all $c \in C$
     for each controller $c$ where $c \in C_w$ // $C_w$ can be
         single or multiple controllers
         Path path = find_shortest_path($w$,$c$);
         solutionSet.add(path);

     solutionSet.sortbyPathCost();
     Provision $w$ on path $p$ = solutionSet.get(0);
     for each link $e$ where $e \in E_p$
         for each node $v$ where $v \in V$ // update link costs.
             $\partial_e^v = \begin{cases} \partial_e^v + 1 & if\ x_w^v = 1 \\ \partial_e^v & otherwise \end{cases}$
```

IV. Mathematical Formulation and Heuristics

Here, we formulate RASCAR into an ILP mathematical model. Then, we propose a heuristic algorithm to solve the problem for larger instances as ILP takes too long.

A. ILP

RASCAR formulation provides the assignment and routing while minimizing network resource usage for a given K, if feasible. To find the minimum K, we run the ILP starting from $K = 1$ and increase *K* by one, until a feasible solution is found.

Input Parameters:

- *G (V, E):* Network topology where *V* is set of nodes and *E* is set of directed links.

- *C:* Set of controllers where the node on which controller $c \in C$ is located is denoted by v_c ($v_c \in V$).
- *Y:* Set of failures, where the set of links that are members of failure y is denoted by $E_y (E_y \subset E)$. All disasters are represented as link failures, i.e., to represent a node failure, all links originated from or destined to that node is added to E_y.
- *K:* Maximum number of stages required to recover any control path after any failure $y \in Y$.
- *M:* A very large number.
- $P = \{p | p =< s_p,\ d_p,\ L_p,\ V_p >\}$*:* Set of k-shortest paths where each path is defined by four tuples:
    - $s_p \in V$ and $d_p \in C$*:* source and destination of path *p*, respectively;
    - L_p*:* number of links on path *p*; and
    - V_p*:* set of nodes on path p ($V_p \subset V$).
- $T = \{t | t =< s_t,\ d_t,\ P_t >\}$: Set of control paths to be routed, where s_t is the source, and d_t is destination nodes where $d_t \in C$, and $P_t \subset P$ is the set of possible paths that can be used to route control path *t*.
- $U_y^{t,p} \in \{0,1\}$: indicator to show whether a candidate path $p \in P_t$ for control path t is unaffected by failure y. If unaffected, then $U_y^{t,p} = 1$, otherwise 0. Note that this parameter is given.

Integer Variables:

- $Z_p^t \in \{0,1\}$: 1 if path $p \in P_t$ is used for control path $t \in T$.
- $A_y^t \in \{0,1\}$: 1 if control path $t \in T$ is not affected by failure $y \in Y$.
- $S_y^{p,t}$: Number of uncontrolled nodes on path $p \in P_t$ after failure $y \in Y$.
- $F_y^{p,t} \in \{0,1\}$: 1 if number of uncontrolled nodes on path p $\in P_t$ is less than *K*.
- $N_t^y \in \{0,1\}$: 1 if there exists at least one path that has at most *K-1* uncontrolled nodes on it after failure *y*.

Objective:

$$Minimize \sum_{t \in T} \sum_{p \in P_t} Z_p^t . L_p \tag{6}$$

The above objective minimizes total resource usage of all control paths. We define resource usage as the total number of links used by control paths throughout this work.

Constraints:

$$\sum_{p \in P_t} Z_p^t = 1\ \forall t \in T \tag{7}$$

$$A_y^t = \sum_{p \in P_t} Z_p^t . U_y^{t,p}\ \forall t \in T,\ \forall y \in Y \tag{8}$$

$$S_y^{p,t} = (L_p - 1 - \sum_{v \in V_p} A_y^v) + (1 - U_y^p).M\ \forall t \in T,\ \forall y \in Y,\ \forall p \in P_t \tag{9}$$

$$1 - F_y^{p,t} \geq \left(S_y^{p,t} - K\right)/M \tag{10a}$$

$$1 - F_y^{p,t} \leq \left(-1 * \left(S_y^{p,t} - K\right)\right)/M\ \forall t \in T,\ \forall y \in Y \forall p \in P_t \tag{10b}$$

$$N_y^t \leq \sum_{p \in P_t} F_y^{p,t} \tag{11a}$$

$$N_y^t \geq \sum_{p \in P_t} F_y^{p,t}/M\ \forall t \in T,\ \forall y \in Y \tag{11b}$$

$$N_y^t \geq 1 - A_y^t\ \forall t \in T,\ \forall y \in Y \tag{12}$$

Eq. (7) enforces that a switch must be connected to exactly one controller (i.e., it has a single control path).

