IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Vol.-7, Issue-1, Jan 2019 E-ISSN: 2347-2693

An overview of SDN Controllers

C. Kalaivani

Department of Computer Applications, Chevalier T.Thomas Elizabeth College for Women, Chennai, India

*Corresponding Author: sreekalai0903@gmail.com

Available online at: www.ijcseonline.org

Accepted: 15/Jan/2019, Published: 31/Jan/2019

Abstract - Software Defined Network is based on separation of network intelligence from packet switching devices and merging network intelligence in a centralized controller. With SDN, Enterprises can manage the entire network from a single SDN controller, irrespective of different vendors of network elements [11].This controller acts as the main brain and act as a strategic control point to decide about routing using OpenFlow protocols. SDN controller is a kind of operating system for network where applications and devices can have communication only through it. In this paper, many SDN controllers are discussed.

Keywords **:** SDN controller, performance, applications, API.

I. INTRODUCTION

In a traditional network architecture each device has a separate control plane, whereas in SDN architecture it is separated and centralized on an isolated process (called controller) running at control layer. This isolated process (controller) provides universal view of the network. The controller resides between network devices and application layer to translate the requirements from the application layer and manage flow control to the network devices (via southbound APIs). It also provides the SDN application with an abstraction view of network and business logic (via northbound APIs). This technology is being considered one of the favorable technologies for isolation of control plane & data plane and logical placement of centralized control with the SDN controller. The following figure (Figure 1) shows the architecture of SDN:

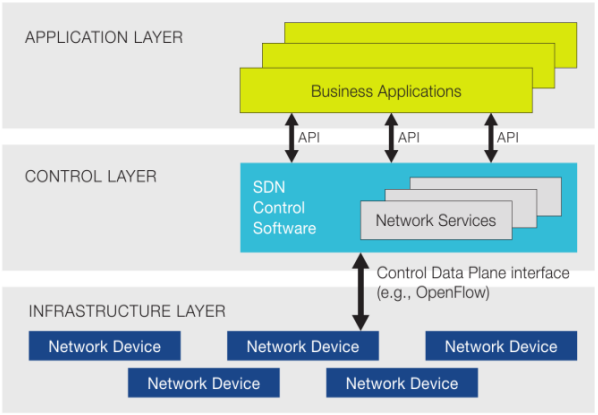

Figure 1 : SDN architecture

The paper focuses on the working principle, features, and types of SDN controllers in the following sections.

II. SDN CONTROLLER ARCHITECTURE

In the SDN architecture, the controller separates the data plane and the control plane where all the computations are done and necessary applications and features can be added whenever needed [1]. Basic modules[2] concerned with a SDN controller are link discovery module, topology module, storage module, strategy making module, flow table module and control data module. The Link Discovery module is responsible for discovering and maintaining the status of physical links in the network which will be triggered only when any unknown traffic enters the domain. The information collected by this module will build a neighbor database in the controller which will contain all the OpenFlow neighbors. This database will be used by the topology manager to build and maintain the topology information in the controller thereby calculates the routes in the network. The Topology Manager builds the global Topology Database at the controller, which contains both the shortest and alternate path information to any OpenFlow node or host. The modules in an SDN controller are shown in the following figure:

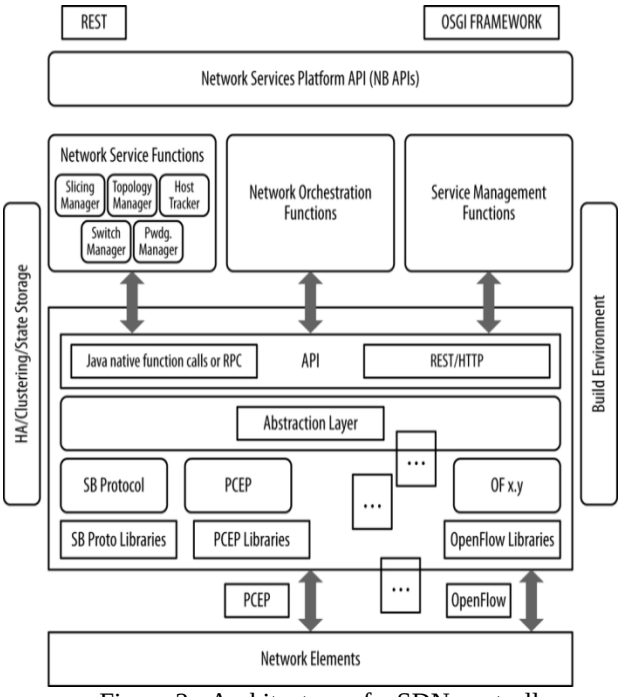

Figure 2 : Architecture of a SDN controller

III. FEATURES OF SDN

1. Programming Language

A controller's performance and development speed will be affected by certain factors of a programming language like platform independence, multithreading, being easy to learn, allowing fast memory access and good memory management. Python, C++, and Java are the most used languages for SDN controllers programming. [1]

2. OpenFlow Support

One of the key factors in SDN is OpenFlow protocol. The forwarding plane of OpenFlow switches will be directly manipulated by this protocol[3]. Before choosing an OpenFlow controller, the OpenFlow functionality that the controller supports as well as the flexibility to implement newer versions of OpenFlow should be understood.

3. Network programmability

An important benefit of SDN is network programmability where the complexity with increased number of connected devices and deployment of new services are managed perfectly by introducing automation and dynamicity in the management process. Automated scripts can be run through 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 command-line interface (CLI) [4].

4. Efficiency

The efficiency of a controller is measured using different parameters like performance, scalability, reliability and security. Performance is defined by various metrics like number of interfaces a controller can handle, latency, throughput, etc. Even though scalability, reliability and security are defined by some other metrics SDN controllers are compared with one another based on performance.

5. Southbound Interfaces

These APIs will have an efficient control over the network as they are used by the controller to make dynamic changes for the forwarding rules installed in the data plane devices consisting of: switches, routers, etc. Some of the southbound APIs are OpenFlow, NETCONF(standardized by IETF), OF-Config (supported by the Open Network Foundation (ONF)), Opflex (supported by Cisco).

6. Northbound Interfaces

The application layer communicates with the controller with the help of the northbound APIs. They seem to be more critical as they have to support different innovative applications. In addition, These APIs allow also the connection with automated stacks such as OpenStack or CloudStack used for Cloud management. Currently, the Representational State Transfer (REST) protocol seems to be the most popular northbound interface.

IV. LIST OF SDN CONTROLLERS

Some of the SDN controllers with the above mentioned features are explained below:

1. NOX:

It was the first SDN controller developed by Nicira networks in the year 2009 that supported Openflow protocol. The NOX controller is made up of APIs which are built using C++ and Python languages and operate on Linux, Ubuntu and Debian systems. NOX mainly provides support modules specific to OpenFlow. The NOX core provides helper methods and APIs for interacting with OpenFlow switches, including a connection handler and an event engine. Additional components are available including host tracking, routing, topology (LLDP), and a Python interface implemented as a wrapper for the component API. Programmatic interfaces also exist for control over the network and high level services.[5]

2. Beacon:

Beacon is an OpenFlow controller developed in Java and released in 2010. It is a multi-threaded controller and considered more suitable for enterprise class networks and data centers. It runs on many platforms, such as Linux servers and Android phones. Beacon's API is designed to be simple and to impose no restrictions as far as available Java constructs, such as threads, timers, sockets, etc. are

concerned. The API for interacting with OpenFlow switches is event based.

