2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)

Proactive Failure Recovery in OpenFlow Based Software Defined Networks

V. Padma, P.Yogesh
Department of Information Science and Technology
CEG campus, Anna University
Chennai, India
padmajs80@gmail.com, yogesh@annauniv.edu

Abstract— **Software Defined Networking (SDN) is a network architecture that decouples the control and data planes. SDN enables network control to become directly programmable and the underlying infrastructure to be abstracted from the network services. The foundation for open standards based software defined networking is the OpenFlow protocol. The OpenFlow architecture which is originally designed for Local Area Networks (LANs), doesn't include effective mechanisms for fast resiliency. But metro, carrier grade Ethernet networks and industrial area networks have to guarantee fast resiliency upon network failure. This paper experiments the link protection scheme that aims to enhance the OpenFlow architecture by adding fast recovery mechanisms in the switch and the controller. This is achieved by enabling the controller to add backup paths proactively along with the working paths and enabling the switches to perform the recovery actions locally. As this avoids controller intervention during recovery, the recovery time solely depends upon the failure detection time of the switch. As this will be less compared to the switch-controller round trip time, this gives better results. The performance of the system is evaluated by finding the packet loss and switch over time and comparing it with the current OpenFlow implementations. The system performs reasonably better than the existing systems in terms of switch over time. However the number of backup path entries increase relatively.**

Keywords—OpenFlow; Fast resiliency; Failure recovery; Link protection; Software Defined Networking.

I. INTRODUCTION

Software Defined Networking (SDN) is a new networking paradigm in which the forwarding hardware is decoupled from control decisions. In traditional networks the control and data planes are combined in a network node. In this traditional approach, the flow management (forwarding policy) is decided by the control plane. The forwarding policy is then pushed down to the data plane and the data plane forwards the packets accordingly. Since the control plane and data plane are tightly coupled, the only way to change the forwarding policy is to reconfigure the devices. Hence it is difficult to scale the networks in response to changing traffic demands, increasing use of mobile devices, and the impact of "big data."

In SDN, control is moved out of the individual network nodes and pushed into the separate, centralized controller. SDN switches are controlled by a network operating system that collects information using the Application Programming Interface (API) and manipulates their forwarding plane. Thus the network operating system running at the controller provides an abstract model of the network topology to the OpenFlow Architecture. The controller can therefore exploit complete knowledge of the network to optimize flow management and support service-user requirements of scalability and flexibility.

OpenFlow is driven by the SDN principle of decoupling the control and data forwarding planes. This standardizes information exchange between the two planes. OpenFlow networks consist of following three components (1) OpenFlow Switch (2) OpenFlow Controller and (3) OpenFlow Protocol. This is shown in Fig 1.

OpenFlow Switch consists of one or more flow tables and a secure channel to an external controller. A flow table consists of a list of flow entries. Each entry has match fields, counters and instructions. Incoming packets are compared with the match fields of each entry and if there is a match, the packet is processed according to the action contained by that entry. When there is no matching flow entry, the packet will be encapsulated and sent to the controller. The controller is a software program which is responsible for manipulating the switch's flow table, using the OpenFlow protocol. It makes a decision on how to handle the packet. It can drop the packet, or it can add a flow entry directing the switch on how to forward similar packets in the future. The OpenFlow protocol deals with defining the format of the messages passed between the control plane and the OpenFlow switch through the secure channel.

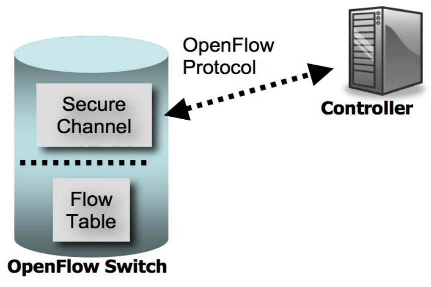

Fig. 1. OpenFlow Components

Since OpenFlow architecture has been designed for LAN, it doesn't include any resiliency mechanism. But fast resiliency is a major requirement in metro and carrier-grade Ethernet network. In this work we have conducted experiments using the link protection scheme that aims to reduce the recovery time during single link network failures. In this scheme the controller calculates the backup path proactively using link protection scheme. The switches are enabled to perform failure recovery operations locally. It avoids controller intervention in the case of failure which not only reduces the time needed for recovery but also the overhead of the controller. As the link is recovered the switches will start using the best path.