Eq. (8) sets A_y^t by checking that control path *t* is "up" after failure *y*, if path $p \in P_t$ reserved for *t* (indicated by Z_p^t) is up after failure *y*.

Eq. (9) sets $S_y^{p,t}$, the number of uncontrolled nodes on path $p \in P_t$ after failure *y*, that is calculated by subtracting the number of controlled nodes from total number of nodes on path *p*. If path *p* is failed, then it cannot be used to reroute any control paths, so Eq. (9) sets $S_y^{p,t}$ to a very large number.

Eq. (10a) sets $F_y^{p,t}$ to 1 if the number of uncontrolled nodes on path $p \in P_t$ (indicated by $S_y^{p,t}$) is less than *K*. This shows if path *p* can be used to restore control path *t* with fewer than *K* stages after failure *y*.

Eq. (11a) sets N_y^t to 1 if $F_y^{p,t}$ is 1 for at least one path $p \in P_t$ to determine if control path *t* can be recovered in *K* stages after failure *y*.

Eq. (12) ensures that, for all nodes that lose their control paths, there exists at least one path to the controller, where at most *K* nodes are uncontrolled after failure *y*. If *t* becomes unreachable after failure *y*, then the constraint ignores that case and does not force N_y^t variable to become 1.

B. Heuristic

The ILP achieves optimal solutions (for the paths allowed), but has high time complexity. For instance, in our numerical examples, it takes several minutes on a Macbook Pro with 2.6 GHz Intel Core i7 Quad-Core by using CPLEX 12.0 optimization tool, for topologies with fewer than 50 nodes. The run time of our ILP solution takes too long for larger instances, so we also develop a faster heuristic for switch-controller assignment and routing, called Neighbor-Disjoint Path Routing (DR), whose details can be found in Algorithm 1.

RASCAR reduces the colocation of nodes that become uncontrolled to avoid large uncontrolled islands. Intuitively, DR tries to route a node's control path disjoint from control paths of its neighboring nodes to reduce the size of uncontrolled islands after failures. Yet, finding disjoint paths for all control paths is not always possible. Therefore, we define a *disjointness cost* parameter to route a node's control path disjoint from its neighbors' whenever possible. Disjointness cost of link *e* for node *v*, denoted as ∂_e^v, indicates the amount of control paths of neighboring nodes of *v* that traverse link *e*.

Since the number of neighbors' control paths that traverse a link may be different for each node, each link might have a different disjointness cost for each node. A path's cost is total cost of the disjointness cost of each link on that path for the node of interest.

DR provisions control paths sequentially. It first sorts all nodes, which are not colocated with a controller, based on their degree in descending order. Hence, high-degree nodes' control paths are routed before low-degree ones. Low-degree nodes are more vulnerable to failures, and it is more difficult to restore their control paths after failures as alternate paths are limited. Thus, provisioning them after other control paths gives better visibility of the network for those control paths.

Neighbor information is given as an input and denoted as $x_w^v \in \{0,\ 1\}$. It is 1 if node $v \in V$ is neighbor of node $w \in V$.

A set of controllers that can control node *v* is denoted as C_v. Note that a strict switch-controller assignment can be regulated by including only the designated controller for *v* in set C_v. Also, switch-controller latency requirements can be assured by including only the controllers located within a certain distance from node *v* in C_v. DR selects the optimal solution from a set of controllers (i.e., selects the controller such that the control path has the lowest disjointness cost). The shortest paths based on disjointness cost selected are denoted as $P = \{p :\ < E_p >\}$, where $\mathrm{E_p}$ is set of links on path *p*.

Step 1 in Algorithm 1 assigns a small value to all links to avoid unnecessarily-long paths. Step 2 finds possible shortest paths between a node to all possible controllers that can control the node. Then, it selects the shortest path with the lowest cost and provisions the control path on it. After provisioning a control path, link costs are updated. This procedure is repeated for all nodes that are not collocated with a controller, starting from the high-degree nodes.

Complexity of the heuristic equals $|V|\times (|E|+ |V| \times|\log|V|)$ because it runs shortest-path algorithm, which has complexity of $(|E| + |V| \times \log |V|)$, for finding each node's control path.

V. Illustrative Numerical Examples

To quantify the impact of our switch-controller assignment and routing algorithms, we run extensive simulations and evaluate our algorithms on a number of real network topologies [23]. This section summarizes our simulation results.

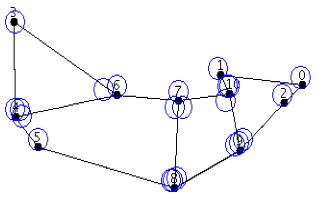
A. Simulation Setup

Performance metrics. Since the aim is to restore disrupted data paths as quickly as possible while using minimum network resources, we demonstrate the improvement achieved by RASCAR using the following two performance metrics.

