Journal of Circuits, Systems, and Computers
Vol. 27, No. 6 (2018) 1850087 (30 pages)

DOI: 10.1142/S0218126618500871

Link Failure Recovery in SDN: High Efficiency, Strong Scalability and Wide Applicability*

Jue Chen†, Jinbang Chen†,§, Junchen Ling†, Junlong Zhou‡ and Wei Zhang†

†*Department of CST,*
East China Normal University,
Shanghai 200062, P. R. China

‡*School of CSE,*
Nanjing University of Science and Technology,
Nanjing 210094, P. R. China
§*chenjinbang@gmail.com*

Received 27 May 2017
Accepted 30 August 2017
Published 28 September 2017

Link failures are commonly observed in computer networks, including the newly emerging Software Defined Network (SDN). Considering that failure recovery methods used in traditional networks cannot be applied to SDN networks directly, we propose a method named pro-VLAN in this paper, which calculates a backup path and assigns a unique VLAN id for each link of the network based on the protection mechanism. It makes the most of SDN's features and can recover a single link failure in SDN with the advantages of high efficiency, strong scalability and wide applicability. More specifically, high efficiency (i.e., a fast failure recovery with a low memory consumption) is achieved by calculating backup paths for each link instead of each flow and using group tables to switch backup paths automatically and locally when failures occur. Strong scalability (i.e., the amount of backup flow entries per switch is stable) is achieved by keeping the amount of links per switch no matter how the network scale extends or how the amount of flows increases. Wide applicability is achieved by always finding a path available without modifying any hardware or protocol as long as the network is still reachable after a link failure. Simulation results and mathematical analysis demonstrate that both pro-VLAN and a flow-based protection method achieve a fast failure recovery, while pro-VLAN consumes less than 1% of the forwarding entries to store backup paths as compared to the flow-based method. Moreover, when the network scale increases from 10 to 60 switches by 500%, the increase of the number of backup flow entries per switch built by pro-VLAN is only less than 50%.

Keywords: Software defined network; openflow; failure recovery; protection mechanism; backup path; VLAN id.

*This paper was recommended by Regional Editor Tongquan Wei.
§Corresponding author.

1. Introduction

With the features of decoupling the data plane from the control plane, and using open programmable interfaces to directly control forwarding behaviors,[1,2] the Software Defined Network (SDN) has gained considerable attention from both academia and industry.[3] A simple and standard SDN is composed of a control plane equipped with one or more controllers and a data plane composed of multiple SDN switches, and we choose OpenFlow as the southbound interface[a] used for communicating between two planes in this paper.

Network fault is a huge stumbling block for both traditional networks and SDN. The fault occurs occasionally but could result in an immeasurable loss of performance, especially for real-time network services. As a result, failure recovery becomes an indispensable part for any network type.[7–9] In traditional networks, routing protocols are able to achieve the automatic reconvergence after a link failure. In addition, the reconvergence time can be further reduced by improving the existing routing protocols or proactively computing backup next hops such as MPLS Fast Reroute (FRR).[10] A router in traditional networks combines the control function with the forwarding function, thus it can run a routing protocol or a failure recovery method and generate the corresponding forwarding rules all by itself in a distributive manner. However, in SDN, the forwarding function and control function are decoupled. More specifically, an SDN switch almost has no intelligence but can only forward packets according to the controller's commands. The controller gathers the network topology information and manipulates forwarding tables of switches in a centralized manner. Due to these aforementioned limitations, the failure recovery methods used in traditional networks cannot be applied to SDN networks directly.

Therefore, we propose a failure recovery method for the data plane in SDN by making the most of SDN's features. Considering that the possibility that multiple network components fail simultaneously in one administrative domain is extremely low (which equals to the product of each component failure's possibility),[11,12] our proposed method is designed for recovering the single link failure in SDN with one-controller model (and we discuss how to recover a switch failure as well).

The failure recovery mechanisms in SDN can be classified into two categories: restoration and protection.[13] In the mechanism of restoration, the controller must be involved into the failure recovery. While in the mechanism of protection, the data plane can switch forwarding paths automatically without the controller. This indicates that the controller should calculate and establish both the working forwarding entries[b]

[a] OpenFlow is the most popular protocol among all the interfaces. Other southbound interfaces include the IEEE P1520 Standards Initiative for Programmable Networks Interfaces,[4] the SoftRouter architecture,[5] ForCES (Forwarding and Control Element Separation),[6] etc.

[b] Working flow entries mean the flow entries to store the information of working paths, backup flow entries mean the flow entries to store the information of backup paths, and backup forwarding entries mean both the flow entries and group entries to store the information of backup paths. In this paper, we regard fast fail-over group entries as a part of backup forwarding entries instead of working flow entries.

and backup forwarding entries in switches before failures. Thus, it is easy to deduce that restoration would take more time for failure recovery than protection due to the inevitable round-trip between the control plane and the data plane in restoration. In other words, protection is more time-efficient. As the network size increases, the recovery time required by restoration will increase dramatically since more flows will be affected and the controller can only handle these flows one by one, while the recovery time required by protection can keep stable since each affected switch can change to its backup path automatically and locally. In other words, protection can guarantee a fast failure recovery regardless of network size while restoration not.

Based on the above discussions, we choose the protection mechanism in order to achieve a fast failure recovery in all the network scales, which mainly consists of four steps as shown in Fig. 1. However, using the protection mechanism in SDN[14–18] has the side effect of a higher memory consumption (i.e., Ternary Content Addressable Memory) as compared to the restoration mechanism. Moreover, most protection-based methods have the drawback of low applicability because they may not be able to always find an available path even though the network is still reachable after a failure,[14] or need to modify the OpenFlow protocol[16] or SDN switches.[18] As a result, an approach that promises a fast failure recovery and overcomes the aforementioned disadvantages is urgent.

In this paper,[c] we propose a new method named pro-VLAN. Based on the protection mechanism, pro-VLAN calculates both working paths and backup paths, and

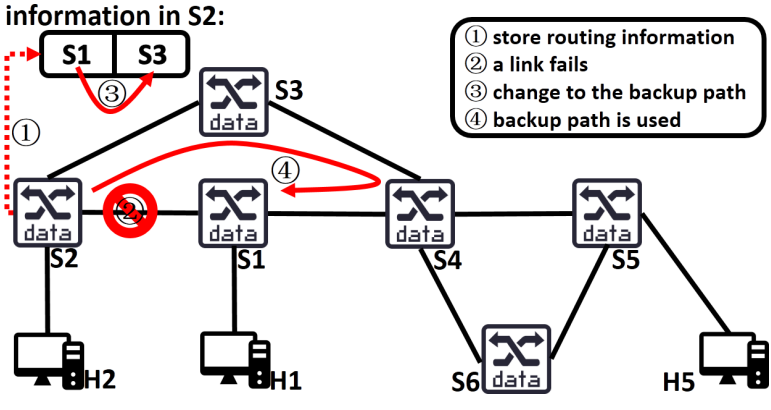


Fig. 1. Recovery of link failures.

[c]This paper is an extension of our previous paper titled "Failure Recovery Using Vlan-tag in SDN: High Speed with Low Memory Requirement".[19] The major differences between this journal paper and our previous IPCCC 2016's paper contain: pro-VLAN is optimized which can reduce the consumption of TCAMs for group entries, and we verify this viewpoint through evaluation and mathematical analysis. Moreover, the discussion which applies pro-VLAN to a multicast scenario and the section of mathematical analysis are added to the journal paper.

stores them into switches before the occurrence of failures. When a link fails, the affected switches can change paths automatically and locally all by themselves, thus promising a fast failure recovery. Considering that the amount of links is fairly less than that of flows, pro-VLAN calculates a backup path for each link instead of each flow, which can greatly reduce the consumption of TCAMs and recover all the single link failures. Moreover, pro-VLAN only needs to reuse VLAN ids to distinguish different links, and does not modify any network components or protocols. The advantages and contributions of pro-VLAN can be summarized as follows.

(1) pro-VLAN is a very efficient method that can use the smallest amount of backup forwarding entries to achieve the most rapid failure recovery.
(2) pro-VLAN is a high scalable method that can keep the amount of backup forwarding entries per switch stable regardless of the network scale and the flow amount.
(3) pro-VLAN is a wide applicable method that does not modify any protocol or hardware and can always find an available path after a link failure as long as the network is still reachable.

Extensive simulation experiments and mathematical analyses are carried out to validate our pro-VLAN. The results demonstrate that pro-VLAN and a flow-based protection method can both achieve a fast failure recovery. They take the similar cost of recovery time, but pro-VLAN consumes less forwarding entries (1%) to store backup paths as compared to the flow-based method. Moreover, when the network scale increases from 10 to 60 switches by 500%, the increase of the number of working flow entries per switch would be more than 1000%, but the increase of the number of backup flow entries per switch built by pro-VLAN is only less than 50%. This result indicates that our pro-VLAN can be widely used in the networks with varying scale.

The rest of this paper is organized as follows. Section 2 introduces related works. Section 3 describes our proposed scheme and algorithm. The effectiveness of our proposed scheme is validated by simulation and mathematical analysis in Secs. 4 and 5, respectively. Section 6 discusses how to extend pro-VLAN to solve the recovery of a switch failure and apply it to a multicast scenario. Concluding remarks are given in Sec. 7.

