Advances in Computer Science and Information Technology (ACSIT)
Print ISSN: 2393-9907; Online ISSN: 2393-9915; Volume 2, Number 2; January-March, 2015 pp. 107-111

http://www.krishisanskriti.org/acsit.html

OpenFlow Software Defined Networking Controllers: A Comparative Study

Avantika Konde[1] and Anita Ganpati[2]

[1]Research Scholar, Department of Computer Sciences Himachal Pradesh University
[2]Department of Computer Sciences Himachal Pradesh University
E-mail: [1]avantika.er91@gmail.com, [2]anitaganpati@gmail.com

Abstract—*The concept of programmable networks has recently gained impetus due to the emergence of Software Defined Networking (SDN) paradigm. SDN architecture separates the network control and forwarding functions allowing the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and the network services. OpenFLow protocol is foundational element for building SDN solution. OpenFlow controller is an application that manages flow control in SDN. This paper presents an extensive study of SDN's open flow controller based on the literature survey and comparison among them. Analysis of these controllers helps to find the most optimal one for future work.*

1. INTRODUCTION

Modern data centers require a network with high cross-section bandwidth, latency, fine-grained security, support for virtualization and simple management that can scale to hundreds of thousands of ports at low cost. Although ethernet is most commonly deployed layer2 (L2) datacenter network, traditional switched ethernet cannot specify these requirements at a large scale. SDN is one promising class of network architecture that is suitable substitutes for traditional switched ethernet. The emergence of SDN [1] has sparked significant interest in rethinking classical approaches to network architecture and design. The SDN [3] [4] is a concept that is to break with the traditional networks where the switch decides the actions to do. The SDN concept was introduced by Nick McKeown [5], a professor at Stanford University and is based on defining a model where all switches move the capacity of decision to a central element, to a controller. The SDN concept is closely related to Network as a Service (NaaS).

SDN makes it possible to control an entire network in software, by writing programs that control network behavior to suit specific applications and environments. SDN gives network designers freedom to refactor the network control plane [2]. All SDN architectures as shown in Fig. 1 have three layers: the infrastructure layer, the control layer and the application layer. The three layers of SDN are connected by two interfaces. The SDN protocol is the switch firmware, and there is a proprietary interface between the hardware and software inside the switch.

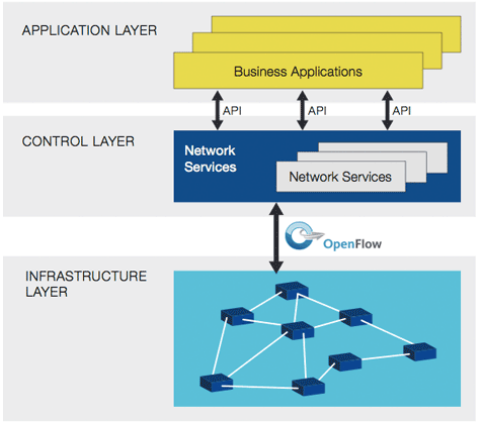

Fig. 1: Three layer Architecture of SDN with OpenFlow.

An OpenFlow controller is an application that manages flow control in an SDN environment. Most current SDN controllers are based on open flow protocol. However, to be able to realize the SDN concept, one must choose a suitable controller. This decision problem can be troublesome as it is difficult to define the right metrics, and the number of controllers keeps increasing. To solve this issue, researcher surveyed literatures, websites, talks, blogs, and any available resources providing information about the existing SDN controllers.

Among them, three controllers have been selected for the survey to gather their properties. These controllers are: NOX [6], Beacon [7] and Floodlight [8].

2. LITERATURE REVIEW

Thomas D. Nadeau et al. [9] described SDN as "an architectural approach that optimizes and simplifies network operations by more closely binding the interaction among applications and network services and devices, whether they

are real or virtualized". Mendonca et al. [10] stated that SDN has been proposed as a way to programmatically control networks, making it easier to deploy new applications and services, as well as tune network policy and performance. The key idea behind SDN is to decouple the data from the control plane by: (1) removing control decisions from the forwarding hardware, (2) allowing the forwarding hardware to be "programmable" via an open interface, and (3) having a separate entity called "controller" defined by software the behavior of the network formed by the forwarding infrastructure, thereby creating a "software defined network".