II. LITERATURE SURVEY

Many schemes had been proposed in the past for reducing the restoration time in OpenFlow networks. This survey discusses the existing approaches used to reduce the failure recovery time and the limitation of the same. The types of approaches used in these schemes are (1) Restoration approach (2) Data Plane Mechanism (3) Path Protection and (4) Link Protection.

R S. Sharma et al. [2][3] proposed a restoration scheme for OpenFlow carrier grade Ethernet networks. In this scheme, the switch connected to the disrupted link directly notifies the controller about the topology change. Upon notification, the controller identifies the disrupted flows, computes the backup paths, and updates the data plane flow tables considering the failure. Obtained results show recovery times around 200 ms. Here the controller should be a full-state controller which stores the complete path of all the flows established in the network. The authors also recognized that in big networks, these full-state controllers could be overloaded by recovery requests. Y. Yu et al. [7] considered OpenFlow resiliency in IP networks. This scheme is also based on a full-state controller that is notified by the switch upon link failure occurrence. In this the emulation results provide a failure recovery time in 200-300ms range.

M. Desai and T. Nandagopal [4] proposed a scheme based on data plane mechanism in OpenFlow networks. In this approach, when a link fails the switch attached to that link sends notification to all other switches that send traffic through this failed link. Hence those switches will drop the packets instead of forwarding the packets. Though it avoids unnecessary traffic, it relies on controller for actual failure recovery. J. Kempf et al. [8] proposed a similar kind of approach in transport networks based on OpenFlow. In this, each established flow is monitored by sending frequent probe messages. P. Fonseca et al. [9] proposed a mechanism that utilizes a backup controller.

R S. Sharma et al. [1] proposed path protection approach using the fast-failover groups functionality of OpenFlow specification 1.1. In path protection approach, backup paths are pre computed along with the working paths. Both are installed in the switches with different priorities. Hence when a link fails, the switch can use the backup path without the intervention of the controller. Here for every working path a single backup path is computed. In their proposal working path aliveness is monitored by the ingress switch using a similar mechanism as the one proposed in [8]. Here the controller intervention is totally avoided. In the case of failure, this achieves recovery in around 50 ms in emulation environment.

Andrea Sgambelluri et al. [6] proposed a scheme based on link protection approach. This is similar to path protection. But instead of computing a single backup path between the source and destination, this computes backup paths from every node in the working path to the destination. This achieves lesser recovery time than path protection scheme, as here every switch in the working path can directly divert the traffic through the back up path. But this increases the number of flow entries in the flow table. Radwan S. Abujassar et al.[5] proposed two algorithms for rerouting the traffic through pre calculated alternate paths during link failure.

Based on these observations, we experiment a link protection mechanism that aims to enhance the OpenFlow architecture by providing fast recovery mechanism in the switch and the controller. We enable the controller to add backup paths proactively along with the working paths. Moreover the switches are enabled to perform recovery locally.

III. SYSTEM ARCHITECTURE FOR OPENFLOW BASED LINK PROTECTION

The system architecture is represented in Fig 2. The system consists of four major modules: recovery module, link failure detection and notification, renewal packet generation and working path restoration. In the architecture, the first module is related to the controller and other modules are related to the switch. The system works as follows. (1) The controller calculates the backup path proactively and installs it in the switch. (2) The switch generates renewal packets to keep the existing backup paths alive irrespective of the idle time of flow entries (3) When a link fails, the switch identifies the failed link and deletes its flow entries which follow the failed link (4) Once the failed link is up, switch notifies the controller to recalculate the best path by sending recovery packets.

A. Recovery module of the Controller

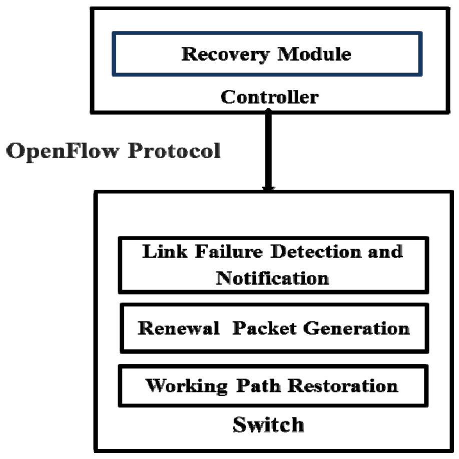

Fig 2. System Architecture

This module is incorporated at the OpenFlow Controller. When a flow request comes, the controller computes the backup path along with the working path proactively. It uses link protection scheme to compute the backup path. In link protection, the backup path is calculated from every node in the working path (towards the destination). In this way, when a failure occurs at any of the link, the corresponding switch can redirect the traffic through its back up path since the backup path from that node is already installed. The flow chart for link protection scheme which is used to calculate the backup path is shown in Fig 3. This algorithm tries to find the backup path for every switch in the working path.

