SDN-based Solution of Path Restoration for Smart Grids

Ma Wei Zhe[1], Jiang Yun Dou[1], Dong Hong Yu[2] , Xue Xiao Ming[1] , Lin Zhong Qiu[3] , Zou, Yu[4]

[1]State Grid Benxi Electric Power Supply Company, Benxi 117000, China;

[2]State Grid Fuxin Electric Power Supply Company, Shenyang 123000, China;

[3]State Grid Dandong Electric Power Supply Company, Shenyang 118000, China.

[4]Liaoning Planning and Designing Institute of Post and Telecommunication Company Limited, Shenyang 110011,China

Keywords: Smart grids; Software defined network; Path restoration.

Abstract. Due to the heterogeneity and the nature of distributed control, the existing electric power data communication network based on Synchronous Digital Hierarchy (SDH) has been unable to meet the requirements of dynamic resource allocation and fine-grained traffic monitoring. The OpenFlow-based Software Defined Network (SDN) allows operators dynamically control and monitor the whole network using the software running on the operating system of the centralized controller (e.g., NOX controller). Thus, in this paper, we propose the SDN-based fault-tolerant solution for the electric power data communication network. The feasibility and efficiency of the proposed solution are verified, and the performances of the dynamic end-to-end path restoration, such as the CPU utilization of NOX and end-to-end delay, are quantitatively evaluated in our experimental platform. The experimental results indicate that the dynamic end-to-end path restoration can be achieved within tens of milliseconds by using our SDN-based solution.

1. Introduction

With the rapid development, the electric power data communication network has become the basic bearing platform of constructing strong and smart power grids, and plays an important role in the electrical power distribution, market operation and modern management. However, due to the heterogeneity and the nature of distributed control, the existing electric power data communication network based on Synchronous Digital Hierarchy (SDH) has been unable to meet the requirements of dynamic resource allocation and fine-grained traffic monitoring. The synergistic construction of information and communication will develop towards the Information Communication Technology (ICT), which will make the electric power data communication network achieve excellent systematic efficiency [1].

To achieve the desired efficiency, the Software Defined Network (SDN) is a promising solution [2]. SDN decouples the control function from the data forwarding plane, and realizes the network virtualization, flow control, centralized control, and fast fault diagnosis, etc. As a result, the SDN-based solution can promote the evolution of the electric power data communication network in the ICT direction.

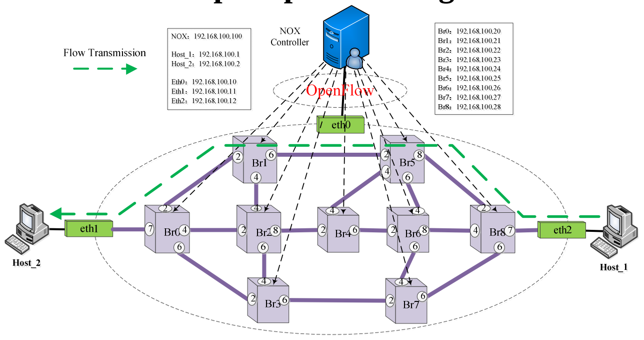
The architecture of the electric power data communication networks makes the global control impossible, which results in the absence of the whole network profile. And it is difficult to real-time monitor the network status for the implementation of the path restoration. Once network fault occurs, if the restore path cannot be found timely, it will cause huge economic losses. Therefore, it is very necessary to design SDN-based fault-tolerant solution in smart grids.

In this paper, the SDN network environment supporting the fast path restoration is built by using NOX controller [3] and OpenvSwitch [4, 5]. OpenvSwitch is used to build a virtual network topology. The NOX controller is used to control all switches in the entire network topology, and achieves traffic routing by virtue of the shortest-path algorithm. Especially for the fault scenario of the electric power data communication network under the SDN environment, such as the failure of the switch in the currently running virtual network, the NOX controller can make the quick fault diagnosis, and calculate the shortest restore path.

