SDN CONTROLLERS COMPARISON

V R SUDARSANA RAJU

SRIT, Bangalore
E-mail: sudarsanaraj@yahoo.com

Abstract - This paper presents about Software Defined Network (SDN) which is a new networking paradigm where the architecture moves from the traditional fully distributed model to a more centralized approach. Software Defined Networking (SDN) is a new networking arch type 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. This approach is also characterized by the separation of the data and control planes. The SDN proposed an approach which involves in moving network's intelligence out from the packet switching devices and putting it into a centralized logical controller. 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 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 utilization, etc.

Keywords - SDN Controller; Software Defined Network.

I. INTRODUCTION

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. The Controllers with associating between SDN Controller domains, using standard application interfaces, such as OpenFlow as the organizations have started to deploy more SDN networks. The data plane includes the forwarding element (switches and routers) and the control plane includes the controller. The controller provides a high abstraction level of the forwarding elements management which is absent in today's networks. Therefore, the controller is a fundamental component of the SDN architecture that will contribute to the success or failure of SDN. Therefore, there is a need to assess and compare the different existing controllers in the market. We are far from a controller (in some cases referred to as network operating system) which is hardware and language independent. However, today's controllers run as monolithic applications and they are highly tied to their programming languages (java, python, C, C++, etc.) Therefore, SDN specific languages such as Pyretic and Frenetic offering high-level abstraction languages were proposed to allow for application portability; however, they are in reality linked to a specific controller platform (POX).

Due to the importance of the controller within the SDN architecture and the diversity of architectures and Implementations in the market, there is a need to assess and benchmark all these choices against different performance indicators.

II. BASIC FUNCTIONS OF SDN CONTROLLERS

As SDN separates the data plane and the control plane, the intelligence of the network is moved to the controller; all computations are done there and many applications and features can be added as needed. The basic modules are discussed in where a lightweight carrier grade controller is proposed. Link discovery module, topology module, storage module, strategy making module, flow table module and control data module are the basic SDN controller's modules. Two modules are responsible for providing the routing service: the topology manager and the link discovery modules. Collecting the physical link status information is the role of the link discovery module. There are two types of link discovery: link discovery between OpenFlow nodes (switches) and link discovery between an end host and an OpenFlow node. The former uses the Link Layer Discovery Protocol (LLDP). Thus, the provided information by the link discovery module is used to build the neighbor database at the controller level. This database is managed by the topology manager module to build the global topology database which relies on the computation of the shortest (and alternate) path to any OpenFlow node or host. Any changes or link ruptures are trackable by the link discovery module. Therefore, the topology manager module has the role to maintain the topology information and to recalculate the routes in the network after any modification in the neighbor database.

III. FEATURES OF SDN CONTROLLERS

1) Cross platform compatibility

Running cross-platform, allowing multithreading, being easy to learn, allowing fast memory access and

good memory management are essential programming languages' characteristics. When choosing a certain controller, we have to take these factors into consideration because they affect the controller's performance and development speed. Python, C++, and Java are the most used languages for SDN controllers programming. In general, the Java coded controllers have the characteristic to run cross-platform and present good modularity, the C coded controllers provide high performance but lack high modularity, good memory management and good GUI and the Python coded ones lack real multi-threading handling.

2) Southbound Interfaces

Southbound APIs allows control over the network. These APIs are used by the controller to dynamically make changes to forwarding rules installed in the data plane devices consisting of: switches, routers, etc. While OpenFlow is the most wellknown of the SDN protocols for southbound APIs, it is not the only one available or in development. NETCONF (standardized by IETF), OF-Config (supported by the Open Network Foundation (ONF)), Opflex (supported by Cisco) and others are examples of southbound interfaces used for managing network devices. Additionally, some routing protocols such as IS-IS,OSPF, BGP are being also developed as southbound interfaces in some controllers in the aim to support hybrid networks (SDN and non-SDN ones) or to apply the traditional networking in a software-defined manner.

3) Northbound Interfaces

The northbound APIs are used by the application layer to communicate with the controller. They are the most critical part in the SDN controller architecture. The most valuable benefit of SDN is derived from its ability to support and enable innovative applications. Because they are so critical, northbound APIs must support a wide variety of applications. These APIs allow also the connection with automated stacks such as OpenStack or CloudStack used for Cloud management. Recently, the ONF turned its focus to the SDN northbound API after working to standardize the southbound interface (OpenFlow). They have stablished a Northbound Working Group that will write code, develop prototypes and look for standards creation. Currently, the Representational State Transfer (REST) protocol seems to be the most used northbound interface and most of the controllers implement it.

4) OpenFlow Support

The OpenFlow protocol is a key enabler for softwaredefined networks. It was the first standardized southbound interface. It allows direct manipulation of the forwarding plane of OpenFlow switches. When choosing an OpenFlow controller, we need to understand the OpenFlow functionality that the controller supports as well as the development roadmap to implement newer versions of OpenFlow, such as v1.3 or v1.4. One reason for needing to take this into consideration is that important functionality such as IPv6 support, for example, is not part of OpenFlow v1.0 but is part of the OpenFlow v1.3 standard.

