Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI)
Vol. 9, No. 3, September 2023, pp. 768-779
ISSN: 2338-3070, DOI: 10.26555/jiteki.v9i3.26533

Network Slicing Using FlowVisor for Enforcement of Bandwidth Isolation in SDN Virtual Networks

Vivi Monita, Wisnu Wendanto, Endang Anggiratih
Faculty of Science and Technology, Pignatelli Triputra University, Surakarta, Indonesia

ARTICLE INFO

Article history:

Received July 06, 2023
Revised August 07, 2023
Published August 11, 2023

Keywords:

SDN;
Network slicing;
FlowVisor;
Bandwidth isolation

ABSTRACT

Software-defined networking (SDN) is becoming increasingly popular because of features such as programming control, embedded monitoring, fine-grained control, flexibility, support for many tenants, and scalability. Problems with the prior design, known as the conventional network, include the need to configure each network device individually, decentralized control, and a persistent issue with tenant enforcement for multitenant support. Tenants are unable to administer their networks without disturbing their neighbours. In this research, network slicing on SDN will ensure tenant isolation using FlowVisor and an SDN controller. Flowspace, which is part of FlowVisor capable of implementing network isolation, is for isolation in this research. Multitenancy is supported in SDN via the network slicing technique. Two types of renters were employed, and two testing procedures connectivity and functionality were run to meet the research objectives. This research produced several findings, including that all hosts were correctly linked, and the connection was achieved without turning on FlowVisor. The host function can only send and receive data from hosts with the same tenant. The research results show that FlowVisor can be applied for isolation enforcement. As a result of each tenant utilising their slice of the network without being interrupted by other slices, this research finds that utilising FlowVisor to construct Flowspace can segment the network to allow multitenancy. Expanding the number of slices for more study and testing in a real-world setting is possible.

Corresponding Author:

Vivi Monita, Pignatelli Triputra University, Surakarta, Indonesia
Email: vivimonita@upitra.ac.id

1. INTRODUCTION

Like now, the millennial period has various technology that make life simpler for people. The internet is one of the numerous technologies now in use. Since it has served as a foundation or a link for developing many modern technologies, the internet plays various roles in their development [1]. Conventional network design suffers from the same issues as the original internet architecture [2]. It requires a lot of device configuration and labour for a typical network to function effectively. These factors make software-defined networking (SDN) technology in the network a likely candidate to solve this issue [3]-[8]. According to the current state of conventional networks, since they need a control plane idea, it will be exceedingly challenging to manage and build networks as they grow vast and complicated, which can impede current developments. Then according to [9], conventional networks are static and need much management effort. Additionally, imposing laws or regulations on conventional networks is challenging, and SDN reduces operating expenses considerably. Why is it supports multitenant network design and is based on the circumstances that now exist in conventional networks. This research is urgent due to the advantages gained from SDN and multitenants. According to [10], adopting SDN in research has advantages such as lower maintenance costs and simpler network administration due to controllers than conventional networks, which need adjustments to each device deployed. This research [11]–[16], multitenancy delivers benefits including cost savings, increased efficiency,

and simpler maintenance, and greater scalability and computing capacity. SDN is a brand-new idea in network design that isolates the hardware's control plane from it. Network configuration is simpler and more flexible because of the fundamental idea of SDN design.

The control and data planes comprise most of the most crucial SDN components [17], [19]. Because the smart grid, including SDN, has been the subject of much research, security is a key concern in SDN [20]. Multitenant is one of the crucial elements enabling SDN, along with security. The multitenant architectural style of a network enables the same infrastructure service provider to manage multiple tenants [21]-[23]. The multitenant approach includes network slicing. According to [24], network slicing is a network strategy based on network virtualization, which is anticipated to offer features like flexibility and modularity. A smart network with greater autonomy and complexity includes SDN [25].

