OpenFlow Version Roadmap

Ching-Hao, Chang and Dr. Ying-Dar Lin
Email: {chchang2222, ydlin}@cs.nctu.edu.tw
September 11, 2015

Abstract

Software-defined networking is an emerging networking architecture that enables flexible network control by separating the control plane and data plane while OpenFlow is the key to wide-spread adoption of SDN. The development of the OpenFlow has an inseparable relation with the prospection of SDN. Thus, in this study, we provide an overview of OpenFlow's evolution by comparing major feature changes, and discuss about the underlying reasons. We will also examine the maturity of products by performing conformance test against two OpenFlow-enabled software switches: Open vSwitch and Lagopus, with two test tools: Ryu test and OFTest. The test results show that both switches have passed conformance test for OpenFlow 1.3, and the number of optional features implemented by Lagopus has outnumbered Open vSwitch; however, Lagopus would crashed under certain test we conducted, thus its stability is still in doubt.

Keywords: SDN, OpenFlow, Conformance Test

1 Introduction

In this study, we will first discuss the basic concept of SDN and provide a brief introduction to OpenFlow and ONF organization. Next, we will present the evolution of OpenFlow and elaborate on major feature changes across versions and discuss the reasons behind it. Then we examine the maturity of product by testing two commonly used OpenFlow-enabled software switches with two testing tools. We will compare the functionality and test coverage of both tools and explain how we setup the testbed along with test results and some observation regarding it. Finally, we present the conclusion of the study.

1.1 Software-Defined Networking

Software-defined networking (SDN) [1] is regarded as the second wave of cloud computing, with the first wave being virtualization and centralization of servers. This

burgeoning network architecture enables more flexible and delicate network control by decoupling the control plane (the head) and the data plane (the body). The control plane, which makes decisions about how packets should flow, is centralized into a controller; the controller is an application running on a server or a virtual machine that sends rules to the data plane with control packets via open standardized protocols such as OpenFlow [2]. The data plane in the switches, where the packets actually move from place to place, accepts rules sent from the controller and forward packets accordingly. Figure 1 shows the architecture of SDN.

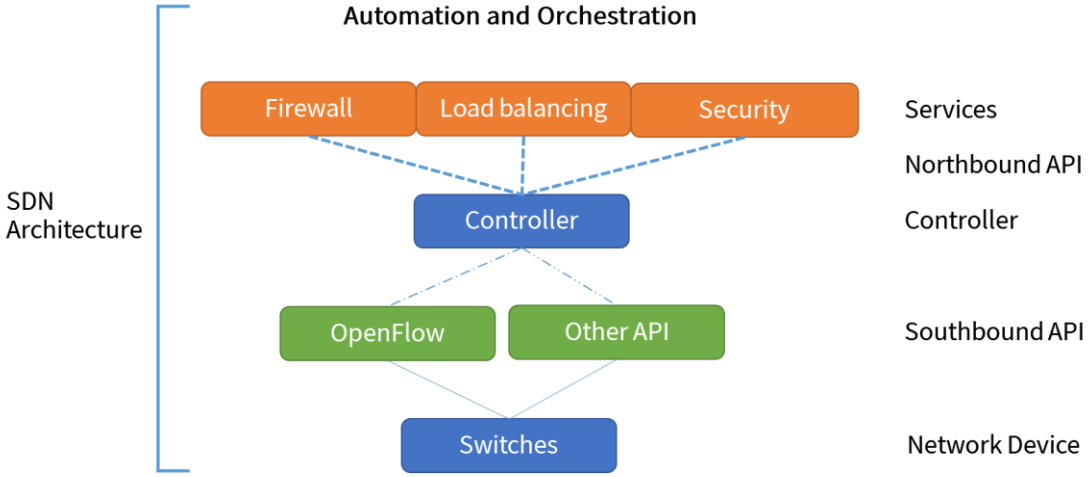

Figure 1 SDN Architecture

In order for controller to communicate with the switches, we need a well-defined and standardized protocol termed southbound API [3]. There are many organizations contributing to the standardization of SDN, and one of which is Open Network Foundation.

