International Journal of Computer Applications (0975 – 8887)
Volume 122 – No.5, July 2015

A Study of Software Defined Networking with OpenFlow

Rekha P M
JSS Academy of Technical Education
Department of Information Science & Engineering

Dakshayini M
BMS College of Engineering
Department of Information Science & Engineering

ABSTRACT

Today's data center complexity has been drastically increased with the widespread of many dynamic services. However, provisioning heterogeneous services to satisfy user's demand is a challenging task for the service providers and as well for the cloud Network administrators. Traditional network architectures were not designed to meet the requirements of today's enterprises and end users. Hence to meet the users demand and to address the difference between market requirements and network capabilities in data centers the industry has come up with the Software-Defined Networking (SDN) architecture and its related standards. With SDN, static network can advance into a wide range of service delivery platform capable of responding rapidly to changing business, end user's demand, and market needs. SDN provides a novel and innovative approach for controlling and managing virtual machines in data centers. In this paper, we discuss the concept of SDN, which can be implemented by the OpenFlow protocol. We discuss the OpenFlow architecture and its components with various OpenFlow versions. Finally we discuss OpenFlow based SDN implementation, testing and present an overview of SDN based applications.

Keywords

Software Defined Networking, Open Flow, Network management

1. INTRODUCTION

Data centers consist of layer 2 and layer 3 devices, namely switches and routers which carry traffic. With the growth of the network, and to meet the growing traffic demands of new applications many efforts have been taking place in configuring these switches and routers. As demand on the data center growing rapidly, so is the network growth. However, the network becomes very much difficult with the addition of hundreds to thousands of devices that must be configured and managed. The change in traffic patterns, rise of cloud services, and growing demand of bandwidth has lead network administrators to look for innovative solutions, since traditional networking technologies are not suited to the dynamic computing and storage needs of data centers. Applications are distributed across many virtual machines (VM), which exchange traffic flows with each other. Migration of VMs to optimize and rebalance workloads causes the physical end points of existing flows to change. Therefore an approach is necessary for maintaining growing network easily. Instead of configuring scattered thousands of devices, network operators and administrators can programmatically configure this simplified network abstraction. Hence, Software Defined Networking (SDN) a new approach in networking technology, was designed to create high level abstractions on top of which hardware and software infrastructure can be built to support new emerging applications. SDN allows network providers to expand their network and services with a common approach and tool set. OpenFlow protocol follows SDN approach which gives programmable control of flows to network administrators. It helps to identify a path that a flow takes from source to destination in spite of the network topology, and utilizes flow based processing for forwarding packets. The objective of this paper is to focus on the concept of SDN from the Open Networking Foundation (ONF).Several versions of OpenFlow specification exist with more powerful features. The rest of the paper is organized as follows. We discuss how SDN supports to network innovation by various networking applications that were implemented and analyzed in recent years. These applications are in the areas of network management and traffic engineering, cloud data center, security, network virtualization. Several surveys of SDN have been made this contribution differs from them through an analysis of architectural design choices for OpenFlow based SDN networks. Section 2 explains the concept of SDN, and Section 3 gives an overview of the OpenFlow architecture, which is currently the major protocol to control network elements in SDN. Section 4 we discuss SDN implementation and testing. Section 5 we discuss various SDN applications. Finally, Section 6 concludes this work.

2. SOFTWARE DEFINED NETWORKING

SDN is the network architecture that helps to ease network control, management, virtualization, multi tenancy and to allow improvement through network programmability.SDN consists of separate control plane and forwarding data plane. The separation of this forwarding hardware from the control logic allows easier operation of new protocols, applications, network visualization and management. Instead of enforcing policies and running protocols on spread devices, the network is reduced to simple forwarding hardware and the decision making network controller. The controller manages all switches, routers, and other network devices as a common logical architecture. This logical architecture may be of different behavior on different vendor equipment and in different types of network devices. SDN architectures support a set of APIs that make it possible to implement common network services, including routing, multicast, security, access control, bandwidth management, traffic engineering, quality of service, processor and storage optimization, energy usage, and all forms of policy management.