B. *Renewal Packet Generation*

As per OpenFlow standards, each flow entry has idle time out and hard time out properties. When the idle time out is set to some value n, and if for n milliseconds no traffic uses that flow entry, then that flow entry is deleted from the flow table. Hence when there is no failure, the backup path won't be used by any traffic and hence these entries will be deleted after idle timeout period. To avoid this, renew packets with specific format is built and periodically sent by each switch that diverts the backup path from working path. A specific field is included in renew packet so that egress switch can identify these uniquely and won't forward it to the destination.

C. *Link Failure Detection*

Even though we install backup flow entries in the switch and the switch is aware of the link failure, it will keep on using the failed working path to route the traffic as this entry is present in the flow table. Hence there should be a mechanism to delete the flow entries that use the failed link. When one of the links connected to a switch fails, the switch identifies the port that is connected to that link. Then it deletes the entries which have that port as input or output port in the flow table. As the working path entries are deleted, now the switch starts using the backup path entries without the intervention of controller.

D. *Working Path Restoration*

When a failed link is repaired, the shortest path using that link has to be used to send the traffic. But as the switches have

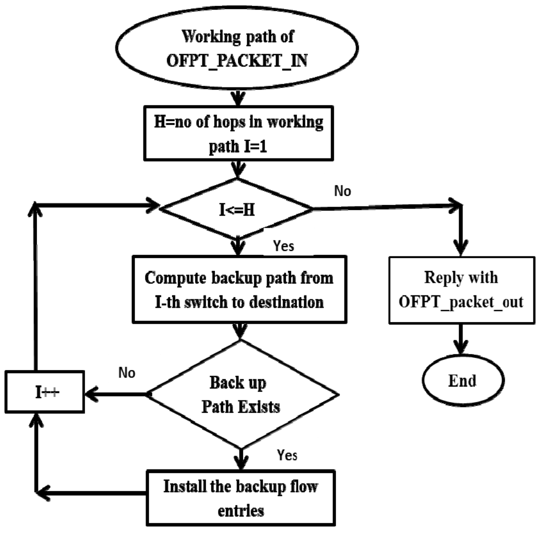
Working path of OFPT_PACKET_IN
H=no of hops in working path I=1
I<=H
No
Yes
Compute backup path from I-th switch to destination
Reply with OFPT_packet_out
End
No
I++
Back up Path Exists
Yes
Install the backup flow entries

Fig. 3. Flow chart for Link Protection

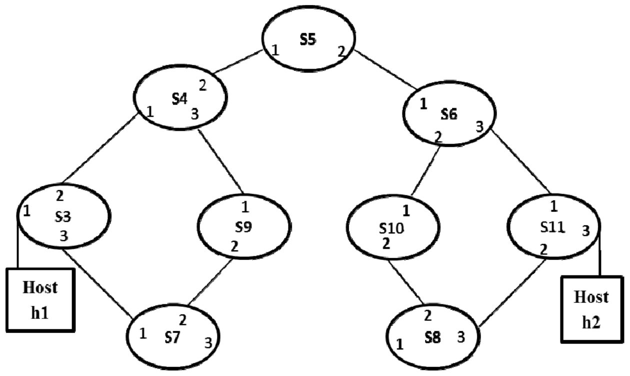

Fig. 4. Sample Network Topology

backup entry, they won't send the packets to the controller to re-compute the working path. To overcome this, when a link is up, the switches connected via that link send restoration packets for each installed flow match. When the controller receives these packets it computes that working path (primary path) and the backup path and installs them in the flow tables. For this a new type of packet is added in the OpenFlow protocol specification. The controller is enabled with a mechanism to process these packets.

IV. IMPLEMENTATION

This link protection scheme is implemented by extending OpenFlow 1.0 specification. OpenFlow controller is extended to include backup path computation. OpenFlow protocol is extended to handle restoration packets. OpenFlow Switch is extended with auto reject mechanism and renewal packet generation modules.

A. *Network Emulation*

In Mininet, a sample network is created with 9 switches and 2 hosts. The sample network is designed in such a way that every switch in the working path has a backup path to the destination. The sample network is shown in Fig 4.