In SDN, the controller is the entity that dictates the network behavior, on this Natasha Gude et al. [6] stated that the logical centralization of the control logic in a software module that runs in a standard server the network operating system offers several benefits. OpenFlow technology moves the control logic to an external controller (typically an external PC) and this controller is responsible for deciding the actions that the switch must perform. This communication between the controller and the data path is made, on the network itself, using the protocol that provides OpenFlow (OpenFlow Protocol).

Chris Tracy [11] mentioned that "OpenFlow is a new technology based on the concept SDN." OpenFlow has been used to implement a wide variety of network tools and protocols, including routing circuit-switch and packet switched traffic over the same switch [12], wave-length path control in optical networks [13], in-network load balancer [14], wireless sensor networks [15], and wireless mesh networks [16].

Marcelo D. D. Moreira et al. [17] concluded that "OpenFlow network virtualization model follows the shared data plane approach by defining a centralized element that controls and programs the forwarding table in each network element".

NOX is a multi-threaded C++-based controller written on top of Boost library. Hardeep et al. [18] defined NOX as "an external controller that is responsible for adding or removing new routing rules into the OpenFlow switch's flow table". The NOX controller decides how packets of a new flow should be handled by the switch. When new flows arrive at the switch, the packet gets redirected to the NOX controller which then decides whether the switch should drop the packet or forward it to a machine connected to the switch. The NOX controller can also delete or modify existing flow entries in the switch.

Beacon [19] is a multi-threaded Java-based controller that relies on OSGi and spring frameworks. Beacon explores new areas of the OpenFlow controller design space, with a focus on being developer friendly, high performance, and having the ability to start and stop existing and new applications at runtime.

Floodlight [20] is a multi-threaded Java-based controller that uses Netty framework. Floodlight is designed to work with the growing number of switches, routers, virtual switches, and access points that support the OpenFlow standard.

3. OPENFLOW CONTROLLERS

In a data center or cloud where virtual machines move swiftly from server to server, networks must respond rapidly to traffic changes. But traditional switch and router path determination algorithms react slowly. SDN aims to reduce network reaction time to traffic changes by moving path allocation from individual devices to centralized controller software that lives on a workstation or server. The controller component communicates with each device in the network, receiving updates on load and link status and then managing the traffic flows among the devices. When a data source begins communication with a destination across the network, the controller determines an optimal path through the network based on existing load and network status. The controller then creates a flow defined by source and destination addresses and communicates with each device along the path, informing them of the new flow and how to handle packets in the flow.

3.1 NOX

NOX is the original openflow controller. It serves as a network control platform that provides a high-level programmatic interface for management and development of network control applications. Its system-wide abstractions turn networking into software platform.

The NOX core as shown in the Fig. 2 provides helper methods such as network packet process, threading and event engine in addition to OpenFlow API's for interacting with OpenFlow switches and input-output support. With the current NOX there are two core applications: OpenFlow and switch and both network and web applications are missing. Dynamic shared object deployer (DSO) scans the directory structure for any components being implemented as DSO's.

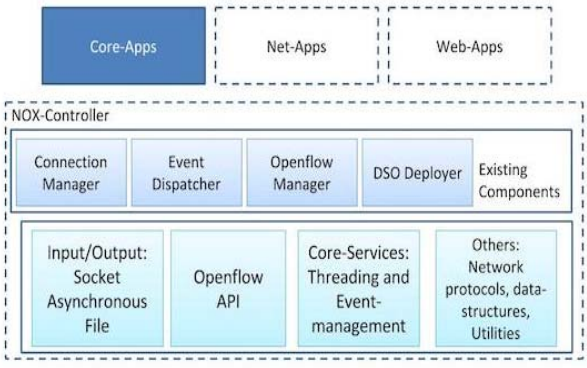

Fig. 2: NOX Architecture [6]

The NOX has an event dispatcher which works on the event system as explained below.