3. OPEN FLOW ARCHITECTURE

OpenFlow is the standard communication interface defined between the control layer and forwarding layer of SDN architecture. OpenFlow allows direct access to and manipulation of the forwarding plane of network devices both physical and virtual devices such as switches and routers. OpenFlow protocol is defined between SDN controllers and network devices and as well as it specifies the logical structure of the network switch functions [1, 2].

The OpenFlow Switch Specification is published by the Open Networking Foundation (ONF). ONF is a group of software providers, content delivery networks, and networking equipment vendors to support software defined networking. The OpenFlow version 1.0 was first developed at Stanford University and was widely implemented. OpenFlow version 1.2 was released from ONF after the project from Stanford. Extended functions were released in OpenFlow versions 1.3 and 1.4 as depicted in table 1.

The OpenFlow architecture consists of three components

- ➢ The data plane network is built up by Open Flow switches
- ➢ The control plane consists of OpenFlow controller
- ➢ A secure control channel connects the switches with the control plane

In the following, we discuss internal structure of Open Flow switches and their interactions with the controller. Figure 1 shows the basic structure of the OpenFlow environment. An SDN controller communicates with OpenFlow switches using the OpenFlow protocol running over the Secure Sockets Layer. Every switch connects to other OpenFlow switches and also to end-user devices.

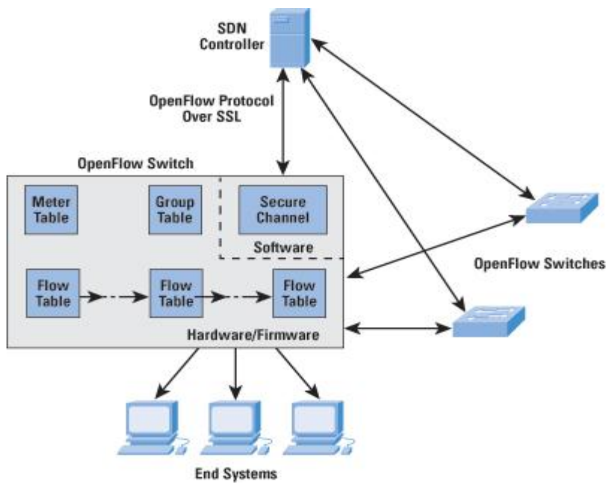

Fig 1: OpenFlow Switch Architecture

3.1 Open Flow Switch

An OpenFlow switch is a basic forwarding device that forwards packets according to its flow table. The OpenFlow switch consists of three types of tables. The flow table, group table and the meter table. In turn, each switch consists of a series of tables, implemented in hardware or firmware to manage the flows of packets through the switch. All incoming packets from a particular flow are matched with the flow table. The flow table describes the functions that are to be performed on the packets. There may be one or more flow tables. A Group table does many actions on one or more flows. Flow table directs a flow to the group table. Flow performance related actions are done on Meter Table.

Table 1. OpenFlow versions and its features

Version	Released in year	Features	Statistics Measured
1.0 [3]	December 2009	Provides QoS support , single flow table	Per table/flow/port/ queue statistics
1.1[4]	February 2011	Provides additional statistics fields due to the changed switch architecture, multiple flow tables	Per table/flow/port/ queue /Group /Action bucket statistics
1.2 [5]	December 2011	Extended protocol support for IPv6, multiple flow tables, it can be configured to simultaneous communication with multiple controllers	Per table/flow/port/ queue /Group /Action bucket statistics
1.3 [6]	June 2012	Multiple flow tables, Introduces new features for monitoring and operations and management	Per table/flow/port/ queue /Group /Action bucket statistics Per-flow meter /meter band
1.4 [7]	October 2013	Multiple flow tables, support for the Open Flow Extensible Match	Group /Action bucket statistics Per-flow meter / meter band

In the following section we discuss in detail open switch components.

3.1.1 Flow- Table components

A flow consists of a sequence of packets that matches a specific entry in a flow table. A combination of flow entries on multiple switches defines a flow that is bound to a specific path. Each flow table in the switch contains six entries as shown in table 2.