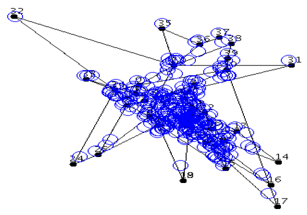
Percentage of restored datapaths (PRD): indicates the percentage of data paths that are restored at each stage. This metric also gives us the number of recovery stages for each scheme, which is an indicator for restoration time. Note that restoration time calculations introduced in Eqs. (1)-(4) depend on many factors (propagation factors, flow installation delays, processing times) and are very complex to make realistic estimation. In Section III-A, we provided an example calculation steps for restoration times. Number of restoration steps is directly proportional with restoration times, and hence for simplicity, we focus on number of recovery stages rather than exact restoration time.

Total Resource Usage for Control Paths (RC): shows total resource usage of all control paths in terms of number of links.

To measure these metric, our simulation includes two phases.



(a) Abilene: All 22 disasters with 100-km radius.



(b) GEANT: All 312 disasters with 100-km radius.

Failure Types	Abilene	GEANT
Single link	14	61
Single node	11	40
r = 100	22	429
r = 200	30	642
r = 400	43	887
r = 800	86	1128
r = 1600	64	1287

(c) Number of failures according to failure type.

Fig. 6. All possible disasters with 100-km radius with a distinct affect.

In the first phase, given a set of failures, we find switch-controller assignment and routing that minimizes K (worst-case number of restoration stages) as discussed in Section IV.

In the second phase:

1. We apply a failure taken from a given set of failures. Some control paths and data paths are down. Recovery stages begin.

2. At the beginning of a stage, we recover data paths using the undamaged and controlled portion of the network. We calculate the number of recovered data paths at this stage.

3. After restoring data paths that can find a route, we recover control paths that are reachable from the controller with a fully-controlled path. When a stage is completed, a larger portion of the network becomes controlled.

If there exists any disrupted data or control path, then we need more stages, so repeat Steps 2 and 3.

4. For each failure in the set, we reset the network to its initial pre-failure state, and start from Step 1 again.

At the end of these steps, we know how many data paths are recovered at the end of each stage, averaged over all possible failure scenarios.

Topologies. We present the results for Abilene (11-node) and GEANT (40-node) topologies (Fig. 6(a) and 6(b), respectively), but the trends and tradeoffs seen for these networks have been confirmed on another 30 topologies taken from [15].

Traffic model. We assume uniform traffic distribution in the network for data traffic, meaning that each route carries one unit of traffic using shortest-path routing. To illustrate resource consumption of control and data paths, we assume average data and control paths use three links.[3] In Abilene, we route $11 \times 10 = 110$ data paths on the network (one path from each node to each other node). Then, resource usage of total data traffic in terms of link usage is 330. Also, there are

[3]Based on separate simulations that we conducted for uniform traffic distribution.

TABLE I
COMPARED ALGORITHMS

	Routing
SR	Shortest-path routing
DR	Disjoint-path routing (Algorithm 1)
RASCAR	Proposed approach (ILP)
	Controller Assignment
SA	The nearest controller is assigned for each switch.
DA	With Algorithm 1's assignment, given multiple controller options, DA routes switches to a controller where the path between them has the minimum *disjointness cost*.
RASCAR	Switch-controller assignment result of ILP.

10 nodes in Abilene in a single-controller scenario (as one of the nodes is the controller), then total control resource usage is $3 \times 10 = 30$.

Disaster model. Figure 6 shows the networks and failures considered in this work.

Our simulations take a set of failures as input. The set of failures that we have considered are:

Single Point of Failures (SPOF)

1. Set of all possible single-link failures (e.g., # of failures in the set is 14 for Abilene network).

2. Set of all possible single-node failures (e.g., # of failures in the set is 11 for Abilene network).

When a disaster occurs, multiple network elements located in the disaster zone (usually defined within a certain radius) fail, and the connections that are traversing the damaged zone need to be reprovisioned. Hence, we model disasters as a circular region, where the center of the circle is the epicenter and radius is the distance of its span.

For instance, the damage of an earthquake may span up to 96 km [24]. Figure 6 shows all possible 100-km radius disasters with a distinct effect for Abilene and GEANT networks. Figure 6(c) shows the number of distinct disasters when the radius r is 100, 200, 400, 800, and 1600 km. For instance, the set of all possible 100-km disasters (# of failures in the set) is 22 for Abilene network.

