Bulletin of Computer Science and Electrical Engineering
Vol. 2, No. 2, December 2021, pp. 56~66
ISSN: 2722-7324, DOI: 10.25008/bcsee.v2i2.1145

Design and Implementation of SDN IP Based on Open Network Operating System and Border Gateway Protocol

I Putu Agus Eka Pratama[1]
[1]Department of Information Technology, Faculty of Engineering, Udayana University, Bali, Indonesia

Article Info

Article history:

Received Jun 06, 2021
Revised Dec 24, 2021
Accepted Dec 29, 2021

Keywords:

Autonomous System (AS)
Border Gateway Protocol (BGP)
Open Network Operating System (ONOS)
Software Defined Networking (SDN)
SDN IP

ABSTRACT

The development of computer network technology in the form of Software Defined Networking (SDN), provides many conveniences for users to be able to develop network control applications, which can separate the functions of the data field from the control field. This separation of routers and switches makes it easier for developers to develop software and devices centrally according to user requirements. However, there are obstacles to implementing SDN on IP networks in a short time. It is necessary to implement SDN in stages by adding SDN to the IP network in the form of SDN IP and learning by SDN IP on existing routes and additional routes or new routes so that SDN can connect and exchange routing information autonomously. This research focuses on the design and implementation of SDN IP using the Open Network Operating System (ONOS) on the Border Gateway Protocol (BGP) along with route learning in it. The results show that the design and implementation of SDN IP based on ONOS and BGP can be done well, where SDN can connect and exchange routing information with the Autonomous System (AS) network based on BGP and SDN IP can learn the additional routes using Intents API on ONOS.

Corresponding Author:

I Putu Agus Eka Pratama
Department of Information Technology,
Faculty of Engineering, Udayana University,
Jl. Raya Kampus Unud, Jimbaran, Badung, Bali, Indonesia.
Email: eka.pratama@unud.ac.id

1. INTRODUCTION

Computer networks have experienced rapid development from time to time, one of which is the Software-Defined Network (SDN). SDN is a new approach to computer networks where design, implement and manage networks separated into control plane (as a network control) and data plane (as the forwarding process) to increase user experience and management at the network [1][2]. From this definition, SDN provides convenience for users and network managers, to be able to develop network controller applications.

This is possible because SDN is able to separate the data plane function from the control plane. There is a separation between the data plane and the control plane on computer network devices (for example on Routers and Switches), making the application and device development process can be done centrally. An illustration of SDN is shown in Figure 1 below:

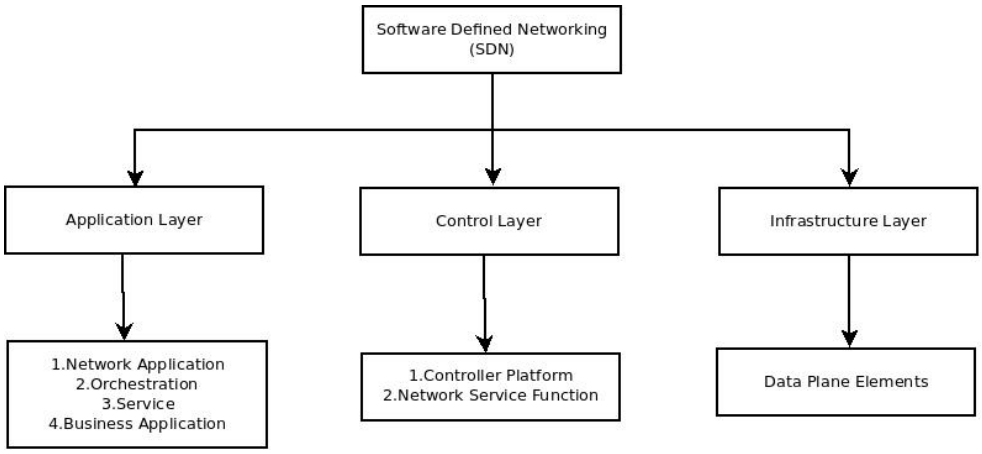

Figure 1. Software-Defined Networking (SDN)

However, there are obstacles in implementing SDN on existing Internet Protocol (IP) based networks. The obstacle is that the SDN network is difficult to connect to the existing IP network, to be able to communicate, exchange data, and exchange routing information automatically using the Autonomous System (AS). The process of communication, data exchange, and information exchange generally also involves the Border Gateway Protocol (BGP) which manages these processes on the network. There needs to be a system that can help SDN connect to the IP network while studying the routes in it.

Based on this problems, this research proposes a solution in the form of design and implementation of SDN IP along with the ability to learn about the existing routes, both existing routes and additional routes. The SDN IP proposed in this study was built using the Open Network Operating System (ONOS) and the Border Gateway Protocol (BGP) on the Autonomous System (AS). The formulation of the problem in this research is stated in the research questions as follows: 1.) How to design and implement SDN IP using ONOS and BGP? 2.) How to test the system built to find out route learning by SDN IP? The purpose of this research is to find out how to design and implement SDN IP based on ONOS and BGP, to overcome problems that occur in terms of connectivity through learning routes that exist in IP networks.

