2015 The 5th International Workshop on Computer Science and Engineering
doi: 10.18178/wcse.2015.04.111

NCFF: A Node Classification Fast Failover Scheme for OpenFlow Networks

Anjian Ai*, Hua Qin, Yingxu Lai

Department of Computer, Beijing University of Technology, Beijing,100124, China

*aaj90816@sina.com

Keywords: OpenFlow, failover, classification

Abstract. The ability to fast failover from failures has always been a crucial goal of network reliability in OpenFlow architecture. In this paper, we proposed a Local Reverse Back algorithm and a NCFF fast failover scheme to ensure fast failover from network link failures. Local Reverse Back algorithm selectively reverse traffic back to upstream node to make a comprehensive failover decision and it implements efficient link failure notification without extra detection mechanism. Based on categorizing switches of data plane into a set of types, NCFF establishes an efficient autonomic coordination mechanism between restoration and protection, which achieve a maximal utilization of pre-planned protection paths. To demonstrate the practical performance of the proposed scheme, we have implemented NCFF in our OpenFlow testbed. The numerical results verify that the NCFF enables faster reaction time and high survivability in various topologies.

Introduction

Software-defined networking has been proposed as an innovative split architecture that decouples control plane from the forwarding data plane. By exploiting applications appropriately, the centralized controller promises better connectivity, resource utilization, optimization opportunities and flexibility network management from a global perspective. The OpenFlow[1] is a predominant deployed protocol for SDN.

Network reliability always the fundamental issue to ensure quality of service. Therefore, many solutions to mitigate the impact of failures have been imposed in the literatures[2,3,4,5,6].When a link failure occurs, communication data plane must direct affected traffic efficiently to alternative path or identify the physical connectivity disruptions quickly. The main challenges on designing a failover mechanism base on OpenFlow networks are:

(1) The lack of efficient link-level failure location and connectivity monitoring scheme is a quite challenging task for notifying source end of switch and controller to perform failover action immediately. The Link Layer Discovery Protocol originated from OpenFlow identify failures in the range of sub-seconds[7].In the scheme such as OAM functions[8]and Bidirectional Forwarding Detection (BFD)[9,10] that the full-state controller is in danger of being overloaded with generating high -frequency probe packets.

(2) Each OpenFlow switch delivers packets to a specific output port according to specific flow entry composed of complicated match fields. Therefore, interaction of the switch with remote controller for re-computing enormous new paths will lead to unacceptable higher latencies.

In this paper, we proposed Local Reverse Back algorithm and a node classification fast failover scheme (NCFF) for OpenFlow networks to deal with random single link or interface failures. The NCFF relies on categorizing switches of data plane into a set of types and a novel reverse back mechanism based on group table feature. Upon failure occurrence, recovery can combine

restoration and protection scheme[11,12] effectively. Meanwhile, our algorithms address the two design challenges mentioned above. In this work, we devise a way to fill the gap in coordinating two well-known strategies.

The remainder of the paper is organized as follows. In Section 2, we present a novel fall back algorithm as foundation of NCFF. In Section 3, we define the node category, and elaborate the relation of failover schemes and three types of switch node. In Section 4, we evaluate the performance of our scheme and compare with other well-known solutions. Finally, conclusion are made in Section 5.

Local Reverse Back algorithm

Typical resilience mechanisms for the OpenFlow network element and link failures are proactive protection and reactive restoration. Compared with restoration solution that the alternative paths are not established until a failure occurs, the protection scheme configure pre-planned backup flow-entries with working flow-entries before the failure occurs. In particular, the numerous number of preconfigured backup entries propose great challenges to storage capacity of flow tables in protection.

Considering the implementation of OpenFlow, we come up with a Local Reverse Back algorithm that would direct affected traffic back to upstream nodes for seeking better recovery approach in the presence of failure. In design, our primary focus is on how to increase utilization of the backup flow-entries in different failure scenarios so as to eliminate unnecessary recalculation of alternative paths.

Overview. Local Reverse back Algorithm selectively transfer flows back to upstream node which to make a decision of failover as depicted in Figure 1(a) and Figure 1(b).

The advantage of Local Reverse Back algorithm reflects in two aspects: (1) the upstream switches can be timely notified of downstream link failure through reversed packets without involving controller. (2)With a broader perspective, the upstream nodes can adopt the more appropriate methods to recover, for example, converting to backup flow entry.