2. Related Works

Considerable research efforts have been devoted to solve the failure recovery in SDN, which can be divided into two major categories: failure recovery for the data plane and for the control plane.[20] From the aspect of control plane, Botelho *et al.*[21,22] attempted to avoid a single point of failure (i.e., the failure of the controller), and proposed that two controllers should be used simultaneously and connected to the same data store for the sake of consistency. However, when a controller fails, not only

the controller's state, but also the switches' states need to be recovered. Katta *et al.*[23] considered all these states after a controller's failure and handled the whole event-processing cycle as a transaction. Based on the fact that different developers need to consider and realize their own failure recovery functions in the control applications, Peresini *et al.*[24] designed an online system which can be combined with any control application to automate failure recovery. From the aspect of the communication channel between two planes, Beheshti *et al.*[25] proposed the concept of "protected switch" which can bypass the failed link/switch to reach the controller. Sharma *et al.*[26] advised control traffic can use both restoration and protection mechanisms to calculate backup paths. From the aspect of data plane, Mark *et al.*[27] invented a declarative language for network administrators to specify working paths as well as backup paths. Unlike in Ref. 27, most of previous works all developed methods to recover link failures either using a restoration[14] or a protection mechanism.[14,16,18,19,28–30] To be specific, Kim *et al.*[30] supposed switches to be hybrid to support regular IP forwarding schemes while others[14,16,18,19,28,29] assumed the data plane to be composed of pure SDN switches without intelligence. The latter assumption is adopted in our work because it has no restrictions on switches and has wider applicability in SDN.

Since our pro-VLAN adopts protection mechanism, we mainly discuss existing protection-based methods in the following. Among them, multiple methods[14,16,18] are applied to recovering a single link or path failure. Sharma *et al.*[14] presented an approach that calculates two nonintersect paths for each flow, and uses the Bidirectional Forwarding Detection (BFD) technology to detect the status of each path. However, the requirement that two paths for the same flow need to be nonintersect is so rigid that cannot always be satisfied. For instance, this approach cannot find two nonintersect paths for the flow from H2 to H5 (shown in Fig. 1) as S4 is a bridge across the topology, while any single link failure in this topology can be recovered in theory. Note that the utilization of group tables is not the only choice to realize a protection mechanism in SDN. Sgambelluri *et al.*[18] pointed out that both working paths and backup paths can be stored into flow tables with different priorities. They also designed a method to realize failure recovery without the controller by implementing an "auto-reject" function for switches that deletes all the entries using failed links and triggers the use of backup paths. Packet headers also can be used to store both working paths and backup paths.[16] However, all these methods are lack of high efficiency, strong scalability and wide applicability.[d]

In addition to recovering a single link failure, SDN also needs to handle multiple link failures.[29] Brent *et al.*[29] proposed a method to provide a sufficient number of backup paths for multiple link failures within the available space. It adopts the protection mechanism to switch forwarding paths quickly and uses a new forwarding table compression algorithm to reduce the consumption of TCAMs. In order to infer

[d]The reasons have been given in Sec. 1, hence are not discussed again.

encountered link failures and each port's status, this method needs to modify both the OpenFlow protocol and SDN switches. It is worth noting that our pro-VLAN can also be extended to recover multiple link failures through iteratively calculating backup paths for each link, which is left to our future work.

3. Our Proposed Scheme and Algorithm

Before presenting our proposed scheme and algorithm, we first give some introduction of a flow, which can be defined as an arbitrary combination of tuples such as IP addresses, Ethernet addresses, and TCP ports involved in the OpenFlow protocol.[31] In this paper, we choose a pair of IP addresses to define a flow for the sake of simplicity. In addition, as long as multiple data packets have the same source and destination IP addresses, we consider that they belong to the same flow, even though they may have other different tuples. Note that all the different types of flow entries used in this paper are listed in Table 1, which are not meant to be comprehensive but already sufficient for our discussion.

In this section, we first propose a method named pro-PATH, which can recover a single link failure with high speed and wide applicability. However, considering that the calculation of backup paths using pro-PATH depends on flows like other protection-based approaches, a large amount of TCAMs for backup paths is required. We then present another method named pro-VLAN, which is independent of the relation between flows and backup paths, and only needs a small number of TCAMs for backup paths when compared to those for working paths. We finally provide a modified version of pro-VLAN such that this method can be applied to recovering multiple link failures in a specific situation. The pro-PATH and the basic and modified version of pro-VLAN are described in the following subsections.

3.1. *pro-PATH: Protection for each link of each path*

The pro-PATH operates at two major steps. In the first step, it calculates the working path for each flow, and then calculates the backup path for each link composing the working path. In the second step, it establishes the working and backup forwarding entries on switches. During the operation of pro-PATH, protection mechanism is adopted to achieve a fast failure recovery, and the OpenFlow protocol or SDN switches are not modified to promise a wide applicability. Moreover, pro-PATH can always find a path available as long as the network is still reachable

Table 1. Different types of flow entries.

Match fields	Functionality
IP addresses, ingress port (optional)	pro-PATH: packets on a working path or backup path
IP addresses, ingress port (optional)	pro-VLAN: packets on a working path
VLAN id, ingress port (optional)	pro-VLAN: packets on a backup path

when a link failure occurs. In other words, pro-PATH tries the best to find a backup path. The reason can be explained as follows. According to the principle of restoration, that is, calculating a backup path for each affected flow after a link failure, it can be inferred that restoration tries the best to find a backup path. In this situation, the working path and the backup path, or some of them compose a circle, and the circle contains the broken link, otherwise the link failure cannot be repaired. As a result, the link which will be broken later has a backup path prepared in the primary network topology (without any link failures), and pro-PATH can find this backup path before link failures. In all, pro-PATH is equivalent to restoration on the feature of trying the best to find a backup path.

Step 1. According to the OpenFlow protocol,[32] the controller can only communicate with switches directly and hosts are invisible to the controller. As a result, when a new flow emerges, the controller first utilizes ARP packets to learn the adjacent switches of source and destination hosts. After the locations of hosts are confirmed, the controller then adopts the shortest path algorithm to calculate a working path for this flow and a backup path for each link involved in the working path by assuming the link fails. Note that we always use the shortest path algorithm in this work to calculate paths without specification. If two or more paths with the same shortest length exist, we always choose the path whose next hop has the smallest index. For example, the shortest path for the flow from H2 to H5 (shown in Fig. 1) selected by our scheme is S2-S1-S4-S5 rather than S2-S3-S4-S5. Therefore, the working path for any flow or the backup path for any link derived by our scheme is always unique. Moreover, the weight of each link is set to be equal in our scheme.

Step 2. After calculating the working and backup paths, the controller distributes forwarding rules among the data plane. Each switch on a working path (except the switch on the tail) establishes a flow entry and a group entry. For the flow entry, the flow-match is a pair of IP addresses, and the instruction set is jumping to a group entry. For the group entry, two action buckets are used to store the next hops of a working path and a backup path, respectively. For instance, the flow entry and the group entry established on S2 for the flow from H2 to H5 (shown in Fig. 1) are illustrated in the left side of Fig. 2. S1 and S4[e] should establish a flow entry and a group entry similar to S2.

The controller then needs to build the backup flow entries on switches. Considering that a switch can exist on both the working path and backup path for the same flow simultaneously, the flow-matches for all the working flow entries are composed of two parts: IP source address and IP destination address, while those for all the backup flow entries are composed of three parts: IP source address, IP destination address, and ingress port. As a result, backup flow entries can be distinguished from

[e]Even though S5 belongs to the working path as well, it just needs to forward packets to the destination host H5 directly. As a result, a flow entry is enough for the switch on the tail of a working or backup path.

S2

Match	Instructions
IPv4 src = H2, IPv4 dst = H5	Group 1

Group	Type	Actions
1	FF	<Fwd S1>, <Fwd S3>

S3

Match	Instructions
IPv4 src = H2, IPv4 dst = H5, input port = port connecting to S2	Fwd S4

S4

Match	Instructions
IPv4 src = H2, IPv4 dst = H5, input port = port connecting to S3	Fwd S1

Fig. 2. Forwarding entries using pro-PATH to protect the link S2-S1 of the flow from H2 to H5.

working flow entries. For instance, as the path S2-S3-S4-S1 is used to protect link S2-S1 (shown in Fig. 1), S3 and S4 need a backup flow entry as shown in the right side of Fig. 2 (S2 has established a group entry, in which the second action bucket is designed for the backup path). It can be inferred that backup flow entries are dependent on flows by using pro-PATH. Although a switch may store multiple forwarding entries for the same flow, it can still handle packets differently and correctly according to whether an ingress port is involved and what the ingress port number is. Once the two steps are completed, any single link failure can be recovered automatically and locally in the data plane.

3.2. *pro-VLAN: Protection for single link failure using VLAN ID*

Even though pro-PATH achieves a fast failure recovery and promises a wide applicability, it still has a shortage which is high consumption of TCAMs as backup paths are calculated for flows. In order to reduce the required amount of TCAMs while maintain advantages of pro-PATH, we propose another method named pro-VLAN (protection using VLAN id). In brief, Pro-VLAN cuts the relation between backup paths and flows, and calculates a backup path for each link only. The details of pro-VLAN are described as follows.

After mastering the connection relationships among switches, the controller then calculates a backup path for each link[f] of the topology by assuming that the link itself fails. As long as backup paths are calculated, all the relevant forwarding entries can be established before the generation of flows. In order to distinguish from working flow entries, backup flow entries use VLAN ids to be the flow-matches. Note the VLAN id assigned to each link is globally unique in the whole network topology. Moreover, a backup flow entry has a higher priority than a working flow entry. For the head of a backup path, a group entry is needed, in which two action buckets are used to store the next hops of the working path and the backup path, respectively. Specifically, the second action bucket contains the action of pushing the link's

[f]Note that the discussion of a link is in conjunction with a direction, hence the link S2-S1 implies the direction is from S2 to S1.

