A STUDY ON SDN CONTROLLERS

1* Siddharth Valluvan, 2 T. Manoranjitham, 3 V. Nagarajan
1 PG student, CSE department, SRM University, Kattankulathur.
2 Assistant Professor, CSE department, SRM University, Kattankulathur.
3 Professor, CSE Department, SRM University, Kattankulathur.

Email: siddharth_v1993@yahoo.co.in

Abstract:

SDN Controller is the brains of the SDN Network. Controllers act as the control point for networks to manage flow between the Application layer and the Data layer through the Southbound API's and the Northbound API's to create a more flexible network. The Controllers with associating between SDN Controller domains, using standard application interfaces, such as OpenFlow as the organisations have started to deploy more SDN networks. It is crucial to give proper attention to any proposal or design of an SDN controller as the control plane is an essential part of the SDN architecture. In the recent years, several controllers have been revised to evaluate, compare and test the performance of these controllers. This paper shows an independent, comprehensive analysis of the efficiency indexes which includes performance, scalability, reliability and security of modern open source SDN controllers (NOX, POX, Beacon, Floodlight, MuL, Maestro, Ryu).

Keywords: Software Defined Network, Controller, SDN, Comparison, POX, NOX, Floodlight.

Introduction

Software Defined Networking (SDN) is a new networking archetype and is also the most discussed topic of networking technology of recent years as it brings a lot of new capabilities and allows to solve many hard problems of legacy networks [5]. The SDN proposed an approach which involves in moving network's intelligence out from the packet switching devices and putting it into a centralised logical controller. The forwarding decisions are primarily done in the controller and then moves down to the switches where it directly executes the logical decisions. Advantages like global controlling and viewing the whole network at a time are being provided, which is helpful for automating several tasks like Operation of a network, better server and network utilisation, etc. SDN's approach is characterised by the separation of the data and control planes [6]. The data plane consists the forwarding element
(switches and routers). The control plane has the controller, which provides a high abstraction level of the forwarding elements management, something which is absent in today's networks. This proves that there is a need to assess and relate the different existing controllers in the market and research domains. And so, the controller is a fundamental component of the SDN architecture that will contribute to the success or failure of SDN. In an SDN Controller, the different network tasks are carried out by a collection of various pluggable modules. Basic tasks like gathering information about capabilities of each device in the network, their statistics and the availability of devices in the network can be accomplished [9]. Some Enhancements can be added that improve the functionality of running algorithms to perform additional tasks such as coordinating new rules throughout the network and perform analytics.

OVSDDB and OpenFlow are the most popular protocols used by the SDN Controllers for communication between the switches/routers [10]. Other protocols include Netconf and Yang. Many other SDN Controllers are currently in development, while established protocols are finding different footholds to run in SDN Environments. For Example, the IETF is developing a routeing system that enables them to make use of traditional network protocols. An SDN/OpenFlow controller for networks is similar to an operating system for computers. Presently, the state of SDN Controllers is comparable to that of early mainframe computer operating systems. Operating systems are ‘control software’ that provide a library of code to help applications during runtime operations, which is similar to the controller's we use, that act as a control software interacting with switching devices while providing an interface for network management applications. It is important to know the difference in controller architectures and their implementations within the research fields and their everyday uses within the market. It, therefore, becomes necessary to tabulate their performance benchmarks against different operational capabilities.

Varying levels of controllers exist currently, a few able to process only a few thousand flows to some being able to handle millions of flows. Such protocols can support multiple languages, approaches, different architectures and APIs. We will compare the most popularly available controllers and list their capabilities. SDN is currently locked in a battle with networking companies wherein; the SDN community wants to build a controller designed to be supported by every equipment, and the companies want to maintain their controller supported only by their equipment. Nicira Networks created the first SDN Controller called NOX, developed alongside OpenFlow. This controller was subsequently donated by Nicira in its entirety to the SDN Community, where it became the basis for many more controllers to come. This controller became the first Open Source SDN Controller. Later, ONIX was co-developed by Nicira along with Google, which became the base for Google's WAN Controller and VMWare's Controller. While ONIX was supposed to be released as open source, they decided not to release the code. Currently,
there are many Controllers under development. POX and Beacon are currently the most modern controllers. Beacon licensed under GPLv2 and FOSS Exception 1.0, is a Java-based OpenFlow controller. Other Controllers include Trema, Ryu, OpenContrail, etc. The Floodlight controller, branched out from Beacon, was the first SDN Controller to support large networks. The first commercial controller on the market was NEC's Flow Controller, and it was not a derivative of any known Open Source Controllers.