Additionally, the POX controller may be employed as an SDN controller. An open-source controller that intends to expand SDN is the network controller. Between the controller and the switch, the controller offers an effective approach to implementing the OpenFlow protocol [15], [17]. A FlowVisor technique exists between hardware and software over the internet network. An operating system-like virtual layer atop the computer, FlowVisor employs a set of instructions to control the hardware. To manage network traffic, FlowVisor makes use of the OpenFlow protocol [26]. The issue emerges when the conventional design has several flaws and the tenant can only control its network if other tenants in the SDN topology disturb it. As a result, the main goal of this study is to determine the best way to isolate network topology by utilizing Flowspace isolation and network slicing with FlowVisor. To enable multi-tenants, the SDN topology is divided into wedge-shaped spaces with the help of flow space isolation, allowing each tenant to govern their own space without interfering with other tenants. To impose Flowspace separation on SDN, researchers will study network slicing using FlowVisor.

2. LITERATURE REVIEW

2.1. Software-Defined Networking

SDN can manage networks more efficiently and flexibly, because SDN is the latest network management technology with an agile, programmable, and centrally managed architecture that offers vendor-agnostic open device management, allowing for more effective and flexible network administration [27]-[29]. In SDN, network administrators can create networks through the central console or controller, making it easier to manage because there is no need to configure each switch and existing network devices. A comparison of SDN and conventional network topologies is shown in Fig. 1. As we know, conventional network architectures are inefficient in meeting current network demands. Traffic management has to be quite adaptable for some applications. The complexity of today's networks constrains scale, security, and flexibility, which is why the SDN design addresses these issues, as shown in Fig. 2.

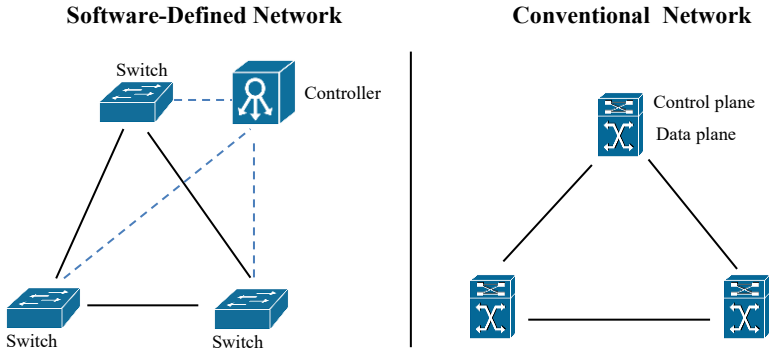


Fig. 1. Comparison of SDN and conventional network topology

With SDN, the control plane and forwarding plane are separated, giving network managers the practical freedom to govern network traffic and other behaviours such that network elements may be managed by a centralized control plane program [30], [31]. The three tiers of the SDN architecture are as follows:

1. The network forwarding plane, which comprises physical infrastructure components like switches and routers, is the data plane layer.
2. A centralized network controller that oversees all network activities and traffic is known as the control plane.

3. Network administrators and operators may directly control the network using the application layer and the controller.

Southbound interfaces are used for the connection between the control fields and the data fields. OpenFlow was the first standard southbound interface [2] that gave the controller direct access to the SDN forwarding plane. The matching flow item in the flow table determines how arriving packets are routed in an OpenFlow network. For the controller to take further action, if the incoming stream does not match any flow entries in the flow table, a situation known as a table miss, the switch sends a packet message to the controller informing it about the flow parameters.

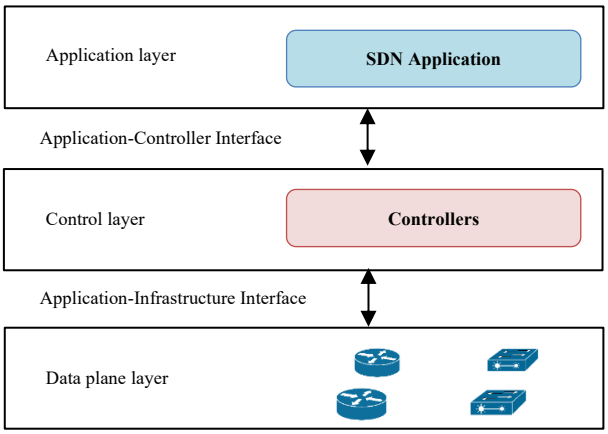


Fig. 2. SDN architecture

2.2. FlowVisor

Between different OpenFlow controllers and OpenFlow switches as a proxy, FlowVisor is an OpenFlow controller that allows viewing the physical OpenFlow architecture as distinct virtual networks [32]-[35]. FlowVisor manages the underlying network by isolating resources into slices and using the OpenFlow protocol to provide a different controller authority over each slice. FlowVisor provides topology and address space isolation. Architecturally, a transparent neutral proxy FlowVisor makes no assumptions about the behavior of switches and controllers.