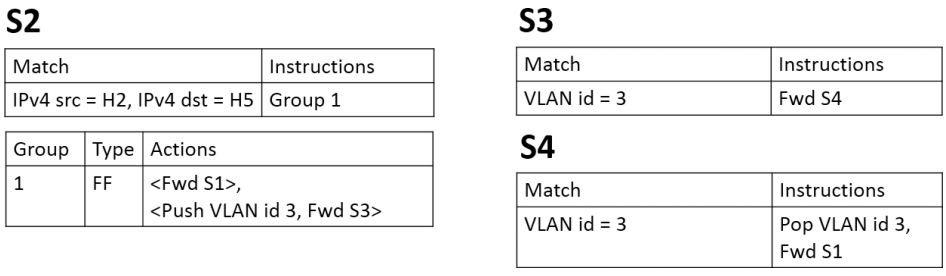


Fig. 3. Forwarding entries using pro-VLAN to protect the link S2-S1 of the flow from H2 to H5.

corresponding VLAN id. For instance, switch S2 on link S2-S1's backup path establishes a group entry as shown in the bottom left corner of Fig. 3. For the intermediate switches of a backup path, a backup flow entry is needed, in which the flow-match is the link's corresponding VLAN id, and the instruction set is to forward to the next hop of the backup path (the backup flow entry established on S3 is shown in the top right corner of Fig. 3). Specifically, the penultimate hop of a backup path is required to strip the VLAN id before forwarding to the next hop (the backup flow entry established on S4 is shown in the bottom right corner of Fig. 3). As a result, the edge switch of the backup path (which equals to the tail of the protected link) then can forward normal packets (without the VLAN id) along the working path again.

When new flows are generated, working paths are calculated and all the remaining forwarding entries are established. For each switch on a working path (except the tail switch), a working flow entry is established, and its instruction set is to jump to the protected link's group entry. For instance, switch S2 on the working path for the flow from H2 to H5 establishes a working flow entry as shown in the top left corner of Fig. 3. It can be concluded that the packet which walks along a backup path is pushed a VLAN id at the first hop, and stripped the VLAN id at the penultimate hop.

After introducing the two algorithms, the similarities and differences between pro-PATH and pro-VLAN are summarized as follows.

(1) When a flow is generated, two methods both use the shortest path algorithm to calculate the working path.
(2) Two methods both calculate backup paths for links instead of flows (even though pro-PATH calculates the backup path for each link composing the working path).
(3) Pro-PATH calculates backup paths after a flow is generated, while pro-VLAN calculates all the backup paths before the generation of any flow.
(4) The flow-match of a backup path using pro-PATH is a pair of IP addresses and an ingress port, while that using pro-VLAN is a VLAN id.
(5) Both backup flow entries and group entries using pro-PATH are related to flows, while those using pro-VLAN are independent from flows.

The detailed algorithm is described in Algorithm 1. It begins by calculating a backup path and assigning a VLAN id for each link (lines 1–4 and 20, 21). Next, pro-VLAN establishes a backup flow entry[g] for each intermediate switch (lines 5–9), a special backup flow entry (popping a VLAN id before forwarding packets) for each penultimate hop (lines 10–14), and a group entry[h] for each head on backup paths (lines 15–19). When new flows are generated, forwarding paths are calculated (lines 22–24 and 40). Finally, pro-VLAN installs a working flow entry (lines 25–29) and a group entry (lines 30–34) for each head, and a special working flow entry (sending to the destination host directly) for each last hop of working paths (lines 35–39).

After establishing these forwarding entries, packets from H2 to H5 (shown in Fig. 1) can still reach S1 along the backup path, even though link S2-S1 fails. If the failed link recovers, packets can return to the working path to reach S1 as long as the status of the group entry's first action bucket on S2 changes to "active" again. It can be observed that all the procedures of switching paths (from a working path to a backup path, or vice-versa) are completed on the data plane locally and automatically without the participant of the controller, hence pro-VLAN can realize a fast failure recovery. Besides, the principle of pro-VLAN which is to calculate a backup path for each link only brings us other benefits described as follows:

(1) Pro-VLAN can reduce the consumption of TCAMs when compared to other protection-based methods[14,16,18] which calculate backup paths dependent on flows, as the amount of links is fairly less than that of flows.
(2) Pro-VLAN can keep the amount of backup flow entries per switch stable regardless of the network scale and the flow diversity, as this amount is only related to the amount of links per switch.
(3) Pro-VLAN can recover any single link failure if the network is still reachable after a link failure.

Moreover, it can be observed from Algorithm 1 that pro-VLAN does not modify any hardware or protocol. As a result, pro-VLAN can recover all the single link failures, with high efficiency, strong scalability and wide applicability.

Pro-VLAN still has restrictions, and the corresponding explanations or solutions are introduced next. Note that the VLAN id field only covers 12 bits, it may be exhausted when applied to a large-scale network. Moreover, to avoid collisions, the VLAN id field cannot be occupied for other purposes. In these situations, we can use another tuple or even combinations of multiple tuples in the OpenFlow protocol, such as the MPLS label field which covers 20 bits.[33] In addition, a new self-defined tuple only used for failure recovery can be added to the OpenFlow protocol. It can

[g] In the pseudo code, the structure of a flow entry such as "(*vid*; send to the next hop of *bp*)" means that *flow-match* field is "*vid*", and *instructions* field is "send to the next hop of *bp*".
[h] In the pseudo code, the structure of a group entry such as "(send to the next hop of *l*; push *vid* and send to *nh_bp_l*)" means that the action of the first action bucket is "send to the next hop of *l*", and the actions of the second action bucket are "push *vid* and send to *nh_bp_l*".

Algorithm 1. Pro-VLAN: Fast Recovery of a Single Link Failure.

Require:
The set of switches, ports, links of the topology;
The set of flows;

for each link l **do**
  calculate a backup path bp for l, and the first next hop of bp: nh_bp_l;
  allocate a VLAN id vid to l;
  record the mapping relation between vid, np_bp_l and l;
  for all the switches on bp except the head and last two hops **do**
    generate a flow entry: (vid; send to the next hop of bp);
  end for
  if the penultimate hop of bp **then**
    generate a flow entry: (vid; pop vid and send to the next hop of bp);
  end if
  for each l's head **do**
    generate a group entry: (send to the next hop of l; push vid and send to nh_bp_l);
  end for
  vid=vid+1;
end for
for each flow f **do**
  calculate the shortest path p for f;
  find out all the links (l_p) contained in p;
  for each l_p's head **do**
    generate a flow entry: (IPv4 source and destination addresses; jump to a group entry);
  end for
  for each l_p's head **do**
    generate a group entry: (send to the next hop of l_p; push vid and send to the next hop of l_p's backup path);
  end for
  if the last one hop of p **then**
    generate a flow entry: (IPv4 source and destination addresses; send to the target host);
  end if
end for

.

eliminate the potential ambiguity of existing tuples, while the price must be paid, which is to modify the OpenFlow protocol.

Similar to other protection-based methods, pro-VLAN cannot promise packets to be forwarded on shortest paths after a link failure. The purpose of pro-VLAN is to

recover link failures as quickly as possible, rather than finding an optimal backup path. However, through a combination of the restoration mechanism with pro-VLAN, the problem can be solved. As port down messages can be transferred to the controller if a link fails, then controller can update the topology information and recalculate new shortest paths for affected flows which involve the failed link into working paths. Simultaneously, the controller can recalculate new backup paths for affected links which involve the failed link into backup paths. Therefore, any link failure can be recovered quickly, and packets can use the shortest paths to reach destinations after the controller's handling. Note that when two links fail successively, the second failed link may lead to continuous packets loss, until the first failed link has been recovered by restoration (as the backup path for the second failed link may involve the first failed link).

3.3. *Extended pro-VLAN: Recovery of multiple link failures*

A more severe situation for the flow from H2 to H5 is that links S2-S1 and S1-S4 fail simultaneously (shown in Fig. 1). As the link S2-S1 fails, its backup path (S2-S3-S4-S1) is enabled and packets can be forwarded to S4 with a VLAN id. Then packets reaching S4 executes the flow entry with VLAN id 3 to be the flow-match, and can only drop packets since S1 cannot be reached. However, an interesting phenomenon is that the path from S4 to H5 is valid, but the corresponding (working) flow entry is disabled as its priority is lower than the backup flow entry. In order to handle this situation, a new strategy needs to be added to our algorithm, which lets packets pop the VLAN id if the next hop of the backup path cannot be reached.

A modification of pro-VLAN is described as follows. Firstly, when S4 finds the next hop of the backup path (S1) cannot be reached, it returns packets to the previous hop of this backup path (S3). Secondly, when S3 finds packets coming from the reverse direction of the backup path (S4), it pops the VLAN id and sends them out from the ingress port again. Thirdly, S4 receives packets again but without any VLAN id, and then forwards packets to H5 along the working path. It can be observed that S3 can receive packets with VLAN ids from two directions, and packets coming from different directions correspond to different instruction sets, hence two backup flow entries are established on this switch as shown in Fig. 4. The backup flow entry which involves an ingress port into the flow-match has a higher priority than the flow entry which only involves a VLAN id into the flow-match. Moreover, S4 needs a group entry instead of a flow entry to handle packets on the backup path, and it stores the backup path's next hop and previous hop into first two action buckets, respectively. Using the modified version of pro-VLAN, packets will be forwarded between S3 and S4 for multiple times, but it does not incur a forwarding loop, as the same switch uses different flow entries or different action buckets (in group entries) to handle each packet by judging whether a VLAN id is involved, and what the ingress port is. Note that pro-VLAN recovers multiple link

S2

Match	Instructions
IPv4 src = H2, IPv4 dst = H5	Group 1

Group	Type	Actions
1	FF	<Fwd S1>, <Push VLAN id 3, Fwd S3>

S3

Match	Instructions
VLAN id = 3	Group 2
VLAN id = 3, port = port connecting to S4	Pop VLAN id 3, Fwd S4

Group	Type	Actions
2	FF	<Fwd S4>, <Fwd S2>

S4

Match	Instructions
VLAN id = 3	Group 3

Group	Type	Actions
3	FF	<Fwd S1>, <Fwd S3>

Fig. 4. Forwarding entries using pro-VLAN to protect the links S2-S1 and S1-S4 of the flow from H2 to H5.

failures conditionally. Take two link failures as an illustration, it requires that both the two links to be on a working path, and one link's backup path to pass through another link. The modified part of Algorithm 1 is described in Algorithm 2, and corresponding explanation is given in the annotation of this algorithm.

