Developing Small Size Low-Cost Software-Defined Networking Switch Using Raspberry Pi

Vipin Gupta, Karamjeet Kaur and Sukhveer Kaur

Abstract Software-defined networking (SDN) is a new emerging technology for networking that separates the forwarding and control plane. With SDN static, inflexible and complex network are replaced by dynamic, scalable, and innovative networks. The motivation of developing low-cost portable SDN switch arose when we were developing load balancing and stateful firewall SDN applications during our research work. To test and measure the performance of our applications, we needed low-cost SDN testbed. Existing solutions were utilizing special hardware such as NetFPGA or real switches. But these were not suitable due to high costs and complexity involved. We could have tested these applications using Mininet emulator but there are performance issues. In this paper, we created a small size, low cost, portable SDN switch for testing our SDN applications using Raspberry Pi. Our low-cost switch supports OpenFlow Specification 1.0–1.4. Raspberry Pi is Linux-based small size low-cost device which can be used as a personal computer as well as for making low-cost portable SDN switch.

Keywords Open vSwitch · OpenFlow · Raspberry Pi · SDN · Controller

1 Introduction

Computer network consists of a large number of network devices such as routers, switches, and various middleboxes such as firewalls, load balancers, network address translators having complex protocols on them. Network operators are

V. Gupta
U-Net Solutions, Moga, Punjab, India
e-mail: vipin2411@gmail.com

K. Kaur (✉) · S. Kaur
AD College, Dharamkot, Punjab, India
e-mail: bhullar1991@gmail.com

S. Kaur
e-mail: bhullarsukh96@gmail.com

D.K. Lobiyal et al. (eds.), *Next-Generation Networks*, Advances in Intelligent Systems and Computing 638, https://doi.org/10.1007/978-981-10-6005-2_16

responsible for configuring individual network devices using configuration interface.

Software-defined networking is a new networking concept in which the data plane is decoupled from control decision plane [1]. Data planes are actually dumb merchant silicon boxes. We can turn them into a simple hub, learning switch, or a router by creating flow entries in flow tables. SDN in part represents logically centralized network intelligence in control plane and data plane become simple packet forwarding device that can be programmed via open interface. OpenFlow is a prominent example of such an interface [2, 3]. OpenFlow switch has flow tables that contain packet handling rules. When a rule matches with the incoming traffic, then corresponding action such as dropping, forwarding, and flooding is taken. According to the flow table rules, OpenFlow switch behaves like a switch, router, hub, or firewall [4].

SDN is getting a lot of attention from research community as well as industry [5]. Open network foundation (ONF) has been created for promoting SDN and standardizes the OpenFlow (Fig. 1).

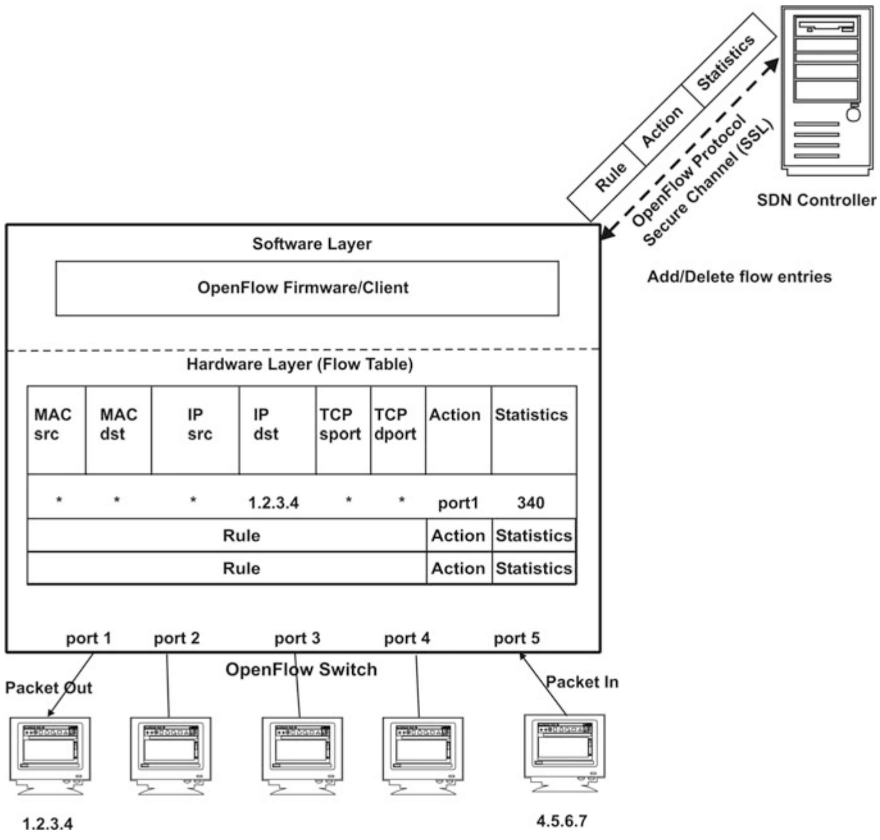


Fig. 1 SDN architecture

2 Related Work

Mininet [6, 7] is a emulator software that enables you in creating large networks on a simple laptop or virtual machine (VM). It allows you to create simple as well complex networking consisting of switches, controllers, hosts, and links. It provides a simple, robust mechanism for testing OpenFlow applications. But there are scalability issues and resource constraints when we run Mininet on a single system. Many other researchers use special devices such as NetFPGA for creating testbeds. But these SDN testbeds are not suitable due to higher cost and complexity [8]. Earlier work supported OpenFlow specification 1.0 [9], while in case of our work, it supports OpenFlow Specification 1.0–1.4.