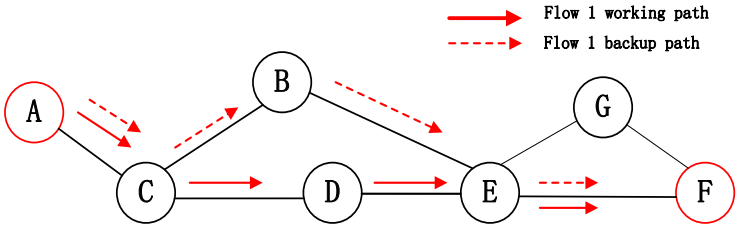

Fig.1 (a): Normal Forwarding

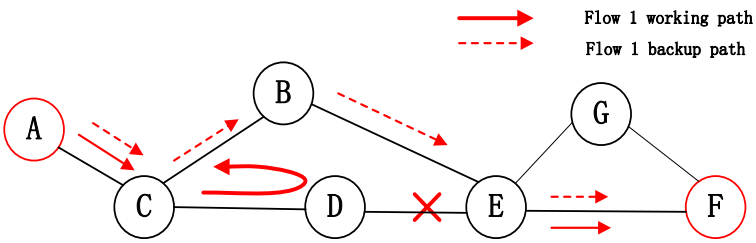

Fig.1 (b): Local Reverse Back When Failure Happens

In addition, the implementation of our proposal is supported by group Table and tagging technique. Beginning with version 1.1, OpenFlow protocol introduced the concept of “group table”. Each group entry contains a set of action buckets and the group will select one bucket for each packet. Furthermore, local fast failover mechanism has been supported by fast failover type of

group entry. Upon the port/link failure is identified, the alternative path as next active bucket are performed immediately, since the fast failover type tables have been monitoring the status of the output ports. Tagging technique is used to set up a flag in packet header that will be sent back from the interface which they were received.

Algorithmic Approach. In the case of link failures, part of affected flow convert into alternative paths if the action field of matched flow entry is directed to a fast failover type group entry. But for most of affected flows, packets will be forwarded in the opposite direction and a flag will be set in packet header. Once the marked packet received, the upstream switch promptly delete or inactivated the relevant entry that forward flow along the direction of fault, and the following packets will not be delivered along the original direction. Note that the packets reversed back without being marked should not be forwarded. The Local Reverse Back algorithm is summarized in Algorithm 1.

The used notations in the algorithm are shown as follows,

Given input constant values:

$k^n : k-th$ interface of switch n occurs interruption

F^n: set of all flow entries in switch n, $f_k^n \in F^n$

T^n: flow table of switch n

G: the action of flow directs to failover type group table

Decision variables:

P_k^n: set of paths affected by local failure k^n, $f_k^n \in P_k^n$

$p^{in}(f^n)$: the in_port field of flow f in switch n

$p^{out}(f^n)$: the original out_port field of flow f in switch n in normal state

$a(f^n)$: the action of flow f in switch n

$pkt(f^n)$: the packets of flow f in switch n

$g(f_k^n)$: the failover out_port in the instructions of group entry for flow f_k^n

$flag(pkt(f^n))$: = 1, if the packet of flow f^n is set to reverse back; = 0 normal forwarding.

$r^{in}(pkt(f^n))$: the actual in_port of flow f in switch n

$\widetilde{p^{out}}(f_k^n)$: the failover out_port after the k^n occurrence, = 0 indicate the packet need to be dropped

Algorithm : Local Reverse Back

```
1  input: pkt(F^n), k^n
2  Output: p̃^out P_k^n
3  set P_k^n := ∅
4  for f^n ∈ F^n
5     if p̃^out(f_k^n) = k^n, then
6           P_k^n := P_k^n ∪ f^n
7  for pkt(f_k^n) ∈ pkt(P_k^n)                    //for received packets
8     if r^in(pkt(f^n)) = p^out(f_k^n)  &&  flag(pkt(f^n)) = 1
9        if a(f^n) = G
10          p̃^out(f_k^n) := g(f_k^n)
11       else
```