Algorithm 2. Pro-VLAN: Fast Recovery of Multiple Link Failures Using VLAN id.

/* for all the switches on *bp* except the head and tail, change the instructions from "send to the next hop of *bp*" to "jump to a group entry", and add a group entry as follows */
for all the switches on *bp* except the head and last two hops **do**
    generate a group entry: (send to the next hop of *bp*; send to the previous hop of *bp*);
end for
if the penultimate hop of *bp* **then**
    generate a group entry: (pop *vid* and send to the next hop of *bp*; send to the previous hop of *bp*);
end if
/* for all the switches on *bp* except the tail, add a flow entry as follows */
for all the switches on *bp* except the tail **do**
    generate a flow entry: (*vid* and ingress port; pop *vid* and send to the next hop of *bp*);
end for

4. Evaluation

4.1. *Simulation setup*

We adopt Mininet 2.1.0+[34] to simulate SDN networks and select Ryu[35] as the controller. All the softwares have been upgraded to support the utilization of group tables. Restoration[14] is selected to compare with pro-PATH and pro-VLAN to validate the effectiveness of pro-PATH and pro-VLAN in terms of the recovery time and the amount of forwarding entries. Note the restoration method for experiments works as follows. As long as a link fails, a port down message is generated by an affected switch and transferred to the controller. The controller then updates the network topology, calculates new forwarding paths for affected flows and updates forwarding entries on corresponding switches.

Ideally, three methods can be applied to all the network topologies, including fat-trees, grid networks and other topologies that have been deployed around the world such as the topology developed within the COST 266 project.[14] Among all the network topologies, grid network[i] is adopted in our simulation with the following considerations.

(1) Unlike other network topologies, the scales of grid networks and fat-trees are easy to change.
(2) In a fat-tree, it is more appropriate to match on an ingress port and a destination address to achieve failure recovery due to the fat-tree's inherent structure, while a grid network does not have similar special handling for failure recovery.

Figure 5 shows a representative grid network, which has 64 switches and hosts. Each host connects to the corresponding switch by the same number. In this figure, all the devices represent switches and hosts are not drawn.

In the experiments, we choose the out-of-band network type with the following considerations.

(1) As protection-based methods calculate backup paths and establish forwarding entries before any link failure, pro-PATH and pro-VLAN can be applied to both out-of-band and in-band networks without distinction.
(2) When restoration is applied to an in-band network, a link failure can lead to a disconnection between one switch of the failed link and the controller if without any extra functions added to switches.
(3) If a new function such as flooding the control traffic[26] is added to switches, it is time-consuming for affected switches to reconnect to the controller before the controller updates forwarding entries.

[i]In fact, we have also run experiments in fat-trees to measure the recovery time and the amount of forwarding entries among three methods: restoration, pro-PATH and pro-VLAN, and the experimental procedures and simulation results in fat-trees are similar to those in grid networks.

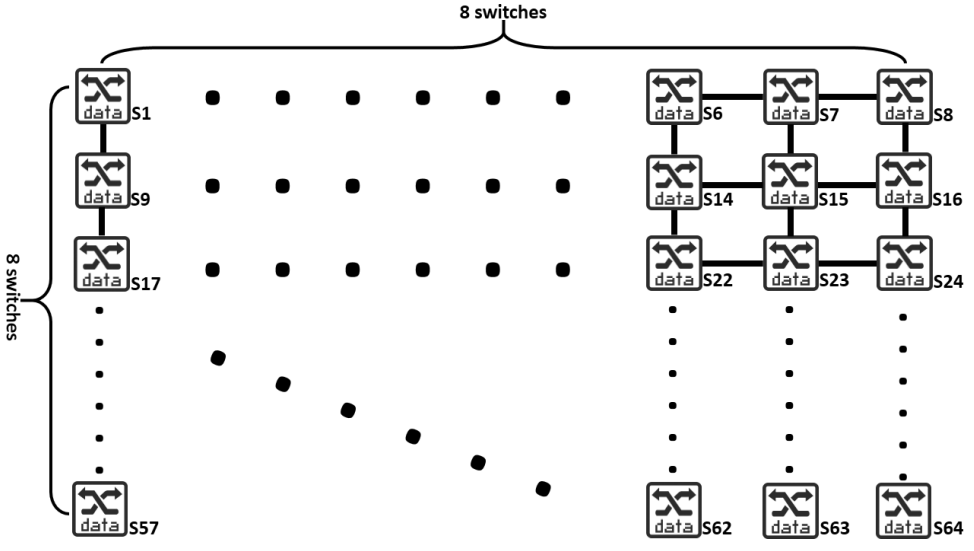


Fig. 5. A grid network with 64 switches.

As a result, restoration leads to an increased recovery time in an in-band network than in an out-of-band one, while protection-based methods can keep the recovery time stable no matter what the network type is.

4.2. *Simulation results*

4.2.1. *Recovery time*

In the first set of simulations, we test a square grid network with 64 switches and hosts as shown in Fig. 5, and measure the recovery time of three methods: restoration, pro-PATH and pro-VLAN. The recovery time is estimated by the amount of lost packets. Since a broken link may affect multiple flows simultaneously, the maximum number of lost packets is used to estimate the recovery time. We set ping requests to be generated every 1 ms, hence the recovery time is estimated to be x ms if x packets are lost, with an error less than 1 ms. Considering that restoration needs tens of milliseconds on the average to recover a single link failure, we set 1 ms to be the frequency.

In order to obtain an accurate simulation result, 250 tests are performed for each method to measure the recovery time. The 250 tests are equally divided into five groups and different groups have different step sizes. In a group, the step size is fixed and used as follows. In a group with the step size of 33, H1 pings H34, H2 pings H35 and so on. Note that a modular arithmetic is needed if the number of a receiver is larger than 64. In a test, each host sends 10,000 ping requests to another one every 1 ms, and we randomly break a link when all the hosts are communicating

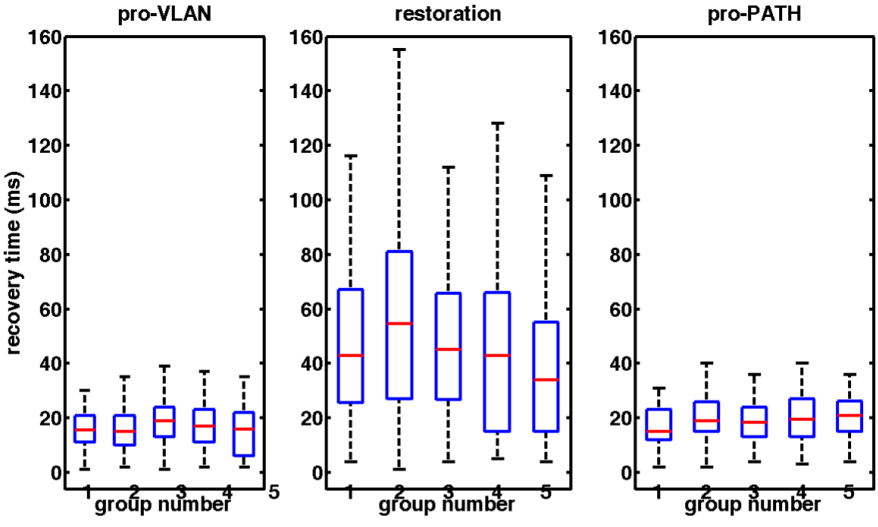


Fig. 6. Recovery time using three methods.

simultaneously. After a link is broken, we record the maximum number of lost packets to estimate the recovery time.

Figure 6 shows the experimental results of recovery time by using three methods. The figure is composed of three sub-figures, each sub-figure contains five box plots, and each box plot is drawn by gathering the data of 50 tests in the same group. Note that each time the step size is randomly decided, hence the same group number of different methods does not mean the same step size. In each box plot, five numerical points are displayed, which are the maximum value (black whisker above the box), 75th percentile (the top of the box), 50th percentile (the band inside the box), 25th percentile (the bottom of the box) and minimum value (black whisker under the box) from top to bottom, respectively. To make the results simpler to observe and compare, the exception values are removed from all the box plots. Two conclusions can be derived from this figure.

Firstly, two protection-based methods recover link failures more quickly than restoration. In the grid network with 64 switches and hosts, the average recovery times for pro-VLAN, pro-PATH and restoration are 20.18 ms, 21.83 ms and 49.71 ms, respectively. Hence two protection-based methods only consume 40% of the recovery time required by a restoration method. Moreover, the maximum recovery time for two protection-based methods does not exceed 40 ms, while that for restoration is even close to 160 ms.

Secondly, it can be observed that the recovery time of two protection-based methods is more stable than that of restoration. The reason is explained as follows. Even though in the same topology, the amount of flows affected by a failed link varies. If restoration is used, the controller can only handle flows one by one, hence

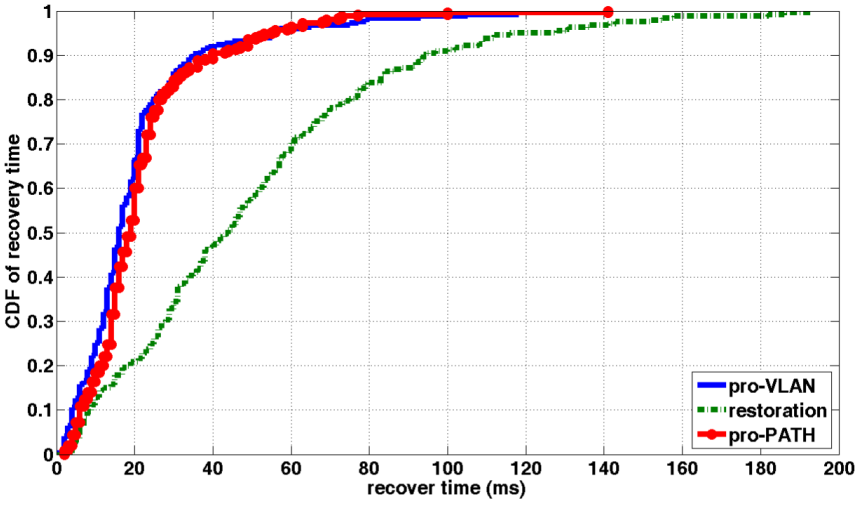


Fig. 7. Cumulative distribution function of recovery time using three methods.

the recovery time will vary as the amount of affected flows varies. On the contrary, if protection is used, each affected flow can switch to the backup path automatically and locally, hence the recovery time can keep stable no matter how many affected flows are.