B. Compared Schemes

RASCAR performs switch-controller assignment and routing. Switch-controller assignment is relevant only for multiple controllers. Since both aspects affect the number of recovery stages, we start by assuming single controller and we observe the control-path routing's effect in isolation. Routing strategies that will be compared with single controller are stated in the upper part of Table I: RASCAR, DR, and SR.

Then, in a multi-controller scenario, we show the performance of different routings under different controller-assignment strategies. Assignment strategies to be compared with single controller are: SA, DA, and RASCAR (Table I). Since assignment strategies cannot be compared independently from the routing, we compare routing algorithms with various assignment to measure the assignment's impact. Combined routing and assignment strategies to be compared are: SR/SA, SR/RASCAR, DR/SA, DR/DA, DR/RASCAR, and RASCAR/RASCAR. We do not evaluate all possible routing/assignment combinations, because some combinations are not applicable (for instance RASCAR cannot be solely used for routing, thus RASCAR/DA and RASCAR/SA are not possible).

C. Results

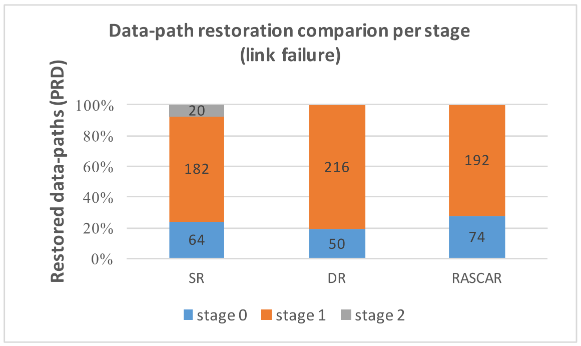
1) Single Controller: Figure 7 compares the amount of data paths recovered at each stage and number of stages needed to recover data paths using SR, DR, and RASCAR in Abilene network.

We consider link, node, and disaster failures, in Figs. 7(a), 7(b), and 7(c), respectively. For both link- and node-failure scenarios, RASCAR and DR recovered all data paths in one stage (i.e., the initial stage, stage 0, right after the failure, before restoring any control paths), whereas SR takes one additional stage. For all disasters with 100-km radius (Fig. 7(c)), RASCAR performs the best with at most two stages, but less than 1% of data paths must wait for the second stage, while the remaining ones are recovered in the first stage. For 20% of disrupted data paths, SA must wait till stages 3 and 4 for full recovery. In terms of resource usage, note that RASCAR performs as well as the shortest path (SR) for single node and link failures, however its resource consumption increases under disaster failures (Fig. 7(d)). RASCAR uses 31 links while SR utilizes 19 links under 100-km radius disaster failures. Note that, although RASCAR's resource consumption has increased more than 50% compared to SR's control-path resource usage, considering total resource consumption of the network (control path (average 30 links) and data paths (average 330 links)), the increase is less than 1% (i.e., 12 more links are used for RASCAR when total link usage is 330 links in the network).

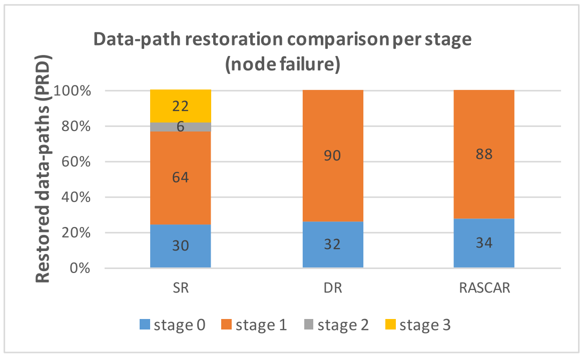
In Section IV, we explained that, in the process of finding a routing with minimum K (number of recovery stages in the worst case), we set K to 1 and try to find a routing. If the solution is infeasible, we increase the value of K and retry until we find a solution. However, the lower the K, the higher the resource consumption of control paths. For rare failures, the network operator may not want to spend more resources to minimize K, but they may want to provide a loose K value that is tolerable. In this case, the objective becomes finding a routing with K that minimizes resources.

In Fig. 8, we compare resource consumption of control paths under disaster-failure scenario with different radius and show how much more resources are used compared to shortest-path (SR) approach. Only for disasters with 25-km radius, all control paths can be recovered in a single stage. For disasters with r = 50, 100, and 200 km, the first feasible solution found using RASCAR is when K is 2; and compared to shortest-path (SR) resource consumption, our solution uses 50% more links. The increase in link usage is from 20 to 30, less than 1% of total resource usage in the network while data-path link usage is around 330. Therefore, the increase in resource usage is not much considering the total network traffic.

