Journal of Interconnection Networks
Vol. 18, No. 4 (2018) 1850014 (17 pages)

DOI: 10.1142/S0219265918500147

World Scientific
www.worldscientific.com

Software-Defined Networking Switches for Fast Single-Link Failure Recovery

DAWEI LI[*,‡], JIE WU[†,§], DAJIN WANG[*,¶] and JIAYIN WANG[*,‖]

[*]*Department of Computer and Science, Montclair State University, Montclair, NJ, 07043, USA*

[†]*Department of Computer and Information Sciences, Temple University, Philadelphia, PA, 19122, USA*

[‡]*dawei.li@montclair.edu*

[§]*jiewu@temple.edu*

[¶]*wangd@montclair.edu*

[‖]*wangji@montclair.edu*

Received 2 April 2018
Accepted 10 December 2018

In this paper, we consider IP fast recovery from single-link failures in a given network topology. The basic idea is to replace some existing routers with a *designated switch*. When a link fails, the affected router will send all the affected traffic to the designated switch (through pre-configured IP tunnels), which will deliver the affected traffic to its destination without using the failed link. The goal of the approach is to achieve faster failure recovery than traditional routing protocols that employ reactive computing upon link failures. Software-Defined Networking (SDN) switches can serve as the designated switches because they can flexibly redirect affected traffic to other routes, instead of only to the shortest paths in the network. However, SDN switches are very expensive. Our objective is to minimize the number of SDN switches needed and to guarantee that the network can still recover from any single-link failure. For networks with uniform link costs, we show that using normal non-SDN switches with IP tunneling capability as designated switches can guarantee recovery from any single-link failure. For networks with general link costs, we find that not all single-link failures can be recovered by using non-SDN switches as designated switches; by using SDN switches only when necessary, we can reduce the total number of SDN switches needed compared to an existing work. We conduct extensive simulations to verify our proposed approaches.

Keywords: Software-defined networking (SDN); failure recovery; shortest paths; equal cost multi-path (ECMP); IP tunneling.

1. Introduction

Traditional IP networks are complex and hard to manage. It is both difficult to configure the IP network according to predefined policies and to reconfigure it

[‡]Corresponding author.

dynamically. The main reason for this difficulty is that the control plane and data plane of the IP router are bundled together. To enable flexible control and management over IP networks, various techniques have been proposed and applied over the past decades. Software-Defined Networking (SDN) [1] [2] is one of the techniques that become popular recently. SDN separates the network's control logic from the underlying routers and switches and promotes logical centralization of network control, allowing optimal policies and flexible management on the network flows. SDN provides many network functionalities including congestion control, failure recovery, network measurement, and traffic engineering, etc. With the growing availability of SDN switches and protocols such as OpenFlow [3], many organizations have started deploying their own SDN networks [4].

SDN switches are usually very expensive; thus, the deployment of SDN is usually a gradual process. In addition, to prevent possible service disruption, we cannot upgrade all the devices at once, and it is sometimes impractical to upgrade all the devices. In practice, we usually see networks where traditional IP routers and SDN switches coexist. This kind of networks are usually referred to as hybrid SDN networks [5] [6]. SDN switches also have IP router functionalities. In the rest of the paper, we use routers and switches interchangeably.

In this paper, we consider using SDN switch to improve the network's fault-tolerance. Specifically, given an IP network topology, we consider replacing some of the IP routers with designated switches, which may or may not be SDN switches, in order to allow the network to recover from any single-link failure. Because SDN switches are expensive, we aim to minimize the number of SDN switches needed, in order to minimize the cost of providing such fault-tolerance. We want to emphasize that the fault tolerance we want to achieve here is very different from that of traditional routing protocols. When a failure happens, in traditional routing protocols, the failure information needs to be propagated to a considerable part of the network, and reactive computing is usually required before the network stablizes. In our design, the affected traffic can be immediately redirected to designated switches; thus, the network can recover from the failure very fast.

An existing work considers using SDN switches to replace some legacy IP routers, so that the network can recover from any single-link failure [5]. When a link failure occurs, the router with the failed link will redirect affected traffic to an SDN switch, which can efficiently figure out a next hop switch, from which the shortest path to the destination can avoid the failed link. The SDN switch used for failure recovery is called a *designated* switch; the authors' objective is to minimize the number of SDN switches in the network, while still allowing the network to recover from any single-link failure. By setting up tunnels between traditional IP routers and the designated switches, the proposed approach allows IP routers to perform failover immediately upon detecting a link failure. Based on the routing decision made by the SDN controller, the designated SDN switches can then help to forward the traffic so that it bypasses the failed link.

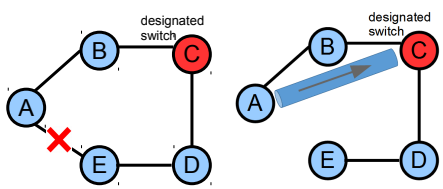
We find that the designated switch does not always need to be an SDN switch. In some cases, we can use a traditional non-SDN designated switch to replace a legacy IP router and then the network can recover from single-link failures. We can succeed in doing so if (1) the shortest path from the affected router to the designated switch does not include the failed link, and (2) the shortest path from the designated switch to the affected destination does not include the failed link. In these cases, the designated switch only needs to have IP tunneling capability but does not need to be an SDN switch. IP tunneling capability is common in non-SDN switches [7]; this makes it possible to use non-SDN switches as designated switches. We still need SDN switches if we cannot find a valid location for a non-SDN designated switch. By using non-SDN switches as designated switches, intuitively, we can further reduce the total number of SDN switches needed while still guaranteeing that the network can recover from any single-link failure.