Fig. 3 demonstrates the FlowVisor's internal workings and the exchange of information between the switch and the controller. The controller issues a command to the OpenFlow switch to initiate the process. A slicer element (1) handling commands and messages from/to the OpenFlow controller is the first to receive controller directives. For each controller virtual network, a separate Slicer exists. The slicer then uses its flow space rules to determine if the command received complies with the virtual network definition (2), edits the command as appropriate, and validates that it does. The classifier components responsible for handling commands and communications to/from the OpenFlow switch are then used to send the command output to the switch (3). Every OpenFlow switch has a classifier.

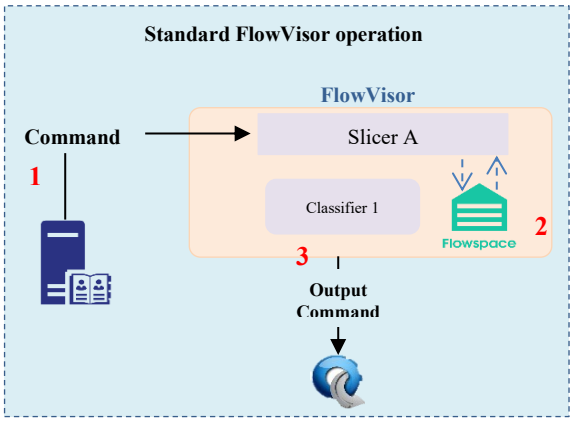


Fig. 3. Operasi internal FlowVisor

The FlowVisor sits between each controller having full access to the flow-maintaining switch, which determines the appropriate slice.

FlowVisor divides the smallest transmission bandwidth for each slice, giving a group of flows from that slice a specific data rate. FlowVisor divides the flow tables across switches and tracks each flow entry for every guest controller. The FlowVisor controller's resource allocation and routing policy slices set up the switches. Slices are autonomous and feature their own "flow spaces" for data flow. These separate slices are breakable, enabling various assaults.

2.3. Network Slicing

Network slicing creates a slice of the targeted network. The distinguishing feature of this technology is that each network slice has distinct characteristics, including very low latency, extremely high bandwidth, and mobile broadband. Slices provide the benefit of handling several situations at once.

Each part of the network can be designed to provide several performance levels, such as Service Level Agreements (SLAs). Testing the required SLAs, such as Quality of Service (QoS) for network sections sharing the same physical network, takes time and effort [36]. Slices must be provided by the control and management plane, which must also be capable of dynamic reconfiguration. The data plane must meet each slice's QoS requirements, such as slice isolation, slice performance isolation, etc. In previous research [15], network slicing on SDN has been implemented in isolating each tenant's flow space, so hosts in different slices cannot communicate with each other. The topology used is a tree topology. Then in research [17] have discussed the application of traffic isolation on the SDN network through the network slicing method using FlowVisor. The results obtained show that each tenant cannot communicate with other tenants.

2.4. Bandwidth Isolation

Bandwidth isolation is one of the methods offered by OpenFlow and SDN to divide the bandwidth speed and regulate the amount of bandwidth per slice of the network. Bandwidth isolation is carried out using FlowVisor by maximizing priority point A virtual LAN (VLAN) whose range is from 0 to 7 and is used to prioritize different classes or traffic [37]-[40]. This Openflow protocol allows VLAN tags such as PCP fields to be managed and gives a certain priority to packets in a flow. The Openflow protocol enforces bandwidth isolation and involves modifying the flow table of each network slice and VLAN priority by FlowVisor. Traffic in each slice is mapped into eight priority groups that allow network administrators to prioritize bandwidth per slice.

