DOI: https://doi.org/10.33103/uot.ijccce.22.4.14

Software Defined Network Implementation by Using OpenDayLight Centralized Controller

Salah Sabah Abed[1], Mohammed Natiq Fadhil[2]
[1,2]Department of Computer Science, University of Technology, Baghdad, Iraq
[1]cs.19.52@grad.uotechnology.edu.iq, [2]Mohammad.n.fadhil@uotechnology.edu.iq

Abstract**—** *The most attractive study framework among academics is Software Define Networking Networking SDN, which aims to create the Internet with an architecture-independent architecture that will lead to the most significant advances in the network field. This can solve many network problems to deal with high demand changes and reduce Replenishment, changes and less manual work. Because of the limited architecture of traditional networks, which requires modifications in the basic design, network expansion has been mature and slow. Since 2010, until now, the ODL - OpenDayLight model has been proposed to solve most of the problems that guide network engineers in the process of managing complex high-volume networks by top research-oriented universities around the world. Now is the time to turn dreams into reality by putting the presented ideas into action, which will result in a solution that meets the expectations of the researcher regarding the process of managing complex networks and all forms of networks. This document is an attempt to assist researchers in implementing a software identification network infrastructure on which the research community may focus on further analysis and development. We demonstrated an incremental approach to implementing ODL - OpenDayLight Controller (ODL is a JVM program and can run from any operating system and device as long as it supports Java) from the Software Define Network, as well as creating and executing required scenarios, and illustrate the working nature of ODL - OpenDayLight compared to ryu contrlloer, in this paper Research.*

Index Terms*— SDN, OpenDayLight, Mininet, Switch, Controller, Performance evaluation.*

I. INTRODUCTION

The internet has been used to interconnect network devices and distribute information for 45 years now. Ethernet protocol and devices are most well-worked, but circumstances changed, the number of endpoint devices that are connected to internet is more than one per person. Again, new technologies, like the IOT, enable remote connection and management of almost all devices over the Internet, rapidly increasing traffic. In such a condition, control of congestion on one side and managing the Datacenter to store users' data on the other side become a momentous subject of research. Shortage of innovation makes academics keen to work on the issue of configuring large and diversified network equipment such as routers and switches, employing low-level to organize the network structure use pre defense and command protocols. New technology (SDN) [1][2] is a unique method in the area of networking that appears to change the architecture of network appliances such as routers devices and switches devices by simplifying the intricate node structure. In lieu of multipartite and messy devices, it provides a single centralized controller (software based) with programmability and easy forwarded devices. Different network society comes up with number of controllers among different features and purposes. But, we choose OpenDaylight from among the several available controllers [3].

In this paper second part talks about the problems of a traditional network, the third part describes SDN architecture, and following part go into detail the architecture of the OpenDayLight (ODL) controller. The last part provides clarify pursuance of a SDN scenario on ODL controller with Mininet.

Received 22/January/2022; Accepted 13/May/2022
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.4.14

II. TRADITIONAL NETWORKING PROBLEM AND ARCHITECTURE

End devices like switches, which are responsible for routing and forwarding packets via networks, must accomplish three separate tasks in today's networks. First: Data plan (process the transit traffic according to a decision made by control plan), second: control plan (figures out and handle what's going on around, depend to its type and configuration) and third: management plan (converse with administrator). Distributed and transport protocols are running inside of network protocol is the base of transiting traffic through IP network [4]. Once the packet reaches a data plan through the port, in accordance with the header information and routing table, the data plan will transmit the packet to the adequate port or take specific decisions such as drop packet or send it to the control plan in case of shortage of information. Control plan or brain of device configures interface, IP subnet, and routing protocol [5]. Moreover, it collects and keeps the information in the waiting list (queuing list), builds route tables and calculates Spanning-Tree Protocol, and sends a copy of the table to a data plan, which it can further deal with the arrival of the packet without involving a control plan later [6]. Using a variety of CLI commands, administrators configure the Control plan through the management plan.

The complicated architecture of traditional devices, in which the control and data plane is vertically tightly coupled, a lack of a comprehensive network vision, results in increased complexity, difficulty managing architecture, distress from dynamic change, a reduction in flexibility, fault persistence, and suffocation of innovation. etc [7][8]. The traditional switch devices are seen in the *Fig. 1.*