The rest of this paper is organized as follows. Section 2 proposes the SDN-based fault-tolerant solution of the electric power data communication network. The experimental results are shown in section 3. Finally, we conclude this paper in section 4.

2. SDN-BASED FAULT-TOLERANT SOLUTION

2.1 End-to-end path provisioning



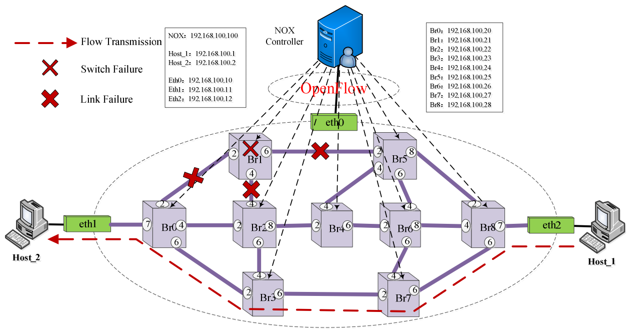


Fig. 1 End-to-end path provisioning

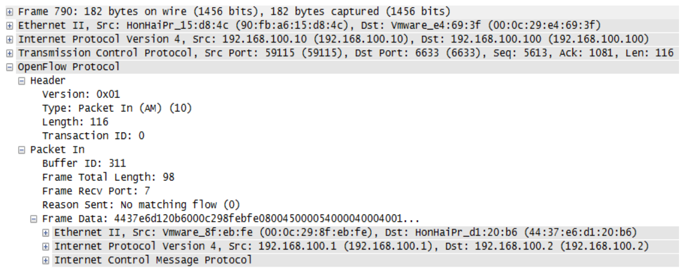
Fig. 2 End–to-end path restoration

We establish the SDN network environment by using the NOX controller and OpenvSwitch as shown in Fig. 1. This network environment consists of a NOX controller, nine OpenFlow-enabled switches and two hosts. The IP addresses of them can be seen in Fig. 1. Host_1 is the source node while Host_2 is the destination node, i.e., Host_1 sends packets to Host_2 by using the Ping command. The NOX controller is responsible for dynamically monitoring the status of the whole network, and determining the approach of handling packets without valid flow entries. And this controller manages the flow tables of the nine switches by adding and removing flow entries according to Flow_Mod message. When a new IP packet arrives at No. 7 ingress port of Br8, no matched flow entry is found, then Packet_In message is sent to the NOX controller along the secure channel. NOX analyzes the state of the whole network topology, and then calculates the route using the dynamic shortest-path algorithm. In Fig. 1, the shortest path from Host_1 to Host_2 is (No. 7 input port) Br8 (No. 2 output port) → (No. 8 input port) Br5 (No. 2 output port) → (No. 6 input port) Br1 (No. 2 output port) → (No. 2 input port) Br0 (No. 7 output port), which is the green line in Fig. 1. Next, the NOX controller inserts the corresponding flow entries into the flow tables of all relative switches (Br8, Br5, Br1, and Br0) along the computed route according to Flow_Mod message. Finally, the packets from Host_1 to Host_2 are transmitted according to these flow tables.