Event system is another important concept of the NOX controller. An event represents a low level or high level event in the network. The event only provides the information, and processing of that information is deferred to handlers. Many events roughly correlate to something which happens on the network that may be of interest to a NOX component. These components consist of a set of event handlers. Events drive all execution in NOX.

The drawback of NOX is that it is single threaded and is neither actively developed nor has an active community.

3.2 Beacon

Beacon is a JAVA based open source OpenFlow controller created in 2010. It is a fast, cross-platform, modular controller that supports both event-based and multithreaded operations.

The key features of Beacon are it has been use in many research projects, networking classes and trial deployment. It powers a 100-vswitch experimental data center and has run for months without downtime. It runs on many platforms, from high end multi-core Linux servers to android phones. Beacon is licensed under a combination of the GPL v2.licenseand the Stanford University FOSS License Exception. Code bundles in Beacon can be started/stopped/refreshed/installed at runtime. It is fast due to multithreaded operation.

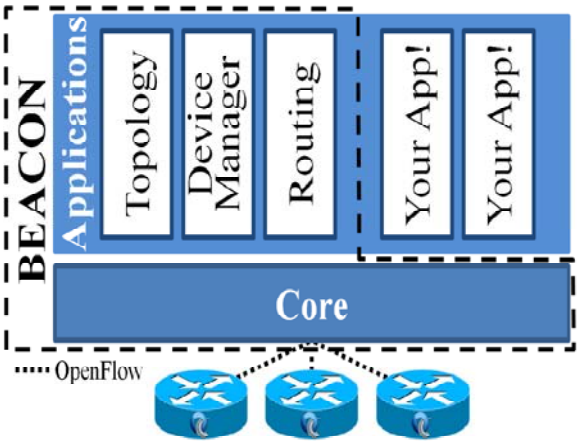

Fig. 3: Architecture of Beacon [7]

The core applications of the Beacon as shown in Fig. 3 includes the decisions to be made on the topology of the network, a device manager which manages the devices in the network including the slices for different networks and the routing of the traffic in the network by analyzing the source and the destination addresses. Apart from these applications which are at the core of the Beacon architecture, it also provides us with the facility to create and run our own applications.

Beacon [17] explores new areas of the OpenFlow controller design space, with a focus on being developer friendly, high performance, and having the ability to start and stop existing and new applications at runtime.

3.3 Floodlight

Floodlight is an OpenFlow controller built on work that has begun at Stanford University and UC at Berkeley and now continues among a community of open source developers along with engineers at SDN and network virtualization startup Big Switch Networks INC. The overview of Floodlight is shown in Fig. 4. It works with physical and virtual switches that speak the OpenFlow Protocol. Apache-licensing lets floodlight to be used for almost any purpose. It is a core of commercial product from Big Switch Networks and is actively tested and improved by a community of professional developers.

While the controller is a key component in SDN, it provides only the means to manage or direct the network that lies beneath.