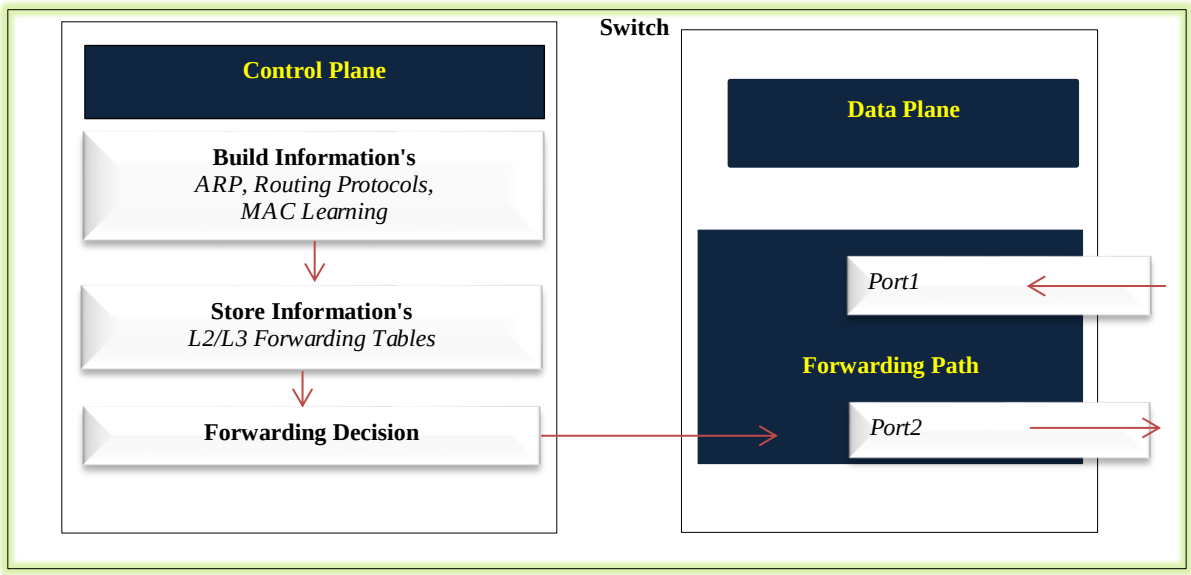

FIG. 1.TRADITIONAL NETWORK DEVICES ARCHITECTURES.

III. ARCHITECTURE OF SOFTWATE DEFINED NETWORKING

As the matter of fact, the traditional network is feeble to face the requirements of today's enterprise network and demanding end-users. The field of Network Management (SDN) stands out as a solution to difficulties with traditional networks. Software Define Networking is transforming networking architecture and providing unprecedented features like programmability, flexibility, stability, and automation by providing programmable controllers.

The idea behind Software Define Networking is to simplify the convoluted network devices by disentangling the control plane from the data plan, where end devices become simple routing devices

Received 22/January/2022; Accepted 13/May/2022

ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.4.14

without complication. This is possible by putting in the network intelligence (control plane) and logically centralized controllers separately. *Fig. 2* clarifies the architecture SDN [9].

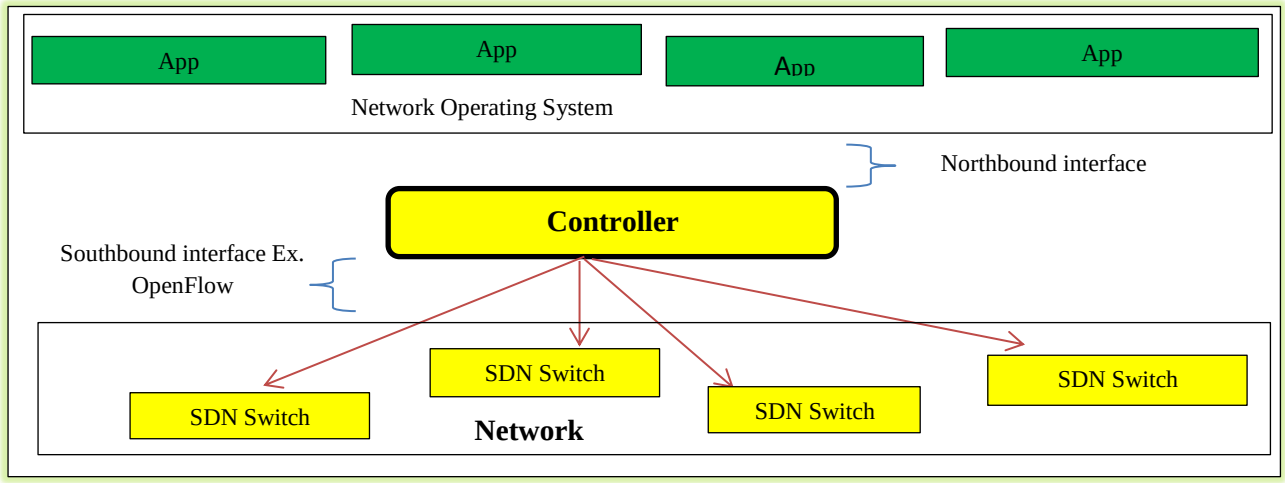

FIG. 2. HIGHER SOFTWARE DEFINE NETWORKING (SDN_HIGH).

The control layer is sometimes positioned as the brain of the network, is located between the data layer which is called (forwarding plane) and the Application plane. It exploits the Application Interfaces (APIs) to control and deal over the data and management plan, the generality well-known like of this southbound API is Open Flow [10],[11] was developed by ONF (Open Network Foundation) [12]. South interface (SI) allows specialized network components to connect with low-level components such as end nodes. It allows the controller to connect with switches and other nodes in order to establish network topology, network flows, and to implement requests delivered to it via north interfaces (NI).

Unlike to the earlier scenario, North interface (NI) provides communication between components at above (higher) level. The services and applications running using the network set up communication via a northbound interface over the controller. The API between the controller and the application provides a platform for the business application to operate without being bound by the implementation specifics.

SDN provides an abstraction for the top layer, allowing researchers and developers to avoid interfering with network element underlying infrastructures. It's a dynamic network architecture that safeguards existing investments while also ensuring the network's long-term viability. SDN also allows for sophisticated orchestration and provisioning solutions to govern the whole network.