3. METHOD

a. System Design

In this research, the system is designed using the topology, as shown in Fig. 4. The topology consists of four switches connected to one controller. In its implementation, the controller is bridged by FlowVisor to apply the network slicing method. The network slicing method used is TCP port slicing with 2 slices. Slice 1 streams data traffic through TCP port 80 on each host, while slice 2 streams data traffic through TCP ports other than port 80. Slice 1 has a link bandwidth of 10 Mbps, while slice 2 has a link bandwidth of 1 Mbps.

b. Device Used

The devices used in this research are as follows:

1. OpenFlow: the communication protocol is used in SDN to control data planes physically separated from the control plane using a controller located on the server. By using OpenFlow, researchers can exercise direct control over routing data packets to be routed on a network [15].
   OpenFlow is the most popular southbound API that allows controller interaction with switches in the SDN architecture [16].
2. FlowVisor: the network virtualization layer separates the architectural layers of hardware and software. To regulate the underlying physical network, FlowVisor makes use of the OpenFlow protocol. To make sure that the guest controller can only see and operate the switches in the data plane it is intended to, FlowVisor serves as a bridge between each OpenFlow controller and the data plane. The objectives of FlowVisor's architecture included tight isolation between network slices, transparent virtualization of the network controller, and comprehensive and flexible slice rules [5].
3. Mininet: network emulators create virtual hosts, switches, and controllers. Mininet can be used for research, development, learning, prototyping, testing, debugging, and other purposes that utilize experimental networks on laptops or PCs [17]. By executing genuine kernels, switches, and application code on a single physical computer, virtual machine, or cloud, Mininet can build realistic virtual networks. This research uses Mininet as an emulator to design a data plane architecture for SDN.
4. Controller: platform based on the Python programming language used to develop SDN controller. POX was originally an OpenFlow controller, but now it also functions as an OpenFlow switch and generally supports software development for SDN [18]. Controllers can run various applications such as hubs,

switches, load balancers, and firewalls. Network analyzer software such as Wireshark can monitor data packet flow between POX controllers and other OpenFlow devices [19].

5. Wireshark: the leading network protocol analyzer application is widely used globally [20]. The main function of a network protocol analyzer is to examine the details of communications in a network by capturing every packet sent to and from a computer and then presenting it in a human-readable format [21].
6. IPerf: a tool to measure the maximum bandwidth achieved in an IP network. Other parameters that can be measured are time, buffer, and protocol (TCP, UDP, SCTP) [22].

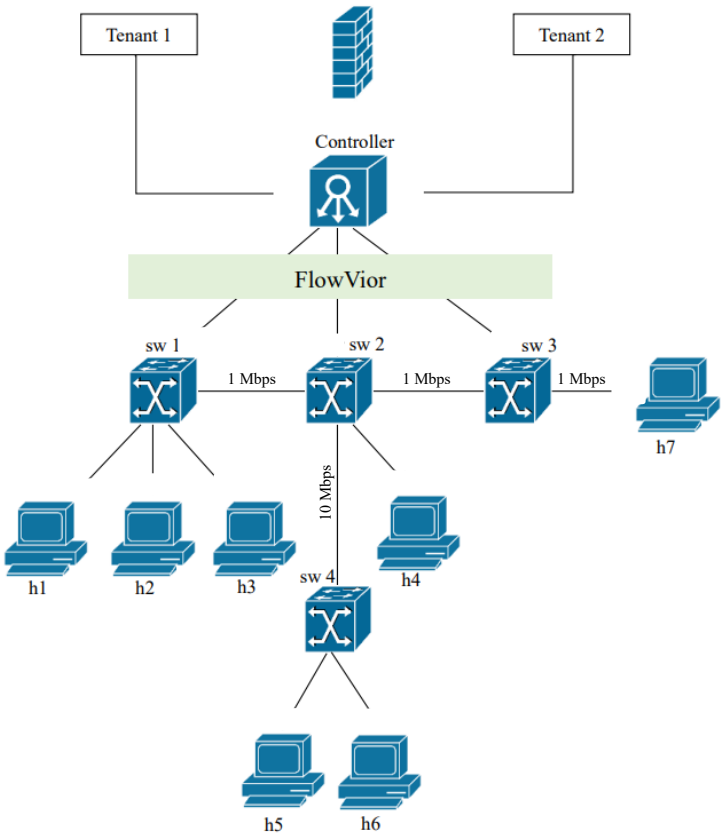


Fig. 4. Network topology

c. System Flowchart

The flowchart of the simulation and testing used in this research, as shown in Fig. 5. Installing the Ubuntu 16.04 LTS Virtual Machine on the Oracle VM VirtualBox hypervisor is the first step. After the VM installation, Mininet is installed as an SDN network emulator at the infrastructure layer. Next, install the controller as the SDN network controller that will be used. After the Mininet and controller installation is successful, the network topology design on Mininet can be done.