2.2 End-to-end path restoration

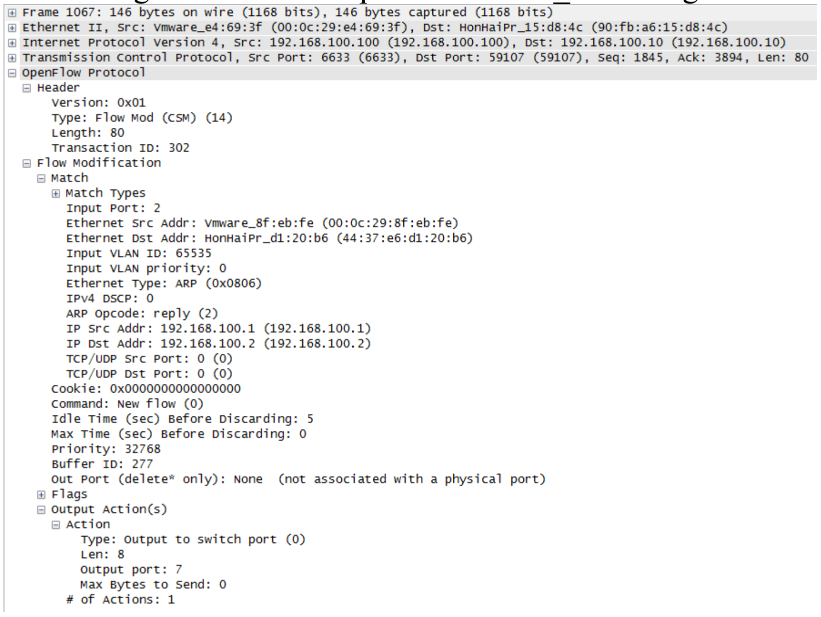
The fault scenario is depicted in Fig. 2. When the switch Br1 breaks down in the currently running virtual network, the links Br1 ↔ Br0, Br1 ↔ Br2, Br1 ↔ Br5 will automatically become invalid. The NOX controller receives the 'datapath-leave' event and link-event. Since the two link failures (Br1 ↔Br0, Br1 ↔ Br5) disconnects the primary path, Host_1 → (No. 7 input port) Br8 (No. 2 output port) → (No. 8 input port) Br5 (No. 2 output port) → (No. 6 input port) Br1 (No. 2 output port) → (No. 2 input port) Br0 (No. 7 output port) → Host_2, the NOX controller dynamically updates the network topology according to these two events, and automatically sends request to the routing application for the computation of the restore path. Routing application calculates the new shortest path, Host_1 → (No. 7 input port) Br8 (No. 6 output port) → (No. 6 input port) Br7 (No. 2 output port) → (No. 6 input port) Br3 (No. 2 output port) → (No. 6 input port) Br0 (No. 7 output port) → Host_2, which is the red line in Fig. 2. And NOX sends flow tables to Br8, Br7, Br3, and Br0, in order to recovery the path transmission and guarantee the high-quality electric power service.

3. EXPERIMENTAL RESULTS AND DISCUSSIONS



```
⊞ Frame 790: 182 bytes on wire (1456 bits), 182 bytes captured (1456 bits)
⊞ Ethernet II, Src: HonHaiPr_15:d8:4c (90:fb:a6:15:d8:4c), Dst: Vmware_e4:69:3f (00:0c:29:e4:69:3f)
⊞ Internet Protocol Version 4, Src: 192.168.100.10 (192.168.100.10), Dst: 192.168.100.100 (192.168.100.100)
⊞ Transmission Control Protocol, Src Port: 59115 (59115), Dst Port: 6633 (6633), Seq: 5613, Ack: 1081, Len: 116
⊟ OpenFlow Protocol
  ⊟ Header
      Version: 0x01
      Type: Packet In (AM) (10)
      Length: 116
      Transaction ID: 0
  ⊟ Packet In
      Buffer ID: 311
      Frame Total Length: 98
      Frame Recv Port: 7
      Reason Sent: No matching flow (0)
    ⊟ Frame Data: 4437e6d120b6000c298febfe08004500005400004000 4001...
      ⊞ Ethernet II, Src: Vmware_8f:eb:fe (00:0c:29:8f:eb:fe), Dst: HonHaiPr_d1:20:b6 (44:37:e6:d1:20:b6)
      ⊞ Internet Protocol Version 4, Src: 192.168.100.1 (192.168.100.1), Dst: 192.168.100.2 (192.168.100.2)
      ⊞ Internet Control Message Protocol
```