1.1. *Motivational example*

We use an example, shown in Fig. 1, to illustrate this idea. Assume that all links in the network have the same cost. Consider the case when the link between A and E fails. If A is the source router, we can see that traffic from A to E, and traffic from A to D will be affected by this link failure.

The authors in [5] propose using SDN switches to replace some routers to improve the network's fault-tolerance. In this example, an SDN switch can replace either B or C. If the SDN replaces C, A will forward affected traffic to the SDN switch at C. The SDN switch at C can choose its neighbor D as the next hot switch, and from there, the traffic can follow the shortest path to E and D without using the failed link $e_{A,E}$.

Alternatively, we can use a traditional non-SDN switch with IP tunneling capability to replace the router at C, to help the network recover from the failure. If we have a designated switch at C, we can setup an IP tunnel from A to C going through B. When the link $e_{A,E}$ fails, we can let the traffic at A go to C using the IP tunnel $A \rightarrow B \rightarrow C$. Then from C, the traffic can still take shortest paths,



(a) Link $e_{A,E}$ fails. (b) IP tunneling.

Fig. 1. Motivational example. We set up a pre-configured IP tunnel from A to C. When link $e_{A,E}$ fails, traffic from A cannot reach E or D through shortest paths. With IP tunneling, affected traffic at A will be first redirected to C through the IP tunnel from A to C. After that C can further deliver the traffic to its destinations.

$C \to D \to E$ and $C \to D$ to arrive at destinations D and E, respectively. Since IP tunneling capability is quite common in modern routers, we actually do not need to physically replace the router; we only need to change the configuration of the router, i.e., enable the IP tunneling capability.

Similarly, for router E, we can use a non-SDN designated switch also at C. We can setup an IP tunnel from E to C going through D. When the link $e_{E,A}$ fails, we can let the traffic at E go to C using the IP tunnel $E \to D \to C$. Then, from C, the traffic can still take shortest paths, $C \to B \to A$ and $C \to B$ to arrive at destinations A and B, respectively.

From this simple example, we can draw an important intuition: if the network can recover from single-link failures using only non-SDN designated switches with IP tunneling capability, we can further reduce the total number of SDN switches needed to guarantee that the entire network can recover from any single-link failure.

1.2. *Contributions and paper organization*

In this paper, we consider IP fast recovery in a given legacy network topology. Our contributions can be outlined as follows:

- First, we find that SDN switches are not always needed to recover from a link failure. Sometimes, a traditional non-SDN switch with IP tunneling capability is sufficient to recover the network from a single-link failure.
- Second, for networks with uniform link costs, we show that by using only non-SDN switches as designated switches, it is guaranteed that the entire network can recover from any single-link failure; thus, no SDN switch is needed.
- Third, for networks with general link costs, we prove that by using SDN switches as designated switches, it is guaranteed that the entire network can recover from any single-link failure. Besides, by using SDN switches as designated switches only when necessary, we can further reduce the number of SDN switches needed compared to an existing work [5].
- We conduct extensive simulations using both practical IP network topologies and randomly generated networks to verify our proposed method. Simulation results show that our proposed method can significantly reduce the number of SDN switches needed compared to the existing work.

The rest of the paper is organized as follows. The problem description and our design approach are presented in Sec. 2. Important preliminary results are presented in Sec. 3. In Sec. 4, we consider minimizing the number of SDN switches needed while guaranteeing recovery from any single-link failure. A heuristic algorithm is proposed to reduce the number of SDN switches. Supporting simulations are conducted in Sec. 5. We discuss related works on single-link failure recovery in Sec. 6. Conclusions and future works are described in Sec. 7.

2. Problem Description and Overall Design

2.1. *Problem description*

Given a network topology, our goal is to replace a subset of routers with designated switches to recover the network from any single-link failure. The fault tolerance we want to achieve here is very different from traditional routing protocols. In traditional routing protocols, such as RIP and OSPF, when a failure happens in the network, it takes a long time for the failure information to be propagated into the network, and some lengthy reactive computation may be required for the network to stablize. In our design with designated switches, affected traffic can be immediately redirected to the designated switch upon detection of the link failure.

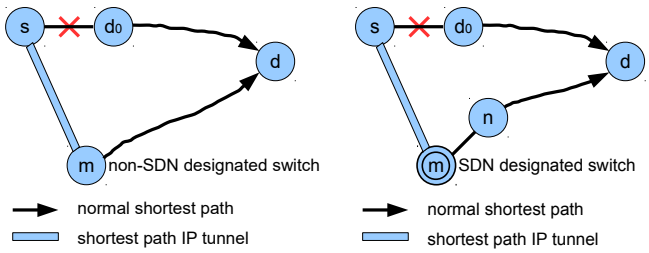
In our consideration, a practical design should have the following features. First, it should take advantage of protocols available on today's routers; it means that routers without SDN functionality should always forward the traffic through shortest paths. When multiple shortest paths are available, Equal-Cost Multi-Path (ECMP) routing can be utilized to achieve traffic balance in the network [8]. Also, ECMP itself can avoid immediate failed links when other shortest paths are available. Second, the network should recover from any single-link failure immediately when the failure is detected.

