AmLight's OpenFlow Sniffer dissected: Troubleshooting production networks

Jeronimo Bezerra[1], Humberto Galiza[2], Julio Ibarra[1], Marcos Schwarz[2]

[1]Florida International University (FIU)
Miami – FL – USA

[2]Rede Nacional de Ensino e Pesquisa (RNP)
Campinas – SP – Brazil

{jbezerra,julio}@fiu.edu, {humberto.galiza,marcos.schwarz}@rnp.br

Abstract. *AmLight network was migrated to an OpenFlow approach in mid-2014. Since its migration, AmLight hosted a few network testbeds, where each testbed presented different challenges to the network operation. As OpenFlow agents are new codes on the network devices, some flaws and crashes are expected. However, as AmLight is a production network, outages have to be handled right away. This paper will present AmLight's OpenFlow Sniffer, an open source tool to be used for troubleshooting and learning purposes. Using the OpenFlow Sniffer improved the overall troubleshooting process on AmLight SDN's network.*

1. Introduction

In OpenFlow-based environments, due to the lack of troubleshooting protocols and tools, packet inspection is frequently used. Packet inspection allows the network engineer to understand what types of OpenFlow [ONF 2009] message are being exchanged.

With the OpenFlow deployment at AmLight [Ibarra et al. 2015], the lack of troubleshooting tools became explicit. Today, the main challenge is to define what approach to use to mitigate any network outage. To handle its operation, AmLight has been developing tools focused on troubleshooting. The OpenFlow Sniffer was developed to troubleshoot OpenFlow messages exchanged between controllers and OpenFlow devices.

This paper is organized as follows: Section 2 will present AmLight's OpenFlow Sniffer. Section 3 will detail the challenges when supporting testbeds and how the AmLight's OpenFlow Sniffer was enhanced to handle these challenges. Section 4 will conclude this paper and Section 5 will detail future work.

2. The AmLight's OpenFlow Sniffer

The OpenFlow Sniffer development at AmLight started as a result of none of the traditional tools available sufficiently support the OpenFlow 1.0 specification. Wireshark tool [Combs et al. 2016], for instance, dissects only 50% of the OpenFlow 1.0 specification, despite of it can dissect 100% of OpenFlow 1.3 [Wireshark 2016]. Furthermore, some packet inspection tools require graphical interfaces, or the captured file has to be moved to another computer to be visualized. In network environments where application servers have no support for graphical user interfaces, or where firewalls sit between users

and servers, using traditional troubleshooting tools might not be the most effective way of troubleshooting. Additionally to Wireshark, [Wundsam et al. 2011] introduced the OFRewind, a tool to help troubleshooting the Stanford University's OpenFlow network, but it was never released as a production tool.

To help the troubleshooting process at AmLight, the OpenFlow Sniffer was developed, with the following objectives: (1) it had to be fully OpenFlow 1.0 compliant; (2) it had to support filters per OpenFlow message type; (3) it had to be text-based and (4) it had to be useful for OpenFlow troubleshooting. All four objectives were part of the version 0.2, released in December 2015. Written in Python 2, with Apache License and available on GitHub [Bezerra 2015], version 0.2 allows the network engineer to read from Libpcap files and network interfaces, without affecting the production traffic, because it is entirely passive. One of the important achievements of the OpenFlow Sniffer is that it colors the most relevant fields of each OpenFlow message. These OpenFlow fields were selected to be highlighted based on the experience from previous troubleshooting processes. The following output shows an OpenFlow OFPT_FLOW_MOD message with some key OpenFlow fields highlighted.

```
2016-01-22 16:42:52 OF_Controller:6633 -> OF_Switch:32975 Size: 146 Bytes
OpenFlow Version: 1.0(1) Type: FlowMod(14) Length: 80  XID: 20
OpenFlow Match - wildcards: 3276782 dl_type: 0x800 nw_dst: 10.10.11.0/25 in_port: 53
OpenFlow Body - Cookie: 0x00 Command: Add(0) Idle/Hard Timeouts: 0/0
          Priority: 32768 Buffer ID: 0xffffffff  Out Port: 65535 Flags: SendFlowRem(1)
OpenFlow Action - Type: OUTPUT Length: 8 Port: 8 Max Length: 0
```

Next section will describe how the OpenFlow Sniffer was enhanced to troubleshooting environments that add a virtualization layer.