Beacon supports both event-based and threaded operation. It uses a static approach in which a fixed number of switches are assigned to a worker thread. Worker threads use static packet batching to serve the requests from the connected switches. Once the packets are processed and ready to be sent, Beacon in its default mode uses write coalescing and allows only one write per I/O select loop to reduce the overhead of socket system calls for each individual OpenFlow message. Alternatively, an immediate mode can be enabled in which the controller attempts a socket write for every outgoing OpenFlow message waiting to be written to the switch to reduce per packet latency. Static partitioning and input batching improve its throughput in the default mode. [5]

3. Maestro:

Maestro is an OpenFlow controller introduced in 2010 as an operating system for some SDN applications. It is developed in Java, which makes it highly portable to various operating systems and architectures. Maestro provides interfaces for implementing modular network control applications to access and modify the state of the network, and coordinate their interactions. Each network control component is represented as one application in Maestro, which is a Java class that contains the code for the control function. A simple and straightforward API is used by all the applications to extend the base abstract class application, and interact with Maestro.

Maestro is a controller, which uses task batching so that worker threads pull a batch of tasks, so as to process multiple flow-requests in a single execution.[5] The output batching technique is used to send packets out in which packets belonging to the same destination are grouped together and sent using a single socket system call [9]. Maestro handles most of the tedious and complicated job of managing work load distribution and worker threads scheduling.

4. Trema

Trema is a programming framework for developing OpenFlow controllers where the modules are created either in C or Ruby. It provides a network emulator and libraries that can create simple and efficient OpenFlow based networks on a system. The main API that the Trema core modules provide to an application is a simple, non-abstracted OpenFlow driver. The base controller design is event-driven and is often compared to the explicit handler dispatch paradigm of other open source products. In addition, the core modules provide a message bus that allows the application modules to communicate with each other and core modules.

The libraries in Trema can be categorized under multiple headings as listed below: protocol (i.e. OpenFlow), interfaces (i.e. OpenFlow application, switch, management), commonly used data structures (i.e. Linked list, doubly-linked list, hash table, timers), utilities (i.e. log, stats, wrapper), network protocols (i.e. CP, IP, UDP, ether and etherIP, ICMP and IGMP).[5]

5. Ryu [5]

Ryu is a component-based, open source framework implemented entirely in Python. It integrates with OpenStack and supports OpenFlow. It provides a logically centralized controller and a well-defined API that make it easy for operators to create new network management and control applications. Components include event management, messaging, in-memory state management, application management, infrastructure services and a series of reusable libraries.

Ryu has an impressive collection of libraries, ranging from support for multiple southbound protocols to various network packet processing operations. With respect to southbound protocols, Ryu supports OF-Config, Open vSwitch Database Management Protocol (OVSDB), NetConf, SFlow (Netflow and Sflow) and other third-party protocols.

6. OpenDaylight:

OpenDaylight, the largest open source SDN controller, is designed for customizing and automating networks of any size and scale with a clear focus on network programmability. The ODL platform is designed to cover a lot of use cases such as [6] Network Resources Optimization (NRO), Automated Service Delivery, Cloud, NFV and others. Using ODL services such as dynamic network optimization, on-demand services (i.e. bandwidth, dynamic VPN services etc.), agile service delivery on cloud infrastructure are provided. ODL may be also used to achieve centralized administration of the network.

The idea of Modular application development in sense that a set of loosely coupled modules can be integrated into large application cause Open Services Gateway Initiative (OSGI) [7], a dynamic module system for Java, defines one such architecture for modular application development which is used in Opendaylight controller.

On the northbound side, the interaction between the controller and the applications is done via Web Services for the request-response type of interaction. The controller exposes open Northbound APIs which are used by applications. The ODL platform supports OpenFlow and OpenFlow extensions such as Table Type Patterns (TTP), as well as traditional protocols including NETCONF, BGP/PCEP and CAPWAP. Additionally, ODL interfaces

with OpenStack and Open vSwitch through the OVSDB Integration Project.

7. ONOS:

ONOS has been designed aiming to fulfill the following goals: Code modularity, separation of concern, configurability, and protocol agnosticism [10]. The ONOS kernel and core services, are written in Java as bundles that are loaded into the Karaf OSGi container. Since ONOS runs in the JVM, it is platform independent. The ONOS platform is designed to support various application categories such as control, configuration and management applications.