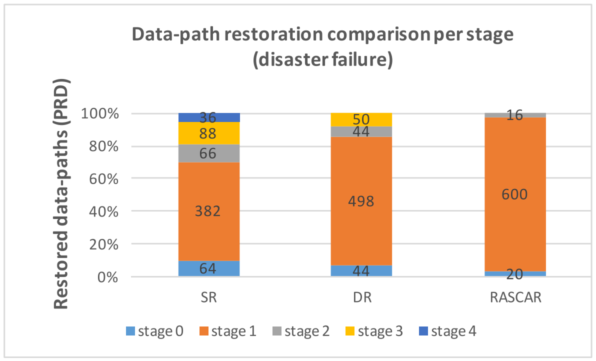
Also, note that relaxing $K = 2$ to $K = 3$ reduces the resource consumption difference with SR drastically. When K is set to 4, RASCAR consumes the same amount of resources



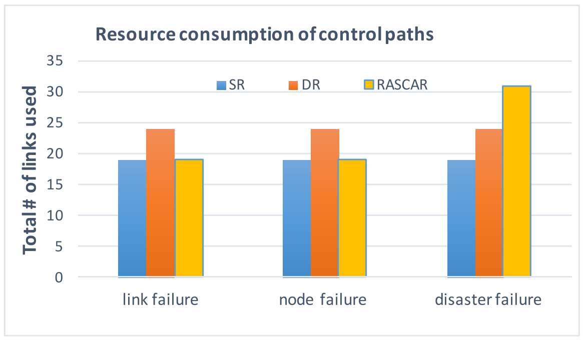
(a) Under all possible link failures.



(b) Under all possible node failures (except the controller).



(c) Under all disasters with 100 km radius.



(d) Resource usage comparison.

Fig. 7. Recovered data paths per stage for different approaches and failures.

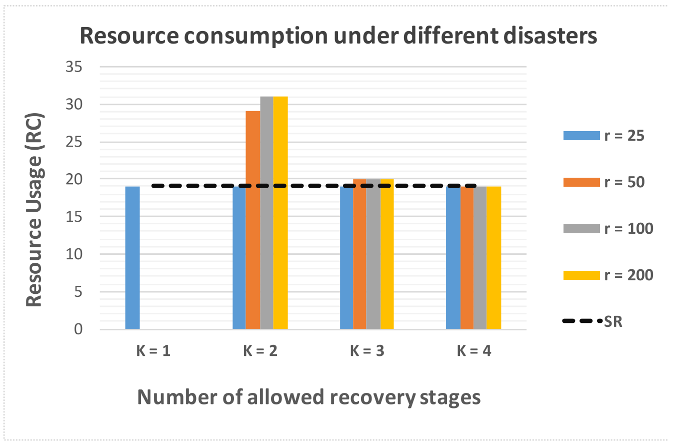


Fig. 8. Abilene: Resource consumption of RASCAR under different K and different-sized disaster.

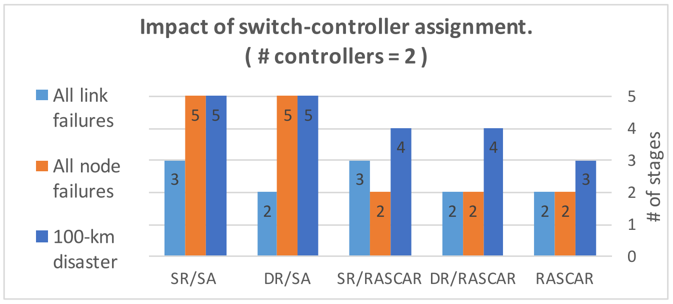


Fig. 9. GEANT: Comparison of different routing and assignment combinations in terms of number of recovery stages.

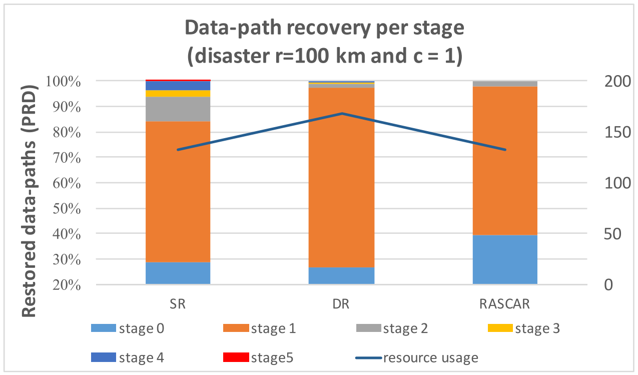
as SR. Thus, larger K decrease the resources needed for control paths.