Table 2. Components of Flow Table

Match Field	Priority	Counter	Instructions	Time outs	Cookies

- ➢ Match Fields: Used to select packets that match the values in the fields.
- ➢ Priority: priority order of each table entries.

- Counters: Many counters are defined in the Open Flow specification namely, number of received bytes, packets per port, per flow table, per flow table entry, number of packets dropped, and period of a flow.
- Time outs: Defines the Maximum amount of idle time before a flow is discarded by the switch.
- Cookie: Data value chosen by the controller to filter flow statistics, flow modification, and flow deletion and is not used when processing packets.
- Instructions: Is associated with the specific flow entry and is executed if a match entry is found else if no match found in a flow table, the outcome depends on configuration of the table-miss flow entry.

3.1.2 Match Fields
The Match Fields component of a table consists of Ingress port, Ethernet source and destination addresses, IPv4 or IPv6 protocol number, IPv4 or IPv6 source Address and destination Address, TCP source and destination Ports, UDP source and destination Ports. Open Flow switch must support these match fields and the optional fields such as physical port, Ethernet Type, VLAN ID ,VLAN user Priority and Traffic Class.

3.1.3 Instructions
The instructions component of a table entry consists of a set of instructions with actions and action set that are executed if the packet matches the entry.

Instructions are of four types:

- Go-to-Table: The packet is directed to a table along the pipeline using the Go to-Table instruction.
- Meter instruction: The packet is directed to a specified meter using the meter instruction.
- The packets modification can be done between two tables and multiple actions can be executed on the packet of the same type when it is matched to a table entry.
- All the actions can be cleared in the action set or specified actions can be combined into the current action set for the packet on this flow.
- Packet consists of metadata value which is used to carry information from one table to the next.

3.1.4 Action Set
An action set is related with each packet which describes packet forwarding, packet modification, and group table processing operations. The action set can be modified using a Write-Action instruction or Clear-Action instruction attached with a particular match. The action set is accepted between flow tables. When the instruction set of a flow entry does not contain a Go-to-Table instruction, pipeline processing is stopped and the actions in the action set of the packet are executed.
The OpenFlow specification includes the following actions:

- Output: The packets are forwarded to the specified port.
- Set-Queue: Queue ID is put for each packet, the queue ID determines which queue attached to port is used for scheduling and forwarding the packet. It is used to provide basic QoS support.
- Group: specifies particular group packet that can be processed.
- Push-Tag/Pop-Tag: Field is applied for virtual network like VLAN or MPLS packet.
- Set-Field: Is used to modify the values of respective header fields in the packet.
- Change-TTL: To used to change the values of the IPv4 Time to Live (TTL), IPv6 Hop limit, or MPLS TTL in the packet.

3.1.5 Flow-Table Pipeline
A switch has one or more flow tables. If there is more than one flow table then the tables are labeled with numbers starting with 0.When a packet is sent to a table for matching, the input to the table consists of the packet, the identity of the ingress port, the associated metadata value, and the associated action set.

For table 0, the metadata value is blank and the action set is null. Processing is done as follows:

- Find the highest-priority matching flow entry. If no match on any entry and no table-miss entry, then the packet is dropped.
- If there is a match only on table-miss entry, then that entry tells one of the below three actions:
    - Send packet to controller to define a new flow for this and similar packets, or decide to drop the packet.
    - Direct packet to another flow table down the pipeline.
    - Drop the packet.
- If there is a match on one or more entries, then the match will be with the highest-priority matching entry. The following actions are performed:
    - Counter is updated associated with this entry.
    - Instructions associated with this entry are executed.
    - Lastly, the packet may be forwarded to either to flow table or to the group table, or to meter table, or it could be send to an output port.

There is no option for forwarding to another flow table from the last table in the pipeline. When a packet is finally directed to an output port, the action set is executed and then the packet is queued for output. Some of OpenFlow switches with design type and versions are summarized in table 3.