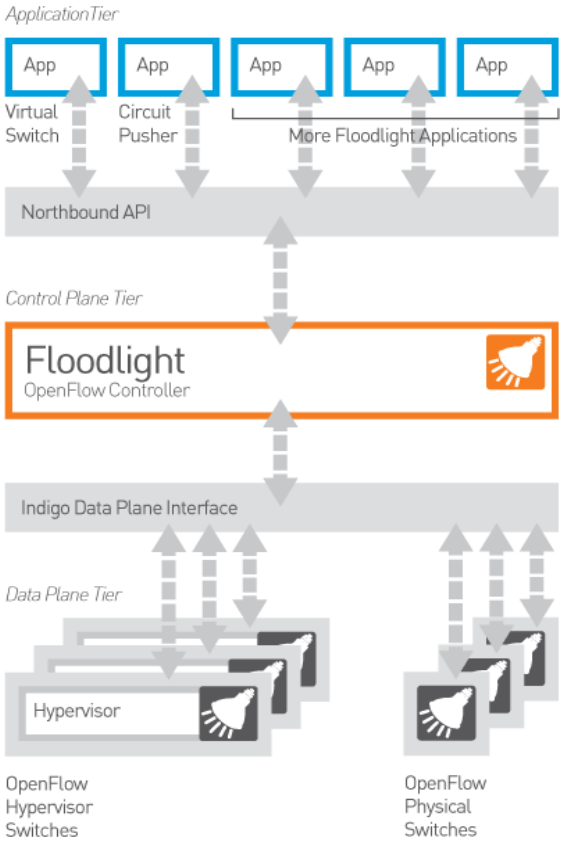

Fig. 4: Architecture of Floodlight [20]

As shown in the Fig. 4 in the control plane of the SDN Floodlight acts as a controller for various applications described below. With Floodlight handling applications it becomes very convenient for the network designers because many of the issues such as routing, controlling flow and security issues are handled by the Floodlight itself making it very viable and multipurpose controller.

Floodlight Applications [20]:

1. Virtual Networking Filter- It identifies the packets that enter the network but do not match an existing flow.
2. Circuit Pusher- It creates a flow and provisions switches along the path to the packet's destination.
3. Static Flow Pusher- It is used to create a flow in advance of the initial packet in the flow entering the network.
4. Firewall modules- It gives the same protection to devices on the SDN as traditional firewalls on a physical network.

Floodlight has unquestionably the most active and responsive community among the F/OSS OpenFlow software. A majority of the floodlight developers working in big switch networks directly participate in the mailing-lists. It was truly a supportive and active community. Floodlight exposes almost all of its functionality through a REST API. One of its kinds, floodlight can also be run as network backend for OpenStack using a Quantum plug-in. Finally, it is the most documented controller project in the ecosystem.

4. OBJECTIVE OF THE STUDY

The objective of the study was to extensively review the openflow controllers and find out the most viable one. Before solving the SDN challenges it is very important to have an in depth knowledge of its components. After studying from various sources of information available an analysis of the three controllers is presented.

5. ANALYSIS

After studying all the three controllers in detail with all the information available whether in the form of research paper or from the internet all the features which are necessary to be in the OpenFlow controller while deployment of SDN are tabulated in the table below. Following is the feature matrix in Table 1

Table 1: Feature matrix for the three controllers.

	NOX	*Beacon*	*Floodlight*
Is actively developed?	×	✓	✓
Has an active community?	×	✓	✓
Easy to install?	×	✓	✓
Easy to program?	×	✓	✓
Documented?	×	✓	✓
Provides REST API?	×	✓	✓
Have utility functions?	×	✓	✓
Has a UI?	Python	Web	Web
Supports hosts with multiple attachment points?	×	×	✓
Topologies with loops?	×	×	✓
Supports OpenStack Quantum?	×	×	✓
Virtual Networking Filter?	×	×	✓
Circuit Pusher?	×	×	✓
Firewall modules?	×	×	✓

When working on an upcoming technology like SDN it is important that the environment which contains many components such as the controllers is actively developed and maintained according to the present scenario. Floodlight has been actively worked upon and is also actively developed. With the support from the Floodlight community it becomes easy for the researchers to provide some solution to the challenges which would be faced in the coming time when SDN would be deployed on a large scale. One such challenge is scalability which involves the controllers to be handling the flow and all the issues related to the control plane. NOX and Beacon are neither actively developed nor they have been documented. With the increase in the data centers the network may be much more complex than the one we are facing now. So it becomes tremendously important that the controller handles even the utmost complicated network with same performance and fault tolerance. NOX and Beacon don't support topologies with loops but Floodlight does. Considering all the parameters and the architectures of the three controllers it was found that Floodlight turned out to be the best in terms of control performance and with respect to future work which can be done to enhance Floodlight.