Raspberry Pi is small size low-cost device which can be used as personal computer as well as for making low-cost SDN switch. Raspberry Pi is basically a device using embedded Linux.

3 Steps for Developing SDN Switch

Our switch was made using Raspberry Pi which comes preloaded with Raspbian operating system. For our switch, we used latest Raspberry Pi model B+ which also comes with 1 GB Ram instead of 512 MB Ram in earlier versions. The Raspberry Pi is a small, powerful, and lightweight ARM-based computer [9]. We loaded the latest Ubuntu MATE 15.04 on Raspberry Pi. Raspberry Pi contains only one LAN card. Since we wanted four ports SDN switch, so we ordered three USB-based low-cost LAN cards online. The following are the steps for converting our Raspberry Pi system to a SDN switch.

1. Attached three USB LAN cards to our Raspberry Pi system thus making total number of LAN cards available to four.
2. We downloaded a pre-built image of Ubuntu MATE which is available on the Internet [10]. We unzipped that image file and wrote the Ubuntu MATE image file on MicroSD card that comes along with Raspberry Pi system. It removed the default Raspbian image on MicroSD. We removed the Raspbian OS because it does not support the latest version of OpenFlow switch (Fig. 2).
3. We used the ‘apt-get install’ for installing the Open vSwitch packages on Raspberry Pi.
4. We used the ‘óvs-vsctl’ for making our Raspberry Pi as SDN switch and added four LAN cards as four ports of our SDN switch.
5. We attached four laptops to four ports of our Raspberry Pi SDN switch. One laptop was used as client system, one laptop as POX/RYU controller and two other laptops as servers.
6. First, we tested our load balancing application using POX controller [11, 12]. Our Raspberry Pi switch was properly working as a load balancer.

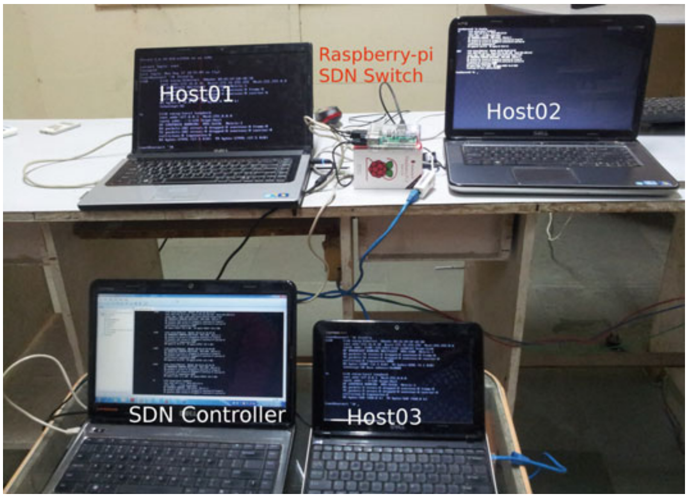


Fig. 2 Raspberry Pi-based SDN laboratory

7. Secondly, we tested our firewall application using RYU controller [13, 14]. Our Raspberry Pi switch was now properly working as firewall.

4 Laboratory Setup

For testing our switch, we created the laboratory as shown figure. Our switch ports were named eth0, eth1, eth2, eth3. We attached our POX controller on eth0 and hosts on eth1, eth2, and eth3 ports. We tested our Raspberry Pi-based SDN switch using load balancing application and firewall application (Fig. 3).

The load balancing architecture consists of OpenFlow switch network with a POX controller and multiple servers connected to the ports of the OpenFlow switch. Each server is assigned static IP address, and the POX controller maintains a list of live servers that are connected to the OpenFlow switch. Web service is running on each server on a well-known port 80.

A firewall allows or rejects a specific type of data. Our firewall application allows or restricts the traffic based on MAC addresses (Layer 2), source and destination IP addresses (Layer 3), ports (Layer 4). When a packet enters into the switch, the packet header is matched against the firewall rules.