1.2 Open Network Foundation

Open Network Foundation (ONF) [4] is a user-driven organization dedicated to the promotion and adoption of SDN and also the design of the OpenFlow protocol. The organizational structure of ONF is shown in Figure 2. Currently, there are 11 working groups in ONF. Among them, the Extensibility working group is the main character in managing the OpenFlow standard, the Configuration and Management working group maintains another protocol called OF-CONFIG, which is used to configure and manage the OpenFlow-enabled switch itself. In addition, the Forwarding Abstraction working group contributes to the open standard of data plane implementations such as Negotiable Dataplane Model (NDM)[5]. Finally, the Testing and Interoperability

working group establishes OpenFlow conformance test specifications [6].

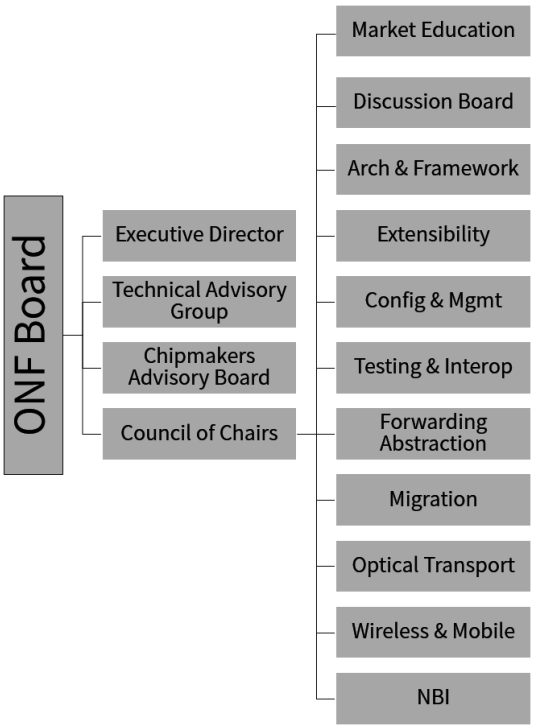

Figure 2 ONF organizational structure

1.3 OpenFlow

OpenFlow is the first standardized interface and the most commonly used protocol designed specifically for SDN. It is an open protocol for communication between controllers and switches. An OpenFlow-enabled switch is called an "OpenFlow Switch" [7]; Figure 3 shows the architecture of an OpenFlow switch.

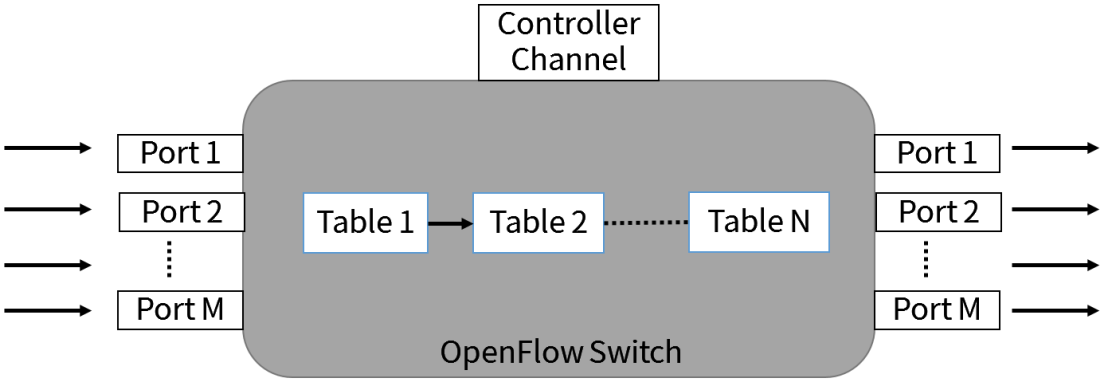

Figure 3 OpenFlow switch architecture [7]

OpenFlow switch is characterized by three components [7]:

- Flow table
- Secured channel
- OpenFlow protocol

An OpenFlow switch may contain one or more flow tables, a secured channel that connects it to the controller, and OpenFlow protocol as the way for it to communicate with the controller. A flow table consists of flow entries. The structure of the flow entry in OpenFlow 1.5 is shown in Table 1.

