Developing a Cost-Effective OpenFlow Testbed for Small-Scale Software Defined Networking

Hyunmin Kim*, Jaebeom Kim**, Young-Bae Ko**

* Graduate School of Software, Ajou University, Korea

** Graduate School of Information and Communication, Ajou University, Korea

kimhm@uns.ajou.ac.kr, jaebeom@uns.ajou.ac.kr, youngko@ajou.ac.kr

***Abstract*—OpenFlow is the first standard interface for realizing Software-Defined Networking (SDN) that can decouple the data and control plane to provide scalable network management. To validate the performance and features of the OpenFlow standard, many researchers have utilized specialized hardware network devices such as NetFPGA. However, these devices are not suitable for implementing a small-scale SDN testbed due to high cost, complexity, and specialized programming languages. The well-known SDN emulator, Mininet[1], is also widely utilized but it is not enough to support network dynamicity and the performance of the virtualized hosts. In this paper, we suggest a more cost-effective alternative of implementing SDN testbed with Open vSwitch (OVS), based on the Raspberry-Pi that is a low-cost embedded Linux machine. We validate our testbed with the OpenFlow specification 1.0 and prove that its maximum network throughput shows almost the same performance compared to the NetFPGA-1G.**

***Keywords*— Software Defined Networking; OpenFlow; open Vswitch; Raspberry-Pi**

I. INTRODUCTION

Recently, amounts of network traffic has explosively increased due to advances in IPTV, smartphone and various smart devices. These user traffic have also become more and more complicated due to service variations and requirements of the users. However, the current network infrastructure cannot handle these service requirements because the traditional network architecture integrates the forward plane and control plane into the same device [2]. To designa more flexible and scalable network, the Software Defined Networking (SDN) paradigm has been proposed in recent years. The main characteristic of the SDN decouples the control and network plane. Thus, network can be dynamically managed depending on networking policies such as routing and service prioritization [3].

There are three evaluation methods of the SDN architecture: Mininet emulator, net-FPGA, and OpenFlow based S/W switch. Mininet is a SDN emulator that includes a collection of virtual end-hosts, switches, and devices, and can be used do design virtual links without using real devices. On the other hand, various research institutes have constructed national-wide SDN testbed such as "OF@TEIN", and "OFELIA" [4]. However, these projects are very large scale, which are not suitable for testbed-based evaluations for smaller lab-scale experimentations. Instead of these larger experiments of SDN, net-FPGA can be utilized for smaller scale, independent analysis of SDN. However net-FPGA also may pose some problems such as high cost, complexity and use of specialized programming languages [5].

Small-scale SDN testbed can validate and test operation of the various OpenFlow applications or functions of the SDN controller more dynamically compared to large-scale testbed. To make a suitable testbed for evaluating small-scale SDN, we suggest a simple and cost-effective testbed using Raspberry-Pi. In our SDN testbed, all of the SDN devices such as SDN controller and host are built on the same device. Thus, reconfiguration and maintenance of the testbed is easier than net-FGPA. The evaluation results of the implemented SDN testbed shows near similar performance compared to SDN implemented on 1Gbps net-FPGA device.

II. BACKGROUND AND RELATED WORK

In this section, we describe the SDN architecture for building SDN testbed. Also, we briefly describe some related SDN testbed implementations. SDN architecture can be divided into three layers; infrastructure layer, control layer, and application layer. The overview of SDN architecture is shown in Figure 1. The SDN infrastructure layer consists of interaction between switches, routers, and network hosts. Only the forwarding plane is implemented in the infrastructure layer, thus routes cannot be found by the infrastructure layer alone. To create routes between devices, each device sends a request message to the SDN controller which is located in control layer through a secure channel [6]. All of the control policy in the SDN are decided by each type of application in the application layer. Also, decided policies are delivered to each devices using secure channel, and southbound APIs that are the communication APIs to make the link between upper layer and bottom layer functions [3].

A. OpenFlow

OpenFlow is a standard interface that allows researchers to directly control how packets are routed in real SDN. OpenFlow is based on Ethernet switch, but maintains an open protocol that can be used to program the flow-table in various switches and routers [6].