There is a various number of Controllers with different platforms are flourishing in last years. Like NOX [13], Ryu [17], Floodlight [15], ONOS [16], POX [14], and so on with different platforms and architecture, different vendors, open-source and commercial, and based on different languages. In this paper, we especially chose Open Daylight and implemented a scenario over that. The following section describes the ODL architecture.

IV. OPENDAYLIGH CONTROLLER ARCHITECTURE

The Linux Foundation [18] manages Open Daylight, a Java-based open-source SDN controller endorsed by over 40 companies. Including IBM, Cisco, Juniper, VMWare, and a number of other significant networking suppliers. The most well-documented controller is Open Daylight. It may be used on any platform that supports Java, including hardware and operating systems. The Open Services Gateway Initiative (OSGI) architecture underpins ODL [19]. The Dynamic Module System for Java, or OSGi, is a standard for developing modular applications.

Open Daylight contains several components and projects and includes three layers. In top layer business and network logic applications reside that use the controller to gather network intelligence, run

Received 22/January/2022; Accepted 13/May/2022
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.4.14

algorithms to control and monitor network behavior. The core is the framework layer where the controller exists. It steads northbound and southbound APIs and unlimbers the application requirement services from the network devices. The controller acts as an abstraction, allowing the developer to concentrate on the application's functionality rather than building device-specific drivers. Physical and forwarded devices make up the bottom layer. *Fig. 3* depicts architecture as consisting mostly of three levels. As shown in *Fig. 3*, the intermediate platform has fundamental network service functions, of which we shall discuss a few in this paper during implementation.

Topology manager is a service for learning the network layout and providing the service to those applications that are looking for network view. Switch manager alongside Topology management are responsible for storing nodes discovered on the physical layer. Forwarding along Topology manager and switch manager provides services for registering and preserving the network flow state.

The southbound interface of OpenDayLight supports several protocols. Such as OpenFlow 1.0, OpenFlow 1.3, BGP-LS, LISP, SNMP [20][21], etc. OpenDayLight is created with the objective of reducing vendor; locking therefore it supports protocols other than Open Flow. The southbound interface is capable of supporting multiple protocols such as OpenFlow and BGP-LS as a separate plug-in. Regardless of the underlying protocol used between the controller and the network devices, the SAL determines how to provide the requested service.

V. SCENARIO IMPLEMENTATION VIA OPENDAYLIGHT (ODL)

Open Daylight (ODL) is a massive platform, includes a large number of plugins and features. Because of this, it is possible to create complexity among new academics. Thus, let's keep things simple and learn step-by-step the procedure of implementation.