12 $\widetilde{p^{out}}(f_k^n) := p^{in}(f_k^n)$

13 delete f^n

14 else if $r^{in}(pkt(f^n)) = p^{out}(f_k^n)$ // $flag(pkt(f^n)) = 0$

15 drop $pkt(f^n)$

16 else // $r^{in}(pkt(f^n)) \neq p^{out}(f_k^n)$

17 if $a(f_k^n) = G$

18 $\widetilde{p^{out}}(f_k^n) := g(f_k^n)$

19 else

20 $\widetilde{p^{out}}(f_k^n) := p^{in}(f_k^n)$

21 $flag(pkt(f^n)) := 1$

Processing with Local Reverse back algorithm, each influenced flow can be forwarded according the output $\widetilde{p^{out}}(f_k^n)$, and we present in detail the recovery scheme in Section III

Node Classification Recovery

We creatively defined 3 types of switch nodes through which the local switch can unilaterally decide to adopt a reactive restoration strategy, a proactive protect strategy or abandon recalculation under random link failures. We will assume that for each flow a reasonable number of pre-compute backup path entries (usually only one) have been configured in advance. Indeed, transit switch nodes of an OpenFlow networks could be classified as follows:

- **Relay Node (RN, type 1):**Relay Node serves a one-to-one flow transfer pattern which only have one original incoming port and one outgoing port for a flow considering a working path and all backup paths (e.g., switch D in Fig 1). For this flow, the node can't forward packets through other prepared outgoing port.
- **Switchover Node (SN, type 2):** Switchover Node serves a one-to-N (N>1) flow transfer pattern which have one original incoming port but more than one potential outgoing port considering a working path and all backup paths (e.g., switch node C in Fig 1).This kind of node is configured with one of more backup entries, and can enable the transmission of packets from other non-primary outgoing port, which indicates that it's a switchover node. With one exception, the N-to-N (N >1) nodes can also be classified in this category.
- **Aggregation Node (AN, type 3):** Aggregation Node serves a N-to-one (N>1) flow transfer pattern which have multiple potential incoming port but only one outgoing port considering a working path and all backup paths (e.g., switch node E in Fig 1). The implication of the node is that at least two of working path and pre-compute backup path converge on a node.

In NCFF, for each admitted flow, the controller configures working entries at relevant switches and one backup path, if it exist, is configured for the case of link failure simultaneously. But as we know, one pre-planned backup path can't cover all failure situations, so different failure scenarios need different failover mechanism in OpenFlow networks.

In situations where arbitrary single failures occur in an OpenFlow networks, the NCFF scheme first analyses the local transit switch type to decide on an appropriate action for the failure. Moreover, there are four potential scenarios that have been considered carefully by NCFF.

(1) If the switch attached to the failed link is a relay node (type 1), the corresponding flows should be reversed according to Local Reverse Back algorithm. Transit switch SW D in figure 2(a) is a RN node for the flow 1, which forwards packets through next-hop SW E. Once the outgoing

link (D, E) fails, SW D reverses the packets back to the sending neighbour (SW C) and transmits along upstream in sequence until the switch type is no longer a RN type.

(2) If the recursive returning of flow meet a SN (type 2) node or the switch adjacent to failure link is the SN node, proactive protection recovery using fast failover group table is performed by the SN switch immediately in NCFF. The SN type of switch imply that pre-planned backup up paths have been installed. In figure 2(a), when the adjacent link(D,E) is unavailable, packets are delivered from SW D back to SW C. SW C is a SN node, which have installed backup path to handle failure locally without remote controller. At SW C, the flow is forwarded through backup outgoing link (SW C, SW B) decisively.