Table 3: OpenFlow Switches

Switch	Design Type	Series and Versions	OpenFlow version
Arista [8]	HW	7050 ,7150, 7500	1.0
Cisco	HW	3750	
Brocade [9]	HW	CES 2000, CER 2000, MLX	1.0, 1.3
HP [10]	HW	3500, 3500yl, 5400zl, 6200yl, 6600	1.0, 1.3
IBM [11,12]	HW	IBM 8264, RackSwitch G8264, G8264T	1.0
LINC [13]	SW	-	1.2,1.3,1.4
NEC [14]	HW	PF5240, PF5248	1.0 , 1.3.1
Open vSwitch [15]	SW	Latest version OVS 2.1.2	1.0 for OVS 1.9 , 1.2 and 1.3 for OVS 1.10 1.1 for OVS 2.0 1.4 for OVS 2.2
Pica8 [16]	HW	P-3290, P-3295, P3930, P-3297, P-3922	1.0, 1,1, 1.2, 1.3, 1.4

3.2 OpenFlow Controllers

Network devices in data center gets instructions and network rules from the controller. In turn, the controller does switch configuration, management, get events from the switch, and sends packets out to the switch. Using the OpenFlow protocol it manages packets, switch flow table entries by adding and removing flow entries over the secure channel. The Controller can be configured either in centralized configuration using single controller for managing and configuring all devices or in distributed configuration with a single controller for each set of switches. The controller operates either in reactive mode or proactive mode [17].

In reactive mode, the table does not have any rules. Hence when the packets appear at a switch, the switch informs the controller about the packet. Then the controller finds the path for the packet and set the suitable rules for all switches along the path, and lastly the packets of that flow are forwarded to their destination. In proactive mode, in prior controller installs the required flow entries in the switches. In this configuration, the controller will have more power on performance than a proactive flow configuration. The controllers are much faster than the switches with respect to performance. Due to timeouts like TLS echo request, session timeouts or other disconnections switch may lose connection with all controllers. The switch must leave at once depending upon the switch implementation and configuration. The packets and messages destined to the controllers are dropped in fail secure mode. According to fail secure mode flow entries have their expire timeouts. In fail standalone mode all packets are processed by the switch using the reserved port where it acts as a legacy router or Ethernet switch. At the start up, switch will operate in fail secure mode or in fail standalone mode, unless successfully connected to the controller. When the controller is connected existing flow entries remain same unless if required the controller has the option of deleting all flow entries. Table 4 provides a list of open source controllers with its main descriptions.

3.3 OpenFlow Channel

The messages are exchanged between OpenFlow switch and OpenFlow controller using the OpenFlow channel. On set up, the switch and controller communicates through a TLS connection which is situated by default on TCP port 6653. While using TCP, Security actions has to be considered to prevent from attacks, eavesdropping, and controller impersonation on the OpenFlow channel. Switches communicate with the controller using a user-specified transport port or the default transport port switches at a user configured IP address. If the switch is configured with the IP address of the controller to connect, the switch initiates a standard TLS or TCP connection to the controller. Authentication between the switch and controller is done by exchanging the certificates signed by a private key. Each switch must be configured by user with one certificate for authenticating the controller. A single controller or several controllers can be connected to the switch. If one controller connection fails then it can be replaced by other controllers. For more reliability multiple controllers are connected. The controller manages the switch, enables load balancing and fast recovery from failure.

3.4 OpenFlow Protocol

The OpenFlow protocol is a key enabler for SDN which directs handling of the forwarding plane of network devices. The messages are exchanged between an OpenFlow controller and an OpenFlow switch using OpenFlow protocol. The controller can delete, add and update actions to the flow entries in the flow table with the OpenFlow protocol. The three classes of communication namely, controller-to-switch, asynchronous, and symmetric is supported by OpenFlow protocol.