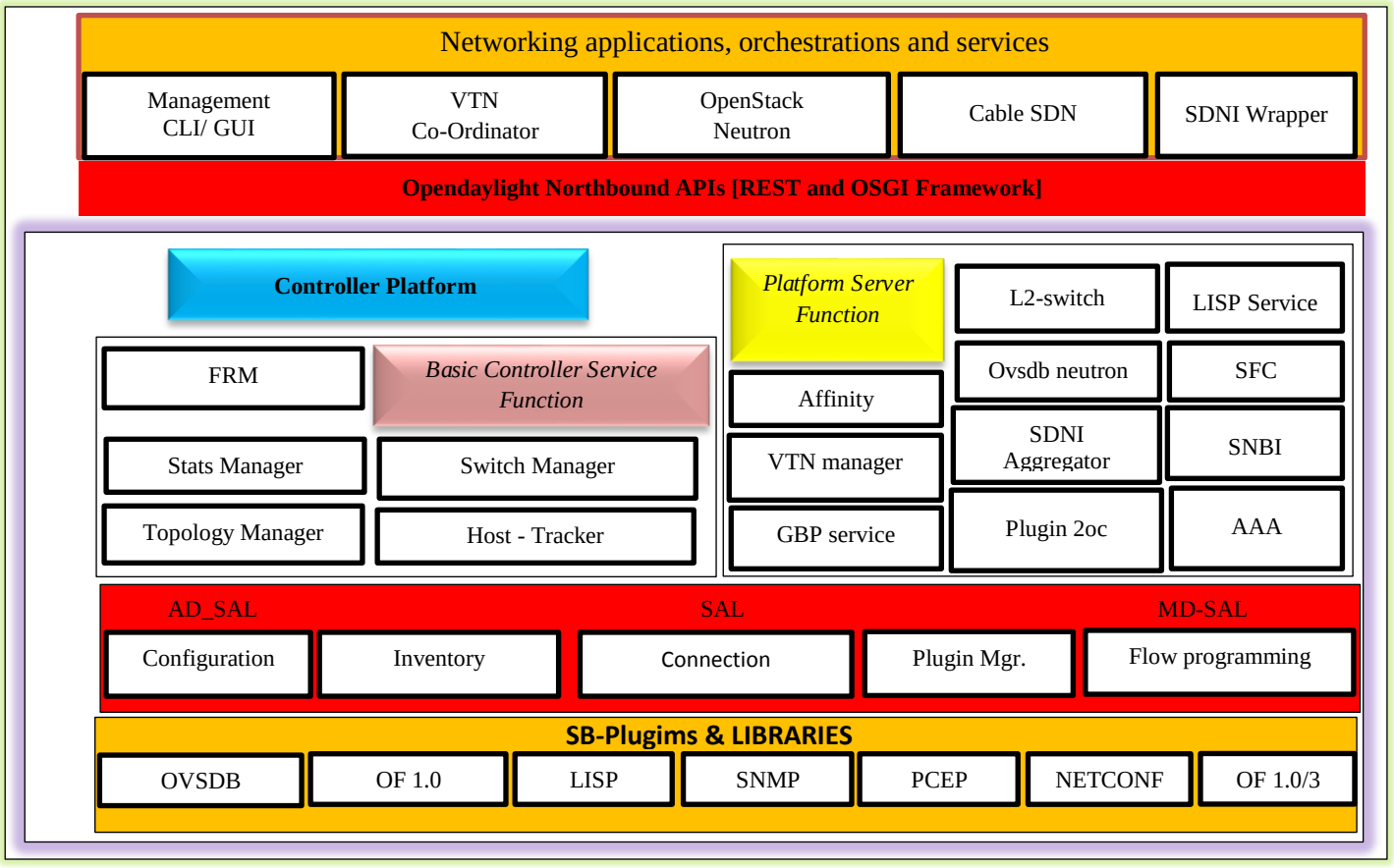

FIG. 3. ODL DETAILED ARCHITECTURE.

Received 22/January/2022; Accepted 13/May/2022

ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.4.14

Step 1 Installation overview: The Karaf distribution is the most convenient way to obtain and install Open DayLight-Beryllium. From Open Daylight's downloads page, we may get the Apache Karaf pre-made package one of two either an x.zip or s.tar file (both with identical tenors). i.e.
https://www.opendaylight.org/downloads, with a NAT connection to the Internet, our new Ubuntu server is up and running. In my example, I am running test1@ubuntu on an eth0 network interface with a DHCP-assigned IP address of 19.16.4.13. This interface will not be configured automatically but manually to get a static IP:

Assign IP to both interfaces public and private for Ubuntu Interface.
eth0 → Nat, public
eht1 → Private

This command can be used to assign the IP to a specific interface through the DHCP service test1@ubuntu:~$ sudo dhclient interface

Step 2: The OpenJDK project is a Java SE Platform implementation that is open-source. This is the default Java version from an Ubuntu repository that is supported. OpenJDK-6 and OpenJDK-7 are the two versions currently available. The Java Runtime Environment must then be installed (JRE). Java 7 or higher JVM versions are required to execute ODL-Be test1@ubuntu:~$ sudo apt-get install openjdk-7-jre

Step 3: Lastly, thus should the JAVA_HOME environment variables editor must be defined which the Apache Karaf server will require. Java comes setup on your computer by default. To maintain access JAVA_HOME variables editor thus should add a JAVA_HOME declaration to our ".profile" file: test1@ubuntu:~$ nano.

profile add blow command to end of this file.
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64 → for 64bit OS
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-i386 → for 32bit OS

Step 4: Download the OpenDaylight Beryllium
test1@USERVER:~$ wget\
https://nexus.opendaylight.org/content/groups/public/org/opendaylight/integration/distribution-karaf/0.4.0-Beryllium/distribution-karaf-0.4.0-Beryllium.tar.gz

Step 5 unzip: We decompress the work in the same place where it is stored ODL-Be:
test1@ubuntu:~$ tar -xvzf distribution- Apache karaf-0.4.0-Beryllium.tar.gz

Step 6 Start the ODL controller: To begin, launch OpenDaylight and the Apache Karaf container in ordin
test1@ubuntu:~$ ./distribution- Apache karaf-0.4.0-Beryllium/bin/ Apache karaf
ODL is now up and running, ready to roll out all of the functionality we need for our scenario., as seen in *Fig. 4*. So, let's get these functionalities installed on ODL-Be. The figure shows that OpenDayLight was successfully installed. We've completed the implementation and are ready to roll out the scenario of our choosing over OpenDayLight. We should investigate a number of elements of OpenDayLight because it offers a large range of options.

Received 22/January/2022; Accepted 13/May/2022
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.4.14