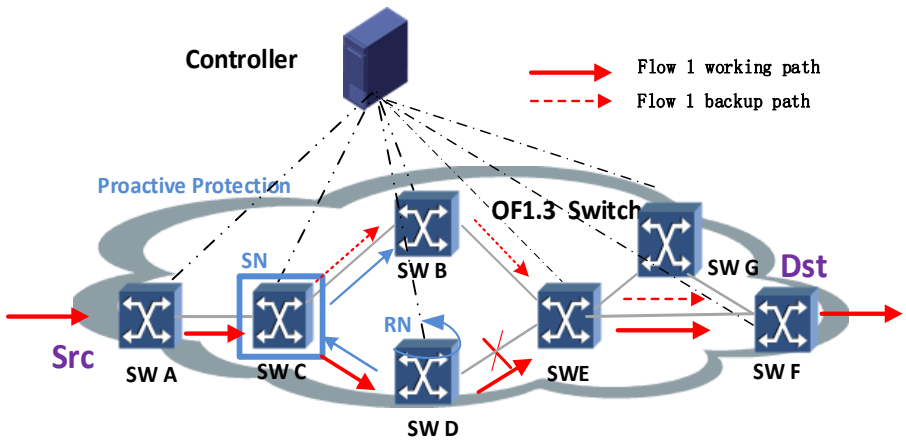

Fig.2(a): Reverse Back and Proactive Protection

(3) If the recursive returning of flow meet an AN (type 3) node or the switch adjacent to failure link is the AN node. The AN type of switch illustrates there are more than one feasible incoming port and only one outgoing port for the flow, but the previous backup paths become invalid for current switch. Requesting alternative path to controller is only possible approach to use in this scenario. Referring to figure 2(b), when SW E, which is an AN node, loses link (E, F) connectivity, predefine failover path (A-C-B-E-F) becomes unavailable, then SW E will inform the controller topological state changes to recalculate an new path rapidly.

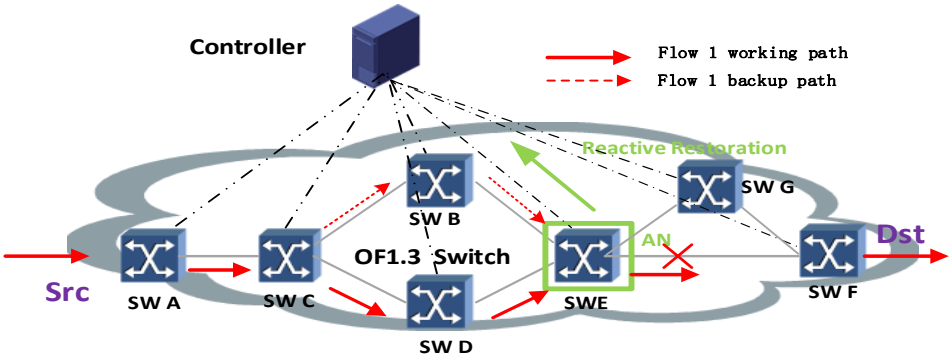

Fig.2(b): Reactive Restoration

(4) If the recursive returning of flow meet an ingress switch src or the switch adjacent to failure link is the ingress switch, whether there is a pre-computed backup entry is the critical factor in determining the approach adoption. (a) If there is an existing backup entries installed on related switch, path re-computation will be requested by src switch (RN type). (b)If not, since there is no backup path in the first place, the date traffic should be dropped immediately.As shown in Figure

2(c), SW A is an RN and ingress switch for flow 1. The working entry is configured with no backup entry as no alternative path is available. The packets are dropped directly upon failure of link (A, B) occurs.

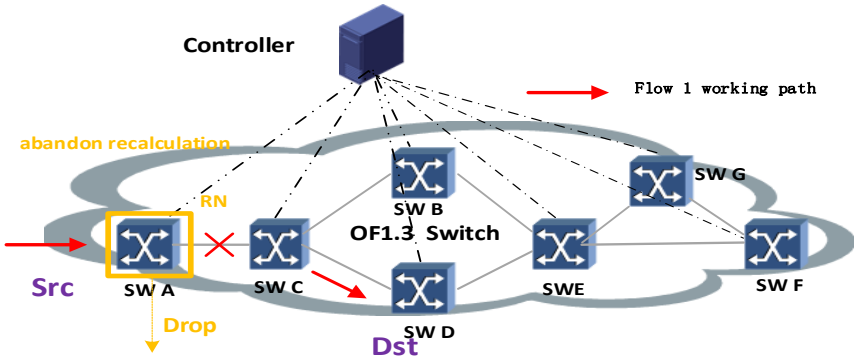

Fig.2(c): Abandon Recalculation

Initially, when each first packet of incoming flows consult the controller for forwarding behaviors, the controller compute a robust failover path, if it exist, as well as primary working flow-entries in NCFF. The computation process provide ability to equip related switch's flow-entries with category detail information. In fact, it is exactly based on this tag information used to identify the switch type for each flow that NCFF scheme can be implemented. As shown in Figure 3, an OpenFlow 1.3 flow table entry contains: match fields, priority, counters, instructions, timeouts and cookie[13]. In instructions field, action header is common to all actions, and we use the unused pad field of action header to describe the switches' category for each flow. Category information is encoded in pad[0]-pad[1]. A value of '00' indicates that there is no available backup path for the flow. '01', '10' and '11' represent the switch is a RN, a SN and an AN node, respectively. Note that the tag information need to be configured along with flow-entries.