Table 1 components of a flow entry in OpenFlow 1.5 [7]

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flag

Each flow entry contains:

- Match fields: match against packet
- Priority: matching precedence of the flow entries
- Instructions: set of instructions that are executed when a packet matches the entry
- Timeouts: maximum amount of idle time
- Cookie: opaque data value chosen by the controller
- Flags: alter the way flow entries are managed

When a packet arrives from a switch port, it is compared with the match fields in the flow entries. If the packet is matched, it will be processed as indicated in the instructions.

2 OpenFlow Evolution

OpenFlow protocol have evolved during ONF's standardization process, from version 1.0 where there are only 12 fixed match fields and a single flow table to the latest version that features multiple tables, over 41 matching fields and a bunch of new functions.[7] The capability and scalability have been largely extended. In this section, we are going to present the major changes in each version as shown in Table 3 and discuss the underlying reasons.

Table 2 major version changes in each version [7]

Version	Major Feature	Reason	Use Cases
1.0 - 1.1	Multiple table	Avoid flow entry explosion	
	Group Table	Enable Applying action sets to group of flows	Load balancing, Failover, Link Aggregation
	Full VLAN and MPLS Support		
1.1 - 1.2	OXM Match	Extend matching flexibility	
	Multiple Controller	HA/Load balancing/Scalability	Controller Failover, Controller Load Balancing
1.2 - 1.3	Meter table	Add QoS and DiffServ capability	
	Table miss entry	Provide flexibility	
1.3 - 1.4	Synchronized Table	Enhance table scalability	Mac Learning/Forwarding
	Bundle	Enhance switch synchronization	Multiple switch configuration
1.4 - 1.5	Egress Table	Enabling processing to be done in output port	
	Scheduled bundle	Further enhance switch synchronization	

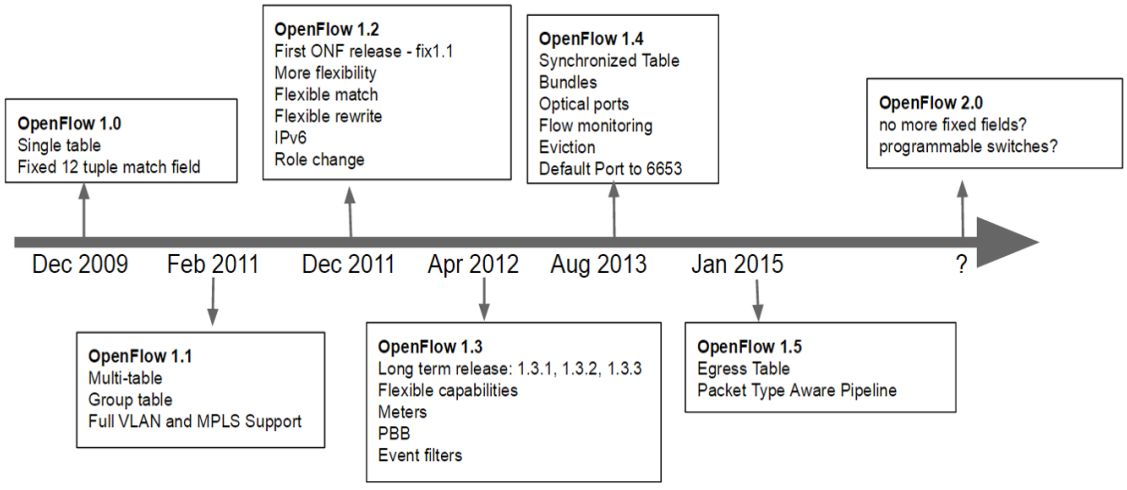

Figure 4 OpenFlow version timeline

OpenFlow 1.0 → OpenFlow 1.1

Back in 2009, with the very first release of OpenFlow, version 1.0, there is only