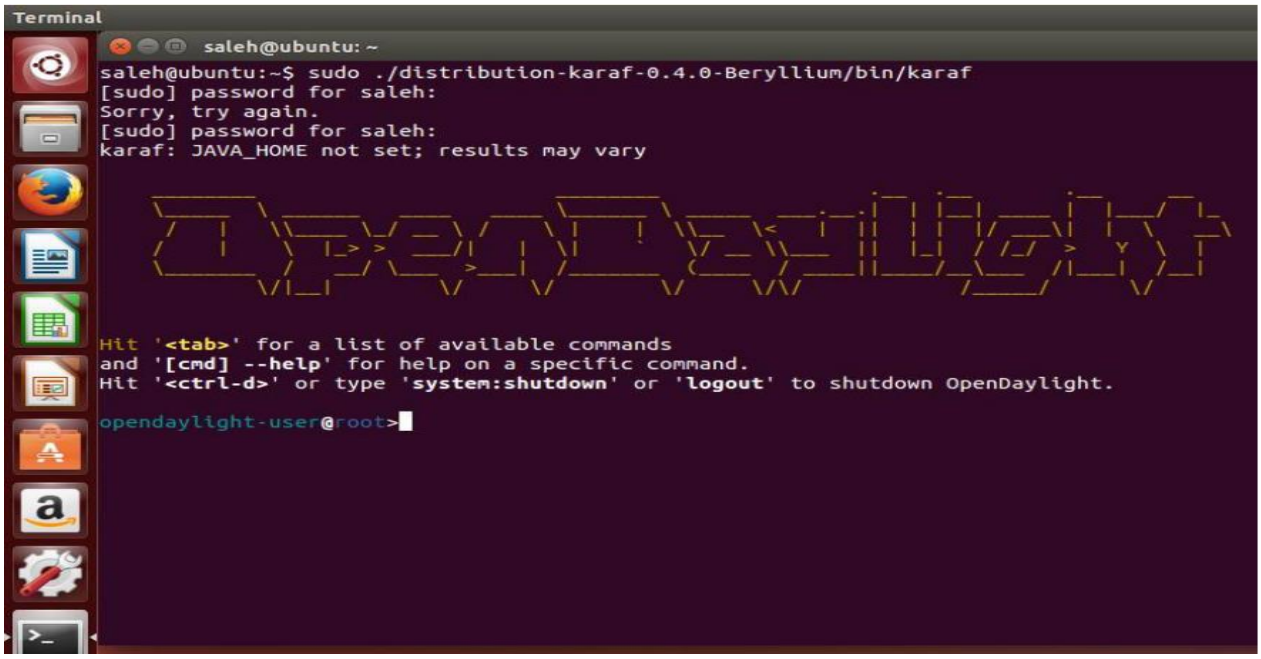

FIG. 4. STARTING ODL CONTROLLER.

In our simple example, we'll employ the following features:

- The OpenFlow southbound plugin communicates with our OpenFlow-enabled switches via the OpenFlow protocol (Version 1.3.x)— Later, we'll pay attention to all switches. There are more southbound components, however, they will not be used in this case.
- The most important component, A layer of Service Abstraction (SAL), will operate as an abstraction layer for any southbound components – in the instance, solely OpenFlow.
- The basic-network-functions bundle, often known as the L2switch bundle, covers things like managing packets, control switches, discovering addresses, and so forth.
- Finally, should be installed the DLUX package, web-based graphical user interface software that allows us to visualize the network topology, including components of networks such as (switches, flows, and hosts), visually. The REST northbound API is used in this software.

ODL-user@root>feature:install odl-l2switch-switch
ODL-user@root>feature:install odl-dlux-all

Step 7 Open ODL through Google Chrome: ODL provide web GUI facility. You may to insert following address to run it.

http:// 19.16.4.13:8181/index.html#/topology

19.16.4.13 (eht0) is the IP of Linux public interface which ODL could run on it.

Received 22/January/2022; Accepted 13/May/2022

ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.4.14

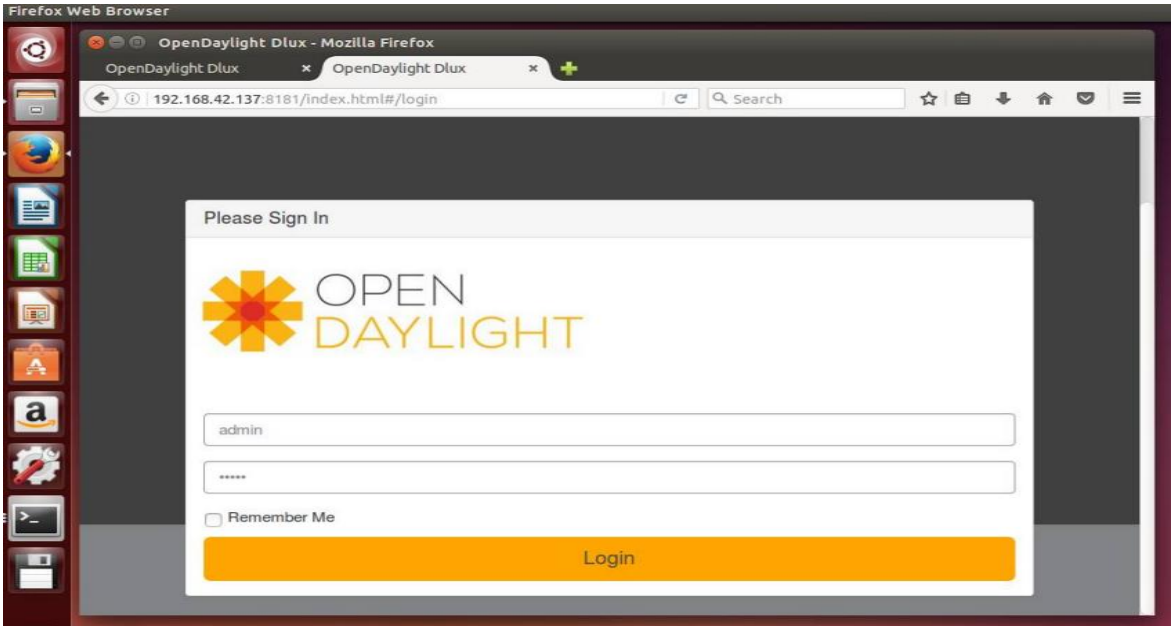

FIG. 5. USING ODL WEB GUI OPTION.