Open Network Operation System is a distributed platform controller that is specialized for scalability and high availability. ONOS support multiple protocol at the southbound interface that provide communication with various devices and exploit right API in northbound in order to hold the needs of service provider use cases and application developers. Synchronization between multiple instances of the controller is implemented using an anti-entropy protocol in ONOX.[8]

8. Floodlight :

Floodlight is a Java multi-threaded OpenFlow controller, initially based on the Beacon implementation. Its last version was released on March 2016. It is intended to be a platform for a wide variety of network applications. The Floodlight core architecture is modular, with components including topology management, device management, path computation, infrastructure for web access, counter store and a generalized storage abstraction for state storage.

The Floodlight OpenFlow controller can interoperate with any element agent that supports OpenFlow, but Big Switch also provides an open source agent that has been incorporated into commercial products. Floodlight can be run as a network plugin for OpenStack using Neutron. The Neutron plugin exposes a Networking-as-a-Service (NaaS) model via a REST API that is implemented by Floodlight. Once a Floodlight controller is integrated into OpenStack, network engineers can dynamically provision network resources alongside other virtual and physical computer resources. This improves overall flexibility and performance.

V. CONCLUSION

SDN controllers have several different properties which affect the efficiency of the network. From the various kinds of controllers discussed above one can be selected using its multiple properties to find out its impact in the efficiency and effectiveness in network is implemented. A SDN controller can be selected using any of the two techniques namely, Multi-Criteria Decision Making (MCDM) and Analytic Hierarchy Process (AHP).

REFERENCES

[1] Ola Salman, Imad H. Elhajj, Ali Chehab SDN controllers: A comparative study https://www.researchgate.net/publication/304457462

[2] F. Alencar, M. Santos, M. Santana and S. Fernandes, '"How Software Aging affects SDN: A view on the controllers,"*Global Information Infrastructure and Networking Symposium (GIIS), 2014*, pp. 1-6.

[3] Feng Wang, Heyu Wang, Baohua Lei and Wenting Ma,"A Research on High-Performance SDN Controller," Cloud Computing and Big Data (CCBD), 2014 International Conference on, pp. 168-174.

[4] O.N. Fundation, '"Software-defined networking: The new norm for networks," *ONF White Paper*.

[5] Dimitra Sakellaropoulou, "A Qualitative Study of SDN Controllers", M.Sc., Thesis, Athens, September, 2017

[6] https://www.opendaylight.org

[7] *OSGi Core Release 5*, OSGi Alliance, San Ramon, CA, USA, Mar. 2012.

[8] Saleh Asadollahi, Dr. Bhargavi Goswami, Dr. Atul M Gonsai, Software Defined Network, Controller Comparison, International Journal of Innovative Research in Computer and Communication Engineering, April 2017 Special Issue

[9] Abhishek Rastogi, Abdul Bais, "Comparative Analysis of Software Defined Networking (SDN) Controllers – In Terms of Traffic Handling Capabilities", Multi-Topic Conference (INMIC), 2016 19th International

[10] https://wiki.onosproject.org

[11]Mandar B. Shinde, Sunil G. Tamhankar, "Review: Software Defined Networking and OpenFlow", International Journal of Scientific Research in Network Security and Communication

Author's Profile

Mrs.C.Kalaivani pursued her Bachelor of Science from Periyar University, Salem in 2001, Master of Science from Bharathidasan University, Tiruchirapalli in 2003 and M.Phil(CS) from Periyar University, Salem in 2008. She is currently working as an Assistant Professor in Department of Computer Applications, Chevalier T.Thomas Elizabeth College for Women, Chennai, India since 2009. She has published more than 3 research papers in international journals/Conferences. Her area of interests is Software Defined Networking, Mobile Networks. She has 11 years of teaching experience.