- ➢ Asynchronous: This class includes various status messages to the controller. The Packet-in message is used by the switch to send a packet to the controller when there is no flow-table match.
- ➢ Controller-to-Switch: The controller sends messages to manage the logical state of the switch, configuration details of flow and group table entries. Packet out message is send when a switch forwards packet to the controller and then the controller directs the packet to a switch output port without dropping it.
- ➢ Symmetric: These messages are sent without the help from either the controller or the switch. Hello messages are sent back and forth when the connection is set up between the controllers and switch. The OpenFlow protocol messages are summarized in table 5, 6 and 7.

Table 4: Controllers in SDN

Controller	Language	Created by	Open source	Open Flow version	Description
NOX	Python, C++	Nicira	Yes	1.0, 1.3	Asynchronous, event-based programming model, component based framework. NOX-MT is multithreaded with improving throughput and response time [18].
POX	Python 2.7	Nicira	Yes	1.0	Component based framework, targets Linux, Mac OS, and Window [19].
Beacon	Java	Stanford university	Yes	1.0.1	Event based and threaded Cross-platform, Dynamic, and Rapid Development [20].
Maestro	Java	Rice university	Yes	1.0	Modular network control applications, multi-thread [21].
Floodlight	Java	Big Switch Networks	Yes	1.0	Based on Beacon, core architecture is modular, open source agent [22].
Floodlight-plus	Java	Big Switch Networks	Yes	1.3	New version of floodlight for supporting OF 1.3 [23].
Ryu	Python	NTT Labs	Yes	1.0, 1.2, 1.3, 1.4	Component based, supporting components development in other languages, event management and reusable NETCONF library, sFlow /Net flow library [24].
Open Daylight	Java	Linux Foundation	Yes	1.0 , 1.3	Modular, pluggable, and flexible controller platform, supporting multiple southbound protocols [25].

Table 5: Asynchronous Messages

Packet-in	Transfer packet to controller, used to configure buffer packets
Packet-out	Message send from the controller
Flow-Removed	Report to the controller about the removal of a flow entry from a flow table
Port-Status	Report to the controller from the switch such as port-status, port configuration or port state changes

Table 6: Controller to Switch Messages

Message	Description
Features	Request the identity and capabilities of a switch. Switch responds by specifying its capabilities
Configuration	Switch responds with parameter settings Set and query configuration parameters
Modify-State	To set switch port properties like Add, delete, and modify flow/group entries
Read-State	To collect status such as current configuration, statistics, and capabilities of the switch
Packet-out	Packet is send to a specified port on the switch and with a list of actions to be applied in the order they are specified, an empty action list drops the packet
Barrier	Barrier request/reply messages are used by the controller to ensure message dependencies have been met or to receive notifications for completed operations
Role-Request	These messages are used by the controller to set the role of its Open Flow channel or query that role. Useful when the switch connects to several controllers
Asynchronous-Configuration	These messages are used by the controller to set filter on the asynchronous messages that it wishes to receive on its Open Flow channel, or to query that filter. This is mostly useful when the switch connects to several controllers

Table 7: Symmetric Messages

Hello	Upon connection set up ,hello message is exchanged between the switch and controller
Error	Switch informs to the controller to notify to notify the connection problems and also indicates failure of a request initiated by the controller
Echo	Echo request/reply messages can be sent from either the switch or the controller verify the liveness of a controller-switch connection as well used to measure its latency or bandwidth
Experimenter	Meant for future Open Flow revisions

4. IMPLEMENTATION AND TESTING

Researchers can test new services, applications and protocols on an emulation of the expected deployment environment before moving to the actual hardware. Mininet is a software tool supports entire OpenFlow v1.0 network can be emulated on a single computer .It uses Linux kernels along with Python language scripts to construct a virtual network of large number of host network, Open Flow switches, and controllers in any network topology over a single desktop or laptop station.EstiNet is simulation as a service which uses the company servers to run the simulation or the emulation projects [27]. Ns-3 supports OpenFlow protocol and its switches in simulator environment but the drawback is it cannot readily run a real Open Flow controller such as NOX, POX, or Floodlight. Trema is an OpenFlow framework for integrated testing and debugging environment. It manages, monitor, and diagnose the entire system with a network emulator and a diagnostic tool like Trema shark, Wireshark plug-in [28]. For testing novel applications using OpenFlow test-beds, experimental environments were developed. To support experimental research in networking Global Environment for Network Innovations (GENI) supported by the National Science Foundation were developed. Researchers can dynamically control and extend the network through OpenFlow OFELIA test-bed.