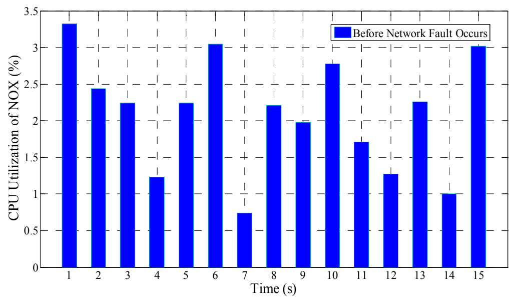
Fig. 3 Wireshark capture of Packet_In message



```
⊞ Frame 1067: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits)
⊞ Ethernet II, Src: Vmware_e4:69:3f (00:0c:29:e4:69:3f), Dst: HonHaiPr_15:d8:4c (90:fb:a6:15:d8:4c)
⊞ Internet Protocol Version 4, Src: 192.168.100.100 (192.168.100.100), Dst: 192.168.100.10 (192.168.100.10)
⊞ Transmission Control Protocol, Src Port: 6633 (6633), Dst Port: 59107 (59107), Seq: 1845, Ack: 3894, Len: 80
⊟ OpenFlow Protocol
  ⊟ Header
      Version: 0x01
      Type: Flow Mod (CSM) (14)
      Length: 80
      Transaction ID: 302
  ⊟ Flow Modification
    ⊟ Match
      ⊞ Match Types
        Input Port: 2
        Ethernet Src Addr: Vmware_8f:eb:fe (00:0c:29:8f:eb:fe)
        Ethernet Dst Addr: HonHaiPr_d1:20:b6 (44:37:e6:d1:20:b6)
        Input VLAN ID: 65535
        Input VLAN priority: 0
        Ethernet Type: ARP (0x0806)
        IPv4 DSCP: 0
        ARP Opcode: reply (2)
        IP Src Addr: 192.168.100.1 (192.168.100.1)
        IP Dst Addr: 192.168.100.2 (192.168.100.2)
        TCP/UDP Src Port: 0 (0)
        TCP/UDP Dst Port: 0 (0)
      Cookie: 0x0000000000000000
      Command: New flow (0)
      Idle Time (sec) Before Discarding: 5
      Max Time (sec) Before Discarding: 0
      Priority: 32768
      Buffer ID: 277
      Out Port (delete* only): None  (not associated with a physical port)
    ⊞ Flags
    ⊟ Output Action(s)
      ⊟ Action
          Type: Output to switch port (0)
          Len: 8
          Output port: 7
          Max Bytes to Send: 0
        # of Actions: 1
```

Fig. 4 Wireshark capture of Flow_Mod message

Figure 3 shows the Wireshark capture of Packet_In message. It can be seen that the source and destination addresses are 192.168.100.1 and 192.168.100.2, respectively. The reason of this message is sent to NOX is that no matched flow entry can be found. Flow_Mod message is shown in Fig. 4. The input port is No. 2, while the output port is No. 7 for forwarding packets.



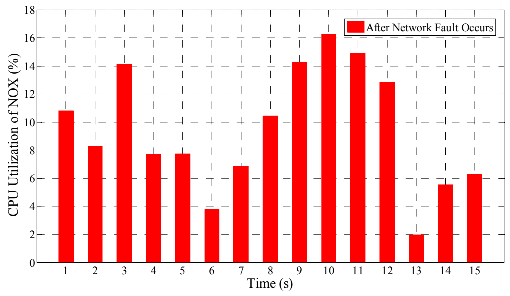
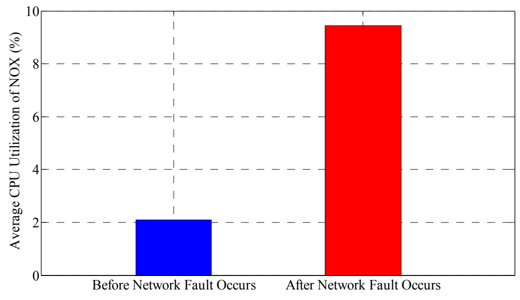