6. CONCLUSION AND FUTURE WORK

Clearly it can be seen from the Table 1 that Floodlight is found to be the most viable controller in the ecosystem as of now. Since it takes a huge amount of time in learning a controller, keeping in mind the analysis of the controllers presented above the development of SDN environment using Floodlight can be begun after it being realized as an optimal openflow controller. The future work includes dealing with the scalability challenge in SDN using Floodlight as a controller for various network slices to be created to scale SDN for huge networks. Supporting a large number of tenants with different abstractions raises scalability challenges. For example, supporting virtual topologies requires a way for tenants to run their own control logic and learn about relevant topology changes. Software Defined Networking (SDN) is an appealing platform for network virtualization, since each tenant's control logic can run on a controller rather than the physical switches. In SDN, a logically centralized controller manages the collection of switches through a standard interface, enabling the software to control switches from a variety of vendors.

REFERENCES

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. *OpenFlow: enabling innovation in campus networks*. ACM Sigcomm CCR, 38(2), 2008.

[2] Dan Levin, Andreas Wundsam, Brandon Heller Nikhil Handigol and Anja Feldmann .*Logically Centralized? State Distribution Trade-offs in Software Defined Networks.* ACM, 2012.

[3] *How the emergence of OpenFlow and Software-Defined Networking (SDN) will change the networking landscape.* Brocade. 2012

[4] *Software-defined Networking: The New Norm for Networks.* Open Networking Foundation. April 13, 2012

[5] Nick McKeown Available at: http://yuba.stanford.edu/[Online] accessed on December 20, 2014

[6] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. *NOX: towards an operating system for networks.* SIGCOMM CCR, 38(3), 2008.

[7] David Erickson in *The Beacon OpenFlow Controller.* ACM, 2013.

[8] FloodlightAvailable:*http://www.projectfloodlight.org/floodlight/.* [Online], accessed on 20-Dec-2014

[9] Thomas D. Nadeau, Ken Gray, *SDN: Software Defined Networks,* O'Reily Publications, 2013.

[10] Marc Mendonca, Bruno Astuto A. Nunesy, Katia Obraczka and Thierry Turlettiy in *Software Defined Networking for Heterogeneous Networks* University of California, Santa Cruz, USA. IEEE. 2012

[11] Chris Tracy, Introduction to OpenFlow: *Bringing Experimental Protocols to a Network Near You,* NANOG50 Conference, 2010.

[12] S. Das, G Parulkar, N McKeown, P Singh, D Getachew, Long in *Optical Fiber Communication (OFC), collocated National Fiber Optic Engineers Conference, 2010 Conference on (OFC/NFOEC)*, IEEE, 2010.

[13] L. Liu Tsuritani, T. Morita, I Hongxiang Guo, Jian Wu *Openow-based wavelength path control in transparent optical networks: a proof-of-concept demonstration*. In Optical Communication (ECOC), 2011 37th European Conference and Exhibition IEEE, 2011.

[14] R. Wang, Dana Butnariu, and Jennifer Rexford *Openow-based server load balancing gone wild*. In Proceedings of the 11th USENIX conference on hot topics in management of internet, cloud, and enterprise networks and services USENIX Association, 2011.

[15] A. Mahmud, Rahmani R. *Exploitation of Openow in wireless sensor networks.* In Computer Science and Network Technology (ICCSNT), 2011 International Conference IEEE, 2011.

[16] P. Dely, Kassler A, Bayer N in *Computer Communications and Networks (ICCCN),* 2011 Proceedings of 20th International *Conference* IEEE, 2011.

[17] Marcelo D. D. Moreira, Natalia C. Fernandes, Hugo E. T. Carvalho, Lyno Henrique G. Ferraz, Rodrigo S. Couto, Igor M. Moraes, Miguel Elias M. Campista, Lus Henrique M. K. Costa, Otto Carlos M. B. Duarte in *Packet Forwarding Using OpenFlow,* IEEE.

[18] Hardeep Uppal and Dane Brandon in OpenFlow *Based Load Balancing*, University of Washington, 2010.

[19] D.Erickson.Beacon. https://openflow.stanford.edu/display/Beacon/Home,2012. [Online], accessed on 23-Dec-2014.

[20] Floodlight. http://Floodlight.openflowhub.org [Online], accessed on 23-Dec-2014.