B. *OpenFlow Controller*

The OpenFlow controller is modified to compute backup path along with the working path. Dijkstra's algorithm is used to compute the backup path. Backup entries are installed with different priorities. Three different priorities (IBACKUP, TBACKUP, EBACKUP) are used in the backup path flow entries. IBACKUP (L_i) is used in ingress and the switches where the backup path diverges from the working path. TBACKUP (L_t) and EBACKUP (L_e) are used in transit switches and egress switches respectively. Fig 5 shows the flow table entries for the switches in the emulated network. Algorithm 1 explains the link protection method used in finding the backup path.

C. *OpenFlow Switch*

OpenvSwitch 2.1.0 is modified to generate renewal packets to keep the backup paths alive. A timer mechanism is added with time duration of 20s. This value is set according the idle time out value of the backup flow entries. For every 20s, in every switch, the flow table is examined to find the rules which have

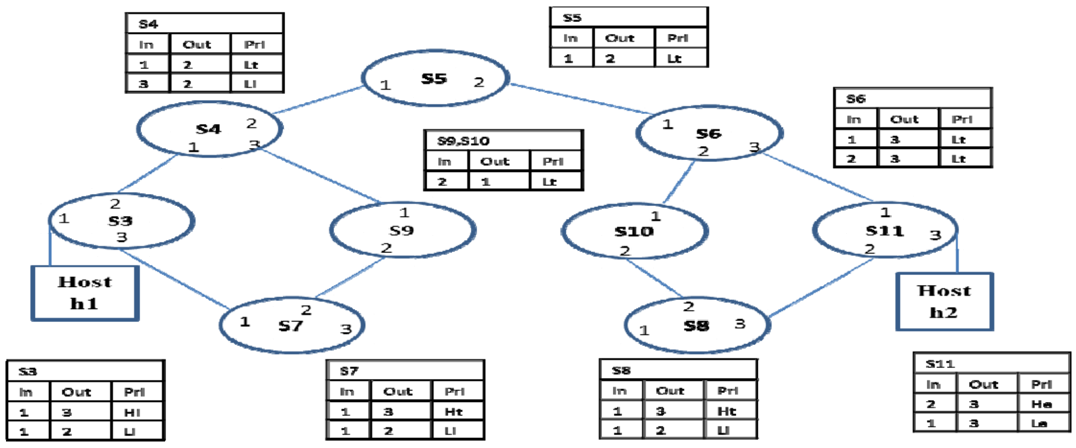

Fig. 5. Protected Traffic Flow with Link Protection

Algorithm 1: Link Protection Method

```
Input:      Tuple <distance, path>  pathmap
            Array[][] adjmatrix
Output:     List<tuple<integer,list>> backuppath
dest ← last node in pathmap
for each node in path(pathmap) except last
            tempadjmatrix ← remove corresponding entry in adjmatrix
end for
for each node in path(pathmap)
            backuppath ← shortest path b/w node to dest in tempadjmatrix
end for
```

priority as IBACKUP(L_i). These switches generate renewal packet which have their EtherType as 0x88dd. Then they forward these packets though the out port specified in their backup path entries with priority IBACKUP(L_i). Further egress switches drop these packets instead of forwarding it to the host. When a link of a backup path fails, these renewal packets are sent to the controller as PACKET_IN messages. The controller is modified to install a null entry in these switches for these messages. The format of the PACKET_IN messages is available in [10].

When a link fails, the two switches (along with the port number) connected by that link identify the link failure. These switches identify the flow entries that have this port as in/out port. Then these entries are deleted by the switches. This removes the working path which follows the failed link. Since the backup paths are already available in the switch's flow table, now the switch redirects the traffic through the backup path. An example scenario is depicted in Fig 6.

D. OpenFlow Protocol

Once a link is recovered, the switches and their port associated with that link are identified. These switches send OFPT_FLOW_RESTORE packets for each of the installed flow, to the controller. The format of this packet is given in Fig 7. Then the controller checks whether there is a better path through the recovered link and installs it if any.