After the topology has been successfully designed, the next step is configuring the network by setting the IP address on each host. After the data plane configuration is complete, it is possible to link the data plane and the control plane. If the connection is successful, the next step is to test the connectivity and functionality of the SDN network without using the network slicing method and using network slicing by following the configuration steps listed in Fig. 6. Next, connectivity and functionality testing is carried out to compare the connectivity and functionality before and after applying the network slicing method. The final step is analyzing the data and making conclusions. After the topology configuration has been successfully executed, the next step is to run FlowVisor by typing the following command in the Ubuntu terminal: "*sudo /etc/init.d/FlowVisor start*". Then configure the FlowVisor for slice creation, as shown in Fig. 7, and configure the flow space according to Table 1. After the flow space has been successfully created, the next step is to run the controllers, each using the port according to the slice configuration. Port 1001 is used for fast slices, while port 1002 is used for slow slices. The command is "*sudo ./pox.py log. level --DEBUG forwarding.l2_learning OpenFlow.of_01 --port=1001*". The same steps are carried out on other terminals by changing the port number to 1002.

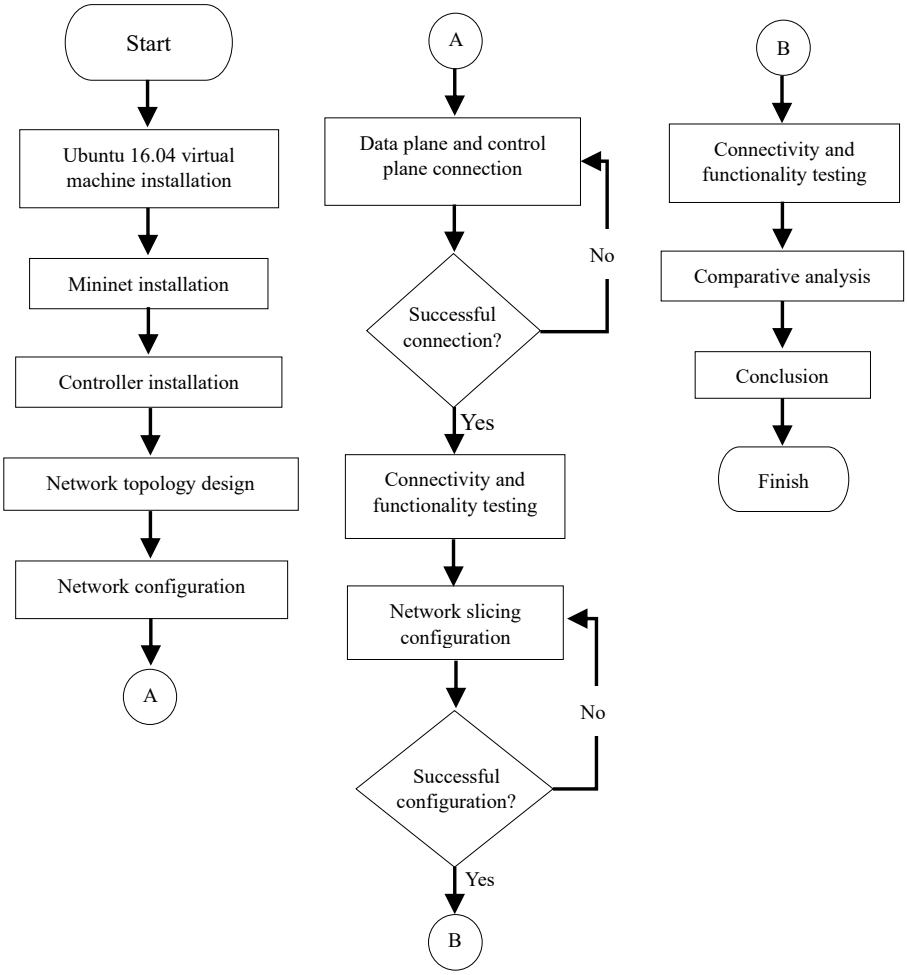


Fig. 5. System design

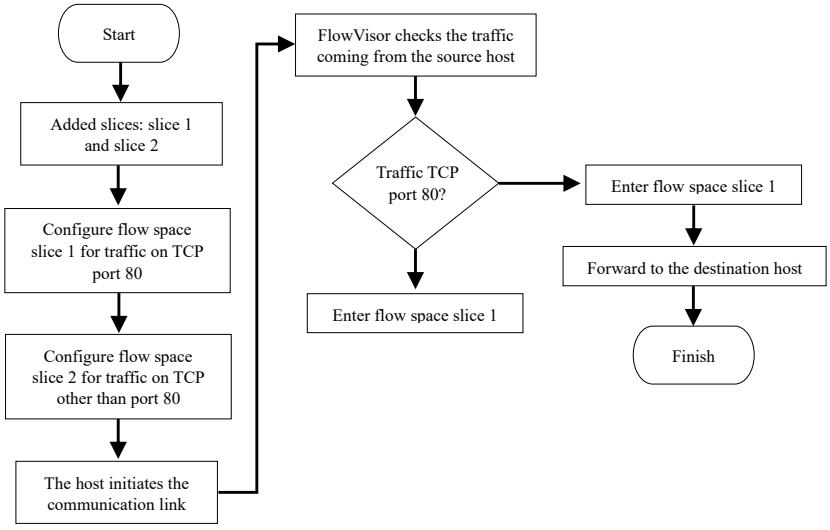