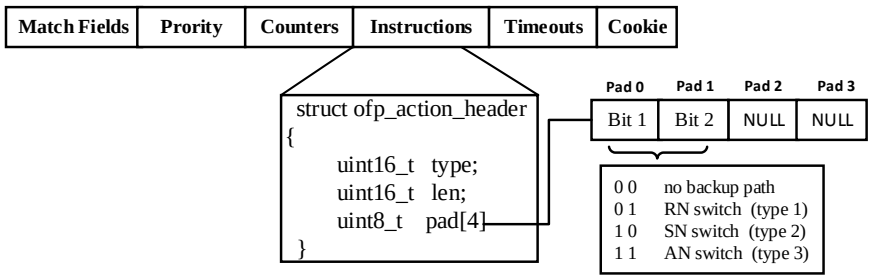

Fig.3: Fields Used for Category Information

Experimental Evaluation

We have evaluated the performance of the proposed NCFF scheme through emulation. Three sets of experiments have been conducted using our physical testbed, which consisted of a remote controller, seven software OpenFlow switches, a Spirent TestCenter and several hosts.

Test Environment. As shown in Figure 2(a), the topology is the most common in OpenFlow Networks, and it can easily simulate four different failure scenarios mentioned above in section 3 to validate the efficiency of our proposed method. Each switch is implemented on a PC (i.e., Intel Core 2 CPU 2.33 GHz, 2GB RAM, Ubuntu 14.04) performing OpenVswitch 2.3.0 [14] supporting OpenFlow 1.3 Failover Group Table. The centralized controller capability is realized using the floodlight implementation[15], installed on a server running Ubuntu 12.04 with an Intel Xeon quad

core 3.00GHz CPU and 16GB RAM. We implemented our NCFF scheme on our testbed to handle failures in OpenFlow Networks.

In our emulation environment, two test end-hosts have been deployed the iperf which can generate packets with a fixed rate at unit time and measure packet loss. Upon a link failure occurs, a break in data transmission and packet loss are detected and recorded by iperf server at Dst host. Therefore, based on packet loss ratio during the outage, the failover time T_f can be computed. In addition, for the purpose of generating a high number of concurrent requests, we use the Spirent TestCenter to generate massive packets with different source and destination IP address, which will cause heavy computing load to controller.

Emulation Scenarios. We have validate the performance of the NCFF under different single link failure situations.

Scenarios 1:With reference to Figure 2(a), before injecting the link failure, the traffic flow from Src host connected with SW A to Dst host connected to SW F has been transmitted along the working path(A-C-D-E-F). Meanwhile, the backup path (A-C-B-E-F) has been computed and installed, and category information defined by our proposal has been set in flow-entries or group entries on related switches. When we deliberately break link (D,E),as shown in Figure 2(a), the data traffic is reversed back at SW D, and transmission path convert to path (A-C-D-B-E-F) since SW D is a RN node and SW C is an SN node. We have repeated this failure on link (D, E) 100 times and obtained c displayed in figure 4(a). Considering that time interval of user packet sent by iperf affects the precision of the measurements, we compared the packet loss under different sending rate in above failure scenario, and obtained Figure 4(b).

Scenarios 2: As shown in Figure 2(b), if the link failure occurs between SW E and SW F, then the pre-planned backup path (A-C-D-E-F)is invalid. SW E will check the tag information in the flow entry to identify the type of itself. As an AN node, SW E immediately interacts with controller for new forwarding decision in updated topology and forwards the traffic along (A-C-D-E-G-F). In this scenario, the recovery time T_f is relatively long, but which is inevitable to reactive restoration.

In order to emulate the real production environments, we use the TestCenter to generate numerous different streams in our testbed. The result is displayed in Figure 4(c).