Moreover, Fig. 7 shows a cumulative distribution function of recovery time by using three methods. It can be observed that if the recovery time is set to be less than 50 ms, more than 90% data of two protection-based methods are located in this range, while the percent does not reach a passing grade (less than 60 percent) when using restoration. Combining these two figures (Figs. 6 and 7), it can be concluded that protection-based methods including pro-VLAN and pro-PATH always recovers link failures more quickly and more stably than restoration.

4.3. *Amount of forwarding entries*

In the second set of experiments, two protection-based methods are compared in terms of the amount of forwarding entries, including the amounts of working flow entries, backup flow entries, whole flow entries and group entries. Note pro-VLAN used in the experiments is the basic algorithm to recover a single link failure, with an optimization of group entries. The network topologies are still grid networks with different scales. The smallest grid network contains 10 switches and hosts, which is composed of 2 rows and 5 columns. Each time we add a new row (5 switches and hosts) to the topology. We use totally 12 grid networks, and the largest grid network contains 65 switches and hosts. In each grid network, we run a "ping-all" command to produce flows. When a "ping-all" command is executed, each host will send ping requests to all the other hosts. For instance, in the grid network with 65 switches and

S2

Match	Instructions
IPv4 src = H2, IPv4 dst = H5	Group 1

Group	Type	Actions
1	FF	<Fwd S1>, <Push VLAN id 3, Fwd S3>

S3

Match	Instructions
VLAN id = 3	Fwd S4

S4

Match	Instructions
VLAN id = 3	Pop VLAN id 3, Fwd S1

S1

Match	Instructions
IPv4 src = H2, IPv4 dst = H5	Group 2
IPv4 src = H2, IPv4 dst = H5, input port = port connecting to S4	OFPP_IN_PORT

Group	Type	Actions
2	FF	<Fwd S4>, <Push VLAN id 4, Fwd S2>

Fig. 8. Forwarding entries using pro-VLAN to protect the link S2-S1 of the flow from H2 to H5, considering the use of "OFPP_IN_PORT."

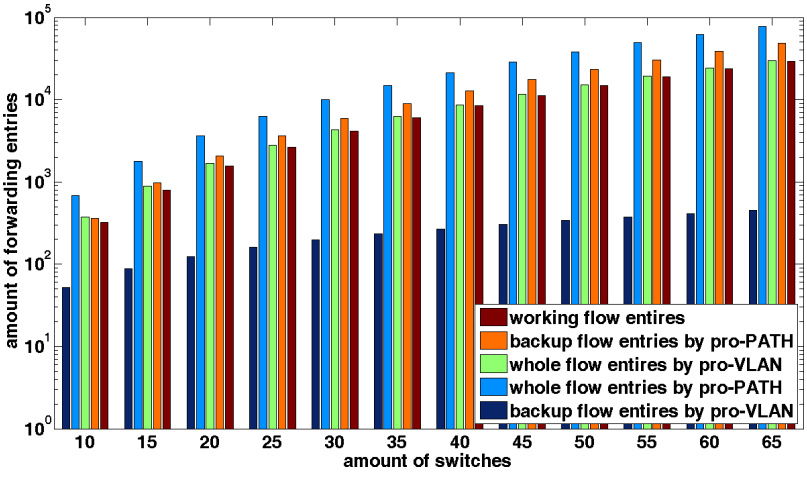
hosts, a "ping-all" command will generate 4160 ($65 \times 64 \div 2$) flows. After executing this command, the controller is responsible for counting the amount of forwarding entries.

To make results simpler to observe and analyze, we do not consider the use of the OFPP_IN_PORT and the modified algorithm to recover multiple link failures in this set of experiments and the mathematical analysis (described in Sec. 5). According to the OpenFlow standard, the OFPP_IN_PORT must be explicitly used if packets should be sent out from the ingress port.[32] Imagine a flow from H2 to H5 (shown in Fig. 1), the working path is S2-S1-S4-S5. If link S2-S1 fails, packets will use the backup path S2-S3-S4-S1 to reach S1, while the next hop for S1 is S4 again. In this situation, a new working flow entry should be established on S1 as shown in Fig. 8 (imagine VLAN id 4 is assigned to the link S1-S4), with an ingress port into consideration. However, as the rule of using the OFPP_IN_PORT is equivalent for two protection-based methods, then the experimental results will not be influenced if the rule is removed. As a result, no matter whether the ingress port is the same as the output port, we only write a specific port number into the *instructions* field, which means we only use one flow entry shown in the left side of Fig. 8 to represent two flow entries in the figure.

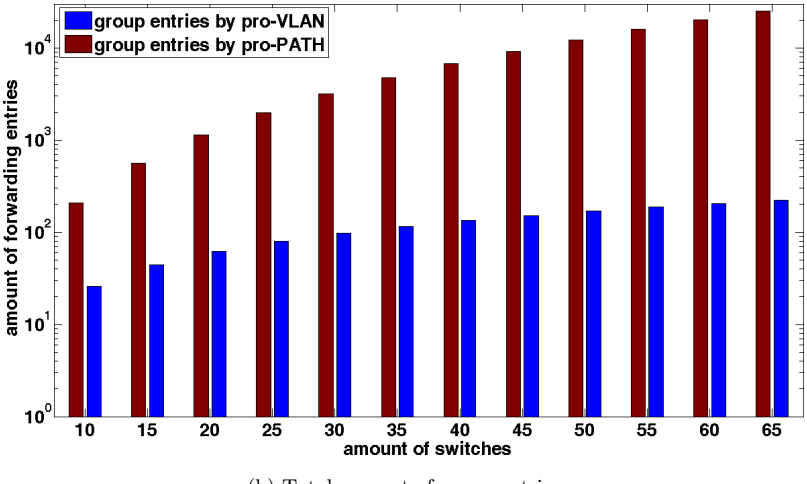
Figure 9 shows the experimental results of forwarding entries' amount by using pro-PATH and pro-VLAN. The total amount of flow entries (including working flow entries, backup flow entries and whole flow entries), the total amount of group entries, the average amount of backup flow entries, and the average amount of group entries are depicted in four sub-figures of Fig. 9, respectively. Three conclusions can be derived from this figure.

Firstly, when comparing the amounts of two methods' backup flow entries, pro-VLAN shows a significant advantage over pro-PATH. If we draw curves by

connecting peaks of 12 bars in the same color, it can be observed that the curve representing pro-PATH's backup flow entries seems like a power function, while the one representing pro-VLAN's backup flow entries seems like parallel to the x axis. As pro-PATH needs to protect each link contained in each flow's working path, then the amount of backup flow entries is decided by the amount of flows and that of links contained in each flow's working path. On the contrary, the amount of backup flow entries by using pro-VLAN is only related to the amount of links. As a result, a huge difference on the amount of backup flow entries generates as shown in this figure.

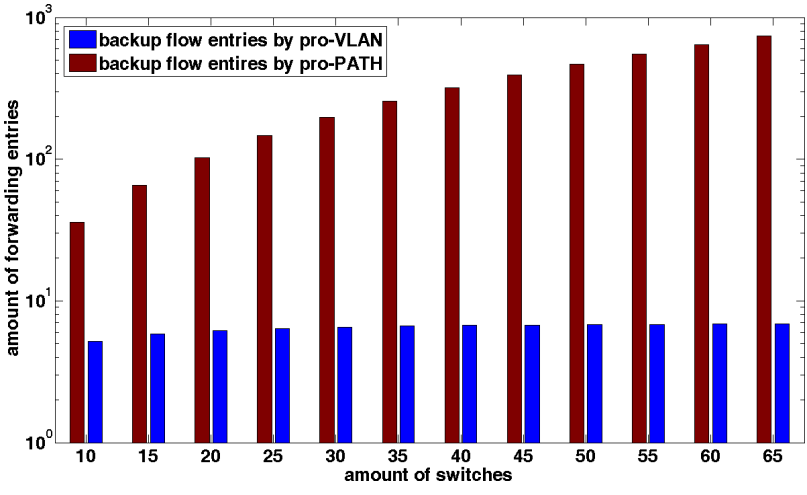


(a) Total amount of flow entries

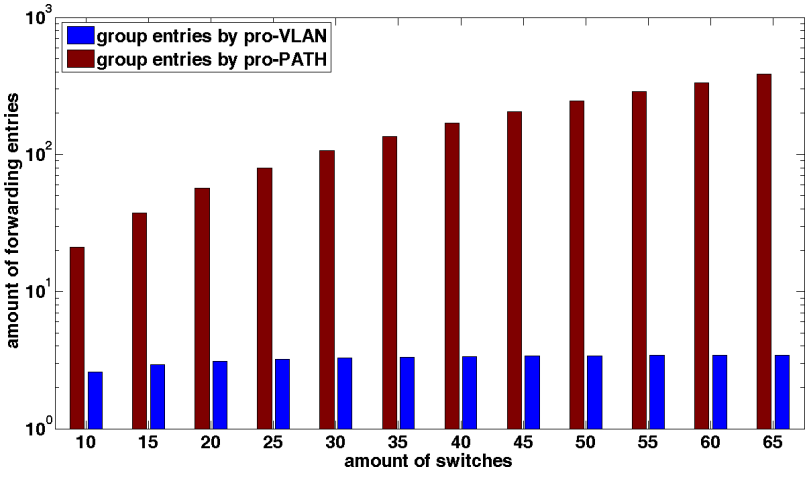


(b) Total amount of group entries

Fig. 9. Forwarding entries' amount using two protection methods via experiments.