one flow table with three components in a flow entry: Header Fields, Counters and Actions, Also, the header fields contains only 12 fixed matching elements. There isn't much flexibility due to the limited matching capability and a single table. Switches using OpenFlow 1.0 forwarding model cannot perform more than one operation during the packet forwarding process because there is only one flow table, and this leads to flow entry explosion[8], which greatly reduces the usability of OpenFlow. As a result, in version 1.1, OpenFlow introduced two major enhancements: Multiple Tables and Group Table. In the meanwhile, they renamed the Header Fields and Actions component to Match Fields and Instructions due to the fact that Header Fields does not exactly match the "headers" of the packet[9] In addition, with more tables, more actions can be added to the action set with each table. The multi-table greatly reduces the entries in a two-stage processing [7] use-case such as mac address learning and VRF.
From a different perspective, the multi-table also brings a new challenge to switch vendors. Current OpenFlow switches commonly use existing chips by overlay the flow tables on original packet processing pipeline. This is easy to do with OpenFlow 1.0 because there is only a single flow table. With OpenFlow 1.1 and above, complicate flow-table pattern and flexible packet process pipeline with multiple table is challenging to implement. To solve this problem, the Forwarding Abstraction WG in ONF proposed Negotiable Dataplane Model; with NDM, switch vendors can provide the switch operators with a list of supported Table Type Patterns. The switch operators may purchase the device with supports of the TTPs which meet their requirements.

Another new feature introduced in OpenFlow 1.1 is the "Group Table". The group table consists of group entries. Group entries can be divided into four types [7]:

- All: execute all action buckets in the group
- Select: execute one action bucket in the group
- Indirect: execute the one defined action bucket in the group
- Fast failover: execute the first live action bucket

The "all" type enables implementations of functions like multicasting in OpenFlow switch because packets are forwarded to multiple port. The "select" type enables implementation of functions such as load-balancing and link aggregation by selecting one action bucket to execute at a time. The "indirect" type increased scalability of flow table by categorizing flows into groups with same actions, which makes it more efficient to implement functions such as default routing. The "fast failover" detects live action bucket to execute, that is, actions with an active port, which is suitable for function such as high availability.

OpenFlow 1.1 → OpenFlow 1.2

With more and more people embracing OpenFlow, users' feedbacks flood in, requesting for more match fields to suit their various needs. Previous versions of OpenFlow uses a fixed length structure for match statements, thus it limits the flexibility. OpenFlow 1.2 introduces a Type-Length-Value (TLV) structure, allowing new match fields to be added in a more modular way, called OpenFlow Extensible Match (OXM). With OXM, new matching criteria can be adopted swiftly, dramatically increasing the match fields' flexibility. In addition, OpenFlow 1.2 adds basic supports for IPv6 based on OXM. The match fields' evolution from OpenFlow 1.2 to OpenFlow 1.4 is shown in Table 2. Since OpenFlow 1.2, the number of match fields has kept increasing. Also, the specification keeps getting complicated as it get extended into different use-cases. This phenomena is referred to as the "match fields' explosion". Instead of changing the specification, some people argue that the future switch should support flexible mechanisms for parsing packet and comparing match fields[10].

In a production environment, having only one controller poses a threat of single point of failure. It is strongly recommended to build more than one controllers either for the purpose of load balancing or failover. Thus, OpenFlow 1.2 introduces the controller role-change mechanism [7], which enables multiple controllers to exist in the same network with three different roles: master, slave or equal; this allows failover from primary to secondary controller, and consequently intensify the availability of the network infrastructure.