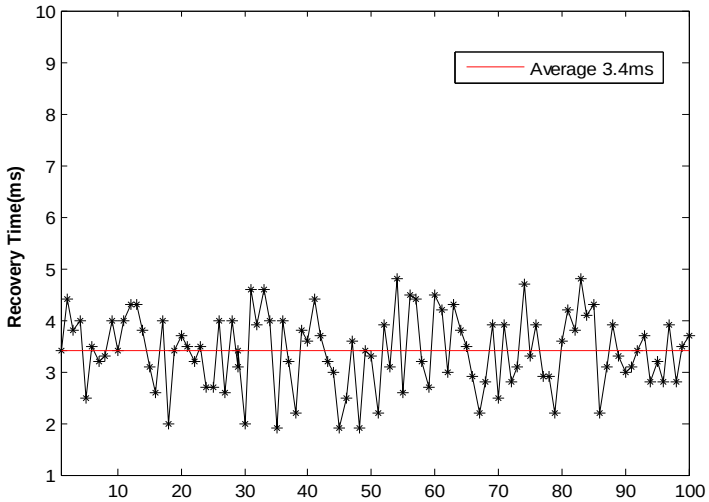

Fig.4 (a): Recovery Times for Scenarios 1

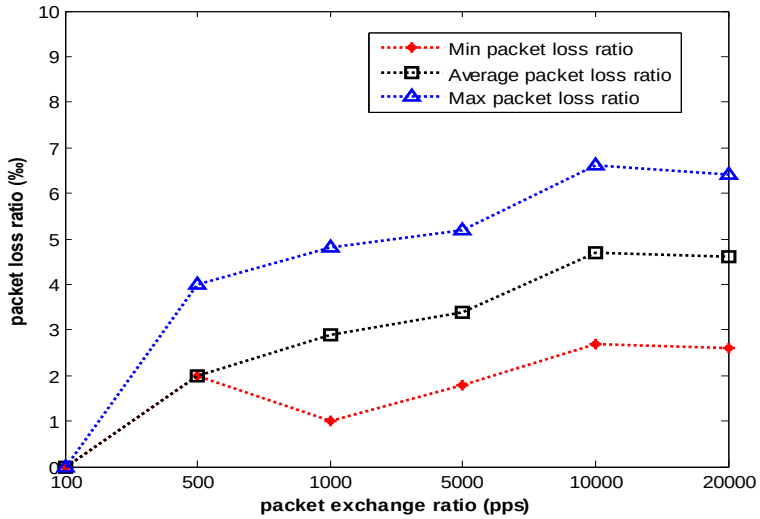

Fig.4 (b): Packet Loss Ratio for Scenarios 1

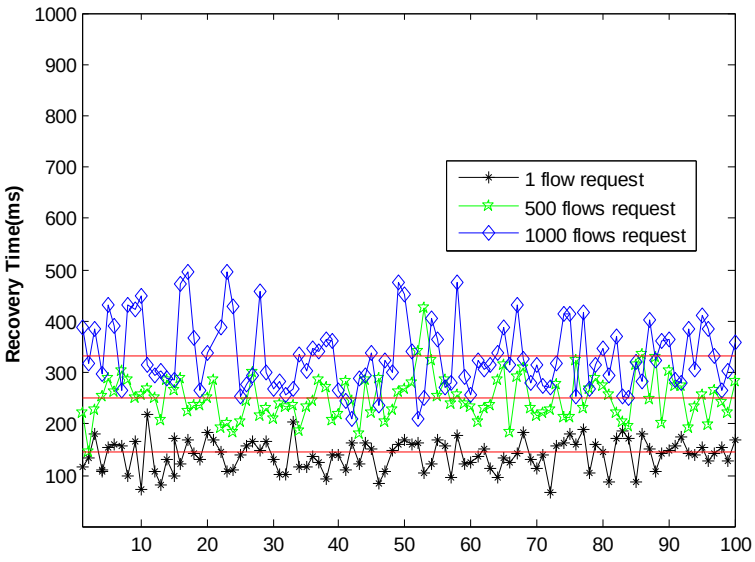

Fig.4(c): Recovery Times for Scenarios 2

Analysis of Results. Figure 4(a) illustrates the recovery time T_f of our scenarios 1 experiments. The rate of data traffic v has been set 10000 pps, in other words, the connectivity was confirmed at an interval of 0.1 ms. Additionally, we used if config command to bring up/down the interface, instead of interrupting physical connections. In this case, we measured the average recovery time T_f of 3.4 ms and T_f was distributed in the range 2.0-4.7 ms. To eliminate the influence of packet exchange ratio, we generated packets at v=100, v=500, v=1000, v=5000, v=10000 and v=20000 pps, respectively. Figure 4(b) shows that packet loss can't be detected at v=100 pps (10ms time interval) and we obtained the packet loss rate of 2.0‰2.9‰,3.4‰,4.3‰,4.4‰ in 1 second of interruption occurred .The result show that packet loss rate generally remained at a level of 4.3‰, which means the alternative path switches fast and steadily under the NCFF protection mechanism.