(c) Average amount of flow entries



(d) Average amount of group entries

Fig. 9. (*Continued*)

In addition, two curves representing group entries are similar to those representing backup flow entries.

Secondly, pro-PATH even requires more forwarding entries for storing backup paths than for storing working paths. For instance, if a flow contains five hops (four links), then five working flow entries are established, and four links need to be protected. In a grid network, each link corresponds to two backup flow entries. As a result, in order to protect this working path, eight backup flow entries are needed

(4×2). As long as a working path contains more than two hops, the amount of backup flow entries will exceed that of working flow entries. As a result, the amount of backup flow entries is larger than that of working flow entries by using pro-PATH.

Thirdly, the amount of working flow entries is almost similar to that of pro-VLAN's whole flow entries. The reasons can be summarized into two points. 1. The amount of links is less than that of flows. 2. A link to be protected just needs two switches to establish backup flow entries, while a flow may need up to 13 switches to establish working flow entries. As a result, by using pro-VLAN, the amount of backup flow entries is fairly less than that of working flow entries, and even can be ignored.

For instance, in the gird network with 10 switches and hosts, pro-VLAN consumes 52 flow entries and 26 group entries to store backup paths, while pro-PATH consumes 358 flow entries and 210 group entries for the same purpose. In the grid network with 65 switches and hosts, pro-VLAN consume 448 flow entries and 224 group entries to store backup paths, while pro-PATH consumes 48,252 flow entries and 24,960 group entries for the same purpose. When the network size increases, the multiple relation between pro-PATH and pro-VLAN increases from less than 10 times to more than 100 times. It can be concluded that pro-VLAN always performs better than pro-PATH on the consumption of backup flow entries and group entries, especially for a large scale network.

Moreover, in the grid network with 10 switches and hosts, pro-VLAN consumes 320 working flow entries, and in the grid network with 65 switches and hosts, this method consumes 29,250 working flow entries. When the network size increases, the multiple relation between the amount of working flow entries and that of backup flow entries increases from about six times to about 65 times. It can be concluded that when compared to the amount of working flow entries, the amount of backup flow entries is low and even can be ignored.

In addition, the average amount of pro-VLAN's backup flow entries increases from 5.2 to 6.89 in 12 grid networks with different scales, which is a slight increase. On the contrary, the average amount of pro-PATH's backup flow entries increases from 35.8 to 742.34. Similarly, the average amount of pro-VLAN's group entries increases from 2.6 to 3.45, and that of pro-PATH's group entries increases from 21 to 384. When we further increase the amount of flows in the same topology, for instance, doubling the amount of flows, pro-VLAN can keep all these backup forwarding entries stable, while pro-PATH will require double backup forwarding entries. It can be explained as pro-VLAN calculates a backup path for each link only, then the average amount of backup forwarding entries has no relation to the amount of flows, or the network scales (of grid networks), while pro-PATH calculates a backup path for each flow independently. As a result, pro-VLAN has a strong scalability, as the amount of backup forwarding entries is fairly less than that of working flow entries, and the average amount of backup forwarding entries can keep stable no matter how the network scale extends or how the amount of flows increases.

5. Mathematical Analysis

In this section, we use the mathematical analysis to calculate the amount of forwarding entries. The topologies are still the grid networks, and methods contain pro-PATH, pro-VLAN and pro-BFD.[j]

We first need to introduce pro-BFD in brief. When a new flow emerges, the controller calculates two disjoint paths for this flow, and adopts BFD to monitor the status of these paths. When the controller distributes forwarding rules among the data plane, both flow entries and group entries are used to store forwarding information. Once BFD declares the failure in the working path, the action bucket associated with this path in the group entry is made unavailable. As a result, the failure of each monitored path can be recovered automatically and locally in the data plane using pro-BFD.

The calculation can be carried out in seven steps totally, including computing the working flow entries' amount (described in **Step 1**), the backup flow entries' amount for pro-PATH, pro-VLAN and pro-BFD (described in **Step 2**–**4**, respectively), and the group entries' amount for the three methods (**Step 5**–**7**, respectively).

Step 1: The amount of working flow entries is calculated, which equals to the product of multiplying the amount of flows and the average hops of a flow. Imagine a grid network composed of M rows and N columns. The amount of hosts or switches is MN, and the needed hops for a node with a coordinate (x, y) to visit all the nodes are

$$\sum_{m=1}^{M}\sum_{n=1}^{N}|x-n|+|y-m|+1\,. \tag{1}$$

Then, the total hops for all the nodes to traverse the whole grid network are

$$\sum_{x=1}^{N}\sum_{y=1}^{M}\sum_{m=1}^{M}\sum_{n=1}^{N}|x-n|+|y-m|+1\,. \tag{2}$$

Note that the amount of flows is A_{MN}^2 (here "A" means a permutation operation), which means a "ping-all" command is executed. As a result, the amount of working flow entries is equal to the total hops for all the nodes to traverse the whole grid network, and the average hops of a flow are

$$\frac{\sum_{x=1}^{N}\sum_{y=1}^{M}\sum_{m=1}^{M}\sum_{n=1}^{N}|x-n|+|y-m|+1}{A_{MN}^2}\,. \tag{3}$$

Step 2: The amount of pro-PATH's backup flow entries is calculated. Similarly, the average length of a backup path needs to be confirmed first. (1) As pro-PATH

[j] As the function of utilizing BFD to detect the status of a path is not supported by Mininet, pro-BFD[14] is not realized in experiments.

calculates a backup path for each link contained in a working path, then the average amount of links contained in a working path is equal to average hops minus one. (2) In a grid network, a link's backup path contains four hops, and only two intermediate switches on a backup path need to establish backup flow entries (as two edge switches are on the working path as well, then they should establish working flow entries and group entries). (3) As a result, the amount of backup flow entries is equal to the product of the amount of flows, the average amount of links, and the hops (of each link's backup path) needed to establish backup flow entries, which is

$$\left(\frac{\sum_{x=1}^{N} \sum_{y=1}^{M} \sum_{m=1}^{M} \sum_{n=1}^{N} |x-n| + |y-m| + 1}{A_{MN}^2} - 1 \right) \times 2 \times A_{MN}^2 . \tag{4}$$

Note this is an upper limit. A special situation should be noticed. For instance, the flow from H1 to H16 has an inflection point on the working path: S8. The relevant links are S7-S8 and S8-S16, and their backup paths are S7-S15-S16-S8 and S8-S7-S15-S16, respectively. As S15 acts as the intermediate switch for two backup paths simultaneously, and has absolutely the same flow-match and instruction set, then only 15 backup flow entries instead of 16 are needed for this flow. However, not all the flows with inflection points can reduce one backup flow entry. For instance, the flow from H9 to H23 has an inflection point on the working path as well: S15, while the backup paths for links S14-S15 and S15-S23 are S14-S6-S7-S15 and S15-S14-S22-S23, respectively. It can be observed that four intermediate switches on backup paths are absolutely different. As a result, if a flow has an inflection point on the working path composed on n links, the needed amount of backup flow entries is $2n-1$ to $2n$. To make our discussion simpler, we calculate the lower limit by supposing that each flow has $2n-1$ backup flow entries, which is

$$A_{MN}^2 \times \left(\left(\frac{\sum_{x=1}^{N} \sum_{y=1}^{M} \sum_{m=1}^{M} \sum_{n=1}^{N} |x-n| + |y-m| + 1}{A_{MN}^2} - 2 \right) \times 2 + 1 \right) . \tag{5}$$

In the same grid network, the experimental result is located between the lower limit and the upper limit.

Step 3: The amount of pro-VLAN's backup flow entries is calculated. The amount of links is

$$M \times (N-1) + N \times (M-1) . \tag{6}$$

Each link corresponds to two directions, and each link of one direction corresponds to two backup flow entries (established on the intermediate switches). Hence, the

amount of pro-VLAN's backup flow entries is

$$4 \times (M \times (N-1) + N \times (M-1)). \tag{7}$$

Step 4: The amount of pro-BFD's backup flow entries is calculated. When pro-BFD is used, a backup path is always two hops longer than its corresponding working path in a grid network. Because only the intermediate switches need to establish backup flow entries, the amount of pro-BFD's backup flow entries is then equal to the amount of working flow entries. The amount of group entries for three protection-based methods can be computed in a similar way.

Step 5: The amount of pro-PATH's group entries is calculated. The head of each link involved in a working path needs a group entry for failure recovery, the amount of pro-PATH's group entries is then equal to the product of the amount of flows and the average amount of links contained in a working path, that is

$$A_{MN}^2 \times \left(\frac{\sum_{x=1}^{N} \sum_{y=1}^{M} \sum_{m=1}^{M} \sum_{n=1}^{N} |x-n| + |y-m| + 1}{A_{MN}^2} - 1 \right). \tag{8}$$

Step 6: The amount of pro-VLAN's group entries is calculated. It is equal to the amount of links with directions to be considered since the head of each link needs a group entry, and can be expressed as

$$2 \times (M \times (N-1) + N \times (M-1)). \tag{9}$$

Step 7: The amount of pro-BFD's group entries is calculated. As the head of each working path needs a group entry, the amount of pro-BFD's group entries is then equal to the amount of flows.

After simplifying these formulas by removing the absolute value sign, the final results on the amount of forwarding entries for three protection-based methods can

Table 2. Forwarding entries' amount using three protection-based methods via mathematical analysis.