2) Multiple Controllers: To show the effect of multiple controllers, we need a larger network. Hence, we use GEANT network for this study. In Fig. 9, we see the number of stages required to recover data paths under different assignment and routing approaches. In general, assignment becomes critical when we also consider controller failures, as all the nodes connected to a failed controller become uncontrolled, and will reconnect to the active controllers. If all nodes in a region are connected to a single controller c, then after c fails, there will be several collocated uncontrolled nodes, so the recovery takes longer, as in Fig. 2(b). As a first observation, under link failures, controller-switch assignment does not have an impact on the number of stages. SR/SA and SR/RASCAR (both 3 stages); DR/SA and DR/RASCAR (both 2 stages) perform the same under link failures.

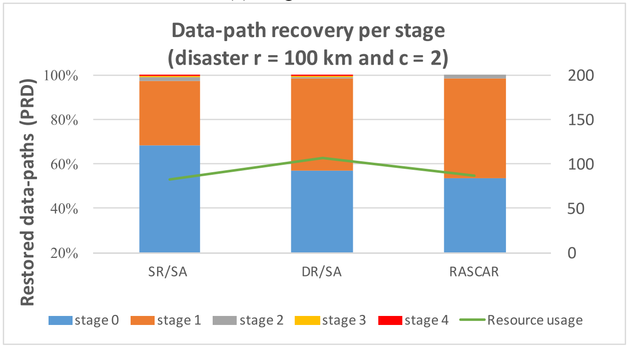
Assigning switches to the nearest controller (SA) performs worst, independent of the routing (see SR/SA and DR/SA), when controller failures are considered (i.e., for all node failures and for 100-km disaster), because uncontrolled region is larger. RASCAR's assignment improves performance of SR and DR significantly. Both SR and DR perform very close to RASCAR assignment and routing.

In Fig. 10, we show the effect of having multiple controllers in a network on the recovery of data paths considering disaster failures. Multiple controllers do not improve RASCAR's performance, but both SR and DR benefit from it.

Note that the placement of controllers is given for our algorithm. While running our simulations, we vary the location



(a) Single Controller



(b) Multiple controller: Two in this case.

Fig. 10. GEANT: Effect of multiple controllers on recovery under disasters (r = 100 km).

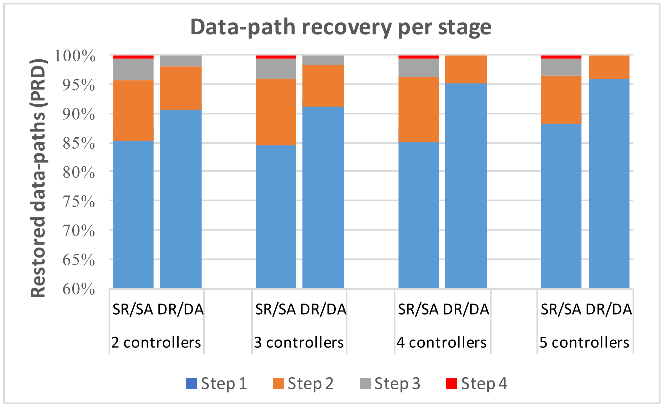


Fig. 11. GEANT: Effect of multiple controllers on recovery under link failures.

of controllers for each possible location set. Our results provided here are the average outcomes of those simulations. The effect of controller location on our approach has not been discussed due to space limitation, however the results show small standard deviations for percentage of restored data path and resource consumption.

In Fig. 11, we compare the performance of SR/SA and DR/DA schemes as the number of controllers increases. DR/DA scheme takes less recovery stages to recover fully as it distributes the control paths across the network and balances controller-switch assignments. In general, controller placement affects the performance of both schemes as well as topology (connectedness of the nodes). DR/DA scheme gets affected less from controller placement as it prevents assigning switches to closest controllers and having large disconnected islands.

VI. Conclusion

In this study, we showed that the current recovery-unaware methods for switch-controller assignment and routing approach may result in high recovery times of both control and data paths in case of both single point of failures and large-scale disaster failures in SDN. We incorporated control-path restoration after failures into controller assignment and control-path routing decisions. Our solution significantly reduces the control-path and data-path recovery times with a slight increase in resource usage consumed by control paths. We also showed that, in an SDN with multiple controllers, switch-controller assignment impacts the recovery performance of control paths more than control-path routing. Also, we observed that increasing the number of controllers significantly improves the performance of shortest-distance controller assignment (SA) and shortest-path routing (SR) algorithm, whereas with RASCAR the same performance can be achieved by less number of controllers.

Acknowledgment

The authors gratefully acknowledge the anonymous reviewers for their constructive criticisms which improved the paper.

References

[1] Y. Hu *et al.*, “Control traffic protection in software-defined networks,” in *Proc. IEEE GLOBECOM*, Austin, TX, USA, Dec. 2014, pp. 1878–1883.