To login, type Admin as the username and password, as illustrated in *Fig. 5*. And now, with the aid of Mininet, we'll put our scenario into action.

Step 8: Connecting our brand new ODL-Be controller to a simple OpenFlow-enabled network is the next step. OpenvSwitch, a free and open-source OpenFlow-enabled virtual switch that we can simply setup on our laptop, is a simple approach to achieve so. Mininet, An even simpler way is to use a fantastic program that allows you to create a virtual network of interconnected It contains all network components on a single machine. So I downloaded the Mininet VM image from their website, configured the network adapter to translate network addresses (NAT), and started running it. Mininet is the username and password by default. We will run the command and check them in *Fig. 6* to create a network topology consisting of four connected OpenFlow1.3 enabled switches, each with a single host, and attach it to our ODL-Be SDN controller.

MININET

Using the sudo mn command, we activate the control and define the network configuration, as well as run ovsk plus protocol OpenFlow like command:

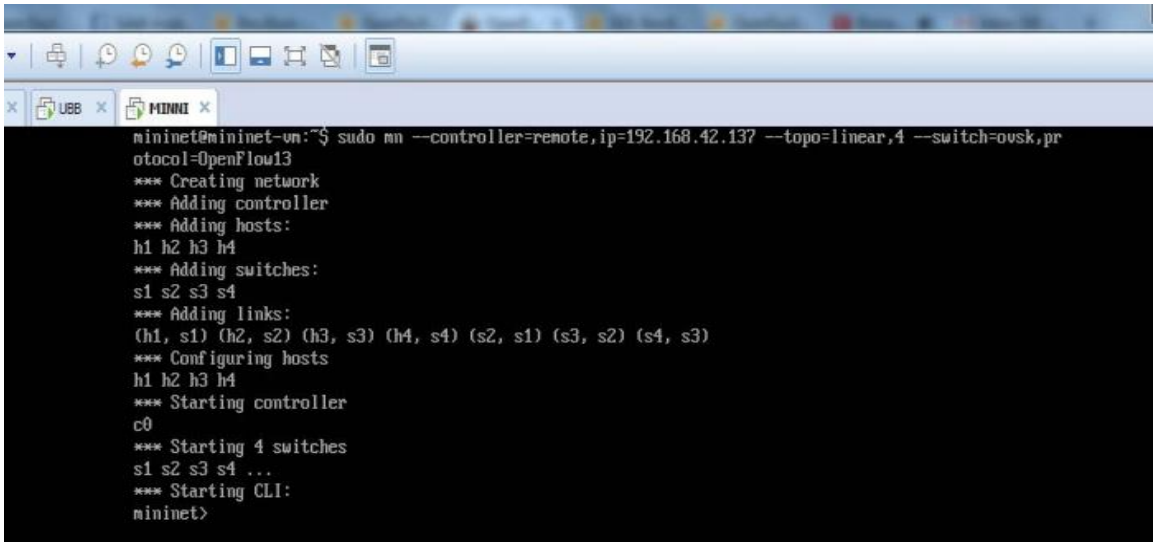

FIG. 6. STARTING THE NETWORK SCENARIO OVER MININET.

Received 22/January/2022; Accepted 13/May/2022

ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.4.14

After logging in, we can see our SDN network architecture, as illustrated in *Fig. 7*.

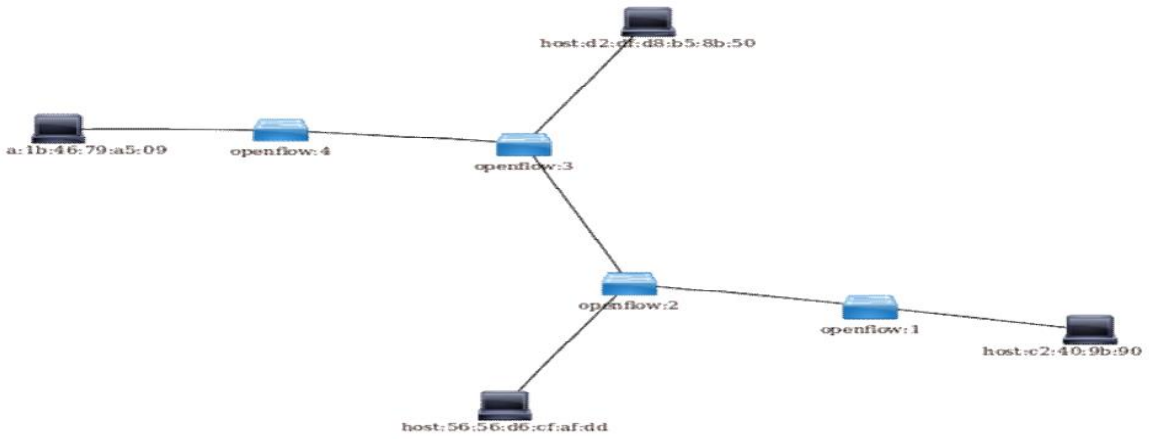

FIG. 7. OBSERVING IMPLEMENTED SCENARIO IN ODL WEB GUI OPTION.