Amount of working flow entries	pro-PATH pro-VLAN pro-BFD	$\frac{1}{3}(M^3N^2 + M^2N^3 - M^2N - MN^2) + M^2N^2$
Amount of backup flow entries	pro-PATH	$\frac{2}{3}(M^3N^2 + M^2N^3 - M^2N - MN^2) - M^2N^2 + 3MN$
	pro-VLAN	$8MN - 4M - 4N$
	pro-BFD	$\frac{1}{3}(M^3N^2 + M^2N^3 - M^2N - MN^2) + M^2N^2$
Amount of group entries	pro-PATH	$\frac{1}{3}(M^3N^2 + M^2N^3 - M^2N - MN^2) + MN$
	pro-VLAN	$4MN - 2M - 2N$
	pro-BFD	$M^2N^2 - MN$

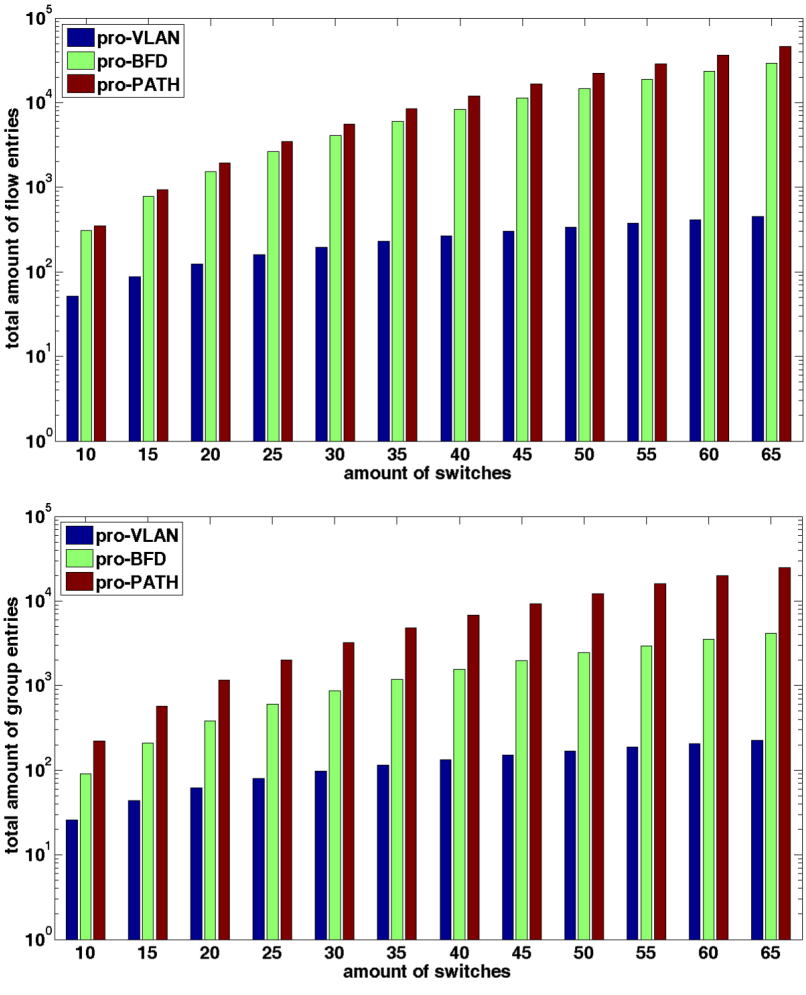


Fig. 10. Forwarding entries' amount using three protection-based methods via mathematical analysis.

be derived, which are also shown in Table 2 and Fig. 10. Note that the amounts of working flow entries, backup flow entries and group entries are displayed in the second, third and forth row of this table, and we use the lower limit when the amount of pro-PATH's backup flow entries is calculated and compared. It can be observed from this figure that pro-PATH consumes the maximum amount of forwarding entries, pro-BFD consumes the next maximum amount, and pro-VLAN consumes the least amount. Moreover, in square grid networks (M equals N), the amounts of backup flow entries required by pro-PATH, pro-BFD and pro-VLAN are M^5, M^5 and M^2 in order of magnitude, respectively. Similarly, the amounts of group entries required by three protection-based methods are M^5, M^4 and M^2, respectively. These

results indicate that pro-VLAN is the most efficient one among the three methods. In all, the mathematical analysis also verifies the simulation results.

6. Extended Discussion

6.1. *Recovery of a switch failure*

A more severe situation is that a switch crashes. In order to solve the recovery of a switch failure, we should calculate a backup path for each three-hop path as well as each link. To make our discussion simpler to understand, we calculate backup paths for all the three-hop paths, without the consideration whether a three-path is calculated through the shortest path algorithm. For instance, the path from S2 to S4 should be S2-S1-S4 instead of S2-S5-S4 (shown in Fig. 11), while we calculate backup paths for both of them. Taking S5 as an example, all of the three-hop paths (S2-S5-S4, S2-S5-S6, S2-S5-S8, S4-S5-S8, S4-S5-S6, S8-S5-S6) and their reverse directions need corresponding backup paths prepared against S5's failure. It should be noticed that the required amount of VLAN ids to cope with S5's failure can be up to 12 (A_4^2) since S5 has four ports to connect with other switches. If a switch has p ports to connect with other switches, the required amount of VLAN ids is A_p^2. Multiplying by the amount of switches, the required amount of VLAN ids is fairly high when compared to the VLAN ids for links only. However, in theory, as long as the following three-hop paths: S2-S5-S4, S4-S5-S8, S8-S5-S6, and S6-S5-S2 are established, any two switches next to S5 can reach each other. Similarly, as long as the corresponding

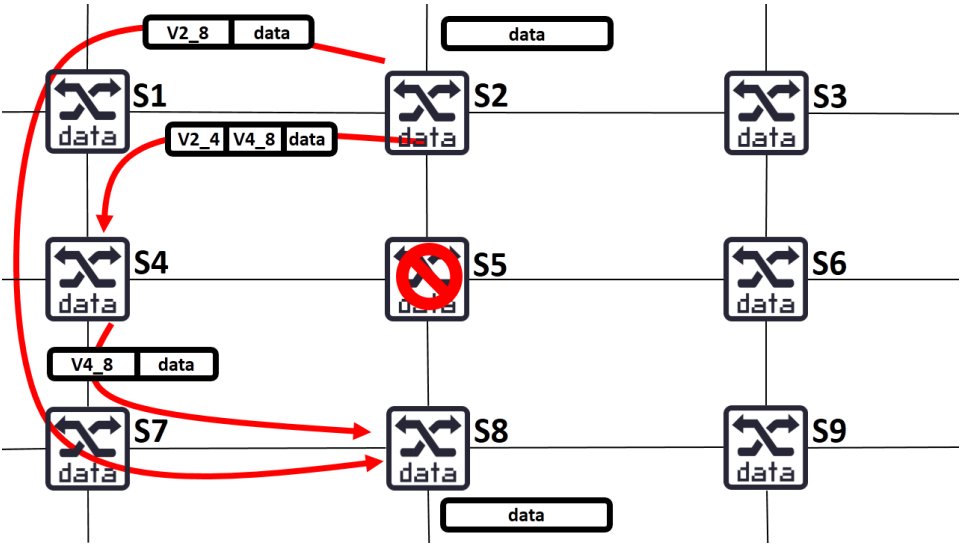


Fig. 11. Recovery of a switch failure.

backup paths of these three-hop paths have been calculated, any two switches next to S5 can still reach each other even though S5 fails. For instance, S2 can use the path S2-S5-S4-S5-S8 to reach S8, and can still reach S8 even though S5 fails, as long as the corresponding backup paths of S2-S5-S4 and S4-S5-S8 have been calculated. In order to realize this modified method designed for recovery of a switch failure, the controller needs to push and strip multiple VLAN ids in sequence, which is similar to the IEEE 802.1AD (QinQ) standard. The benefit is that the required amount of VLAN ids for a switch with p ports can be decreased from A_p^2 to p. The algorithm of the modified method is left for our future work.

6.2. *Recovery in a multicast scenario*

Figure 12 shows a multicast group composed of nine switches, and a multicast tree represented by bold lines. Note that hosts are not drawn while each of them connects to the corresponding switch by the same number. Among the hosts, H1 is the sender, and three hosts (H7, H8 and H9) in the bottom are receivers. In order to realize the multicast in SDN networks, each multicast flow is defined by using a sender's IP address and a multicast address, and an "all" group entry is used to support a multicast or broadcast forwarding. By combining with pro-VLAN, even though a link belonging to a multicast tree fails, multicast packets can still be forwarded to receivers. For instance, as shown in Fig. 12, H1 is required to forward multicast packets to three hosts in the bottom, in the situation where the link S1–S4 is prone to fail. The forwarding entries established on S1 for this purpose are shown in Fig. 13.

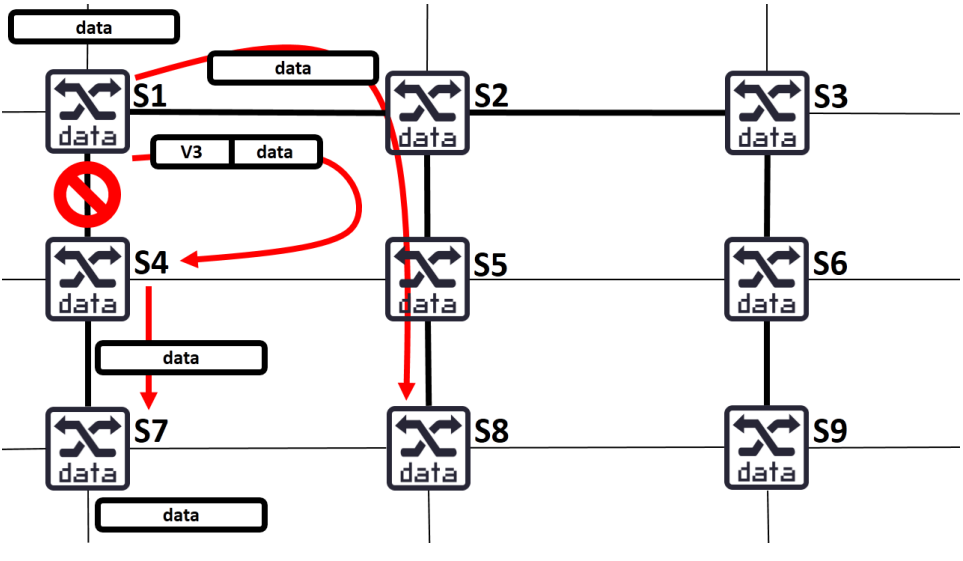


Fig. 12. Recovery in a multicast scenario.