[2] Y. Zhang, N. Beheshti, and M. Tatipamula, “On resilience of split-architecture networks,” in *Proc. IEEE GLOBECOM*, Houston, TX, USA, Dec. 2011, pp. 1–6.
[3] N. Feamster, J. Rexford, and E. Zegura, “The road to SDN,” *Queue*, vol. 11, no. 12, p. 20, 2013.
[4] R. Bifulco and A. Matsiuk, “Towards scalable SDN switches: Enabling faster flow table entries installation,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 343–344, Aug. 2015.
[5] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, “A roadmap for traffic engineering in SDN-OpenFlow networks,” *Comput. Netw.*, vol. 71, pp. 1–30, Oct. 2014.
[6] H. Huang *et al.*, “Near-optimal routing protection for in-band software-defined heterogeneous networks,” *IEEE J. Sel. Areas Commun.*, vol. 34, no. 11, pp. 2918–2934, Nov. 2016.
[7] S. Lange *et al.*, “Heuristic approaches to the controller placement problem in large scale SDN networks,” *IEEE Trans. Netw. Service Manag.*, vol. 12, no. 1, pp. 4–17, Mar. 2015.
[8] G. Yao, J. Bi, Y. Li, and L. Guo, “On the capacitated controller placement problem in software defined networks,” *IEEE Commun. Lett.*, vol. 18, no. 8, pp. 1339–1342, Aug. 2014.
[9] D. Hock, S. Gebert, M. Hartmann, T. Zinner, and P. Tran-Gia, “POCO-framework for Pareto-optimal resilient controller placement in SDN-based core networks,” in *Proc. IEEE/IFIP NOMS*, Kraków, Poland, May 2014, pp. 1–2.
[10] 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 HPSR*, Budapest, Hungary, Jul. 2015, pp. 1–6.
[11] P. Vizarreta, C. M. Machuca, and W. Kellerer “Controller placement strategies for a resilient SDN control plane,” in *Proc. RNDM*, Halmstad, Sweden, Sep. 2016, pp. 253–259.
[12] X. Zhang *et al.*, “Local fast reroute with flow aggregation in software defined networks,” *IEEE Commun. Lett.*, vol. 21, no. 4, pp. 785–788, Apr. 2017.
[13] C. Cascone, L. Pollini, D. Sanvito, and A. Capone, “Traffic management applications for stateful SDN data plane,” in *Proc. EWSDN*, Bilbao, Spain, Sep. 2015, pp. 85–90.

[14] M. Borokhovich, L. Schiff, and S. Schmid, "Provable data plane connectivity with local fast failover: Introducing openflow graph algorithms," in *Proc. ACM HotSDN*, Chicago, IL, USA, Aug. 2014, pp. 121–126.
[15] J. Liu *et al.*, "Ensuring connectivity via data plane mechanisms," in *Proc. USENIX NSDI*, Lombard, IL, USA, Apr. 2013, pp. 113–126.
[16] A. Marsico, R. Doriguzzi-Corin, and D. Siracusa, "An effective swapping mechanism to overcome the memory limitation of SDN devices," in *Proc. IEEE/IFIP IM*, Lisbon, Portugal, May 2017, pp. 247–254.
[17] P. M. Mohan, T. Truong-Huu, and M. Gurusamy, "TCAM-aware local rerouting for fast and efficient failure recovery in software defined networks," in *Proc. IEEE GLOBECOM*, San Diego, CA, USA, Dec. 2015, pp. 1–6.
[18] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "On reliability-optimized controller placement for software-defined networks," *China Commun.*, vol. 11, no. 2, pp. 38–54, Feb. 2014.
[19] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.
[20] J. Khalid *et al.*, "Measuring control plane latency in SDN-enabled switches," in *Proc. ACM SIGCOMM SOSR*, Santa Clara, CA, USA, Jun. 2015, pp. 1–6.
[21] 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.
[22] M. Moshref, A. Bhargava, A. Gupta, M. Yu, and R. Govindan, "Flow-level state transition as a new switch primitive for SDN," in *Proc. ACM SIGCOMM HotSDN*, Chicago, IL, USA, Aug. 2014, pp. 377–378.
[23] P. Berde *et al.*, "ONOS: Towards an open, distributed SDN OS," in *Proc. ACM SIGCOMM HotSDN*, Chicago, IL, USA, Aug. 2014, pp. 1–6.
[24] T. L. Weems, "How far is far enough," *Disaster Recovery J.*, vol. 16, no. 2, p. 749, 2003.

S. Sedef Savas received the B.S. degree in computer science and engineering from Sabanci University, Istanbul, Turkey, in 2009, the M.S. degree in computer science from Koç University, Istanbul, in 2011, and the Ph.D. degree in computer science from the University of California at Davis. Her research interests include adaptability and survivability of communication networks against disasters, QoS-aware service provisioning schemes, and software-defined networking.