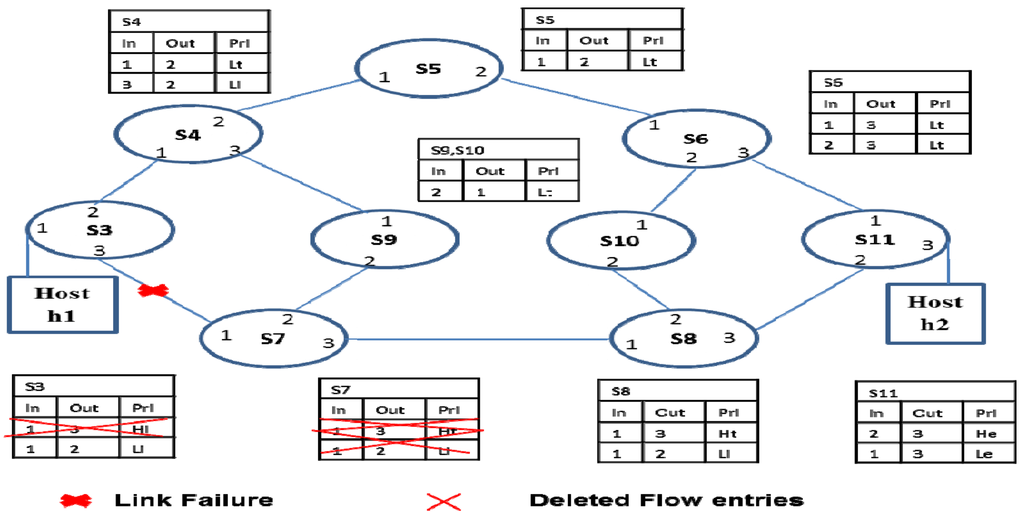

Fig. 6. Failure of Link A-B and the Effect in the Flow Tables

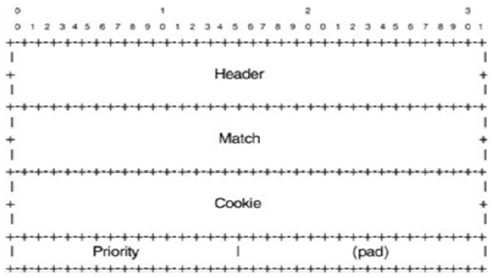

Fig. 7. OFPT_FLOW_RESTORE packet format

V. PERFORMANCE EVALUATION

The link protection scheme is tested in the emulated environment. For this two servers are used. (1) In a system, POX OpenFlow controller is installed on Ubuntu 12. (2) In that system Virtual Box is installed to create a virtual machine with Ubuntu 12. In this virtual machine Mininet 2.1.0 version is installed. The default OpenvSwitch is removed and OpenvSwitch 2.1.0 is installed. The OpenFlow controller, switch and protocol are extended to incorporate the required changes.

The performance of the system is evaluated using the number of packets lost during a continuous flow. To find this value, continuous flow of ping requests are given between the two hosts of the emulated network. Then link failure is simulated by giving a link down command in the Mininet. Now the flow of the ping request is observed using Wireshark and Xterm window. We find the number of packets lost during the flow by explicitly terminating the ping request. Now this shows the statistics of the flow in which the number of packets transmitted and the number of packets received are specified. Using this we find the number of packets lost during the transmission. This ping test is performed several times by specifying different values for ping intervals and the observations are recorded. The results are plotted in a graph as shown in Fig 8.

It could be observed from the graph that the packet loss is zero when the ping interval is more than 50 milliseconds. Hence, the system is able to identify the link failure and re-route the traffic within 50 milliseconds. This is the time taken by the switch to identify any local link failure and delete the corresponding flow entries.

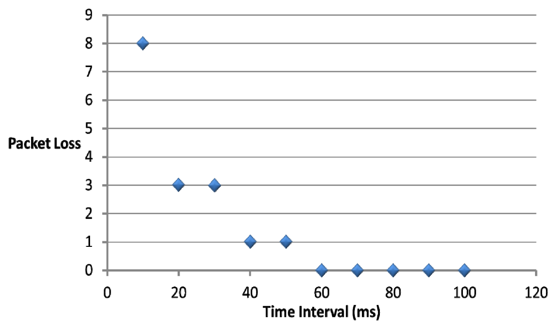

Fig. 8. Ping interval (milliseconds) vs Packet loss

The performance of the system is compared with the path protection scheme by recording the number of packets lost during ping requests with various ping intervals for both schemes. This is shown in Fig 9. The graph clearly depicts that the number of packets lost in path protection scheme is more than that of link protection scheme for all ping intervals. Hence link protection provides better result than path protection scheme. This is due to the avoidance of extra overhead of intimating the source switch about the local link failure.