OpenFlow organizes three components, which are Flow table, secure channel and OpenFlow protocol. Flow table consists of flow entries that decides how data flows are

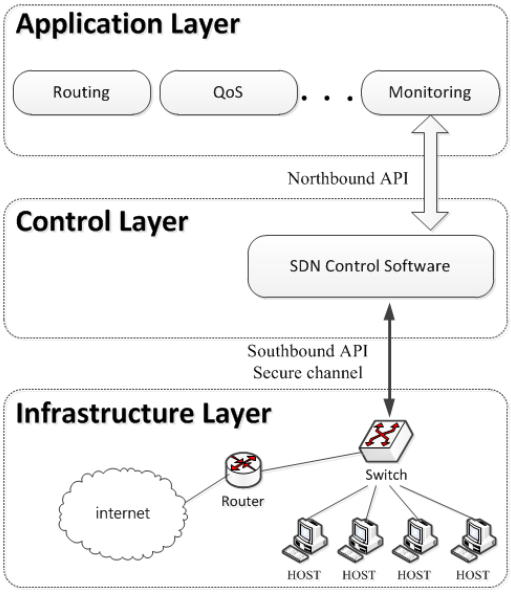

Figure 1. Software Defined Networking architecture

processed in the network. Through flow entries, data flows can be dynamically adjusted and transmitted in the network infrastructure [7]. Secure channel is used as a communication means between the SDN switch and controller to establish a secure connection. OpenFlow protocol provides a standard interface that can be defined externally by researchers, thus avoiding additional programming in switches

B. Net-FPGA and other commercial testbeds

Net-FPGA is a platform for build high performance networking systems in hardware.

The advantage of Net-FPGA based programmable router is that the packets can be processed at line-rate in a user way. Net-FPGA consists of PCI based board and it has to plug-in linux based PC. Net-FPGA includes two Static RAMs (SRAMs) that operate synchronously with the Field Programmable Gate Array (FPGA). A quad-port physical-layer transceiver (PHY) is provided enabling the platform to send and receive packets over four, standard twisted-pair Ethernet cables. Two Serial Advanced Technology Attachment (SATA) connectors on the platform can be attach multiple Net-FPGAs within a system to exchange data at high speeds [8]. It can be efficiently deployed in enclosed areas such as office, building and laboratory alongside OpenFlow enabled switches.

Net-FPGA can be considered as a good component in OpenFlow environment. However, it has two weakness points. Firstly, developer must be proficient in the low-level programming language and design tools [9]. This is because the Net-FPGA library consists of Verilog skeleton design. Also The Net-FPGA employ standard Computer Aided Design (CAD) tool flows to implement the circuits that run on the FPGA [10].A large number of companies have also developed commercial network switches that enable OpenFlow. Several of these products use commercial OpenFlow-enabled switch to build testbeds such as NEC IP8800, WiMax, HP Procurve 5400, Cisco Catalyst 6k and Quanta LB4G [11]. However, these switches are not suitable for small scale SDN testbed due to high cost and restricted modification.

C. Linux PC based software switch

Generally, the underlying architecture of OpenFlow uses linux PC-based software switch. B. Pfa et al. [7] implement OpenFlow kernel module under general purpose linux PC using OVS. OVS is one of the OpenFlow software switches which provides open and accessible designs by using open source software [12]. The OVS provides connectivity between the virtual machines and the physical interfaces.

Raspberry-Pi is the arm-based embedded system witch is more suitable than Net-FPGA in small-scale SDN environment because of low-cost, easy-programing, standardized device drivers. The cost of the Raspberry-Pi is only 35 dollars [13] but a 1Gbps Net-FPGA interface card is 1300 dollars also it requires a specialized main platform to equip them.

III. PROPOSED TEST-BED ARCHITECTURE

The proposed testbed includes three network devices which are SDN controller, SDN switch and host device. All devices in proposed testbed are built on standard raspberry-pi embedded machine and uses general linux kernel based operating system called Raspbian [14]. Thus, all devices can be easily reconfigured to evaluate various network environments. In this chapter we describe the network design and software architecture of each network devices in our testbed.

A. Network design

Network substrate consider of interconnections between SDN Hosts as shown in Figure 2, if the host that is connected with OVS, then OVS identifies which flow-entry is matched. If the OVS cannot find any entry, then OVS sends request message to Floodlight based SDN controller.