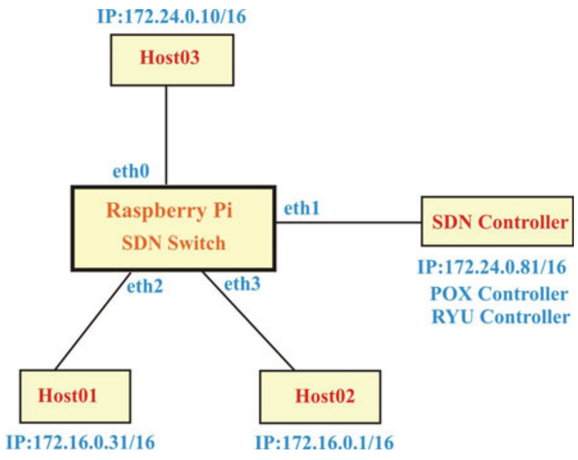
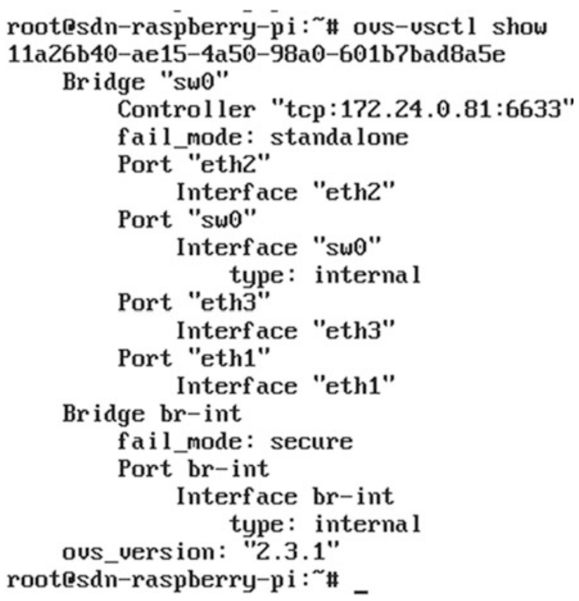


Fig. 3 Laboratory setup



```
root@sdn-raspberry-pi:~# ovs-vsctl show
11a26b40-ae15-4a50-98a0-601b7bad8a5e
    Bridge "sw0"
        Controller "tcp:172.24.0.81:6633"
        fail_mode: standalone
        Port "eth2"
            Interface "eth2"
        Port "sw0"
            Interface "sw0"
                type: internal
        Port "eth3"
            Interface "eth3"
        Port "eth1"
            Interface "eth1"
    Bridge br-int
        fail_mode: secure
        Port br-int
            Interface br-int
                type: internal
    ovs_version: "2.3.1"
root@sdn-raspberry-pi:~# _
```

Fig. 4 Switch ports for connecting hosts

We tested our load balancing application using POX controller installed on ‘172.24.0.81’ host. Firewall application was tested using Ryu controller installed on ‘172.24.0.81’. Raspberry Pi SDN switch was configured to use remote controller as shown in figure. Both of these applications are working properly on Raspberry Pi-based SDN switch (Fig. 4).

5 Conclusion

Here we have successfully developed a SDN switch using Raspberry Pi by installing Ubuntu MATE Linux and other open source softwares. We were able to successfully test our load balancing and firewall applications. This switch is very low cost and portable as compared to other available alternatives in the market. Future work can involve creating a SDN testbed consisting of Raspberry-based switches and hosts.

References

1. Mendonca, M., Nunes, B.A.A., Nguyen, X.N., Obraczka, K., Turletti, T.: A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks (2013)
2. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. **38**(2), 69–74 (2008)
3. Hegr, T., Bohac, L., Uhlir, V., Chlumsky, P.: OpenFlow deployment and concept analysis. Adv. Electr. Electron. Eng. **11**(5), 327–335 (2013)
4. Lara, A., Kolasani, A., Ramamurthy, B.: Network Innovation Using Openflow: A Survey, pp. 1–20 (2013)
5. Feamster, N., Rexford, J., Zegura, E.: The road to SDN: an intellectual history of programmable networks. ACM SIGCOMM Comput. Commun. Rev. **44**(2), 87–98 (2014)
6. Handigol, N., Heller, B., Jeyakumar, V., Lantz, B., McKeown, N.: Reproducible network experiments using container-based emulation. In: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, pp. 253–264. ACM (2012)
7. Lantz, B., Heller, B., McKeown, N.: A network in a laptop: rapid prototyping for software-defined networks. In: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, p. 19. ACM (2010)
8. Naous, J., Erickson, D., Adam Covington, G., Appenzeller, G., McKeown, N.: Implementing an OpenFlow switch on the NetFPGA platform. In: Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, pp. 1–9. ACM (2008)
9. Kim, H., Kim, J., Ko, Y.-B.: Developing a cost-effective OpenFlow testbed for small-scale software defined networking. In: 2014 16th International Conference on Advanced Communication Technology (ICACT), pp. 758–761. IEEE (2014)
10. https://ubuntu-mate.org/raspberry-pi/
11. Kaur, S., Singh, J., Ghumman, N.S.: Network Programmability Using POX Controller
12. POX at https://openflow.stanford.edu/display/ONL/POX+Wiki
13. Shalimov, A., Zuikov, D., Zimarina, D., Pashkov, V., Smeliansky, R.: Advanced study of SDN/OpenFlow controllers. In: Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia, p. 1. ACM (2013)
14. Lin, T., Kang, J.-M., Bannazadeh, H., Leon-Garcia, A.: Enabling SDN applications on software-defined infrastructure. In: Network Operations and Management Symposium (NOMS), 2014 IEEE, pp. 1–7. IEEE (2014)