Lastly, we start it up by displaying the screen displayed in *Fig. 8.* You may rejoice at the successful implementation of ODL's controller, which is now ready to implement any scenario selected by the user. Please keep in mind that OpenDayLight is a Linux-based operating system. Beryllium-SR4 is an OpenDayLight version that was published on October 26, 2016, and includes a number of features such as authentication, BGP, BMP, DIDM, Centinel, L2 Switch, NetIDE, Repository of Time Series Data (TSDR), and more.

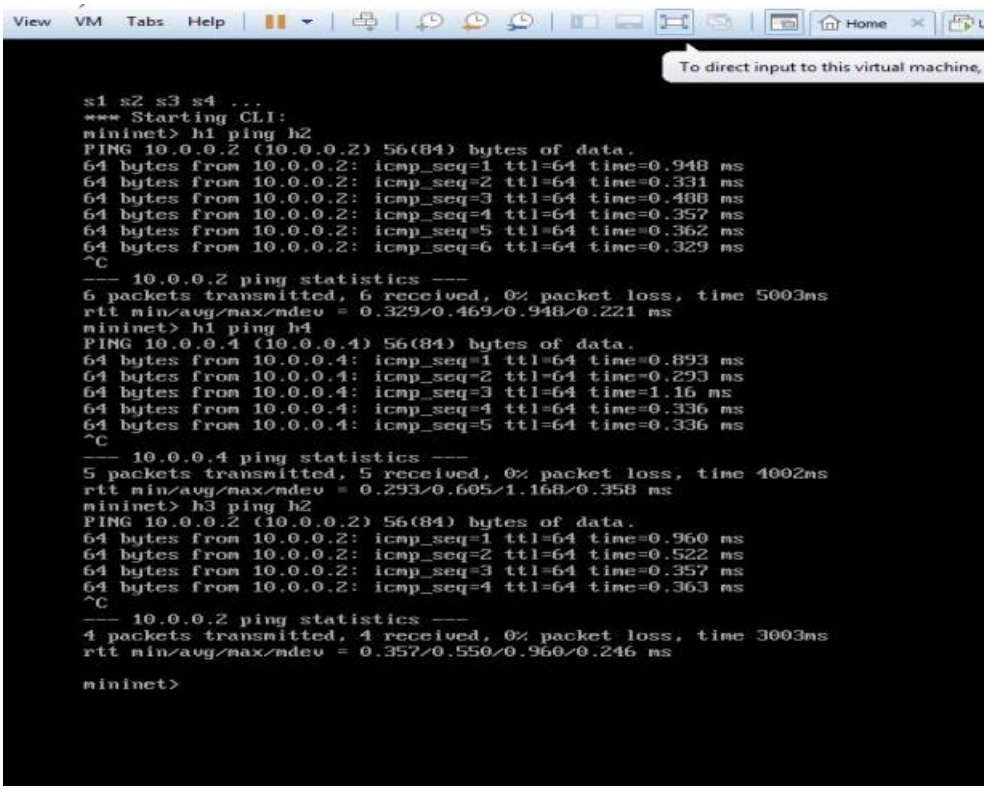

FIG. 8. CONNECTIVITY ALL THE HOST.

Received 22/January/2022; Accepted 13/May/2022

ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.4.14

VI. NODE TO NODE PERFORMANCE EVALUATION THROUGH SDN CONTROLLERS

SDN (Software Defined Networking) heralds a new era in the evolution of business on the Internet. The data plane and the control plane are separated in the SDN model. It uses controllers, which are responsible for central control of several network devices at the same time. We observe a performance test in three open source SDN consoles based on the OpenFlow protocol. The performance of the Floodlight [22], OpenDayLight (ODL) and Ryu controllers is specifically checked for latency [23], throughput, scalability, manual configuration, debugging, security, navigation, etc [24]. The goal is to evaluate performance metrics such as bandwidth, throughput, round-trip time, instability, and packet loss. In order to achieve this, the Clench tool is used in an environment emulated with Mininet [25]. The results showed that the Ryu controller had the lowest performance across all parameters tested; ODL has lower latency and floodlight has higher throughput [26][27]. In terms of scalability, we conclude that Floodlight is suitable for use in small networks, while ODL is suitable for use in dense networks.With the RYU controller, there is no loss between nodes [28].

VII. CONCLUSIONS

OpenDayLight (ODL) is one of the outstanding frameworks for implementing SDN, and it is discussed in depth in this article. We have provided a complete step-by-step implementation of SDN-based architecture in this article, which researchers may use to create their desired topological situation from basic to complicated and undertake additional performance assessment if they follow the methods outlined in section V. This work is an attempt to practically bring forth a solution to solve the concerns of a researcher who has a good understanding of SDN but is unable to put it into practice. For them, this paper will motivate them to take the next step and adopt experimentation more fully. This study will open doors for researchers who want to work on security, scalability, efficiency, congestion, speed, configuration, protocol, and other topics by using the ODL framework offered in this paper to conduct their experiments.

REFERENCES

[1] P. Wei Tsai, C. Wei Tsai, C. Wei Hsu, & C. Sing Yang, (2018). Network monitoring in software-defined networking: A review. IEEE Systems Journal, 12(4), 3958-3969.