S1

Match	Instructions
IPv4 src = H1, IPv4 dst = a multicast address	Group 1

Group	Type	Actions
1	all	<Group 2>, <Fwd S2>

Group	Type	Actions
2	FF	<Fwd S4>, <Push VLAN id 1, Fwd S2>

Fig. 13. Forwarding entries on s1 using pro-VLAN to protect the link S1-S4 of the multicast flow from H1 to three hosts in the bottom.

Note that no packet will enter a loop, as a normal packet and the packet used for failure recovery have different headers, and then match against different forwarding entries. However, in order to maintain the feature of a tree, we should try to transform the forwarding paths after a failure recovery into a new multicast tree, which is a part of our work in future.

7. Conclusion

In this paper, we propose two protection-based methods to recover a single link failure in SDN. The first method is named pro-PATH which calculates the working path for each flow and then calculates the backup path for each link composing the working path. Considering that the calculation and establishment of backup paths are dependent on flows, pro-PATH will consume a huge number of TCAMs for working paths and backup paths. In order to overcome this drawback, the second method named pro-VLAN is presented to calculate a backup path and assign a VLAN id for each link. By using pro-VLAN, any single link failure can be recovered with the advantages of high efficiency, strong scalability and wide applicability. The simulation results and mathematical analysis show that both pro-VLAN and pro-PATH achieve a fast failure recovery. They have the similar cost of the recovery time, while pro-VLAN consumes less than 1% of the forwarding entries to store backup paths as compared to pro-PATH. Moreover, when the network scale increases from 10 to 60 switches by 500%, the increase of the number of working flow entries per switch would be more than 1000%, while the increase of the number of backup flow entries per switch built by pro-VLAN is only less than 50%.

Acknowledgments

This work was supported by China Postdoctoral Science Foundation under Grant No. 2014M561438.

References

1. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J. Turner, Openflow: Enabling innovation in campus networks, *SIGCOMM Comput. Commun. Rev.* **38** (2008) 69–74.
2. D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, Software-defined networking: A comprehensive survey, *Proc. IEEE* **103** (2015) 14–76.
3. S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zhu, U. Hölzle, S. Stuart and A. Vahdat, B4: Experience with a globally-deployed software defined wan, *Proc. ACM SIGCOMM 2013 Conf. SIGCOMM* (Hong Kong, China, August 2013), pp. 3–14.
4. J. Biswas, A. A. Lazar, J. F. Huard, K. Lim, S. Mahjoub, L. F. Pau, M. Suzuki, S. Torstensson, W. Wang and S. Weinstein, The ieee p1520 standards initiative for programmable network interfaces, *IEEE Commun. Mag.* **36** (1998) 64–70.
5. R. Ramjee, K. Sabnani, T. V. Lakshman, T. Nandagopal and T. Woo, The softrouter architecture, *ACM SIGCOMM Workshop on Hot Topics in Networking*, San Diego, USA, November 2014, pp. 1–6.
6. J. Hadi Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, A. Doria and J. Halpern, Forwarding and control element separation (forces) protocol specification, *RFC-5810*, March 2010.
7. J. Zhou, M. Yin, Z. Li, K. Cao, J. Yan, T. Wei, M. Chen and X. Fu, Fault-tolerant task scheduling for mixed-criticality real-time systems, *J. Circuits Syst. Comput.* **26** (2017) 1750016.
8. H. Zhao, Q. Wang, K. Xiong and S. Pei, A path-counter method for fault-tolerant minimal routing algorithms in 2D mesh, *J. Circuits Syst. Comput.* **27** (2018) 1850054.
9. Z. Han and W. Jian, A fault diagnosis method based on active example selection, *J. Circuits Syst. Comput.* **27** (2018) 1850013.
10. P. Pan, G. Swallow and A. Atlas, Fast reroute extensions to rsvp-te for lsp tunnels, *RFC-4090*, May 2005.
11. D. Zhou and S. Subramaniam, Survivability in optical networks, *IEEE Netw.* **14** (2000) 16–23.
12. A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. N. Chuah, Y. Ganjali and C. Diot, Characterization of failures in an operational ip backbone network, *IEEE/ACM Trans. Netw.* **16** (2008) 749–762.
13. J.-P. Vasseur, M. Pickavet and P. Demeester, *Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*, (Elsevier, 2004).
14. S. Sharma, D. Staessens, D. Colle, M. Pickavet and P. Demeester, Openflow: Meeting carrier-grade recovery requirements, *Comput. Commun.* **36** (2013) 656–665.
15. V. Sekar, S. Ratnasamy, M. K. Reiter, N. Egi and G. Shi, The middlebox manifesto: Enabling innovation in middlebox deployment, *ACM Workshop on Hot Topics in Networks*, Cambridge, Massachusetts, November 2011, pp. 1–6.
16. R. M. Ramos, M. Martinello and C. Esteve Rothenberg, Slickflow: Resilient source routing in data center networks unlocked by openflow, *38th Annual IEEE Conf. Local Computer Networks*, Sydney, Australia, October 2013, pp. 606–613.
17. H. Huang, S. Guo, P. Li, W. Liang and A. Y. Zomaya, Cost minimization for rule caching in software defined networking, *IEEE Trans. Parallel Distrib. Syst.* **27** (2016) 1007–1016.
18. A. Sgambelluri, A. Giorgetti, F. Cugini, F. Paolucci and P. Castoldi, Openflow-based segment protection in ethernet networks, *IEEE/OSA J. Opt. Commun. Netw.* **5** (2013) 1066–1075.

19. J. Chen, J. Chen, J. Ling and W. Zhang, Failure recovery using vlan-tag in SDN: High speed with low memory requirement, *2016 IEEE 35th Int. Performance Computing and Communications Conference (IPCCC)*, Los Alamitos, CA, USA, 2016, pp. 1–9.
20. J. Chen, J. Chen, F. Xu, M. Yin and W. Zhang, When software defined networks meet fault tolerance: A survey, *15th Int. Conf. Algorithms and Architectures for Parallel Processing (ICA3PP 2015)*, Zhangjiajie, 18–20 November 2015, pp. 351–368.
21. F. Botelho, A. Bessani, F. M. V. Ramos and P. Ferreira, On the design of practical fault-tolerant sdn controllers, *2014 Third European Workshop on Software Defined Networks*, London, UK, September 2014, pp. 73–78.
22. F. A. Botelho, F. M. V. Ramos, D. Kreutz and A. N. Bessani, On the feasibility of a consistent and fault-tolerant data store for SDNs, *2013 Second European Workshop on Software Defined Networks*, Berlin, Germany, October 2013, pp. 38–43.
23. N. Katta, H. Zhang, M. Freedman and J. Rexford, Ravana: Controller fault-tolerance in software-defined networking, *Proc. 1st ACM SIGCOMM Symp. Software Defined Networking Research*, Santa Clara, California, June 2015, pp. 1–4.
24. M. Kuźniar, P. Perešíni, N. Vasić, M. Canini and D. Kostić, Automatic failure recovery for software-defined networks, *Proc. Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, Hong Kong, China, August 2013, pp. 1–6.
25. N. Beheshti and Y. Zhang, Fast failover for control traffic in software-defined networks, *2012 IEEE Global Communications Conf. (GLOBECOM)*, 3–7 December 2012, pp. 2665–2670.
26. S. Sharma, D. Staessens, D. Colle, M. Pickavet and P. Demeester, Fast failure recovery for in-band openflow networks, *Design of Reliable Communication Networks* (Budapest, Hungary), 4–7 March 2013, pp. 52–59.
27. M. Reitblatt, M. Canini, A. Guha and N. Foster, Fattire: Declarative fault tolerance for software-defined networks, *Proc. Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, Hong Kong, China, August 2013, pp. 109–114.
28. D. Staessens, S. Sharma, D. Colle, M. Pickavet and P. Demeester, Software defined networking: Meeting carrier grade requirements, *2011 18th IEEE Workshop on Local Metropolitan Area Networks (LANMAN)*, Chapel Hill, NC, USA, October 2011, pp. 1–6.
29. B. Stephens, A. L. Cox and S. Rixner, Scalable multi-failure fast failover via forwarding table compression, *Proc. Symp. SDN Research*, Santa Clara, CA, USA, March 2016, pp. 1–12.
30. H. Kim, M. Schlansker, J. R. Santos, J. Tourrilhes, Y. Turner and N. Feamster, Coronet: Fault tolerance for software defined networks, *2012 20th IEEE Int. Conf. Network Protocols (ICNP)*, Austin, TX, USA, October 2012, pp. 1–2.
31. B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka and T. Turletti, A survey of software-defined networking: Past, present, and future of programmable networks, *IEEE Commun. Surv. Tut.* **16** (2014) 1617–1634.
32. Openflow 1.0.0, http://archive.openflow.org/documents/openflow-spec-v1.0.0.pdf.
33. Openflow 1.1.0, http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf.
34. Mininet, http://mininet.org/.
35. Ryu, http://osrg.github.io/ryu/.