Our network under consideration can be modeled as an undirected simple bi-connected graph. A bi-connected graph is a graph which will remain connected if any one vertex were to be removed [9]. The bi-connected graph model for a network guarantees that the network will still be connected upon any single-link failure. This is the foundation for our single-link failure recovery scheme.

2.2. *Overall design*

In our approach, each interface of a router is configured to have a backup IP tunnel to provide failover upon detecting a link failure on its interface. The IP tunnel is established between the router and a designated switch, which may or may not be an SDN switch. Whenever a link failure is detected, the router, which is directly connected to the failed link, would immediately encapsulate and forward all affected traffic to its designated switch through the backup IP tunnel. As shown in Fig. 2(a) and Fig. 2(b), when link e_{s,d_0} fails, router s will redirect all affected traffic to a designated switch at m. Because from s to m, all traversed routers in the IP tunnel are normal routers, the IP tunnel should be a shortest path. The designated switch m takes further actions to make sure that the traffic gets to its destination without using the failed link.

We use two types of designated switches to recover the network from single-link failures. The first type is the traditional *non-SDN designated switch*, and the second type is the *SDN designated switch*. Non-SDN designated switches can only set up IP tunnels. From the affected router to the non-SDN designated switch, the IP tunnel



(a) Use non-SDN designated switch. (b) Use SDN designated switch.

Fig. 2. An example of recovering from a single-link failure. In figure (a), m is a non-SDN designated switch; affected traffic from s will take a shortest path from m to the destination d. In figure (b), m is an SDN designated switch; affected traffic from s can first go to a neighbor switch of m, for example n, from which a shortest path to d is taken. The solid line between n and m means that n is a direct neighbor of m.

must be a shortest path; from the non-SDN designated switch to the destination switch, the path must also be a shortest path.

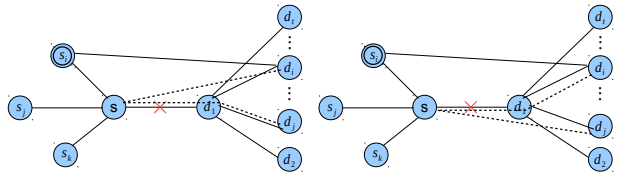
SDN designated switches have more flexibility in terms of redirecting affected traffic. From the affected router to the SDN designated switch, the tunnel must be a shortest path; from the SDN designated switch to the destination switch, the path is not necessarily a shortest path. The SDN switch can select any of its neighbor switches, and then from the selected neighbor switch to the destination, the traffic must take a shortest path.

We require the assignment of designated switches to be destination independent. The advantage of this requirement is that the complexity of configuring failover is minimized since routers do not need to deal with each individual destination. Therefore, for a designated switch to be eligible, it must be able to accommodate all the possible destinations tunneled from an affected router. Since all the above IP tunnels and next hops are pre-configured before the failures really take place, this approach allows all the devices to perform failover right after the failure is detected. Thus, this approach can achieve fast failure recovery.

3. Preliminaries

In this section, we investigate important characteristics of the problem. For networks with uniform link costs, there exist several methods for recovering any single link failure by just using IP routers' tunneling capability. Examples include IP Fast Reroute (IPFRR) Not-Via [10] and a more recent method proposed in [11]. In other words, we do not need any SDN switch to recover from any single link failure.

Next, we consider networks with general link costs. Given the failed link $e_{s,d}$, the nodes in the graph can also be partitioned into two parts. The first part consists of all the nodes where the shortest paths from s to these nodes include the failed link $e_{s,d}$. The second consists of all the nodes where the shortest paths from s to these nodes do not include the failed link $e_{s,d}$. Denote all nodes in the first part by



(a) Case I (b) Case II

Fig. 3. Two possible cases of the shortest paths for networks with general link costs. We use an SDN designated switch at s_i for s when e_{s,d_1} fails.

$\mathcal{D} = \{d_1, d_2, \ldots, d_k\}$[a]. Denote all nodes in the second part by $s_1, s_2, \ldots, s_l$, where $s_1 = s$. It is easy to find examples where networks with general link cost cannot recover from all single-link failures with only non-SDN designated switches. Next, we show that using SDN designated switches, it is guaranteed that networks with general link costs can recover from any single-link failure.

First, there exists at least one node in the first part that has a link to a node in the second part. Otherwise, it violates the properties of a bi-connected graph. Denote the node in the first part as d_i and the node in the second part as s_i.

We can then choose s_i as the designated SDN switch for s_1 when $e_{s,d}$ fails, and choose d_i as the next hop switch for all the destination nodes in the first part.

Lemma 3.1. *The shortest path from d_i to d_j will never use the failed link $e_{s,d}$.*

Proof. We prove this by contradiction. Assume that the shortest path from d_i to d_j includes $e_{s,d}$ as one link in it. We have two cases to investigate.

In the first case, as shown in Fig. 3(a), the shortest path p_{d_i,d_j} consists of the shortest path from d_i to s, i.e., $p_{d_i,s}$, the link, $e_{s,d}$, and the shortest path from d to d_j, i.e., p_{d,d_j}. However, since d_i is affected by the failed link $e_{d,s}$, there is a shortest path from s to d_i that consists of $e_{s,d}$. The length of the shortest path $p_{d_i,s}$ is equal to that of $e_{s,d} + p_{d,d_i}$. Also, the length of the path $e_{s,d} + p_{d,d_j}$ is greater than that of p_{d,d_j}. Then, the length of p_{d_i,d_j} is greater than the length of $p_{d_i,d} + p_{d,d_j}$, which violates the fact that p_{d_i,d_j} is a shortest path.