[2] T. Bakhshi, (2017). State of the art and recent research advances in software defined networking. Wireless Communications and Mobile Computing, 2017.

[3] Z. K. Khattak , M. Awais , A. Iqbal "Performance Evaluation of OpenDaylight SDNController," in Procceding of the 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS), 16-19 Dec. 2014. DOI: 10.1109/PADSW.2014.7097868.

[4] E. R. Jimson, K. Nisar, & M. H. A. Hijazi, (2019). The state of the art of software defined networking (SDN) issues in current network architecture and a solution for network management using the SDN. International Journal of Technology Diffusion (IJTD), 10(3), 33-48.

[5] S. Sahni & K. Suk Kim. (2018). IP Router Tables. In Handbook of Data Structures and Applications (pp. 765-781). Chapman and Hall/CRC.

[6] F .Wu, Y Geng, X .Tian. (2021, May). RXstp: A topology discovery mechanism based on rapid spanning tree for SDN in-band control. In 2021 International Conference on Communications, Information System and Computer Engineering (CISCE) (pp. 703-706). IEEE.

[7] T. Benson, A. Akella, and D. Maltz, "Unraveling the complexity of network management," in Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, ser. NSDI'09,Berkeley, CA, USA, 2009, pp. 335–348.

[8] M., Mohamad Alsaeedi, M. Murtadha Mohamad, &, A. Al-Roubaiey (2019). Toward adaptive and scalable OpenFlow-SDN flow control: A survey. IEEE Access, 7, 107346-107379.

[9] M.K. Shin, K.H. Nam, H. J. Kim, "Software-Defined Networking (SDN): A Reference Architecture and Open APIs," In Proceedings of International Conference on ICT Convergence (ICTC), pp.360–361, Oct. 2012.

Received 22/January/2022; Accepted 13/May/2022
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.4.14

[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, Mar. 2008.

[11] ONF, "Open networking foundation," 2014. [Online]. Available: https://www.opennetworking.org.

[12] ONF, "OpenFlow management and configuration protocol (OF-Config 1.1.1)," March 2014. [Online]. Available:https://www.opennetworking.org/images/stories/downloads /sdnresources/onf-specifications/openflow-config/of- config-1-1-1.pdf.

[13] NOX detailed implementation, available online: http://www.noxrepo.org.

[14] POX detailed implementation, available online: http://www.noxrepo.org.

[15] Floodlight detailed implementation, available online: http://floodlight.openflowhub.org.

[16] O. Salman, I.H. Elhajj, A. Kayssi, A. Chehab "SDN Controllers: A Comparative Study". Mediterranean Electro technical Conference MELECON 2016, Limassol, Cyprus, 2016.

[17] Rya detailed implementation, available online: http://ryu.readthedocs.io/en/latest/getting_started.html#what-s- ryu.

[18] Linux Foundation, "Open platform for NFV," https://www.opnfv.org,Sep 2014.

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

[20] D. Harrington, R. Presuhn, and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," Internet Engineering Task Force, dec 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3411.txt.

[21] R. I. Renaldo, (2018). Analisis Simple Network Management Protocol (SNMP) Menggunakan Wireshark dan Visualisasi Traffic Data Menggunakan Orange. Analisis Simple Network Management Protocol (SNMP) Menggunakan Wireshark dan Visualisasi Traffic Data Menggunakan Orange.

[22] J. P. Duque, D. D. Beltrán & G. P. Leguizamón, (2018). OpenDaylight vs. floodlight: Comparative analysis of a load balancing algorithm for software defined networking. International Journal of Communication Networks and Information Security, 10(2), 348-357.

[23] Y. Li, X. Guo, X. Pang, B. Peng, X. Li, & P. Zhang, (2020, August). Performance analysis of floodlight and Ryu SDN controllers under mininet simulator. In 2020 IEEE/CIC International Conference on Communications in China (ICCC Workshops) (pp. 85-90). IEEE.

[24] M. Islam, N. Islam & M. Refat, (2020). Node to node performance evaluation through RYU SDN controller. Wireless Personal Communications, 112(1), 555-570.

[25] M. N. A. Sheikh, (2019). SDN-Based approach to evaluate the best controller: internal controller NOX and external controllers POX, ONOS, RYU. Global Journal of Computer Science and Technology.

[26] K. Smida, H. Tounsi, M. Frikha, & Y. Q. Song, (2020, October). Efficient SDN Controller for Safety Applications in SDN-Based Vehicular Networks: POX, Floodlight, ONOS or OpenDaylight?. In 2020 IEEE Eighth International Conference on Communications and Networking (ComNet) (pp. 1-6). IEEE.

[27] M. Latah & L. Toker, (2020). Load and stress testing for SDN's northbound API. SN Applied Sciences, 2(1), 1-8.

[28] R. K. Chouhan, M. Atulkar, & N. K. Nagwani, (2019, March). Performance comparison of Ryu and floodlight controllers in different SDN topologies. In 2019 1st International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE) (pp. 188-191). IEEE.

Received 22/January/2022; Accepted 13/May/2022
 ISSN (Print) 1811-9212 *ISSN (Online) 2617-3352*