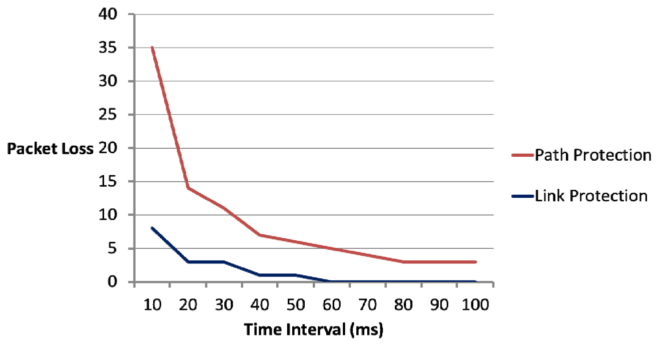

Fig. 9. Path Protection Vs Link Protection

VI. CONCLUSION AND FUTURE WORK

In this work, link protection scheme is implemented and tested to enable fast recovery in OpenFlow networks. This approach does not require a full-state controller. Further the controller is not involved during failure recovery. This solely relies on working path entries and backup path entries that are configured in the network switches with different priorities and the auto reject mechanism that deletes the entries involved in the failure. This also enables the switches to revert back to the working path once the link failure is recovered. Here the achieved recovery time is determined only by the failure detection time. This is possible since our scheme avoids the unnecessary intervention of the controller in the process of deleting defunct entries. In emulated environment most of the time this gives zero packet loss. However, this increases the number of flow entries in the switch.

Though the link protection reduces the recovery time, it increases the number of backup path entries. This results in low bandwidth efficiency. Hence as a future work, we can use the backup paths to distribute the traffic when the link load of the traffic is more than some threshold. Further instead of link protection, segment protection scheme can be applied to achieve moderate bandwidth efficiency and moderate recovery time. This work has been implemented with OpenFlow 1.0 as there are no standardized open source controller versions that support higher OpenFlow specifications. This failure recovery can be improved by the fast fail-over group type of higher versions of OpenFlow.

REFERENCES

[1] Sachin Sharma, Dimitri Staessens, Didier Colle, Mario Pickavet and Piet Demeester, "Enabling Fast Failure Recovery in OpenFlow Networks", Proceedings of 8th International Workshop on the Design of Reliable Communication Networks (DRCN), pp. 164–171, October 2011.

[2] Sachin Sharma, Dimitri Staessens, Didier Colle, Mario Pickavet and Piet Demeester, "Software Defined Networking: Meeting Carrier Grade

Requirements", Proceedings of 18th IEEE Workshop on Local Metropolitan Area Networks (LANMAN), pp. 1–6, October 2011.

[3] Sachin Sharma, Dimitri Staessens, Didier Colle, Mario Pickavet and Piet Demeester, "OpenFlow: Meeting Carrier-Grade Recovery Requirements", Journal of Computer Communications, Vol. 36, No. 6, pp. 656–665, March 2013.

[4] M. Desai and T. Nandagopal, "Coping with Link Failures in Centralized Control Plane Architectures", Proceedings of 2nd International Conference on Communication Systems and Networks (COMSNETS), pp. 1–10, January 2010.

[5] Radwan S. Abujassar and Mohammed Ghanbari, "Efficient Algorithms to Enhance Recovery Schema in Link State Protocols", International Journal of UbiComp (IJU), Vol.2, No.3, pp. 53-58, July 2011.

[6] Andrea Sgambelluri, Alessio Giorgetti, Filippo Cugini, Francesco Paolucci and Piero Castoldi, "OpenFlow-Based Segment Protection in Ethernet Networks", Journal of Optical Communications and Networking, Vol. 5, No. 9, pp. 1066 -1075, September 2013.

[7] Y. Yu, L. Xin, C. Shanzhi, and W. Yan, "A framework of usingOpenFlow to handle transient link failure," in Int. Conf. on Transportation, Mechanical, and Electrical Engineering(TMEE), Dec. 2011, pp. 2050–2053.

[8] J. Kempf, E. Bellagamba, A. Kern, D. Jocha, A. Takacs, and P.Skoldstrom, "Scalable fault management for OpenFlow," in IEEE Int. Conf. on Communications (ICC), pp. 6606–6610, June 2011.

[9] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "A replication component for resilient OpenFlow-based networking," in IEEE Network Operations and Management Symp. (NOMS), pp. 933–939, Apr. 2010.

[10] Open vSwitch documentation [Online]. Available: http://openvswitch.org/support/.