**Literature Survey**

The first study on controllers was done on a limited number of SDN controllers like NOX, Beacon, Maestro. These studies focused only on its performance in the network. These controllers have already been over-ridden by other controllers like OpenDaylight, Ryu, POX, etc [8]. The basis for comparison in the previous study included factors like documentation, virtualization support, support for OpenFlow, documentations, modularity and user interface. Analytic Hierarchy Process (AHP) was used as the comparison mechanism for different controllers, which mapped the different properties values to a value on a scale. By using this tool, an impartial study was performed on these controllers: NOX, Floodlight, POX, Maestro, Beacon, MuL and Ryu. The tool used for this process is called hcprobe[9]. The study showed that all the controllers were able to deal with the average workload in the network, but faced certain security leaks. The Beacon controller demonstrated maximum throughput. Mainly two modes of operation were compared, Proactive - where the rules of the network were already preloaded into the network and Reactive - where the rules are deployed each time the switch receives a packet from another device with no other matching rules in the flow table.

The Programming language used is essential in the development of the controllers since everything is software based. Java is usually the best choice of language since it has support for multithreading and can be used cross-platform with other operating systems. On the other hand, Python has some issues with multithreading. C/C++ has issues with memory management. .Net languages have issues dependency on runtime platforms. Beacon consumes less memory as compared to other controllers. Therefore it shows better performance. Memory leak is the main problem highlighted among the controllers.

**Different Controllers**

A. OpenDaylight Controller

OpenDaylight is a community-based Open Source project, where its goal is to improve SDN by giving features and protocols that hold up to the industry standard. This controller has been recently renamed as the OpenDaylight Platform [8]. It is open to all, including end users and customers, thereby providing a common platform for those
with motivations and goals in SDN to work together to find newer innovative solutions. Under the Linux Foundation, OpenDaylight consists of support for the OpenFlow protocol, however, can also support other open SDN standards. The OpenFlow protocol is considered as the first SDN standard where it defines the open communications protocol that allows controllers to work with the data forwarding plane and allows it to make changes to the network. This protocol gives businesses an opportunity to have superior control over their networks and the ability to adapt well to their changing needs. The OpenDaylight Controller is utilized in a wide variety of environments. It supports a modular controller framework, then provides support for other SDN standards and forthcoming protocols. The OpenDaylight Controller applications can collect network information, perform analytics by running algorithms, and create new rules throughout the network, where it also exposes open northbound APIs.

B. NOX

NOX is a piece of the software-defined networking (SDN) ecosystem, an explicit platform for building network control applications[8]. OpenFlow was the first SDN technology to get real name recognition. NOX was the first OpenFlow controller which was primarily developed at NiciraNetworks alongside OpenFlow, serving as a network control platform that provides a high-level programmatic interface for management solutions and the advancements in newer control applications.

Its system-wide perceptions turned networking into a software issue. Ever since Nicira donated NOX to the research community in 2008, it has been the basis for various research projects in the early exploration of the SDN space. NOX aims to provide a platform for developers and researchers which give them the capability develop novel applications that innovate within the industrial and business networks. NOX's applications usually determines how each flow is routed or not routed in the network.

C. POX

POX is an open source Python-based development platform for software-defined networking (SDN) control applications, for instance, OpenFlow SDN controllers. POX is becoming more commonly used than NOX; which is a sister project. It allows rapid development and prototyping. POX uses OpenFlow or OVSDB protocol for providing a framework for communicating with SDN switches. Using the Python programming language, developers can use POX to create an SDN controller. POX is a tool to educate people about SDN and is also used for research purposes and for building network-related applications. POX Components can be invoked directly from the Command Line Interface[2]. The Network functionality is implemented by using these components in SDN. POX can be utilized as a
primary SDN controller by loading the readily available default components. Developers can create a more sophisticated controller by using new components, or they might write applications that target the API itself.