Programmable network can be dynamically provided by composing the flow's route. For example, packets can be routed via physical node for a specific service or light loaded path based on their priority. The way for providing programmable network is to make the QoS metrics, or to make traffic route manually by user's control. When these invoked user requirements are satisfied, Floodlight controller responses a

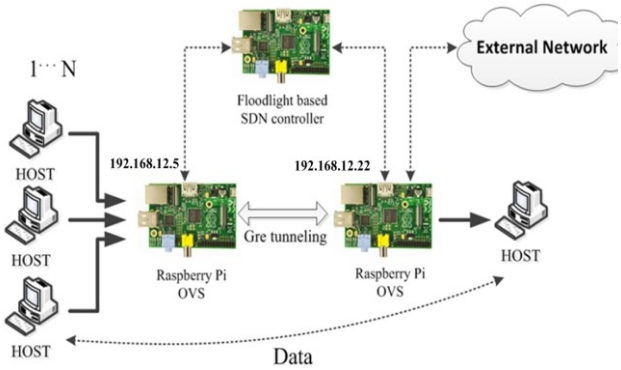

Figure 2. Network design of the Raspberry-Pi based SDN testbed

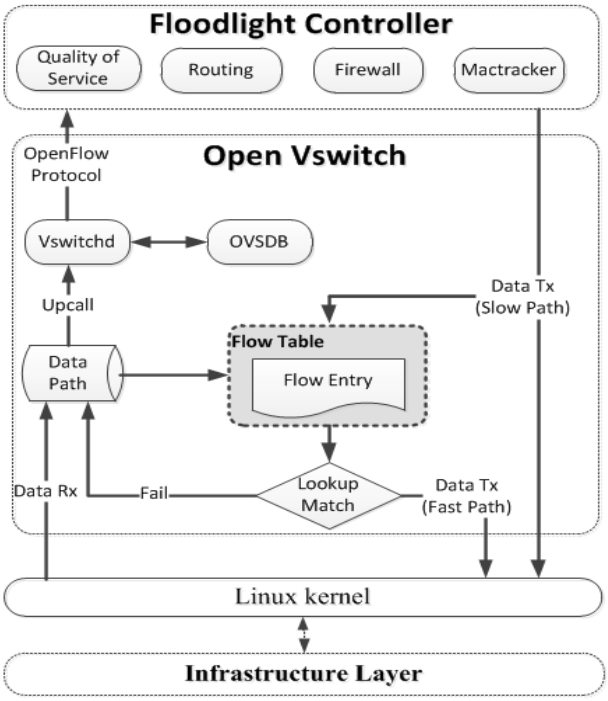

Figure 3. Software architecture of the raspberry-pi based SDN testbed

message to OVS. It updates the flow entry and transmits to the destination host.

B. Software design

Figure 3 shows the software design architecture of the proposed testbed. We consider the interactions between Raspberry-Pi, OVS and Floodlight controller on the Raspbian linux kernel [15]. Raspberry-Pi linux kernel is 3.7.11+ version based on Ubuntu and OVS version is 2.0.90. The proposed Floodlight controller consists of the basic modules: QoS, basic routing, firewall, and MACtracker. Thus, all of the devices can be dynamically changed as a network switch or host.

C. SDN controller device

The SDN controller in our testbed uses Floodlight controller software. The Floodlight controller can handle a large of amount equipment while maintaining a high level of service. Thus, various applications such as QoS control, Load-balance, and topology viewer of the SDN can be utilized in our testbed. Also it provides a rich set of APIs to perform operations on the underlying OpenFlow network. Floodlight controller can not only easily control list of the modules but also write on them using the popular JAVA language.

D. SDN switch device

Raspberry-Pi has a 1Gbps Ethernet interface that is not sufficient to process multiple connections individually. To solve this problem, we create the virtual interfaces using OVS open source program.

IV. PERFORMANCE EVALUATION

We implement the suggested SDN testbed and evaluate the maximum throughput. Also, operation of the SDN function is validated using OpenFlow white paper. Table 1 and Figure4 shows the validation result of the OpenFlow functions in our testbed. The items in the check list of Table 1 was decided depending on the mandatory function of the SDN. The result shows that our testbed successfully operate SDN functions and validation of each result are done using controller applications. Figure 4 shows the terminal screenshot of the MACtracking result in controller side that includes connected MAC and secure channel information of the SDN switch.