As shown in Figure 4(c), we measured recovery time T_f of scenarios 2 experiments against increasing number of requesting flow. In the case of requesting flows W=1, the recovery time was distributed in the 80ms-170ms and the average value is 145 ms within 100 trials. Due to interaction

with the controller, the failover time of restoration strategy is much longer than that of protection strategy. Therefore, when the requesting flows W=500 flows/s and W= 1000 flows/s, we obtained averages failover time of 251ms and 334ms, which shows that processing time increases with the number of concurrent requesting number. The large variation range of recovery time indicates the controller's real-time load has a profound effect on failover time under the NCFF reactive restoration mechanism.

From the experiments we conducted above, it is clear that the single link failure should be recovered with protection as much as possible. Indeed, the NCFF make it easy to achieve this through transferring failover point to an upstream node by Local Reverse Back algorithm.

Conclusions

In this paper, we have proposed a node classification fast failover scheme for survivable OpenFlow networks. The NCFF scheme combine restoration and protection scheme effectively based on Local Reverse Back algorithm and analysis of relevant switch's type. In the presence of failure, local failover decision can avoid imposing additional communication overhead on controller. Moreover, flexible failover point is allowed so as to achieve the maximum utilization of pre-installed protection entries, which can reduce path re-computation requests by 50% with no extra backup paths. Our experimental results demonstrate that the NCFF scheme guarantee the seamless failover within 4ms and low packet loss ratioduring the period of recovery in protection condition. Moreover, compared with the existing methods, the NCFF has higher survivability than OSP[15] and the scheme based on BFD[16] for various topologies and more efficient decision-making process than CogMan[12].

References

[1] N. McKeown, T. Anderson,et al., "OpenFlow:Enabling innovation in campus networks," Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, Aug. 2008.

[2] J. Kempf, E. Bellagamba, et al., "Scalable fault management for openflow," in Communications(ICC), 2012 IEEE International Conference on, 2012.

[3] D. Staessens, et al., "Software Defined Networking: Meeting Carrier Grade Requirement," in Proc. IEEE LANMAN, 2011.

[4] E. Bellagamba, J. Kempf, et al., "Link failure detection and traffic redirection in an openflow network," May 19 2011, uS Patent App.13/111,609.

[5] S. Sharma, et al., "OpenFlow: Meeting carrier-grade recovery requirements," Computer Communications, vol. 36, no. 6, pp. 656-665, Mar. 2013.

[6] Junda Liu "Data-Driven Network Connectivity" in Proc.HotNets-X, 14–15,November, 2011

[7] S. Sharma, et al., "Enabling Fast Failure Recovery in OpenFlow Networks," in Proc. DRCN, 2011.

[8] J. Kempf, et al., "Scalable Fault Management for OpenFlow," in Proc. IEEE ICC, 2012.

[9] D. Katz, D. Ward, Bidirectional Forwarding Detection (bfd), RFC5880, Tech. Rep., June 2010

[10] Van Adrichem, N.L.M."Fast Recovery in Software-Defined Networks." in Proc EWSDN, 2014 Third European Workshop, 1-3 sept.2014

[11] S. Sharma, D. Staessens, et al., "Openflow: meeting carrier-grade recovery requirements", Comp. Commun. 36 (6) (2012) 656–665

[12] S.Kim, J.Kang, et al.,"A cognitive model-based approach for autonomic fault management in OpenFlow networks." International Journal of Network Management Int. J. Network Mgmt 2013; 23: 383–401.

[13] OpenFlow Switch Specification 1.3.0, October, 2013 [Online]. Available: http://www.openflow.org/
[14] Open vSwitch documentation [Online]. Available: http://openvswitch.org
[15] Floodlight. August 2013,Available: http://floodlight.openflowhub.org
[16] A. Sgambelluri, et al., "OpenFlow-Based Segment Protection in Ethernet Networks," IEEE/OSA Journal of Optical Communications and Networking, vol. 5, no. 9, pp. 1066-1075, Sept. 2013.
[17] S. Sharma, et al., "Fast failure recovery for in-band OpenFlow netwoks," in Proc. DRCN, 2013