Table 3 match fields and statistics in each version [12]

Version	Match Fields	Statistics
1.0	Ingress Port	Per table statistics
	Ethernet: src, dst, type, VLAN	Per flow statistics
	IPv4: src, dst, proto, ToS	Per port statistics
	TCP/UDP: src port, dst port	Per queue statistics
1.1	Metadata, SCTP, VLAN tagging	Group statistics
	MPLS: label, traffic class	Action bucket statistics
1.2	OpenFlow Extensible Match (OXM)	
	IPv6: src, dst, flow label, ICMPv6	
1.3	PBB, IPv6 Extension Headers	Per-flow meter
		Per-flow meter band
1.4	--	Optical port properties

OpenFlow 1.2 → OpenFlow 1.3

Quality of service is an essential feature in computer networking to provide better services for applications such as IP telephony and video streaming. To implement QoS in OpenFlow switches[13], OpenFlow 1.0 provides an optional "enqueue" action to forward packets through a queue attached to port. In addition, VLAN Priority and IP ToS are included in the header fields so that they can be matched and rewritten. OpenFlow 1.3 further introduces a new table called "Meter table" to extend QoS capability; the meter table consists of meter entries identified by meter identifier. Furthermore, each entry in meter table contains a list of "Meter bands", which specify the rate and behavior (drop or remark DSCP). When a packet matches a meter entry, the meter band with the highest configured rate that is lower than the current measured rate will be applied, thus it supports DiffServ [14] model.

Furthermore, OpenFlow 1.3 has extended the flow table with a table-miss entry [7]. In previous version of OpenFlow, a packet would either be dropped or sent to controller in a packet-in message; with table-miss entry, the processing behavior of non-matched packets would be more flexible than that of previous version.

OpenFlow 1.3 → OpenFlow 1.4

As mentioned before, the entry explosion problem has been resolved owning to the multi-table feature introduced in OpenFlow 1.1. OpenFlow 1.4 further extends the flow table scalability by introducing a new "Synchronized table". With synchronized table, flow tables can be synchronized bidirectionally or unidirectionally. If table synchronization is bidirectional, then the changes done by controller must be reflected on the source table. This can be effective when switches are doing multiple lookup upon the same lookup data. One common example is the MAC learning table and MAC forwarding table on an Ethernet learning switch; in this case they both lookup on the same set of MAC addresses [15].

Moreover, OpenFlow 1.4 introduces a new feature called "Bundle". Sometimes it is necessary to group related state modifications together into a transactional group, that is, all modifications in the group are applied or none of them are [15]; this is when bundle come into use. In other words, bundle is similar to converting multiple operations into an atomic operation. Moreover, bundle can be used to prepare and pre-validate OpenFlow messages when applying it to multiple switches. Controllers can create a bundle and add messages one by one into the bundle on multiple switches. If the Nth message fails, the switch would reply with an error message, then the controller can instruct the switches to discard the bundle including all the messages in it; this it much easier for the messages to rollback and largely boost the synchronization of

among switches.

OpenFlow 1.4 → OpenFlow 1.5

OpenFlow 1.5 further extends bundle with "Scheduled bundle" [7] by including an execution time property. A switch that received scheduled bundle will apply the messages as close to execution time as possible. This further strengthen the synchronization among multiple switches.

In the previous version of OpenFlow, the switch only matches and processes ingress packet. In order words, it is difficult to post process the packet without chaining another switch after the output port. This challenge is resolved in OpenFlow 1.5 with the new "Egress table" [7] by allowing matching packet based on its output port.

3 Maturity of Product

There are some major problems confronted by OpenFlow during the evolvement process. First of all, with the fast-changing specification, hardware vendors have found it hard to keep up with the pace of OpenFlow especially in the early age. Secondly, one of the major goals of OpenFlow is to create a networking ecosystem with vendor-independent switches, where switches from different vendors can cooperate under the same controllers. Yet there are too many optional features to decide whether to support or not. Last but not least, switches implemented by various vendors may behave slightly different, and some of them even diverse from the specification. As a result, we conducted a series of conformance test on two commonly used software switches: Open vSwitch [16] and Lagopus [17] to evaluate the maturity of product.

3.1 Conformance Test Tools

As shown in Table 4, the tools we choose to perform conformance test are Ryu Test [18] and OFTest [19]. Both of them support OpenFlow up to version 1.4. The test pattern in Ryu Test is formatted in JSON. It is considered to be slightly easier to customize test patterns in Ryu Test than in OFTest because the test patterns are written in python and may require programming skills.