The performance evaluation is done by throughput comparison using performance evaluation result of the net-FPGA tested by M. K. Park et al [16]. They are experiment environment is heterogenetic such as internet environment not support OpenFlow. According to this paper we build similar test environment as Figure 2. In our test scenario, a host sends the number of traffics using iperf [17] depending on various types of maximum segment size. We validate the throughput using iperf tool also same configuration the MSS. Figure 5 shows the maximum throughput between various schemes as Mininet emulator, net-FGPA, Raspberry-Pi. Mininet is a software emulator which is combining virtualization with an extensible CLI and API, and it also provides a rapid prototyping workflow to create, customize and share a SDN to running on real hardware. We customize the Mininet from same environment our testbed and connect our Floodlight controller.

TABLE 1. VALIDATION RESULT OPENFLOW FUNCTIONS

Function	Result
OpenFlow interface creation	OpenFlow interfaces such as OVS, basic applications and controller.
SDN Controller connection	Validate using ping test from host to end-host. The ping message is used to reach the end-host.
Flow table creation/remove	When connected to the floodlight controller, flow entry is created and propagated to OVS which updates it into the Flow table.
Port state control	Port status using OVS.
MAC address learning control	Floodlight controller application can be MAC address learning real raspberry-pi hosts and also MAC address too.
Backward compatibility	Through connection of public internet, host can access through Open Vswitch.

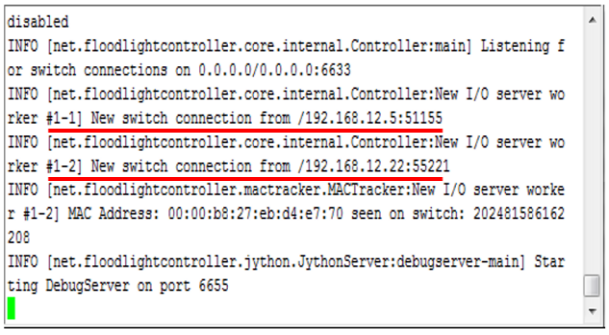

Figure 4. MACtracking operation result of the Raspberry-Pi based SDN testbed measrured by SDN controller.

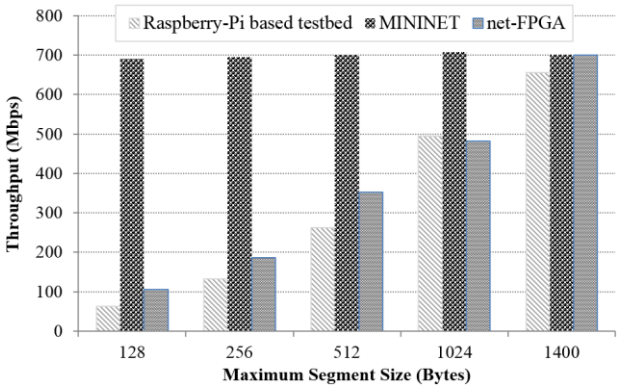

Figure 5. Maximum throughput comparison in various packet size

The experiment result of our testbed shows the similar performance compared to 1Gbps net-FPGA hardware. However, the Mininet emulator does not show accurate results. This is because the Mininet generates a virtualized network infrastructure, which cannot consider various real-life factors and parameters.

V. CONCLUSIONS

In this paper, we suggest a Rasberry-pi based open source software using a small-scale SDN testbed and software architecture. Proposed testbed has several benefits such as low-cost, low-complexity and easy programmability. Also, the evaluation result shows the similar performance with 1Gpbs net-FGPA device. Also, important OpenFlow functionalities are successfully operated. For future works, we will extend the proposed SDN functionalities from wired to wireless networks using the proposed testbed and software architecture.

ACKNOWLEDGMENT

This research was partially supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2012R1A1B3003573) and MKE (The Ministry of Knowledge Economy), Korea, under IT/SW Creative research program supervised by the NIPA (National IT Industry Promotion Agency)" (NIPA-2013- H0502-13-1059)

REFERENCES

[1] The Mininet website. [Online]. Available: http://yuba.stanford.edu/foswiki/bin/view/OpenFlow/Mininet.

[2] Y. Yiakoumis, K.-K. Yap, S. Katti, G. Parulkar, and N. MCKEOWN, “Slicing home networks.” In *Proc. HomeNets’11*, 2011.