D. Ryu

Ryu is an Open Source SDN Controller that is used to increase the flexibility of the network by making the task of handling the traffic easier. Ryu provides several components with full program interfaces that allow developers to create new network management and control applications with ease conveniently. These Components can be used to customise deployments by organisations to meet their particular needs; such that existing components can be quickly and easily modified and implemented into current networks to meet the changing needs of different applications.

E. Trema

Trema is an OpenFlow controller framework written in Ruby that provides many solutions to create a controller in the network. It provides a network emulator and libraries that can create simple OpenFlow-based networks on a system. These features are an efficient way to provide development and testing environments for networks. It allows developers to build custom controllers by adding messaging scripts. Trema focuses on precise coding methods to reduce the possibility of errors and for easier code maintenance.

F. Beacon

Beacon is a Java-Language based SDN Controller that supports Multi-threads and event handling. It is modular, supports multiple platforms and is very fast. Its development began in the early 2010s and had been used in several trials and projects. It is capable enough to run a data centre and can run for months without any downtime. Beacon is Open Source and is currently licensed under GPLv2 and FOSS Exception v1.0. Code packages can be installed even during runtime without interrupting other packages. Beacon can optionally support custom UI Frameworks and can embed webserver enterprises.

G. Floodlight

Floodlight Open SDN Controller is an enterprise-class, Apache-licensed, Java-based OpenFlow Controller developed by Big Switch Networks; that works with OpenFlow protocol to manage the flow of traffic in an SDN environment. Floodlight is simple to use, build, maintain and run. It can also run with any switches, both physical and virtual, as long as they support the OpenFlow Protocol. Floodlight is currently open source; Beacon Controller is a fork of Floodlight.

H. MuL

Mul is a C Language based SDN OpenFlow Controller. It supports a multi-threading infrastructure and has a multi-
leveled northbound interface for hosting various applications. Currently, it aims to support southbound interfaces such as OpenFlow 1.3, 1.4 and of-config, ovsdb, etc. It is designed with reliability and performance in mind which is essential for critical networks. Mul is easy to learn and implement, making it highly flexible.

I. Maestro

Maestro is a controller for implementing network control applications. It provides interfaces for implementation of modular network applications which can control the state of the network and can be used to coordinate interactions between devices. Maestro can improve a machine's throughput performance by exploiting its parallelism. Maestro requires as little effort as possible to manage the parallelization since it does the complex job of managing the workloads as well as the scheduling of threads.

Controller Features

A. Programming Language

Programming for a controller make use of common languages like Python, Java and C++, and may also use languages like Ruby and Javascript to a certain extent [4][3]. A few characteristics of these languages are that they are very easy to learn, allows faster memory access, runs cross-platform and allows multiple threads. Java displays a more rapid runtime when it comes to business applications.

B. OpenFlow Support

The OpenFlow protocol enables the direct manipulation of OpenFlow switches in the data forwarding plane. It was the first standard in the Southbound Interface. Newer versions of OpenFlow include v1.3 or v1.4. This version provides support for IPv6. Before choosing a controller, its functionality needs to be understood [7].

C. Network programmability

The ease of programmability of the network is the most significant advantage of using SDN in networks; it allows the management of complex networks with a large number of connected devices and also lets users quickly deploy new networks. Older networks managed devices in the system on a device-by-device basis, which is time-consuming and can lead errors. SDN introduces automation of these devices which reduces the difficulty and increases the dynamic of the management process. The support for network programmability relies on a good GUI and command-line interface and the ability to integrate some northbound interfaces.

D. Efficiency

Different parameters such as throughput, network security, reliability and scalability fall under the umbrella term of Efficiency. Various metrics such as throughput, the number of interfaces, latency, etc. defines the performance of a
controller. Similarly, some parameters determine its scalability and reliability. These parameters are used as the factor for comparison, which can be used to calculate the efficiency [10].

E. Southbound Interfaces

Southbound APIs allows the controller to make dynamic changes to the rules in the forwarding plane which allows efficient control over the network. OpenFlow is not the only API available; other APIs are also currently in development. Other such APIs include NETCONF, which was standardised by IETF and OF-CONFIG, supported by the Open Network Foundation which can be used to manage devices. Traditional routeing protocols such as OSPF, BGP, etc. are also being emulated into the southbound interfaces in controllers with the aim of supporting a network hybrid that works with both traditional networks and SDN [11].