Table 4 comparison of Ryu test and OFTest

	Ryu Test	**OFTest**
Support version	1.4	1.4
Test pattern format	JSON	Python
Usability	Easy	Medium (Require programming skills)
Flexibility	High	High
Testing Templates	144	187

We made a brief comparison to the test templates provided by two test tools over the test coverages. As shown in Figure 5, over half of the tests overlaps, and most of these test patterns are designed for testing the required features in the specification. Another interesting fact to notice is that the test templates in OFTest contain test for basic OpenFlow messages such as “Echo”, “Feature Request”, etc. On the other hand, Ryu Test seems to treats them as mandatory features and includes no test for them. Furthermore, OFTest has made a lot of effort on test for group table especially implementation details such as adding group with invalid id or modifying group with invalid id. In contrast, Ryu Test only focuses on forwarding behavior of “select” group type. Furthermore, Ryu Test emphasizes on tests for new features and supports for MPLS and SCTP. The comparison of the test templates for MPLS, SCTP and role request message is shown in Table 5.

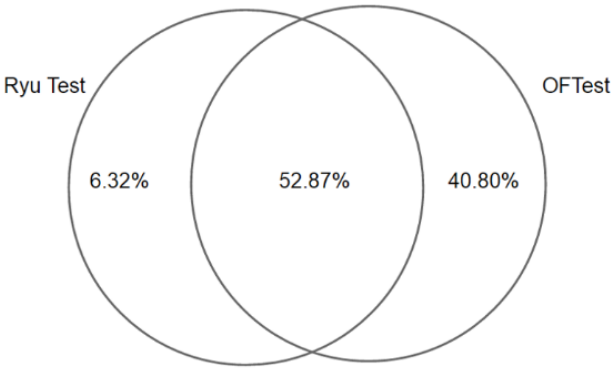

Figure 5 Ryu test and OFTest test case template coverage

Table 5 comparison of test template

	Ryu Test	**OFTest**
MPLS	YES	NO
SCTP	YES	NO
Role Request Message	NO	YES

Testbed Setup for Open vSwitch

Figure 6 shows the architecture for Open vSwitch with Ryu Test and OFTest

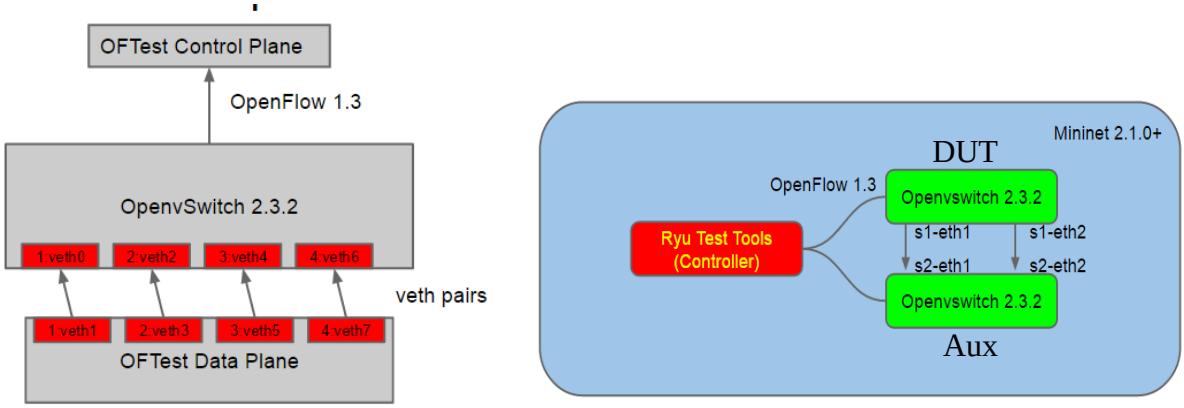

Figure 6 OFTest testbed architecture for Open vSwitch

In the architecture of OFTest[19], both the control plane and data plane are connected to OFTest. We create four virtual Ethernet pairs[20], and connect one end to Open vSwitch with the other end connects to data plane of OFTest. The data plane of OFTest serves the purpose of generating packets and receiving packets from Open vSwitch. It coordinates OpenFlow commands with dataplane stimulus and monitoring.