5. SDN APPLICATIONS

5.1 Network Management

Network management in software defined network can be done from logical flow tables which are in the centralized logical controllers and these flow tables can be used in distributed switches. Due to high-level policies in a distributed low-level configuration using CLI mode network state is difficult to achieve. Authors solved problems in an event-driven control framework which is on functional reactive programming and OpenFlow 1.0.0 [29].Another control system was proposed that does integrated network management, control system discovery and fault detection [30]. Security management is another aspect of network management. The algorithm of information security management system combined with fuzzy logic and a prototype of intrusion detection system was designed [31]. For event-based network control approach, Lithium system was developed [32]. For minimal visibility of performance and control, a programmable measurement platform approach was Bismarck was introduced [33].

5.2 Network Virtualization

Network virtualization is one of the important key research areas of network that allow multiple users to share resource and infrastructure. A system for managing virtual networks and sharing infrastructure based on layer 2 and OF were introduced [35]. For managing and monitoring slices infrastructure operations an approach for network isolation based on Flow Visor and Slice Control and Management Layer was presented [36]. To manage and allocate resource optimally a test-bed for isolated virtual network with different QoS to help administrator was presented [37]. Xen hypervisor is used for virtualization with the high performance and to manage resources [38].

5.3 Cloud Data Center

SDN gives better solution for cloud Services and data center. In cloud, users gain the enough resources based on requirement in real time [39]. The cloud data center was implemented using SDN based implementation which is faster and easier to configure [40]. NetGraph, software architecture to manage SDN based cloud systems were introduced which provides a module with a set of API for monitoring and diagnostics cloud system [41]. Hedera defines dynamic Flow Scheduling for Data Center Networks with a central scheduler to balance load based on dynamic flow condition on whole system [42]. Cross stratum architecture was proposed for data center interconnection designed for Flexi-Grid optical networks. Authors argued that the system has improved end to end responsiveness and optimized resource usage [43].

5.4. Security

A system was introduced in which OpenFlow switch on edge is used to pass only safe traffic defined on flow table to recognize trusty user [44].OpenFlow Random Host Mutation system was introduced in which real IP of each host is changed with the random virtual IP [45]. Multilevel security system was developed that prevents information of specific level which gathered by same or lower level host to monitor packet and check content so filter the packet with security problem [46]. The authors suggested architecture with IDS component responsible for detecting attacks and recognizing unsecure devises and inform OpenFlow controller [47].
One problem of SDN approach is that when network traffic is high gathering full data from flow table is not resourceful so author suggested solution to optimal real time detection system [48].

6. CONCLUSION

In this study, we discussed the concept of software defined networks and the open Flow architecture with three components namely, OpenFlow switch, Open Flow controller and OpenFlow protocols. We discussed switches and their capabilities, communication between switch and controller. We presented different SDN implementations and testing platforms. At last we briefed SDN applications in different areas to improve network management, virtualization, cloud data centers and the works done on security of SDN. Finally, we conclude that SDN promises to change static networks into flexible, programmable platforms with the intelligence to allocate resources dynamically in data centers and it supports dynamic, highly automated, and secure cloud environments.

7. REFERENCES

[1] Thomas A. Limoncelli. OpenFlow: a radical new idea in networking.Communication, ACM, 55(8):42–47, August 2012.

[2] Open networking foundation. https://www.opennetworking.org

[3] OpenFlow Switch Consortium and Others, “OpenFlow Switch Specification Version 1.0.0”, 2009.

[4] OpenFlow Switch Consortium and Others, “OpenFlow Switch Specification Version 1.1.0”, 2011.

[5] Open Flow Switch Consortium and Others, “Open Flow Switch Specification Version 1.2.0”, 2011. Available online: https://www.opennetworking.org.