As a state of the art, there are twenty-five previous researches related to this research, which discuss about SDN, BGP, OpenFlow, and Autonomous Systems (AS). Mousa, et al., in their research paper, introduce the concepts and applications of SDN, with a focus on the open research challenges in it [3]. Lukas and Windha in their publication, presenting SDN simulation results by involving BGP in it, to be able to find out how BGP works in the selection of routing paths as well as the possibility of broken links and their impact on the performance of BGP on the Autonomous System in SDN [4]. Kaur et al., describe in their paper propose an architecture named SD-IoAV for the integration of SDN with IoAV, which SD-IoAV is geographically dispersed by nature, by deploying multiple SDN controllers across the widely dispersed SDN domains using Controller Placement Problem (CPP) in the context of SD-IoAV for energy minimization and load balancing [5] In the aiming at the authentication and security issues of SDN architecture in autonomous decentralized system (ADS) applications, Zhou, et al., introduce the secure mutual trust among the autonomous controllers, by combining both trusted technology and SDN architecture, and introduce an authentication protocol based on SDN architecture without any trusted third party between trusted domains in Autonomous Systems [6]. Monika, et al., in their research examines the impact of using Intent Monitor and Reroute (IMR) with a custom topology on ONOS, to find the best scenario by performing traffic management on a data plane, with the results obtained are IMR able to optimize the use of each link and maximize bandwidth usage in a network when distributing data and following TIPHON standards [7]. Pramudita and Suardana in their research describe the comparison between the Opendaylight controller (ODL) and Ryu on the Software-Defined Network (SDN), with test results showing that the Ryu controller performance is better than ODL with an average throughput value of 325,682 Mb/s, the average value of delay is 0.313395 s, and

the average value of Packet loss is 4.59%, in 10 tests using a traffic load variation of 100Mb-10Gb [8].

Nugroho utilizes the Route Flow controller, Djikstra routing algorithm, Full-Mesh topology, that provides full connections to all switches, TCP and UDP are used to transmit data in SDN networks, with the result of network performance measurements show that the use of the UDP obtains better values in delay, jitter, and throughput than TCP [9]. In his paper, Goswami is contributing to the evaluation of the performance and the scalability of the ONOS controller by taking many scenarios which are experimented on Mininet, ONOS Controller, Docker, and iPerf [10]. Kartadie and Panggayuh in their research, present a test on Floodlight and ONOS controllers, where the test results show that the Floodlight controller is more stable than ONOS in terms of overcoming switch and host loads [11]. Pratama and Wikantyasa on their research, implementing and analyzing the simulation of Quality of Service (QoS) and performance device OpenFlow switch using ONOS as controller, to monitor the device performance and iperf3 as a Quality of Service (QoS) testing [12] Halomoan, et al., explained in their research about the operational Open Flow controller using the Open Network Operating System (ONOS) [13]. In her research, Annisa offers a solution for the development of SDN-based network management at Sekayu Polytechnic using the concept of network programming, hardware efficiency, and centralized control, with the results of research in the form of Quality of Service (QoS) of 18 ms on SDN networks and 37,7 ms on a conventional network [14]. Friyanto on his paper, analyze the high availability aspects of SDN IP reactive routing, by making multiple ONOS controllers in one ONOS cluster system, then the testing process is carried out on aspects of BGP speakers, ONOS controller, and links between the components [15]. Purwiadi, Yahya, and Basuki in their research publication present a proposed solution in the form of a mechanism to maintain the availability of a controller called the High Availability Controller, which is carried out using the Heartbeat and Distributed Replication Block Device (DRBD) based failover mechanism, where the test results show the average value of POX controller downtime during the failover process of 23 seconds and 59.6 seconds during the failback process [16].

Xia, et al., in their paper, surveys the latest developments in this active research area of SDN, including a generally accepted definition for SDN with the two characteristic features and potential benefits of SDN, three-layer architecture (infrastructure layer, control layer, application layer), de facto SDN implementation (Open Flow), and some of the open research challenges [17]. Rizky, et al, in their paper, review some literature related to routing optimization methods on networks, using Software Defined Networking-Wide Area Network (SDN-WAN) [18]. In his research, Adrian focuses on OSPF implementation and QoS performance analysis on SDN networks, using cost settings and without cost settings, to obtain the best routing configuration on SDN networks using OSPF [19]. Jamal and Purnomowati compared the performance of SDN networks with OpenDayLight controllers, against traditional architectures (TCP / IP) using Mininet emulator for SDN architecture, CORE emulator for traditional architecture (TCP / IP), and iPerf for generating traffic on the network, with parameters in the form of latency, throughput and packet loss, where the results show that the performance of latency, throughput, and packet-loss has a better value on the traditional architecture than on the SDN architecture, with Mesh Topology having the best value in all tests [20]. The research from Varadharajan has proposed a policy based security architecture for securing the communication in multiple Autonomous System (AS) domains with Software Defined Networks (SDN), to enable end-to-end secure communication within a single AS domain and for multiple AS domains using ONOS controller as a model [21]. Ghivani in his research conducted tests that focused on the analysis of the routing performance of BGP and EIGRP in terms of Quality of Service (QoS) in the form of throughput and delay, where the test results show that EIGRP routing has a greater throughput than BGP and more delay. smaller than BGP [22]. Baggan and Panda in their paper discuss the literature review, issues, and challenges for existing Network Path Restoration Mechanisms, Border Gateway Protocol, Multi-Protocol Label Switching, Software Defined Networks and emulated the topology in GNS3 Using the BGP-MPLS [23]. Also, Baggan and Panda in their other paper explored various dimensions of Border Gateway Routing Protocol (BGP) as de-facto of Routing Protocol for inter Autonomous

Systems (AS) [24]. The last, Jingjing, et al., in their publication paper, learned about the distributed characteristics of Kandoo architecture, improved, and optimized it based on Routing Control Platform (RCP), and analyzed the deployment strategies of BGP and OSPF protocol in a distributed control plane of SDN, in which the simulation results show that the deployment strategies are superior to the traditional routing strategies [25].

Two research questions must be answered at the conclusion, namely: 1.) How is the technical implementation of SDN IP using ONOS on BGP? 2.) How technical is SDN IP testing to see if it is running well to see connections between networks and how to SDN IP to learn additional routes?