In the second case, as shown in Fig. 3(b), the shortest path p_{d_i,d_j} consists of the shortest path from d_i to d, $e_{d,s}$, and the shortest path from s to d_j. However, since d_j is affected by the failed link $e_{d,s}$, there is a shortest path from s to d_j that consists of $e_{s,d}$. The length of the shortest path p_{s,d_j} is equal to that of $e_{s,d} + p_{d,d_j}$. The length of $p_{d_i,s}$ is greater than that of $p_{d_i,d}$. Then, the length of p_{d_i,d_j} is greater than the length of $p_{d_i,d} + p_{d,d_j}$, which again violates the fact that p_{d_i,d_j} is a shortest path.

Thus, the shortest path from d_i to d_j will never use the failed link $e_{s,d}$. □

[a]It is possible for $\mathcal{D}$ to be empty, for example, when the cost of $e_{s,d}$ is very high, and not any shortest path includes it. In this case, the network can be considered unaffected. In the rest of the paper, we will not consider this special case.

Theorem 3.1. A bi-connected network can recover from any single-link failure by using SDN designated switches.

Proof. The theorem follows from Lemma 3.1. By choosing s_i as the designated SDN switch for s_1 when e_{s_1,d_1} fails, and choosing d_i as the next hop switch for all the affected destination nodes, Lemma 3.1 guarantees that the shortest path from d_i to d_j will never use the failed link e_{s_1,d_1}. This means that we can always find an SDN switch that allows for the network to recover from this link failure. Note that e_{s_1,d_1} is a general link in the network; thus, all the analyses in Lemma 3.1 apply to any link in the network. Thus, the entire network can recover from any single-link failure by using SDN designated switches. □

4. Minimizing the Number of SDN Switches

In this section, we consider minimizing the number of SDN switches needed in detail. Recall that networks with uniform link costs can recover from any single-link failure by using only non-SDN designated switches. In other words, the minimal number of SDN switches is zero. From now on, we focus on networks with general link costs.

Recall that we can use two types of switches to recover the network from link failures. The first type is the normal *non-SDN designated switch*, and the second type is the *SDN designated switch*. To minimize the number of needed SDN switches, an intuitive approach is to use SDN switches only when they are required. When we consider a single-link failure, if all affected traffic can be recovered by a non-SDN designated switch, then we do not require an SDN switch to recover from this link failure.

Definition: A link is called a Non-SDN Recoverable Link (NSRL) if the failure of this link does not need an SDN switch to be the designated switch.

Notice that when a link fails, the two end nodes of the link can be configured to take different actions. Thus, for each link failure, we need to consider two directions of the failure.

4.1. *Collecting non-SDN recoverable links*

The first step of our algorithm is to identify all directed NSRLs. Given a link failure, say $\vec{e}_{s,d}$, we assume that the source is s, and we denote all the destinations that are affected by this link failure as $\mathcal{D} = \{d_i\}$. The link, $\vec{e}_{s,d}$, can be regarded as an NSRL if and only if $\forall d_i \in \mathcal{D}$ there exists a switch in the network, m ($m \notin \mathcal{D}$), such that the shortest path from s to m, $p_{s,m}$ and the shortest path from m to d_i, p_{m,d_i} both do not include the failed link.

We can first collect all such NSRLs, denoted as L_{NSRL}. We denote all link failures in the network as L_U, where $L_U = \{\vec{e}_{s,d} | e_{s,d} \in E\} \cup \{\vec{e}_{d,s} | e_{s,d} \in E\}$. After collecting NSRLs, we need to consider recovering from failures in $L_R = L_U - L_{NSRL}$ using SDN switches.

Figure 4 shows a simple example on how to collect the NSRLs in a network. The link costs of $e_{A,B}$, $e_{B,C}$, $e_{C,D}$, $e_{D,E}$, and $e_{E,A}$ are 8, 7, 2, 1, and 4, respectively. There are five links in total. Considering the link directions, there are 10 link failure cases. The table in Fig. 4 shows the information for each of the link failures. Each link failure is represented by a row in the table. The first column is the failed link; the second column is the source node; the third column is the destinations that are affected by this link failure; the fourth column indicates whether the network can recover from this link failure only using non-SDN designated switches. A "Y" in the fourth column means that the failed link is an NSRL. An "N" in the fourth column means that the failed link is not an NSRL; thus, an SDN designated switch must be used for the network to recover from this link failure.

Consider the link failure of $\vec{e}_{B,C}$. Traffic from B to C, D, and E will be affected. This is because the shortest paths from B to C, from B to D, and from B to E all include $\vec{e}_{B,C}$ as a link. Though three destinations are affected, the network can still recover from this link failure by using a non-SDN switch at node A. So, the traffic from B destined at C, D, and E will be redirected to node A. Since the shortest paths from A to C, from A to D, and from A to E do not include $\vec{e}_{B,C}$ as a link. The traffic can successfully arrive at the destinations using shortest paths. Thus, link $\vec{e}_{B,C}$ is an NSRL.

Consider the link failure of $\vec{e}_{C,D}$. Traffic from C to D, E, and A will be affected. This is because the shortest paths from C to D, from C to E, and from C to A all include $\vec{e}_{C,D}$ as a link. We can easily tell that only node B can serve as the designated switch for C. If B is a non-SDN switch, even if the traffic from C destined at D is redirected to B, B cannot deliver the traffic to D using a shortest path, because the shortest path from B to D also includes the failed link $\vec{e}_{C,D}$. For the same reason, even if the traffic from C destined at E is redirected to B, B cannot deliver the traffic to E using a shortest path. Thus, $\vec{e}_{C,D}$ is not an NSRL. We must use an SDN switch to allow the network recover from this link failure. And B is the only valid location for C.