5) Network programmability

Network programmability is the most important benefit of the SDN introduction to deal with the unprecedented management complexities in today's network with the explosion in the number of connected devices and the deployment of new services. Using the device-by-device paradigm to manage the high scale future networks will not be feasible. The old static way of managing network devices is time consuming, error prone and leads to inconsistencies. Software defined paradigm comes to hide these management difficulties introducing automation and dynamicity in the management process. Automated scripts can be run through command-line interfaces (CLIs) and applications can be deployed on top of the controller platform to perform predefined tasks and management functions. The controller support of network programmability relies essentially on its degree of integration of a wide number of northbound interfaces, a good graphical user interface and a CLI.

6) Efficiency (Performance, Reliability, Scalability, and Security)

The controller efficiency is an umbrella term used to refer to the different parameters – performance, scalability, reliability and security. Various metrics, such as number of interfaces a controller can handle, latency, throughput, etc. define what we call performance. Similarly, there are various metrics defining the scalability, reliability and security. Most of the work done to compare the controllers consider only the performance criteria. Additionally, the centralization of the control in the SDN scheme will present a serious challenge from the reliability and the performance perspectives. Thus, the distributed scheme, supported by some controllers, aims to cope with this issue.

7) Partnership

Being under good partnership oversight, an SDN controller will have chances to be maintained and enhanced for a long time]. The experience in the network and computer domains, and the economic capacity of the partner's organization are the main criteria biasing trust and use of products. Cisco, Linux Foundation, Intel, IBM, Juniper, etc. are examples of reputable organizations entering the SDN market and participating in controllers development. Several surveys have been done in the previous two years providing us with lists of the most commonly known controllers. Essentially, most of the

listed features are taken into consideration when comparing the controllers.

8) 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. 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.

We note that ONOS and OpenDaylight are the most featured controllers. These two Java coded controllers run cross-platforms and present high modularity using the OSGI container that allows loading bundles at runtime. Inheriting the power of Java/Javascript in the graphical user interfaces programming, they present good GUI feature. Being under the partnership of well-known network providers and research communities, they have a clear development plan and good documentation. Additionally, their support for the distributed scheme make them able to conduct a real SDN wide deployment.

The ONOS controller is principally designed for carrier networks. It gives them the ability to provide new SDN services along with their initial proprietary services. ONOS architecture is designed to maintain high speed and large scale networks. Its main distinguishing characteristic is its support for hybrid networks. However, OpenDaylight was primarily datacenter focused but its latest release (Lithium) shows a capability to support different kinds of applications. Many southbound interfaces have been added (HTTP, COAP, PCEP, LACP, OpFlex, SNMP, etc.) and new modules have been implemented (IoT data broker (IoTDM), unified secure channel of USC, etc.). Thus, it is the first controller entering the IoT domain. Supporting a wide range of southbound interfaces and the distributed control paradigm, it seems to be the controller of the Internet of the Future. The support of OpenStack Neutron plugin in its architecture has also a remarkable importance when deploying software-defined edges responding to the Edge computing proliferation.

IV. 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. 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. 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. 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. Ryu, presenting fair features, is a good choice for small businesses and research applications. Being Python coded, this controller presents facilities for applications and modules development. However, its lack of high modularity and its inability to run cross-platforms limit its wide use in real market applications.we will try to compare these controllers efficiency. Even though it will not be the only criterion to choose a controller, but processing requests at high rates with minimum latency is a key requirement of any controller.

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.

CONCLUSION

In this paper, we conducted a comparison of severalcontrollers based on multiple criteria. Thus, due to the diversity of SDNapplications and the controller'suses, the choice of the best-fitted controller will be somehow application dependent. We have found that OpenDaylight is a good choice as a full-featured controller. Supporting wide range of applications with a good ecosystem gives it a real chance to become the useful controller.

REFERENCES

Books:

[1] Thomas D. Nadeau & Ken Gray (2016), SDN: Software Defined Networks, O`REILLY, Beijing.

Websites:

[1] https://onosproject.org/

[2] D. Kreutz F. M. V. Ramos P. Esteves C. Esteve S. Azodolmolky "Software-Defined Networking: A Comprehensive Survey" Proceedings of the IEEE vol. 103 pp. 10-13 2016.

[3] Q. Y. Zuo M. Chen G. S. Zhao C. Y. Xing G. M. Zhang P. C. Jiang "Research on OpenFlow-based SDN technologies" Journal of Software vol. 24 pp. 1078-1097 2015.

[4] S. Jain A. Kumar et al. "B4: experience with a globally-deployed software defined wan" ACM SIGCOMM 2016 Conference on SIGCOMM. pp. 3-14 2016.

[5] Singh J. Ong A. Agarwal et al. "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's

Datacenter Network" ACM Conference on Special Interest Group on Data Communication pp. 183-197 2015.

[6] K. Zhang Y. Cui H. Y. Tang J. P. Wu "State-of-the-Art survey on software-defined networking (SDN)" Journal of Software vol. 26 pp. 62-81 2015.

[7] T. Q. Zhou Z. P. Cai J. Xia M. Xu "Traffic engineering for software defined networks" Journal of Software vol. 27 pp. 394-417 2016.

[8] S. Scott-Hayward S. Natarajan S. Sezer "A Survey of Security in Software Defined Networks" IEEE Communications Surveys & Tutorials vol. 18 pp. 623-654 2016.

[9] Blen A. Basta M. Reisslein "Survey on Network Virtualization Hypervisors for Software Defined Networking" IEEE Communications Surveys & Tutorials vol. 18 pp. 655-685 2016.

★★★