The architecture of Ryu Test and OFTest are similar; however, Ryu Test takes advantage of another OpenFlow switch called the "Auxiliary switch" (shown in Figure 6 as Aux), which is a packet generator and monitor. The controller in Ryu Test controls auxiliary switch to generate packet to DUT switch with a packet-out message and monitor the packet received on the corresponding port using packet-in message.

Testbed Setup for Lagopus

We have tried to replicate the testbed using Lagopus[21]; however we were confronted with some challenges; though we have succeeded testing some OpenFlow messages, we failed to receive packets from it, see appendix A for more information.

3.2 Conformance Test Result

Table 6 shows the test result of Open vSwitch with OFTest. We compare the result with the OpenFlow specification 1.3 and confirm that Open vSwitch implements all of the required feature in OpenFlow 1.3 correctly.

Table 6 Test Result of Open vSwitch with OFTest

	Total	Basic	Match	Groups	Actions	Flow_related	Pktin_match	Role_request
Ok	131	25	65	6	28	2	2	3
Fail	30	0	1	2	5	1	0	3

Table 7 shows the test result of Open vSwitch and Lagopus with Ryu Test. We can derive the same conclusion from the test result; Open vSwitch and Lagopus both passed the OpenFlow 1.3. On the other hand, we can see that Lagopus implements more optional features than Open vSwitch.

Table 7 Test Result of OVS and Lagopus with Ryu Test [22]

OVS			Lagopus		
	OK	ERROR		OK	ERROR
Total	670	321	Total	976	15
Required	114	0	Required	114	0
Optional	556	321	Optional	862	15

4 Conclusion

Though OpenFlow has been around for several years and has been adopted and deployed by vendors, there is still a lot of challenges waiting to be resolved:

- Hardware Implementation
- Interoperability
- Conformance

The multi-table challenge in OpenFlow 1.1 has yet found a perfect solution. It is hard to implement merchant silicon for hardware switches. In addition, with so many features around, the interoperability of various switch is a problem because different switches implement different features; not to mention vendors may add proprietary enhancements that are not compatible with each other. Furthermore, OpenFlow specifications change swiftly and some of the feature are not well-defined, thus, switches or controllers may behave differently.

The conformance tests we performed shows that Open vSwitch and Lagopus support all of the required feature in OpenFlow 1.3; however, the their supports for optional features differs. Lagopus support a lot more optional features than Open vSwitch. Also, in view of stability, Open vSwitch seems to be more robust than Lagopus.

5 References

[1]. Lin, Ying-Dar, et al. "Software-Defined Networking: Standardization for Cloud Computing's Second Wave." Computer 11 (2014): 19-21.

[2]. McKeown, Nick, et al. "OpenFlow: enabling innovation in campus networks." ACM SIGCOMM Computer Communication Review 38.2 (2008): 69-74.