4.2. *Using SDN switches for remaining link failures*

After identifying all NSRLs, we need to consider using SDN switches to recover from failures in $L_R = L_U - L_{NSRL}$. The problem of minimizing the number of SDN switches can be formulated as a binary integer programming problem which is a well-known NP-complete problem [5] [12]. This means that the computation may not complete in a reasonable time for large networks. We use a heuristic algorithm for this problem with polynomial time complexity. It includes two phases: candidate table construction and column selection.

For candidate table construction, the goal is to identify eligible candidate SDN locations for each link failure in L_R. Figure 5 shows the candidate table for the topology in Fig. 4. Note that, we only consider links in L_R because all links in L_{NRSL} do not require SDN switches for failure recovery. For each link failure $i \rightarrow$

Link Failure	Source	Affected Destinations	NSRL
A->B	A	B	Y
B->A	B	A	Y
B->C	B	C, D, E	Y
C->B	C	B	Y
C->D	C	D, E, A	N
D->C	D	C, B	Y
D->E	D	E, A	N
E->D	E	D, C, B	N
E->A	E	A	Y
A->E	A	E, D, C	Y

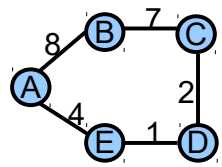


Fig. 4. Illustration of NSRLs.

Link Failure	Source	Affected Destinations	SDN candidate locations				
			A	B	C	D	E
C->D	C	D, E, A	0	1	0	0	0
D->E	D	E, A	0	1	1	0	0
E->D	E	D, C, B	1	0	0	0	0

Fig. 5. Illustration of candidate table construction.

j, all affected destinations are again denoted by $\mathcal{D}$. A node $k \notin \mathcal{D}$ is an eligible candidate location for an SDN designated switch for this failure if it meets the following conditions:

(1) the shortest path from i to k does not include the failed link, and
(2) for each affected destination $d \in \mathcal{D}$, k must have at least one neighbor m, such that the shortest path from m to d does not include the failed link.

If node k meets the above conditions, it is labeled "1," otherwise, it is labeled "0." After constructing the candidate table, the problem is reduced to a column selection problem. The goal is to select the least number of SDN locations (columns) so that all failures in L_R (all rows in the table) are covered. For the selected columns to cover all rows, the sum of each row should be greater than 0.

A greedy approach is used to select the columns. For each column, we add up all the elements and the sum is called the weight of the column. This weight indicates the number of failures that can be covered if an SDN switch is placed at the corresponding location. We then choose the column with the largest weight and remove this column as well as rows that are covered by this column. The weights are then updated for the remaining columns and rows, and another column with the largest weight is selected. We repeat this process until all the rows are removed which means that all the possible link failures are covered. These selected columns are the locations where we should place SDN switches.

5. Evaluations

We conduct simulations with different topologies to compare the number of SDN switches needed using different methods. We call the existing method in [5] the

"*base*" method, where all designated switches are SDN switches. We call ours the "*proposed*" method, where we use SDN switches as designated switches only when required. Notice that, for networks with uniform link costs, our *proposed* method requires no SDN switches, while the *base* method always requires some number of SDN switches. The advantage of our method is obvious. In this section, we only conduct comparisons for networks with general link costs.

5.1. *Basic network topologies: Specific examples*

Three practical network topologies are used for the evaluation: the 11-node COST239 topology [13] with 26 links, the 14-node NSFNET topology [14] with 21 links, and the 22-node Exodus topology [15] with 51 links. Their basic profiles are listed in Table 1. All the topologies are represented in undirected graphs. The degree of each topology stands for the average number of links on a router.

We assign different costs to the links in the network. We start from a network with uniform link costs, i.e., all links with a cost value of 1. Then, we pick some links and assign different cost values to them. In our simulations, we find that, if we select a small number of links, and assign different link costs to them, it is highly probable that our *proposed* method still does not require any SDN switches to achieve the desired fault-tolerance. In the following, we will present several examples for each of the three topologies, where our *proposed* method does require some SDN switch(es) to achieve the desired fault-tolerance.

Table 1. Topology Profile.

Topology	Number of Nodes	Number of Links	Degree
COST239	11	26	4.7
NSFNET	14	21	3
Exodus	22	51	4.6

Table 2. Number of SDN Switches Needed for the Three Specific Examples.

Topology	*base*	*proposed*
COST239	2	1
NSFNET	3	1
Exodus	4	2

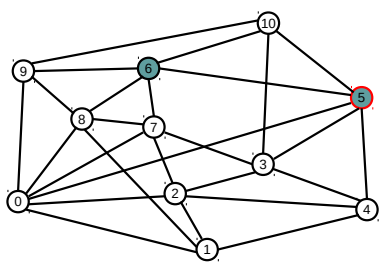


Fig. 6. COST239 topology.