3. Handling a Virtualization Layer

Some OpenFlow-based networks, such as AmLight, have support for network slices [Sherwood et al. 2009], where a virtualization layer sits between controllers and switches, which increases the troubleshooting complexity. Slicing or Network Virtualization allows multiple controllers to share the same physical infrastructure.

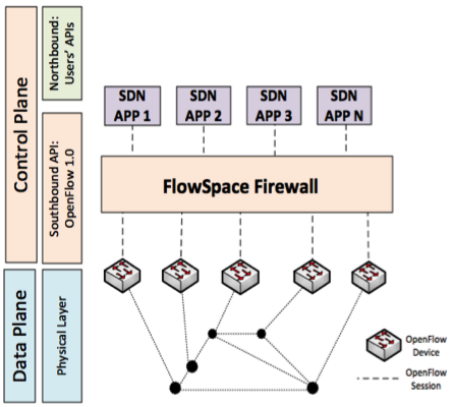

Figure 1. AmLight SDN stack

Slicing was achieved at the AmLight network using Internet2's Flow Space Firewall (FSFW) [GRNOC 2016] - an OpenFlow proxy that controls what OpenFlow controllers can do to the OpenFlow devices. To enable slicing, all OpenFlow devices must be configured to have the FSFW as its OpenFlow controller. Once an OpenFlow connection is initiated by the OpenFlow device, FSFW checks which slices have the new connected OpenFlow device included in its configuration. Each slice has a manually inserted OpenFlow controller entry associated with it; a connection between the FSFW and the controller is then established. If more than one slice has the OpenFlow device in its configuration, more OpenFlow sessions are initiated by the FSFW.

Having a TCP/OpenFlow session per-device per-controller makes it easier to manage slices, especially because OpenFlow messages do not have any field that identifies the target device once the OpenFlow session is established. The OpenFlow controller knows which switch is associated with the TCP session through the OpenFlow message OFPT_FEATURES_REPLY, used in the moment the session is established, which has the Datapath ID (DPID). The Datapath ID uniquely represents a single OpenFlow device.

Figure 1 represents the current SDN stack at AmLight after deploying OpenFlow 1.0, Flow Space Firewall and SDN applications. In the SDN stack, FSFW acts as a proxy between the physical layer and the control layer, represented by SDN applications. With FSFW, the OpenFlow communication does not happen between controllers and OpenFlow devices directly anymore. The OpenFlow Sniffer was extended to handle this peculiar situation created by OpenFlow proxies.

Figure 2 represents the process developed for the OpenFlow Sniffer to make the association between the IP and TCP ports in use by FSFW and the OpenFlow devices: (1) The OpenFlow Sniffer collects an IP packet from a file or from a network interface; (2) Once the IP packet is dissected, the OpenFlow Sniffer checks if the OpenFlow message is an OFPT_PACKET_OUT message and if it carries Link Layer Discovery Protocol (LLDP) [IEEE 2004] data; (3) If the packet is not an OFPT_PACKET_OUT message or it does not carry LLDP data, the packet is parsed and (4) the association dictionary is checked. If the source IP and source TCP port were inserted before, (5) the packet is printed with the device identification; (6) otherwise the packet is printed without any OpenFlow device identification; (7) If the packet is an OFPT_PACKET_OUT message and it carriers LLDP data, the DPID is collected. From the same IP packet, the source IP and TCP port are collected and all three are inserted in a DPID dictionary. (8) This DPID dictionary is then merged with a name dictionary created by the network admin, having the DPID and the name of the switch. In the end, an association dictionary is created having the device's name, source IP and source port. From now on, all packets will be checked against the association dictionary.