[6] OpenFlow Switch Consortium and Others, “Open Flow Switch Specification Version 1.3.0”, 2012. Available online: https://www.opennetworking.org

[7] OpenFlow Switch Consortium and Others, “Open Flow Switch Specification Version 1.4.0”, 2013. Available online: https://www.opennetworking.org

[8] Arista Networks, available online: http://www.aristanetworks.com/en/products/eos/openflow, last visit: 18.10.2014.

[9] Brocade, available online: http://www.brocade.com/products/all/switches/product-details/netiron-ces-2000-series/index.page, last visit: 18.10.2014.

[10] HP switches, available online: http://h17007.www1.hp.com/us/en/networking/products/switches/HP_3500_and_3500_yl_Switch_Series/index.aspx, last visit: 18.10.2014.

[11] IBM, available online: http://www-03.ibm.com/systems/networking/software/sdnve last visit: 8.8.2014.

[12] Juniper, http://www.juniper.net/us/en/products-services/sdn , last visit:8.8.2014

[13] Y. Cheng, V. Ganti and V. Lubsey, “Open Data Center Alliance Usage Model: Software-Defined Networking rev. 2.0”, Open Data Center Alliance, 2014, [Online].

[14] NEC, available online: http://www.necam.com/sdn/doc.cfm?t=PFlowPF5240Switch, last visit: 18.10.2014

[15] OpenvSwitch, available online: http://openvswitch.org, last visit: 18.10.2014

[16] Pica8 open networking, http://www.pica8.org/open-switching/1gbe-10gbe-40gbe-open-switches.php, last visit: 18.10.2014

[17] Evaluation of OpenFlow Controllers Guillermo Romero de Tejada Muntaner, October 15, 2012.

[18] NOX, available online: http://www.noxrepo.org/nox/about-nox/, last visit: 18.10.2014.

[19] POX, available online: http://www.noxrepo.org/pox/about-pox/, last visit: 18.10.2014.

[20] Beacon, available online: https://openflow.stanford.edu/display/Beacon/Home, last visit: 18.10.2014.

[21] E. Ng, “Maestro: A System for Scalable OpenFlow Control”, available online: www.cs.rice.edu/~eugeneng/papers/TR10-11.pdf, last visit: 18.10.2014.

[22] Floodlight OpenFlow Controller - Project Floodlight, available online: http://www.projectfloodlight.org/floodlight/, last visit: 18.10.2014.

[23] Announcing release of Floodlight with OF 1.3 support, available online: http://sdnhub.org/releases/floodlight-plus-openflow13-support/, last visit: 18.10.2014.

[24] Ryu 3.9 documentation, available online: http://ryu.readthedocs.org/en/latest/getting_started.html#what-s-ryu, last visit: 18.10.2014.

[25] Opendaylight, available online: http://www.opendaylight.org/, last visit: 18.10.2014.

[26] S. Azodolmolky, Software Defined Networking with OpenFlow, Packt Publishing, 1st ed., October 2013.

[27] EstiNet Technologies Inc., “The GUI user Manual for the EstiNet 8.0 Network Simulator and Emulator”, January 2013.

[28] T. Dietz, “Trema tutorial”, NEC Corporation, March 2012.Available at: http://www.fp7-ofelia.eu/assets/Uploads/201203xx-Trema Tutorial.pdf, accessed on 23/3/2014.

[29] H. Kim and N. Feamster, “Improving network management with software defined networking”, Communications Magazine, IEEE, Vol.51, No.2, 2013, pp.114-119.

[30] P. Sharma, S. Banerjee, S. Tandel, R. Aguiar, R. Amorim and D. Pinheiro, “Enhancing network management frameworks with SDN-like control”, 2013 IFIP/IEEE International Symposium on Integrated Network Management ,2013, pp.688-691.

[31] S. Dotcenko, A. Vladyko, and I. Letenko, “A fuzzy logic-based information security management for software-defined networks”, 16th International Conference on Advanced Communication Technology (ICACT), 2014, pp. 167-171.