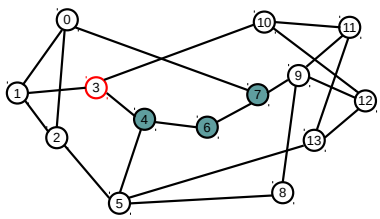


Fig. 7. NSFNET topology.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0							1				1						1					
1						1		1				1										1
2					1	1										1						
3						1								1								
4			1						1								1					1
5		1	1	1					1							1	1					
6	1									1												
7		1							1				1	1			1					1
8						1	1		1								1					
9								1			1		1		1		1				1	1
10	1									1					1							1
11		1											1				1					
12									1		1		1									1
13				1					1						1	1	1		1		1	1
14										1	1			1			1					1
15			1			1								1			1	1				
16	1				1	1		1	1	1		1		1	1	1		1		1		1
17																1	1					
18														1							1	
19				1													1					
20										1				1					1			
21		1			1			1		1	1		1	1	1		1					

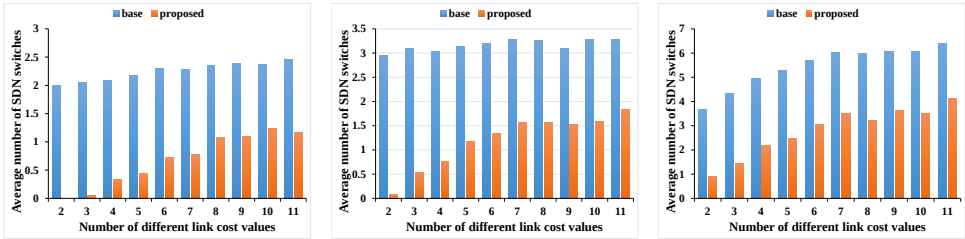
Fig. 8. Adjacency matrix for the Exodus topology.

For the COST239 topology, if the costs of $e_{0,1}$, $e_{1,4}$, $e_{3,4}$, $e_{3,10}$ and $e_{6,9}$ are 2, the costs of $e_{2,4}$, $e_{3,5}$, $e_{5,6}$, $e_{6,7}$ and $e_{7,8}$ are 3, the costs of $e_{6,10}$ is 4, while all other links' costs remain 1, then, the *base* method will replace the switches at 6 and 5 with SDN switches as shown in Fig. 8, and our *proposed* method only need to replace the switch at 5 with one SDN switch.

For the NSFNET topology, if the costs of links $e_{0,1}$, $e_{3,10}$, $e_{4,5}$, $e_{5,8}$, $e_{6,7}$ and $e_{6,7}$ are 2, the cost of link $e_{8,9}$ is 3, the cost of link $e_{4,6}$ is 4, while all other links' costs remain 1, then, the *base* method will replace the switches at 4, 7, and 6 with SDN switches, as shown in Fig. 7, and our *proposed* method only needs to replace the switch at 3 with one SDN switch.

For the Exodus topology, if the cost of link $e_{3,13}$ is 2, the costs of links $e_{5,15}$, $e_{5,16}$, $e_{11,16}$ and $e_{11,16}$ are 3, the costs of links $e_{3,5}$ and $e_{10,21}$ are 4, the costs of links $e_{4,21}$ and $e_{9,12}$ are 5, while all other links' costs remain 1, then, the *base* method will replace the switches 16, 5, 6 and 3 with SDN switches, and our *proposed* method only needs to replace switches at 3 and 5 with SDN switches.

Table 2 summarizes the results. We can observe that, in all cases, our *proposed* method uses fewer SDN switches than the *base* method.



(a) COST239. (b) NSFNET. (c) Exodus.

Fig. 9. Simulations for the COST239, NSFNET, and Exodus topologies.

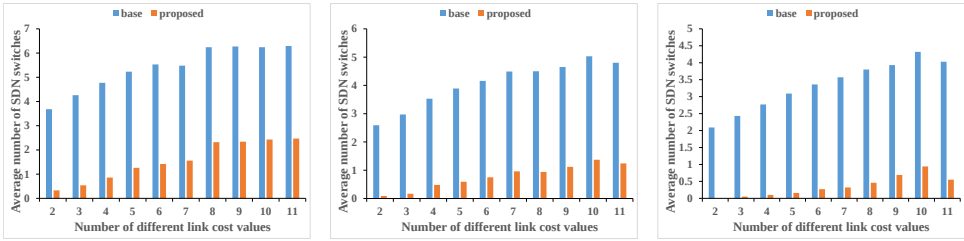
5.2. *Basic network topologies: Random link costs*

We continue to conduct simulations using the three basic network topologies. Our goal is to investigate how the link cost distribution influence the number of SDN switches needed using both methods. For each of the three basic network topologies, we conduct 10 groups of simulations. Each of the 10 groups differ from each other in how the network link costs are assigned. From group 1 to group 10, we let the network links to choose cost values from $2, 3, ..., 11$ distinct values, respectively. Specifically, in group 1, all network link costs will be either 1 or 2; in group 2, network link costs will randomly choose values from $\{1, 2, 3\}$. Generally, in group i, all network link costs will randomly choose values from $\{1, 2, \cdots, i + 1\}$. The simulation results for the COST239, NSFNET, and Exodus topologies are presented in Fig. 9(a), Fig. 9(b), and Fig. 9(c), respectively.

We can see that, when the number of different network link cost values increases, the number of SDN switched needed by both methods increases. However, the increase in the number of SDN switches will stop when it reaches some number. This indicates that given a fixed network topology, the number of SDN switches needed is limited no matter how the link cost values vary. The COST239 topology has the densest connections and fewer nodes. Thus, the number of SDN switches needed is less than both that of NSFNET and that of Exodus. The number of SDN switches needed in Exodus is greater than that of COST239, though they have a similar average node degree; this is because, the Exodus network also has much more network nodes and links. In all cases, the number of SDN switches needed using our *proposed* method is much less than that using the *base* method.