F. Northbound Interfaces

This interface is the interface between the controller and the application layer. Development of newer applications depends on this interface. Since northbound APIs support a variety of applications, they are a critical part of the overall architecture. The existence of this interface is directly tied to the innovations that occur in the community and the improvements that it can bring. They also support various Cloud management applications like Cloudstack and Openstack. Currently, the Representational Stat Transfer or REST is the most frequently used northbound interface by the controllers [11].

Methodology

Cbench, also known as, controller benchmarker, is a program used for testing OpenFlow controllers through generating packet-in events for new flows. It emulates a group of switches which connects to a controller, sends packet-in messages, and watch for flow-mods to get pushed down. Cbench measures the changes that might have a performance impact.

Cbench is usually used as a testing tool for SDN controllers. It primarily runs in two modes: throughput and latency modes. In the throughput mode, Cbench sends as many packets as possible to compute the maximum number of packets handled by the controller while in the latency mode, Cbench sends a packet and waits for the reply to calculate the time taken to process a single packet by the controller. Cbench and the tested controller would be able to run on the same machine or different machines. Therefore, due to the limitation of having a 10 Gbps link, tests are being performed on a single computer.
<table>
<thead>
<tr>
<th>Controller</th>
<th>Supported Language</th>
<th>Interface</th>
<th>Supported Platforms</th>
<th>Southbound API</th>
<th>Northbound API</th>
<th>Modular</th>
<th>OpenStack Support</th>
<th>Multithreading</th>
</tr>
</thead>
<tbody>
<tr>
<td>POX</td>
<td>Python</td>
<td>Python</td>
<td>Mac, Linux, Win</td>
<td>OF 1.0</td>
<td>REST</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NOX</td>
<td>C++</td>
<td>Python</td>
<td>Linux</td>
<td>OF 1.0</td>
<td>REST</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>FloodLight</td>
<td>Java</td>
<td>Java Web UI</td>
<td>Linux</td>
<td>OF 1.0,1.3</td>
<td>REST</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Beacon</td>
<td>Java</td>
<td>Web UI</td>
<td>Linux, Mac, Win</td>
<td>OF 1.0</td>
<td>REST</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Ryu</td>
<td>Python</td>
<td>Web UI</td>
<td>Linux</td>
<td>OF 1.0,1.2,1.3,1.4 and NETCONF</td>
<td>REST</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MuL</td>
<td>C</td>
<td>Web UI</td>
<td>Linux</td>
<td>OF 1.0,1.3,1.4, OVSDB, OF-CONFIG</td>
<td>REST</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>OpenDaylight</td>
<td>Java</td>
<td>Web UI</td>
<td>Win, Linux, Mac</td>
<td>OF 1.0,1.3,1.4, NETCONF</td>
<td>REST</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Maestro</td>
<td>Java</td>
<td>Web UI</td>
<td>Win, Linux, Mac</td>
<td>OF 1.0</td>
<td>REST</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Conclusion**

After conducting a comparison of several controllers based on multiple criteria, a performance test using Cbench was prepared. Therefore, because of the diversity of SDN applications and the controller's methods, the choice of the best-fitted controller will be somehow application dependent. Open Daylight is an excellent choice as a full-featured controller as it supports a broad range of applications with a healthy ecosystem giving it a real opportunity to become the über-controller. Its integration of the IoT data broker and new IoT-specific southbound interfaces in its last release made it the first competitor in the election of the "Internet of the future" controller.
Reference

8. SDN controllers: A comparative study, Ola Salman Imad Elhajj Ayman Kayssi Ali Chehab Electrical and Computer Engineering Department American University of Beirut, Beirut 1107 2020, Lebanon
10. Advanced Study of SDN/OpenFlow controllers, Alexander Shalimov, Dmitry Zuikov, Daria Zimarea, Vasily Pashkov, Ruslan Smeliansky
11. Open Networking Foundation. OpenFlow Switch Specication
   https://www.opennetworking.org/sdn-resources/onf-specifications/openflow