Fig. 5 CPU utilization of NOX before network fault occurs Fig. 6 CPU utilization of NOX after network fault occurs



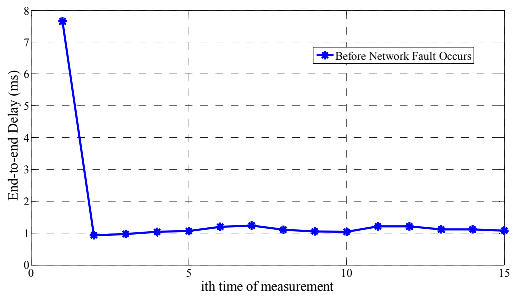


Fig. 7 Average CPU utilization of NOX   Fig. 8 End-to-end delay before network fault occurs

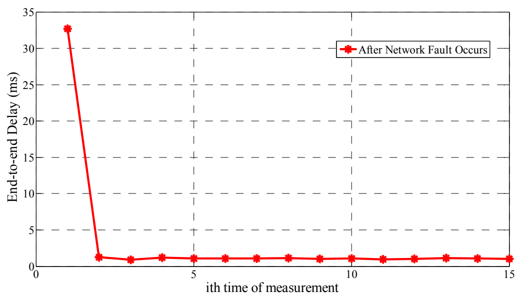


Fig. 9 End-to-end delay after network fault occurs

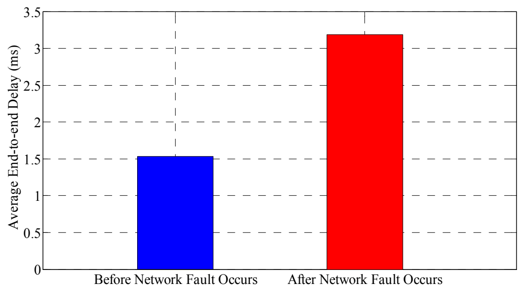


Fig. 10 Average end-to-end delay

The NOX controller accounts for the path computation, and runs on the virtual machine with 2 processors. Figures 5 and 6 demonstrate the CPU utilization of NOX before and after the network fault occurs, respectively. The horizontal axis is the time, i.e., one sampling one second, and the vertical axis records the CPU utilization. Figure 7 compares the average CPU utilization before and after the network fault occurs. The average CPU utilization of NOX increases under the path restoration, but it varies within the acceptable range.

Figures 8 and 9 demonstrate the end-to-end delay before and after the network fault occurs. Among which, the horizontal axis is the packet arrival sequence, while the vertical axis records the end-to-end delay including the propagation time of OpenFlow message and the processing latency of the NOX controller. We find that the dynamic end-to-end path restoration can be achieved within tens of milliseconds by using the SDN-based solution, which is very promising. Figure 10 compares the end-to-end delay before and after the network fault occurs. The average delay increases after the fault happens, because NOX consumes time to dynamically update the network topology and send flow tables to relative switches for the path restoration.

4. CONCLUSION

In this paper, we experimentally demonstrated the feasibility of the dynamic end-to-end path restoration of the electric power data communication network under the real SDN environment by using NOX controller and OpenFlow-enabled switches. The experimental results indicated that the dynamic end-to-end path restoration can be achieved within tens of milliseconds.

References

[1] Jianchao Zhang, Boon-Chong Seet, Tek-Tjing Lie et al. Opportunities for Software-Defined Networking in Smart Grid. 2013 9th International Conference on Information, Communications and Signal Processing (ICICS), Tai Nan, Dec. 2013, pp.1-5.

[2] McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: Enabling Innovation in Campus Networks. SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.

[3] Gude N, Koponen T, Pettit J, et al. Nox: Towards an Operating System for Networks. ACM SIGCOMM Computer Communication Review, 2008, 38(3): 105−110.

[4] OpenvSwitch: http://openvswitch.org/.

[5] OpenFlow:https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf, 2009-12-31.