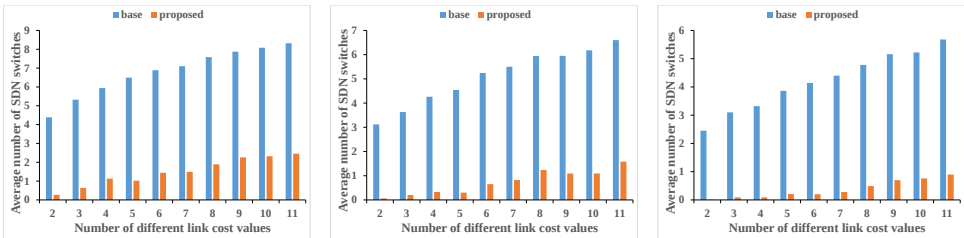
5.3. *Randomly generated network*

We also conduct simulations for various randomly generated networks. Recall that the candidate networks are all modeled as bi-connected graphs. Thus, after we generate the graph, we only collect bi-connected graphs to perform our simulations on. We aim to investigate how the node degree of a network influences the performances of both methods. We design our simulation groups as follows. First, we fixed the number of nodes in the network. They, we randomly generate 100 networks with



(a) Average node degree = 4.432 (b) Average node degree = 6.022 (c) Average node degree = 7.838

Fig. 10. Simulations for networks with 50 nodes.



(a) Average node degree = 4.512 (b) Average node degree = 6.155 (c) Average node degree = 7.949

Fig. 11. Simulations for networks with 100 nodes.

a certain average node degree. For all these 100 networks, we further conduct 10 groups of simulations. In the first simulation group, links in the 100 networks choose random link cost values from $\{1, 2\}$. In the 2nd simulation group, network link costs choose random values from $\{1, 2, 3\}$, and so on. In the 10th group, network link costs choose randomly value from $\{1, 2, \cdots, 11\}$.

For a network with a fixed number of nodes, we intend to generate links such that the network will have some fixed node degree, i.e., 4, 6, or 8, by giving a fixed parameter. However, the links are randomly generated, and the resulting number of links is not fixed. Thus, each network's average node degree is not exactly equal to the given parameter, though they are very close. In the simulation results, we will explicitly write the actual average node degree of the networks. We conduct simulations for networks with 50 nodes, and 100 nodes respectively.

The simulation results for networks with 50 nodes are presented in Fig. 10. In Fig. 10(a), Fig. 10(b), Fig. 10(c), the average node degrees for the networks are 4.432, 6.022, and 7.838, respectively. We can see that, besides the network link costs, the network node degree has a significance influence on the number of SDN switches needed using both methods. In Fig. 10(a), the average number of SDN switches can be greater than 6 using the *base* method; in Fig. 10(c), this number drops to less than 4.5. Similarly, in Fig. 10(a), the average number of SDN switches can be greater than 2 using our *proposed* method; in Fig. 10(c), this number drops to less than 1, meaning that when the network node degree is 8, many of the networks do not need

an SDN designated switch to achieve single-link failure recoverability. From these simulation results, we can still see the influence of network link cost distribution on the number of SDN switches needed. When the deviation of the link cost values is small, i.e., when the link costs choose values from a small set of values, most of the networks do not require SDN switches to have the desired fault-tolerance using our *proposed* method, as shown in the first three simulation groups of Fig. 10(c). The simulation results for networks with 100 nodes are presented in Fig. 11; it shows similar properties and patterns as Fig. 10. In all simulation cases, our proposed method reduces the number of SDN switches needed significantly. Through these simulations, we can see that our proposed method can reduce the number of SDN switches in general networks with general link costs.

6. Related Works

6.1. *Hybrid SDNs*

Hybrid SDN networks have made significant presence in recent years [5], 6]. Promising benefits of hybrid SDN networks have been shown by early research works. More investigation is needed to explore the benefits of hybrid SDN networks in other aspects. Safely updating hybrid SDN networks is considered in [16]. Authors in [17] propose a lightweight control plane, which takes legacy switches into full consideration and allows SDN control applications run on it transparently. The authors in [18] propose an inter-domain routing platform with hybrid SDN architecture. The authors of [19] investigate the use of SDN switches in a hybrid SDN network to dynamically adjust routing decisions based on measured traffic; it is shown that the performance of traffic engineering can be significantly improved with only a few SDN switches. [20] presents a comprehensive survey on various types of hybrid SDN networks; opportunities, challenges, and trade-offs in different types of hybrid SDN networks are also discussed.

6.2. *Fast failure recovery for single link failures*

Failure recovery has long been a subject in communication and computer networks, and many approaches have been proposed for traditional networks with legacy routers. Typical examples include ECMP [8], IP Fast Reroute - Loop Free Alternates (IPFRR-LFA) [21], IPFRR Not-Via [10] and Multiple Routing Configuration (MRC)[22], etc.

The authors in [5] consider using SDN switches to recover from single-link failures. The idea is to replace some legacy IP router(s) in the network with SDN switches, such that when a link failure occurs, the router with the failed link will redirect affected traffic to an SDN designated switch.