[32] H. Kim, A. Voellmy, S. Burnett, N. Feamster, and R. Clark, “Lithium: Event-driven network control”, Georgia Institute of Technology, 2012.

[33] Bismarck, Available online: http://projectbismark.net, last visit: 18.10.2014.

[34] S. Song, S. Hong, X. Guan, B.-Y. Choi, and C. Choi, "NEOD: network embedded on-line disaster management framework for software defined networking," Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on, (2013), pp.492-498.

[35] J. Matias, B. Tornero, A. Mendiola, E. Jacob, and N. Toledo, “Implementing Layer 2 Network Virtualization using OpenFlow: Challenges and Solutions”, Proceedings of European Workshop on Software Defined Networking (EWSDN), 2012, pp: 30-35.

[36] R. Nejbati, S. Azodolmolky and D. Simeonidou, “Role of Network Virtualization in Future Internet Innovation”, Proceedings of 17th European Conference on Networks and Optical Communications (NOC), 2012, pp: 1-4.

[37] I. M. Moraes, D. M. Mattos, L. H. G. Ferraz, M. E. M. Campista, M. G. Rubinstein, L. H. M. Costa, *et al.*, “FITS: A Flexible Virtual Network Test bed Architecture”, Computer Networks, Vol.63, 2014, pp: 221–237.

[38] P. Barham, B. Dragovic, K. Fraser, S. H and T. Harris, A. Ho, “Xen and the Art of Virtualization”, Proceedings of the nineteenth ACM symposium on Operating systems principles, Vol.37, 2003, pp:164-177.

[39] J. Hurwitz, A. Nugent, F. Halper and M. Kaufman, Big Data for Dummies, John Wiley & Sons, Inc., 201, pp: 7-35.

[40] C. Baker, A. Anjum, R. Hill, N. Bessis and S. L. Kiani, “Improving Cloud Datacenter Scalability, Agility and Performance using OpenFlow”, Proceedings of 4th IEEE International Conference on Intelligent Networking and Collaborative Systems (INCoS), 2012.

[41] R. Raghavendra, J. Lobo and K.-W.Lee, “Dynamic Graph Query Primitives for SDN-based Cloud Network Management”, Proceedings of the first workshop on Hot topics in software defined networks, 2012, pp.97-102.

[42] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang and A. Vahdat, “Hedera: Dynamic Flow Scheduling for Data Center Networks”, Proceedings of 7th USENIX conference on Networked systems design and implementation, 2010.

[43] H. Yang, J. Zhang, Y. Zhao, H. Li, S. Huang, Y. Ji, *et al.*, “Cross Stratum Resilience for OpenFlow enabled Data Center Interconnection with Flexi-Grid Optical Networks”, Optical Switching and Networking, Vol.11, 2014, pp:72-82.

[44] C. YuHunag, T. MinChi, C. YaoTing, C. YuChieh, and C. YanRen, “A Novel Design for Future on-demand Service and Security”, Proceedings of 12th IEEE International Conference on Communication Technology, 2010, pp:385-388.

[45] J. H. Jafarian, E. Al-Shaer and Q. Duan, “Openflow Random Host Mutation: Transparent Moving Target Defense using Software Defined Networking”, Proceedings of the first workshop on Hot topics in software defined networks, 2012, pp: 127-132.

[46] X. Liu, H. Xue, X. Feng and Y. Dai, “Design of the Multi-level Security Network Switch System Which Restricts Covert Channel”, Proceedings of 3rd IEEE International Conference on communication software and networks (ICCSN), 2011, pp:233-237.

[47] Y. Juba, H.-H. Huang and K. Kawagoe, “Dynamic Isolation of Network Devices Using OpenFlow for Keeping LAN Secure from Intra-LAN Attack”, Procedia computer science, Vol. 22, 2013, pp: 810-819.

[48] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining OpenFlow and sFlow for an Effective and Scalable Anomaly Detection and Mitigation Mechanism on SDN Environments", Computer Networks, (2013), available online: http://dx.doi.org/10.1016/j.bjp.2013.10.014.