Massimo Tornatore (S'03–M'06–SM'13) received the Ph.D. degree in information engineering from Politecnico di Milano, Italy, in 2006, where he is currently an Associate Professor with the Department of Electronics, Information, and Bioengineering. He also holds appointment as an Adjunct Professor with the Department of Computer Science, University of California, Davis, USA.

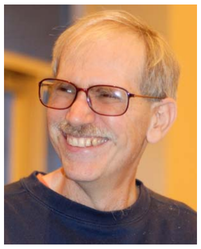
He has authored over 300 peer-reviewed conference and journal papers.

Dr. Tornatore's research interests include performance evaluation, optimization and design of communication networks, cloud computing, and privacy. He is an Editorial Board Member of *Photonic Network Communications*, *Optical Switching and Networking*, and the IEEE COMMUNICATION SURVEYS AND TUTORIALS.

He was a co-recipient of eleven best-paper awards. He is an active member of the technical program committee of various networking conferences, such as INFOCOM, OFC, ICC, and GLOBECOM.

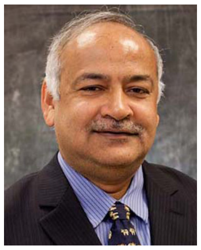
Ferhat Dikbiyik received the M.S. and Ph.D. degrees in electrical and computer engineering from the University of California, Davis, in 2009 and 2013, respectively. He is a Research and Development Manager with NormShield, a cyber security company. He was an Assistant Professor with the Department of Computer Engineering, Sakarya University, Turkey, from 2013 to 2017. His research interests include cyber risk assessment of organizations and design, development, and analysis of next-generation lightwave networks, especially survivability against and recovery from disasters.

Aysegul (Gencata) Yayimli received the M.Sc. and Ph.D. degrees in control and computer engineering from Istanbul Technical University, Turkey, in 1995 and 2003, respectively, where she was an Assistant Professor and later as an Associate Professor until she joined Valparaiso University, IN, USA. Her research interests include routing optimization, architecture and protocol design, and survivability issues in optical networks. She served in technical committees of several conferences, including ICC, Globecom, and ONDM.



Charles U. Martel received the B.S. degree in computer science from the Massachusetts Institute of Technology in 1975 and the Ph.D. degree in computer science from the University of California (UC), Berkeley, in 1980. Since 1980, he has been a Professor with the UC Davis and was one of the founders of their Computer Science Department. He has been an Emeritus Professor since 2012.

He has worked on a broad range of combinatorial algorithms, but recently is focusing on algorithms for computer networks. As a seven time world bridge champion and a member of the American Contract Bridge League hall of fame, he also has an interest in computer bridge playing programs.



Biswanath Mukherjee (S'82–M'83–SM'05–F'07) received the B.Tech. degree (Hons.) from the Indian Institute of Technology, Kharagpur, in 1980 and the Ph.D. degree from the University of Washington, Seattle, in 1987. He is a Distinguished Professor with the University of California (UC), Davis, where he has been a Faculty Member since 1987 and was the Chairman of Computer Science from 1997 to 2000. He is the Founder and the President of Ennetix, Inc., a startup company incubated at UC Davis and developing cloud-based network performance analytics and management software. He has supervised 76 Ph.D. students to completion and currently mentors eight advisees, mainly Ph.D. students. He has authored the graduate-level textbook entitled *Optical WDM Networks* (Springer, 2006). He served a 5-year term on the Board of Directors of IPLocks, a Silicon Valley startup company (acquired by Fortinet). He has served on Technical Advisory Board of several startup companies, including Teknovus (acquired by Broadcom). He was a recipient of the 2004 Distinguished Graduate Mentoring Award and the 2009 College of Engineering Outstanding Senior Faculty Award at UC Davis and also a co-recipient of 11 best paper awards from various conferences, including the Optical Networking Symposium Best Paper Awards at IEEE Globecom in 2007 and 2008. He was the General Co-Chair of the IEEE/OSA Optical Fiber Communications Conference in 2011, the Technical Program Co-Chair of OFC'2009, and the Technical Program Chair of the IEEE INFOCOM'96 Conference. He is an Editor of Optical Networks Book Series (Springer). He has served on eight journal editorial boards, most notably the IEEE/ACM TRANSACTIONS ON NETWORKING and the IEEE NETWORK. He has guest edited special issues of *Proceedings of the IEEE*, the *IEEE/OSA Journal of Lightwave Technology*, the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, and the IEEE TRANSACTIONS ON COMMUNICATIONS.