Compared to [5], we find that the designated switch does not always need to be an SDN switch. In some cases, we can use a traditional non-SDN designated switch to help network recover from single-link failures. We can succeed in doing

so if the shortest path from the affected router to the designated switch does not include the failed link, and that the shortest path from the designated switch to the affected destination does not include the failed link. In these cases, the designated switch only needs to have IP tunneling capability. By using SDN switches only when required, we can further reduce the total number of SDN switches needed. Besides, in our work, we consider network with general link costs explicitly, while [5] only considers networks with uniform link costs.

7. Conclusions and Future Works

In this paper, we consider IP fast recovery from single-link failures. By redirecting traffic from the failed link to designated switches through pre-configured IP tunnels, our proposed approach is able to react to the failures very fast. For networks with uniform link costs, we show that we can use normal non-SDN switches with IP tunneling capability as designated switches to recover the network from any single-link failure. For networks with general link costs, we show that using SDN switches as designated switches can always allow the network to recover from any single-link failure. By using SDN switches only when necessary, we can further reduce the total number of SDN switches needed compared to an existing method. Extensive simulations have been conducted to verify the applicability of our approaches. Future works can be conducted in the following directions: (1) consider single-node failure recovery using non-SDN designated switches; (2) consider failure recovery when two or more links fail at the same time.

References

1. D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: A comprehensive survey, Proceedings of the IEEE **103**(1) (2015) 14–76.
2. M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, S. Shenker, Ethane: Taking control of the enterprise, in: Proceedings of SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, (2007) pp. 1–12.
3. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, Openflow: Enabling innovation in campus networks, SIGCOMM Comput. Commun. Rev. **38**(2) (2008) 69–74.
4. S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, A. Vahdat, B4: Experience with a globally-deployed software defined WAN, in: Proceedings of the ACM SIGCOMM Conference on Computer Communications, (2013) pp. 3–14.
5. C. Y. Chu, K. Xi, M. Luo, H. J. Chao, Congestion-aware single link failure recovery in hybrid sdn networks, in Proceedings of IEEE Conference on Computer Communications (INFOCOM), (2015) pp. 1086–1094.

6. H. Xu, X. Y. Li, L. Huang, H. Deng, H. Huang, H. Wang, Incremental deployment and throughput maximization routing for a hybrid sdn, IEEE/ACM Transactions on Networking PP **99** (2017) 1–15.
7. S. Kini, S. Ramasubramanian, A. Kvalbein, A. F. Hansen, Fast recovery from dual-link or single-node failures in ip networks using tunneling, IEEE/ACM Transactions on Networking **18**(6) (2010) 1988–1999.
8. A. Iselt, A. Kirstadter, A. Pardigon, T. Schwabe, Resilient routing using mpls and ecmp, in Workshop on High Performance Switching and Routing (HPSR), (2004) pp. 345–349.
9. P. E. Black, Dictionary of algorithms and data structures, U.S. National Institute of Standards and Technology, 2004.
10. A. Atlas, A. Zinin, IP fast reroute using not-via addresses, in IETF Draft, draft-ietf-rtgwg-ipfrr-notvia-addresses-05, (2010).
11. S. Bryant, C. Filsfils, S. Previdi, M. Shand, N. So, Remote loop-free alternate (LFA) fast reroute (FRR), in: RFC 7490, (2015).
12. R. M. Karp, Reducibility among combinatorial problems, in: Springer, 1972.
13. P. Batchelor, Ultra High Capacity Optical Transmission Networks: Final Report of Action COST 239, Faculty of Electrical Engineering and Computing, (1999).
14. D. Banerjee and B. Mukherjee, Wavelength-routed optical networks: linear formulation, resource budgeting tradeoffs, and a reconfiguration study, IEEE/ACM Transactions on Networking **8**(5) (2000) 598–607.
15. D. Hock, M. Hartmann, C. Schwartz, M. Menth, Effectiveness of link cost optimization for ip rerouting and ip fast reroute, in: Proceedings of the 15th International GI/ITG Conference on Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance, (2010) pp. 78–90.
16. S. Vissicchio, L. Vanbever, L. Cittadini, G. G. Xie, O. Bonaventure, Safe update of hybrid sdn networks, IEEE/ACM Transactions on Networking PP **99** (2017) 1–14.
17. S. Huang, J. Zhao, X. Wang, Hybridflow: A lightweight control plane for hybrid sdn in enterprise networks, in: Proceedings of IEEE/ACM 24th International Symposium on Quality of Service (IWQoS), (2016) pp. 1–2.
18. C. E. Rothenberg, M. R. Nascimento, M. R. Salvador, C. N. A. Corrêa, S. Cunha de Lucena, R. Raszuk, Revisiting routing control platforms with the eyes and muscles of software-defined networking, in Proceedings of the First Workshop on Hot Topics in Software Defined Networking (HotSDN), (2012) pp. 13–18.
19. S. Agarwal, M. Kodialam, T. V. Lakshman, Traffic engineering in software defined networks, in Proceedings of IEEE Internatioal Conference on Computer Communications (INFOCOM), (2013) pp. 2211–2219.
20. S. Vissicchio, L. Vanbever, O. Bonaventure, Opportunities and research challenges of hybrid software defined networks, SIGCOMM Comput. Commun. Rev. **44**(2) (2014) 70–75.
21. A. Atlas, A. Zinin, Basic specification for ip fast reroute: Loop-free alternates, tech. rep. in: RFC 5286, (2008).
22. A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing, O. Lysne, Multiple routing configurations for fast ip network recovery, IEEE/ACM Transactions on Networking **17**(2) (2009) 473–486.