2. RESEARCH METHOD

This research using Experimental Methodology, which consists of some steps of implementation and analysis of the result, including literature review, design of a solution, testing scenario, implementation, testing, analyzing, discussion, documentation, and publication [26]. According to the Experimental Methodology, the sequence of steps in this research starting from the background and formulation of the problem, then carried out literature review from paper (journal, proceeding) and state of the art.

After that, design of a solution and testing scenario were determined, followed by implementation and testing using the open-source software ONOS at Ubuntu Linux [27] based on BGP with some of configurations in it [28] to implement SDN and SDN IP [29]. The test results are then observed, discussed, conclusion, and ended with documentation and research publications (paper). The flowchart of the research process is shown in Figure 2. below:

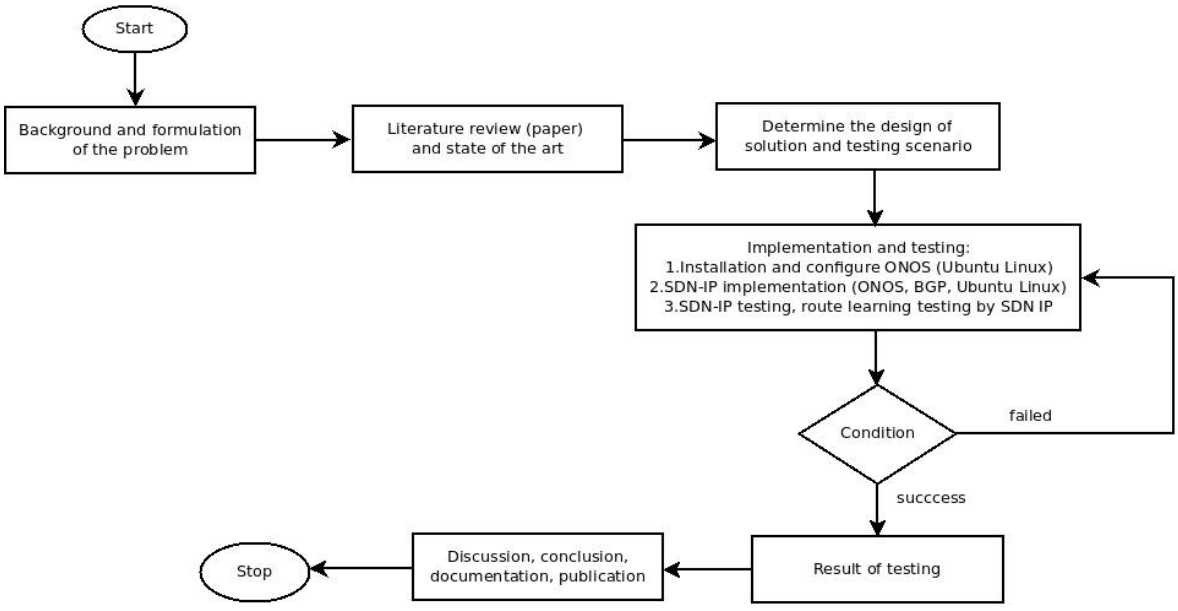

Figure 2. Flowchart of the research process

The SDN IP proposed in this research is a part of the ONOS open source project to connect the SDN network to an external network via BGP, where SDN acts as an Autonomous System that integrates ONOS with BGP. SDN IP acts as an Autonomous System that connects various other IP-based networks. In SDN and SDN IP networks, there are one or more BGP speakers in the form of BGP routers, which play a role in the exchange and dissemination of information. The SDN IP architecture is shown in Figure 3. below:

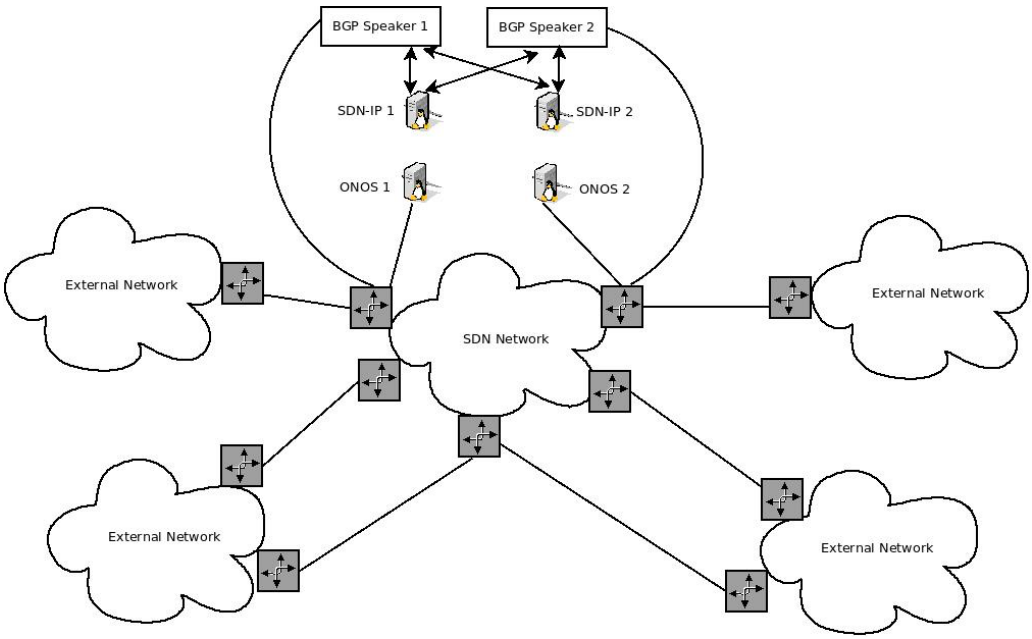

Figure 3. SDN IP architecture

The test scenario between the user (researcher) and the system (Ubuntu Linux with ONOS and SDN IP in it), both connected to the network, the user accessing ONOS and SDN IP remotely, is shown in Figure 4. below:

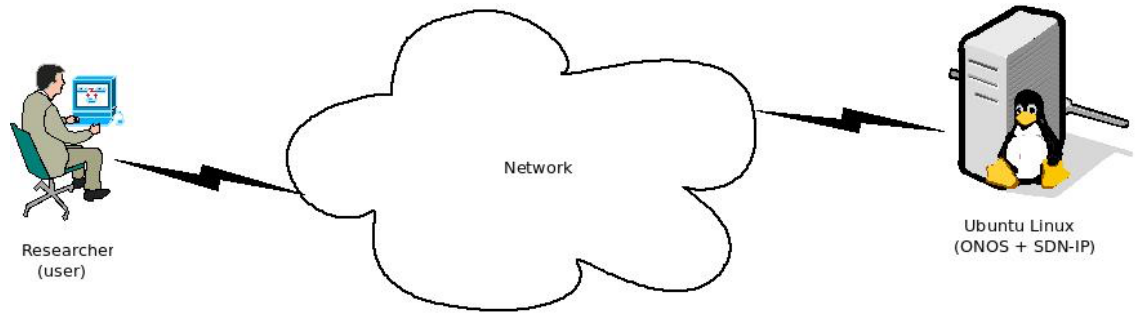

Figure 4. Testing scenario of SDN IP

In this research, several supporting hardware and software are needed for the implementation and testing. The hardware used is Dell Latitude E6440 with specification: Intel Core i7-4610M (4) @ 3.700GHz, 16GB of RAM. The software used is Ubuntu Linux, VirtualBox, ONOS (GUI and CLI based), and Mininet. Internet connection also including in this research.

SDN IP is implemented by first running ONOS CLI through Virtual Box on Ubuntu Linux, then logging into the system, using the sdnip User ID. ONOS provides three menus to be used in this research, namely: ONOS GUI, ONOS CLI, and SDN IP Mininet. In this implementation, six interconnected devices are used, which are shown in Figure 5 below:

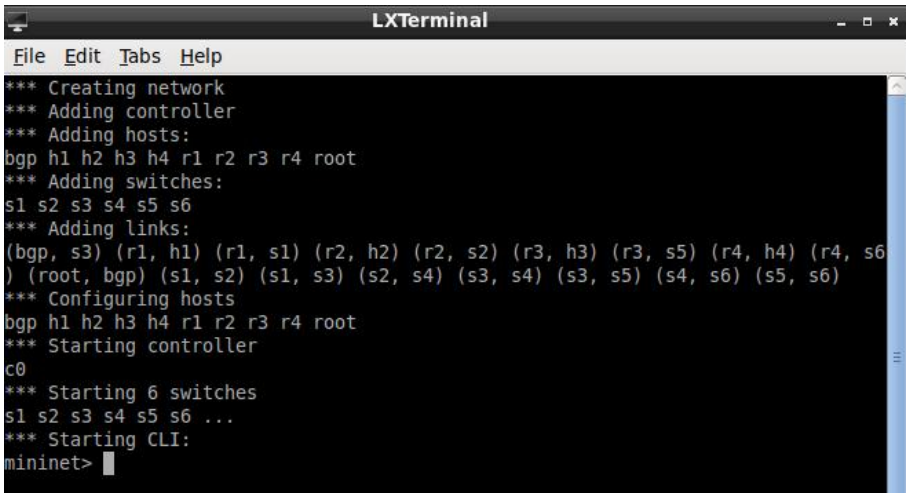

Figure 5. SDN IP Mininet

Based on Figure 5. above, the SDN IP topology development stages are carried out with a number of processes in them. The processes carried out started with creating a network, then continued with adding controller, then continued with adding hosts BGP which included h1, h2, h3, and h4 on root r1, r,2, r3, and r4. Then proceed with adding switches that include s1, s2, s3, s4, s5, and s6. After all processes are completed, then 17 links are added to the SDN IP topology, which include (BGP, s3), (r1, h1), (r1, s1), (r2, h2), (r2, s2), (r3, h3), (r3, s5), (r4, h4), (r4, s6), (root, BGP), (s1, s2), (s1, s3), (s2, s4), (s3, s4), (s3, s5), (s4, s6), and (s5, s6).

Finally, configuring hosts on BGP, h1, h2, h3, h4, r1, r,2, r3, and r4, as well as starting controller c0 and starting the six switches s1, s2, s3, s4, s5, and s6. The CLI display of the SDN IP topology above for the six connected devices can be displayed in GUI form via the ONOS GUI, as shown in Figure 6. below:

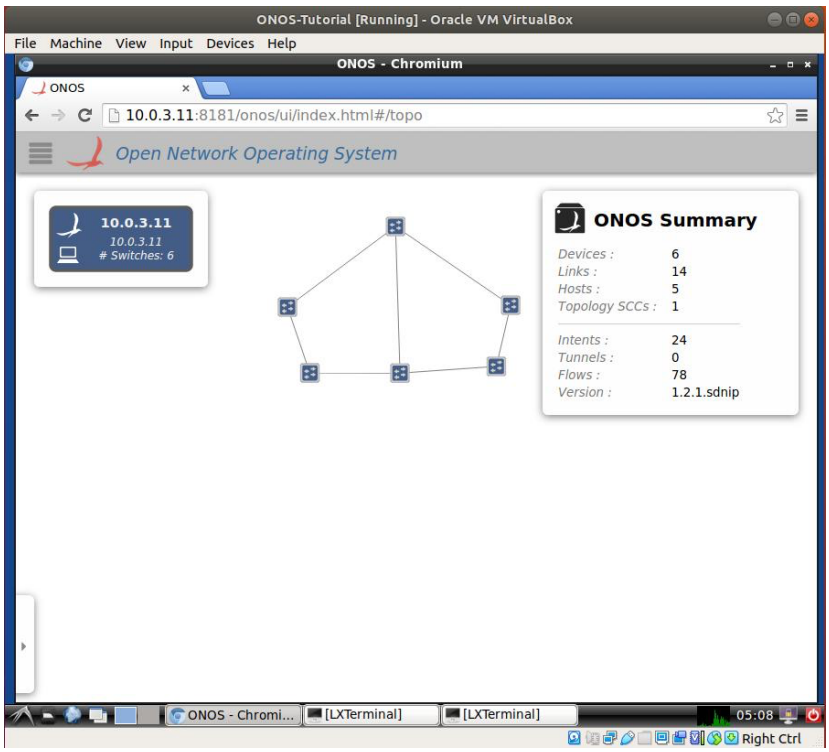

Figure 6. SDN IP Mininet

3. TESTING, RESULTS AND DISCUSSION

3.1. Testing Phase and Result

There are eight phases of testing carried out in this research, from view the configuration of the host used to run SDN IP on host h1 using the command h1 ip addr show until view four routes on SDN IP implemented on ONOS using the command routes. The eight test phases along with the command and their results, are shown in Table 1. below:

Table 1. Testing phase and result

No	Testing Phase	Result
1	View the configuration of the host used to run SDN IP on host h1 using the command h1 ip addr show	Visible IP subnet of host h1, different subnet for each host
2	View networks that have been connected to ONOS and networks that will be connected to ONOS using the command devices	There are six networks that have different IDs
3	Running SDN IP by pinging hosts h1 and h2 using the command h1 ping h2	The results obtained are ping process between hosts h1 and h2 successful
4	Running SDN IP by pinging hosts h1 and h3 using the command h1 ping h3	The results obtained are pinged between hosts h1 and h3 succcessful
5	Running SDN IP by pinging hosts h2 and h3 using the command h2 ping h3	The results obtained are pinged between hosts h2 and h3 successful
6	Running SDN IP by pinging hosts h1 and h4 using the command h1 ping h4	The results obtained are pinged between hosts h1 and h4 successful
7	View Intents API on ONOS using the command intents -s	ONOS can display a list of all Intent APIs used to learn the route to the switch
8	View 4 routes on SDN IP implemented on ONOS using the command routes	The 4 routes can be displayed along with the network, subnet, and next-hop

Based on Table 1. above, the first testing phase is to see the host configuration used to run SDN IP on host h1 using the h1 ip addr show command, with the results obtained are the visible IP subnet of host h1 with a different subnet for each host. In this case, information on the IP Address range and subnet are obtained, namely, the inet line 192.168.1.1/24 brd 192.168.1.255 scope global h1-eth0, as shown in Figure 7. below:

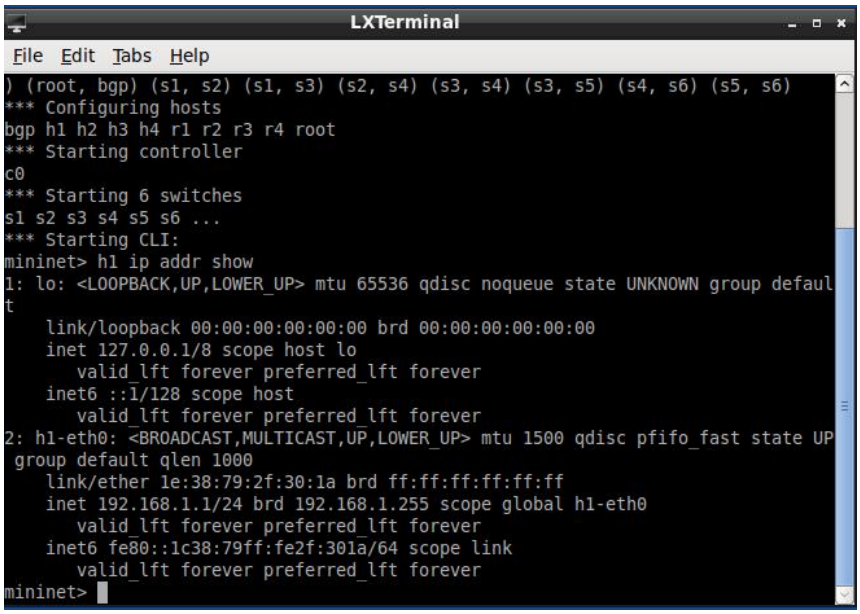

Figure 7. View the configuration of the host used to run SDN IP on host h1 (h1 ip addr show)

The second testing stage is to see the network that has been connected to ONOS and the network that will be connected to ONOS, using the devices command. The results of the devices command show that there are six networks with IDs 00000000000000a1 00000000000000a2, 00000000000000a3, 00000000000000a4, 00000000000000a5, and 00000000000000a6, as shown in Figure 8. below:

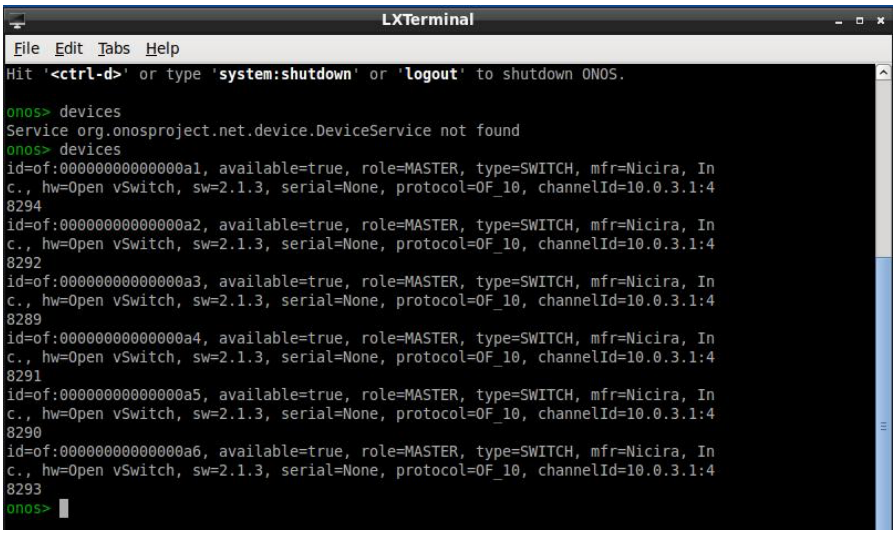

Figure 8. View networks that have been connected to ONOS

The third testing phase is running SDN-IP by pinging hosts h1 and h2 using the command h1 ping h2. The results obtained are ping process between hosts h1 and h2 successful. The fourth testing phase, the fifth testing phase, and the sixth testing phase, also perform SDN-IP by pinging between two hosts using the ping command. In the fourth testing phase, run SDN IP by pinging hosts h1 and h3 using the h1 ping h3 command, with the result that the ping between hosts h1 and h3 is successful. In the fifth test stage, running SDN IP by pinging hosts h2 and h3 using the h2 ping h3 command, with the results obtained that the pinging between hosts h2 and h3 is successful. In the sixth testing stage, running SDN IP by pinging hosts h1 and h4 using the h1 ping h4 command, with the results obtained that the ping between hosts h1 and h4 was successful. The display of some test results is shown in the Figure 9. below:

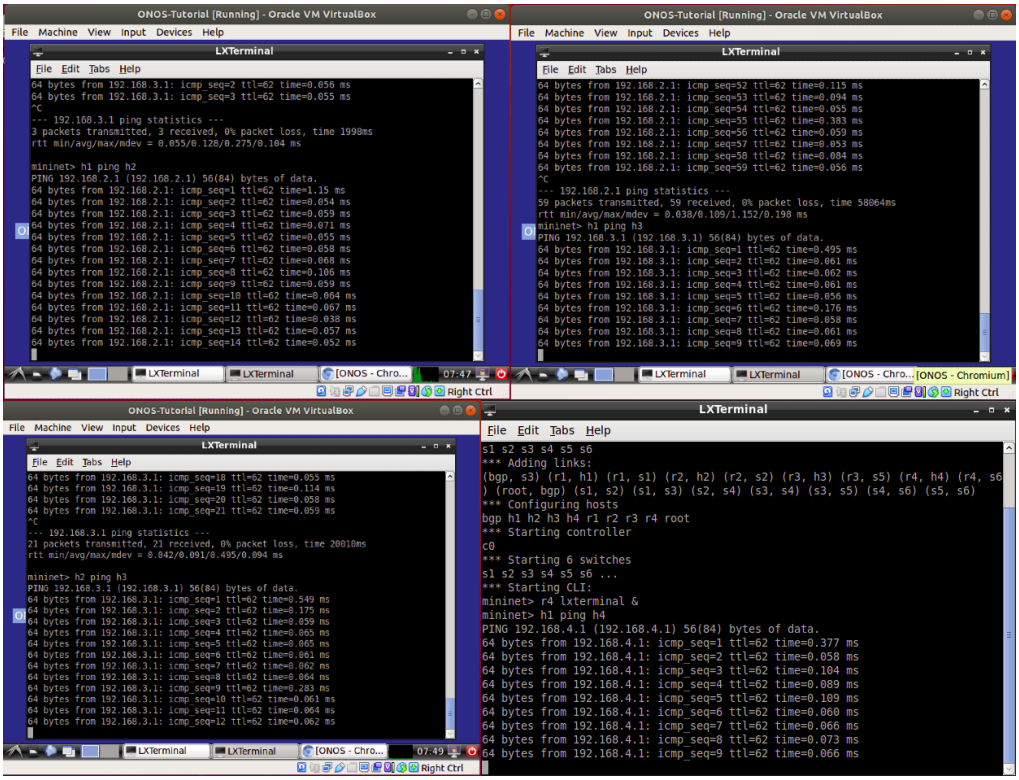

Figure 9. Running SDN IP by pinging between hosts (h1 and h2, h1 and h3, h2 and h3, h1 and h4)

In the seventh stage of testing, the Intents API display on ONOS is performed using the intent -s command, whereas a result ONOS can display a list of all API Intents used to learn the route to the switch. From the list displayed by the Intent API, there is an additional entry in MultiPointToSinglePoint (total and installed) which in this case acts as a route, from the previous three to four with the addition of one additional route learned by SDN IP through the Intents API. Figure 10. below shows the Intents API:

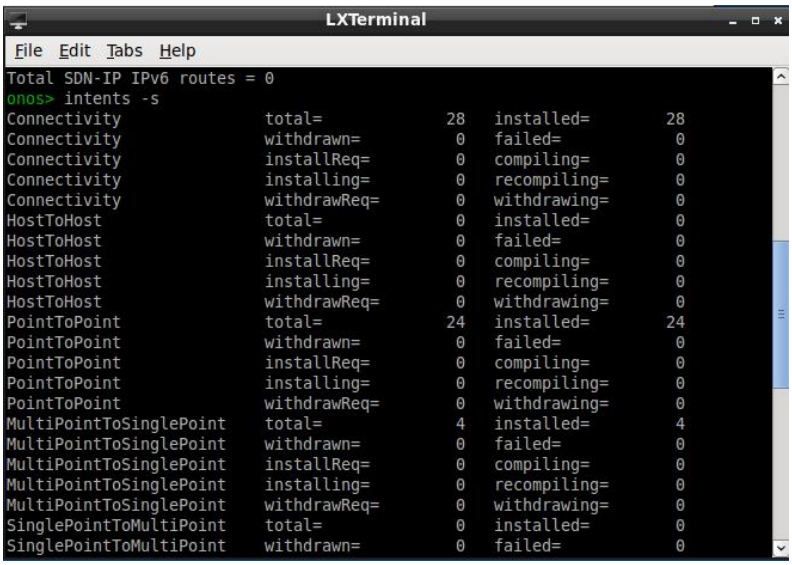

Figure 10. ONOS Intents API list

In the last test, we examined four routes on SDN IP implemented on ONOS using the route command, where the result is that ONOS can display a list of the four routes that SDN IP successfully learned including network, subnet, and next-hop.

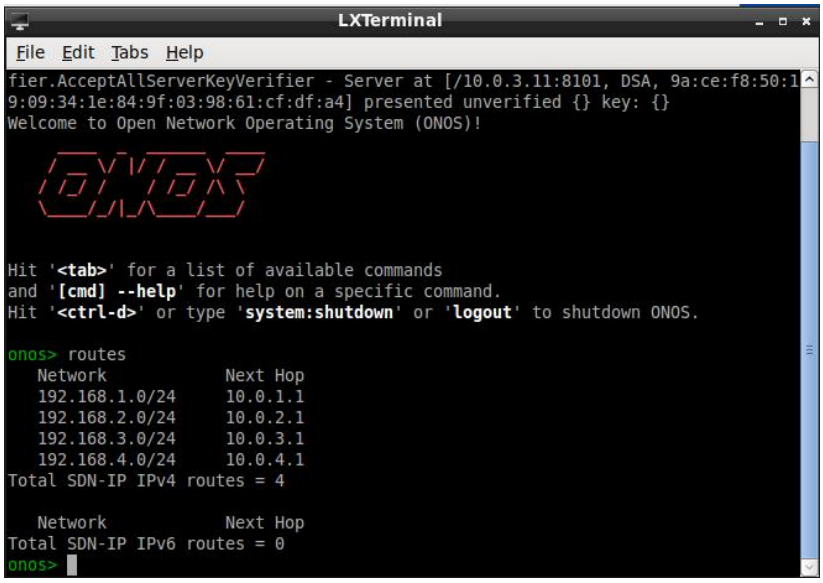

Figure 11. View four routes on SDN IP implemented on ONOS

From the Figure 11. above, it shows the results of the routes command which is used to view the new routes that have been learned by SDN IP, where the routes that have been learned by SDN IP are 4 routes, consisting of three routes in SDN IP IPv4 and one route addition. The existing

routes are 192.168.1.0/24, 192.168.2.0/24, and 192.168.3.0/24. The new route that has been learned by SDN IP is 192.168.4.0/24.

3.2. Discussion

Based on the tests carried out, the results obtained that SDN IP can be well designed and implemented using ONOS on the Border Gateway Protocol (BGP). Networks on SDN can connect to available IP networks, hosts, and entities. In the test, the four available hosts (h1, h2, h3, and h4) were able to ping each other successfully and also exchange routing information on a BGP-based Autonomous System (AS) network. The routing process and learning of routing by SDN IP are done by software on ONOS, using the Intents API. From the Intents API, it can also be seen that the four existing routes consist of three existing routes and an additional route that has been successfully learned by SDN IP.

4. CONCLUSION

The design and technical implementation of SDN IP using ONOS on BGP can be done by installing ONOS (ONOS CLI, ONOS GUI, SDN IP Mininet) via VirtualBox on a Linux operating system, and then activating SDN IP. SDN IP testing to check connections between networks can be carried out in seven phases, starting from checking the host configuration on SDN IP to checking the four routes on the implemented SDN IP, where the Intents API plays an important role on the software side to help ONOS in recognizing and studying every route and host connected in SDN IP so that hosts and networks on SDN can connect to the available IP network. SDN can connect and exchange routing information with the Autonomous System (AS) native BGP-based network.

ACKNOWLEDGEMENTS

This research was carried out independently in 2021 during the Covid-19 pandemic, using Ubuntu Linux operating systems and ONOS, Mininet, and some open-source software. Gratitude to Software Defined Network Research Group Institut Teknologi Bandung, Indonesia Linux Open Source Community, and Udayana University during this research.