[3] S. Sezer, S. Scott-hayward, P.K Fraser, D. Lake, J. Finnegan, N. Vijoen, M. Miller, and N. Rao, "Are we ready for SDN? Implementation challenges for software-defined networks." *Communications Magazine, IEEE* Vol. 51(7), 2013.

[4] A. Köpsel and H. Woesner, “OFELIA – Pan-European Test Facility for OpenFlow Experimentation”, *Lecture Notes in Computer Science*. Vol. 6994/2011. 2011

[5] G. Lu, C. Guo, Y. Li, Z. Zhou, T. Yuan, H. Wu, Y. Xiong, R. Gao, and Y. Zhang. “ServerSwitch: A Programmable and High Performance Platform for Data Center Networks,” in *Proc. NSDI’11*, 2011.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. “Openflow: enabling innovation in campus networks.” *SIGCOMM Computer Communication Review*., vol. 38, pp. 69–74, Apr. 2008.

[7] B. Pfa, J. Pettit, T. Koponen, K. Amidon, M. Casado, and S. Shenker. “Extending Networking into the Virtualization Layer.” In *Proc. HotNets-VIII’09*, 2009.

[8] The net-FPGA website. [Online]. Available: http://netfpga.org/

[9] G. Ibáñez, B. De Schuymer, J. Naous, D. Rivera, E. Rojas, and J. A. Carral, “Implementation of arp-path low latency bridges in linux and openflow/netfpga.” in *Proc. HPSR’11*,2011.

[10] J. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, and J. Luo, “NetFPGA–An Open Platform for Gigabit-rate Network Switching and Routing,” in *Proc MSE’07*, 2007.

[11] N. McKeown, “ Software-defined networking,” INFOCOM keynote talk. Available:http://www.cs.rutgers.edu/~badri/552dir/papers/intro/nick09.pdf, Apr. 2009.

[12] V. Tanyingyong, M. Hidell, and P. Sj¨odin, “Improving pc-based openflow switching performance,” in *Proc. ANCS ’10*, 2010.

[13] J. D Brock, F. B Rebecca, and E. C Marietta, "Changing the world with a Raspberry Pi." *Journal of Computing Sciences in Colleges,* vol. 29(2) pp. 151-153, 2013.

[14] The Raspbian website. [Online]. Available: http://www.raspbian.org.

[15] G. Ibáñez, B. De Schuymer, J. Naous, D. Rivera, E. Rojas, and J. A. Carral, “Implementation of arp-path low latency bridges in linux and openflow/netfpga.” in *Proc. HPSR’11*,2011.

[16] M. K. Park, J. Y. Lee, B. C. Kim and D. Y. Kim, “Implementation of a Future Internet Testbed on KOREN based on NetFPGA/OpenFlow Switches,” *NetFPGA Developers Workshop*, stanford, CA, 2009.

[17] The iperf website. [Online]. Available: http://sourcerforge.net/projects/iperf/.

Hyunmin Kim received his B.S degree in Electronic Engineering from the KyungHee University, Korea, in 2013. He is currently a M.S course student in the Software Engineering of Ajou University, Korea. His research interests are in the areas of Software Defined Networking, Wireless LAN, and Smart Grid Communications.

Jaebeom Kim received his B.S degree in Computer Engineering from the Korea Polytechnic University, Korea, in 2010. He is currently a Ph.D candidate in the School of Information and Computer Engineering of Ajou University, Korea. His research interests are in the areas of network virtualization, software defined networking, wireless multi-hop networking, and Smart Grid Communications.

Young-Bae Ko is currently a Professor in the School of Information. He was also a visiting professor of Coordinated Science Lab at University of Illinois, Urbana Champaign (UIUC) for the 2008–2009 academic year. Prior to joining Ajou University in 2002, he was with the IBM T. J. Watson Research Center, Hawthorne, New York, as a research staff member in the Department of Ubiquitous Networking and Security. He received his Ph.D. degree in computer science from Texas A&M University, and B.S. and M.B.A. degrees from Ajou University. His research interests are in the areas of mobile computing and wireless networking. In particular, he is actively working on mobile ad hoc networks, wireless mesh/ sensor networks, and various ubiquitous networked system issues. He was the recipient of a Best Paper award from ACM Mobicom 1998. He has served on the program committees of several conferences and workshops. He also serves on the editorial board of ACM Mobile.