[3]. “What are SDN Southbound APIs?”
https://www.sdxcentral.com/resources/sdn/southbound-interface-api/
[4]. “ONF Overview”
https://www.opennetworking.org/about/onf-overview
[5]. “Interoperable OpenFlow with NDM and TTP”
http://www.slideshare.net/US-Ignite/interoperable-open-flow-with-nd-ms-and-ttpsbeckmann
[6]. “Conformance Test Specification for OpenFlow Switch Specification 1.0.1”
https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-test/conformance-test-spec-openflow-1.0.1.pdf
[7]. “OpenFlow Switch Specification 1.5.0”
[8]. “Flow Table Explosion With OpenFlow 1.0”
http://blog.ipspace.net/2013/10/flow-table-explosion-with-openflow-10.html
[9]. “[SDN Protocols] Part 2 – OpenFlow Deep-Dive”
http://keepingitclassless.net/2014/07/sdn-protocols-2-openflow-deep-dive/
[10]. “OpenFlow Table Type Patterns”
https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/OpenFlow%20Table%20Type%20Patterns%20v1.0.pdf
[11]. Bosshart, Pat, et al. "P4: Programming protocol-independent packet processors." ACM SIGCOMM Computer Communication Review 44.3 (2014)
[12]. D. Kreutz, F.M.V. Ramos, P. Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S.Uhlig, “Software-Defined Networking: A Comprehensive Survey,” ArXiv e-prints, Jun. 2014.
[13]. Wallner, Ryan, and Robert Cannistra. "An SDN approach: quality of service using big switch’s floodlight open-source controller." Proceedings of the Asia-Pacific Advanced Network 35 (2013)
[14]. “DiffServ -- The Scalable End-to-End QoS Model”
http://www.cisco.com/en/US/technologies/tk543/tk766/technologies_white_paper09186a00800a3e2f.html
[15]. Tiantian Ren , Yanwei Xu “Analysis of the New Features of OpenFlow 1.4”, Atlantis Press
[16]. “Open vSwitch”
http://openvswitch.org/support/
[17]. “Lagopus”
https://lagopus.github.io/
[18]. “Ryu Book - OpenFlow Test Tools”
https://osrg.github.io/ryu-book/en/html/switch_test_tool.html

[19]. “OFTest -- Validating OpenFlow Swtiches”
https://floodlight.atlassian.net/wiki/display/OFTest
[20]. “Linux Switch – Interconnecting Namespaces”
http://www.opencloudblog.com/?p=66
[21]. “Lagopus running on small factor machine”
http://www.slideshare.net/lagopus/lagos-runningonsmallfactormachine
[22]. “Ryu Certification”
https://osrg.github.io/ryu/certification.html

Appendix A

Figure 7 shows the testbed architecture for Lagopus. We create two virtual machines with five virtual network interfaces; one of the VMs is for Lagopus switch and the other for OFTest. Four of the virtual network interfaces in Lagopus VM are set to DPDK compatible driver, and the promiscuous mode is enabled on all of them in VirtualBox. We connect the four interfaces accordingly as shown in Figure 7 using internal network feature in VirtualBox. The fifth interface is used to connect the OFTest controller to Lagopus switch in an out-of-band manner.

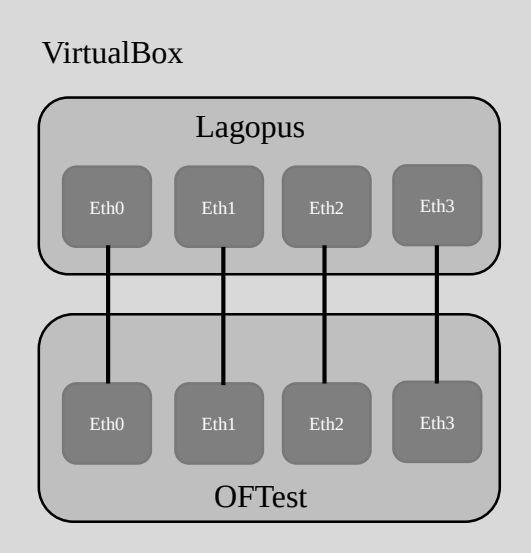

Figure 7 OFTest testbed architecture for Lagopus

When performing the test, we monitor the interfaces on OFTest VM with tcpdump; however we only see packets going out and receive no packet. We further exam the flow table Lagopus switch; it turns out that the flow entries are applied as expected. We suspect this is the due to use of DPDK driver but we haven't been able to confirm it.

Another interesting observation is that, whenever we perform test of basic OpenFlow messages, Lagopus terminated unexpectedly with "segmentation fault"; so we split up the test suit and perform them one by one. At last, we found that the error occurs when we perform test for Packet-Out message. This issue is reported to Lagopus by us. Unfortunately, we haven't heard from them by the end of the study.