REFERENCES

[1] K. Benzekki, A.E. Fergougui, A.E. Belrhiti, "Software-Defined Networking (SDN): A Survey," Journal of Security and Communication Networks, Vol.9, No.18, 2017.
[2] S. Badotra, J. Singh, "A Review Paper on Software Defined Networking," International Journal of Advanced Research in Computer Science, Vol.8, No.2, pp. 1-9, 2017.
[3] M. Mousa, A. M. Bahaa-Eldin, M. Sobh, "Software Defined Networking Concepts and Challenges," 11th International Conference on Computer Engineering and Systems (ICCES), pp. 79-90, 2016.
[4] L. Lukas, V. Windha, "Border Gateway Protocol Pada Teknologi Software Defined Network," Jurnal Teknik dan Ilmu Komputer, Vol.06, No.24, pp. 389-400, 2017.
[5] K. Kaur, S. Garg, G. Kaddoum, N. Kumar, F. Gagnon, "SDN-Based Internet of Autonomous Vehicles: An Energy-Efficient Approach for Controller Placement," IEEE Wireless Communications, Vol.26, No.6, pp. 72-79, 2019.
[6] R. Zhou, Y. Lai, Z. Liu, Y. Chen, X. Yao, J. Gong, "A Security Authentication Protocol for Trusted Domains in an Autonomous Decentralized System," International Journal of Distributed Sensor Networks, Vol.2016, pp.1-13, 2016.
[7] P. Monika, R.M. Negara, D.D. Sanjoyo, "Performance Analysis of Software Defined Network Using Intent Monitor and Reroute Method on ONOS Controller," Bulletin of Electrical Engineering and Informatics, Vol.9, No.5, pp.2065~2073, 2020.
[8] A.Z. Pramudita, I.M. Suartana, "Perbandingan Performa Controller OpenDayLight dan Ryu pada Arsitektur Software Defined Network," JINACS: Journal of Informatics and Computer Science, Vol.01, No.04, pp.174-178, 2020.
[9] K. Nugroho, D.P. Setyanugroho, "Analisis Kinerja Route Flow Pada Jaringan SDN (Software Defined Network) Menggunakan Topologi Full-Mesh," ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, dan Teknik Elektronika, Vol.7, No.3, 2019.
[10] B. Goswami, "Experimenting with ONOS Scalability on Software Defined Network," Journal of Advanced Research in Dynamical and Control Systems, Vol.10, No.14, pp.1820-1830, 2019.
[11] R. Kartadie, V. Panggayuh, "Floodlight VS ONOS Dalam Unjuk Kerja," JIPI: Jurnal Ilmiah Penelitian dan Pembelajaran Informatika, Vol.4, No.2, 2019.

[12] I.P.A.E. Pratama, I.M.A. Wikantyasa, "Implementasi dan Analisis Simulasi QOS dan Perfomance Device dengan Menggunakan ONOS dan Iperf3," Jurnal Informatika Universitas Pamulang, Vol.4, No.2, 2019.
[13] D. Halomoan, M. Zarlis, Tulus, "Karakteristik Open Flow Controller Dengan ONOS," JTIK:Jurnal Teknik Informatika Kaputama, Vol.1, No.1, 2017.
[14] R. Annisa, "Pengembangan Manajemen Jaringan Berbasis Software-Defined Network di Politeknik Sekayu," TIPS: Jurnal Teknik Informatika Politeknik Sekayu, Vol.VII, No.2, pp.33-43, 2017.
[15] A. Friyanto, "High Availability Aspects of SDN IP Reactive Routing," IOP Conference Series Material Science Engineering, 2020.
[16] M. Purwiadi, W. Yahya, A. Basuki, "High Availability Controller Software Defined Network Menggunakan Heartbeat dan DRBD," Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, Vol.2, No.8, pp.2297-2306, 2018.
[17] W. Xia, Y. Wen, C.H. Foh, D. Niyato, H. Xie, "A Survey on Software-Defined Networking," IEEE Communication Surveys and Tutorials, Vol.17, No.1, 2015.
[18] E. Rizky, E.M. Safitri, S.A. Priyambada, N.D. Angresti, "Optimasi Rute Untuk Software Defined Networking-Wide Area Network (SDN-WAN) Dengan Openflow Protocol," Informatika Mulawarman Jurnal Ilmiah Ilmu Komputer, Vol.13, No.1, 2018.
[19] R. Adrian, "Cost Optimization on Open Shortest Path First in Software Defined-Network," Techno.COM, Vol.16, No.4, pp.421-434, 2017.
[20] F.A. Jamal, E.B. Purnomowati, "Perbandingan Performansi Arsitektur Tradisional Terhadap Arsitektur Software-Defined Network (SDN) Dengan Menggunakan Controller Opendaylight," Jurnal Mahasiswa TEUB, Vol.6, No.6, 2018.
[21] V. Varadharajan, K.K. Karmakar, U. Tupakula, "Securing Communication in Multiple Autonomous System Domains With Software Defined Networking," IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pp. 195-203, 2017.
[22] A.Z.A. Ghivani, "Studi Perbandingan Routing Protocol BGP dan EIGRP, Evaluasi Kinerja Performansi Pada Autonomous System Berbeda," SISTEMASI: Jurnal Sistem Informasi, Vol.7, No.2, 2018.
[23] V. Baggan, S.N. Panda, "Enhancing Network Path Restoration With Software Defined Networking," International Journal of Applied Engineering Research, Vol.14, N0.8, pp.1910-1916, 2019.
[24] V. Baggan, S.N. Panda, "Enhancing Border Gateway Routing Protocol with Software Defined Networking," International Journal of Innovative Technology and Exploring Engineering (IJITEE), Vol.8, Issue.8, 2019.
[25] Z. Jingjing, C. Di, W. Weiming, J. Rong, W. Xiaochun, "The Deployment of Routing Protocols in Distributed Control Plane of SDN," Hindawi: The Scientific World Journal, Vol. 2014.
[26] L.B. Christensen, "Experimental Methodology, 10th Edition," Willey tPublisher. 2012.
[27] ONOS, "Basic ONOS Tutorial," [online]. Available: https://wiki.onosproject.org/display/ONOS/Basic+ONOS+Tutorial (accessed: 20 May 2021).
[28] V. Jain, B. Edgeworth, "Troubleshooting BGP: A Practical Guide to Understanding and Troubleshooting BGP (Networking Technology) 1st Edition," CISCO Press, 2016.
[29] P. Goransson, C. Black, T. Culver, "Software Defined Networks: A Comprehensive Approach 2nd Edition," Morgan Kaufmann, 2016.