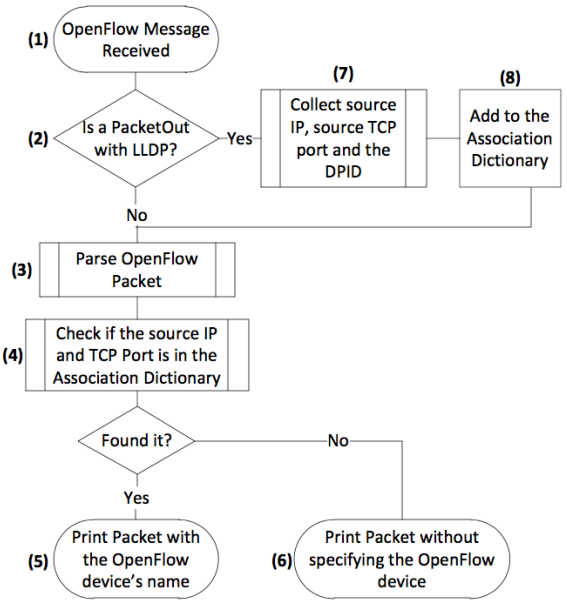

Figure 2. OpenFlow Sniffer process to discover the OpenFlow device

With this extension, network engineers will quickly identify the OpenFlow device associated to the OpenFlow message received or sent, especially in cases of OFPT_ERROR messages. As it was described, the OpenFlow Sniffer is completely passive, it does not get involved in the OpenFlow connection. With this approach, its installation brings minimum risks and complexity to the OpenFlow controller's environment.

4. Conclusion

The OpenFlow Sniffer provided troubleshooting visibility to AmLight's network operation. Real-time and historical information are available when debugging. Communication with vendors already occurs using output collected from the OpenFlow Sniffer. The skills acquired in its development, as well as the developed features are also helping the AmLight network team to validate the OpenFlow implementation of different network devices available on the market.

5. Future Work

AmLight's OpenFlow Sniffer version 0.3 is planned to be released in mid-2016. Version 0.3 will be entirely rewritten to add support for OpenFlow 1.3, for new filters, including OFPT_PACKET_OUT and LLDP. Version 0.4 has in its roadmap to add support for OpenVSwitch specification and SSL dissection.

References

[Bezerra 2015] Bezerra, J. (2015). Openflow protocol sniffer. *Available online: https://goo.gl/402xpp*.

[Combs et al. 2016] Combs, G. et al. (2016). Wireshark. *Online website: http://www.wireshark.org*.

[GRNOC 2016] GRNOC (2016). Flowspace firewall. *Online website: http://globalnoc.iu.edu/sdn/fsfw.html*.

[Ibarra et al. 2015] Ibarra, J., Bezerra, J., Morgan, H., Fernandez Lopez, L., Stanton, M., Machado, I., Grizendi, E., and Cox, D. A. (2015). Benefits brought by the use of openflow/sdn on the amlight intercontinental research and education network. pages 942–947.

[IEEE 2004] IEEE (2004). 802.1ab standard: Station and media access control connectivity discovery. *Available online: http://www.ieee802.org/1/pages/802.1ab.html*.

[ONF 2009] ONF, O. N. F. (2009). Openflow switch specification 1.0.0 (wire protocol 0x01). *Online website: http://archive.openflow.org/documents/openflow-spec-v1.0.0.pdf*.

[Sherwood et al. 2009] Sherwood, R., Gibb, G., Yap, K.-K., Appenzeller, G., Casado, M., McKeown, N., and Parulkar, G. (2009). Flowvisor: A network virtualization layer. *OpenFlow Switch Consortium, Tech. Rep*, pages 1–13.

[Wireshark 2016] Wireshark (2016). Openflow support on wireshark. *Online website: https://wiki.wireshark.org/OpenFlow*.

[Wundsam et al. 2011] Wundsam, A., Levin, D., Seetharaman, S., Feldmann, A., et al. (2011). Ofrewind: Enabling record and replay troubleshooting for networks. In *USENIX Annual Technical Conference*.