Fig. 6. Network slicing configuration flowchart

```
Configured slices:
fvadmin         --> enabled
fast            --> enabled
slow            --> enabled
```

Fig. 7. Command to create slices in FlowVisor

Table 1. Flow space configuration each switch

Flowsapce	Commands
Switch 1	fvctl -n add-flowspace dpid1-p3-fast-src 1 100 in_port=3,dl_type=0x0800,nw_proto=6,tp_src=80 fast=7
	fvctl -n add-flowspace dpid1-p3-fast-dst 1 100 in_port=3,dl_type=0x0800,nw_proto=6,tp_dst=80 fast=7
	fvctl -n add-flowspace dpid1-p3-slow 1 1 in_port=3 slow=7
	fvctl -n add-flowspace dpid1-p1-fast 1 100 in_port=1 fast=7
	fvctl -n add-flowspace dpid1-p2-slow 1 1 in_port=2 slow=7
Switch 2	fvctl -n add-flowspace dpid2 2 100 any fast=7
Switch 3	fvctl -n add-flowspace dpid3-p4-fast-src 3 100 in_port=4,dl_type=0x0800,nw_proto=6,tp_src=80 fast=7
	fvctl -n add-flowspace dpid3-p4-fast-dst 3 100 in_port=4,dl_type=0x0800,nw_proto=6,tp_dst=80 fast=7
	fvctl -n add-flowspace dpid3-p4-slow 3 1 in_port=4 slow=7
	fvctl -n add-flowspace dpid3-p1-fast 3 100 in_port=1 fast=7
	fvctl -n add-flowspace dpid3-p2-fast 3 100 in_port=2 fast=7
	fvctl -n add-flowspace dpid3-p3-slow 3 1 in_port=3 slow=7
Switch 4	fvctl -n add-flowspace dpid4-p4-fast-src 4 100 in_port=4,dl_type=0x0800,nw_proto=6,tp_src=80 fast=7
	fvctl -n add-flowspace dpid4-p4-fast-dst 4 100 in_port=4,dl_type=0x0800,nw_proto=6,tp_dst=80 fast=7
	fvctl -n add-flowspace dpid4-p4-slow 4 1 in_port=4 slow=7
	fvctl -n add-flowspace dpid4-p1-fast 4 100 in_port=1 fast=7
	fvctl -n add-flowspace dpid4-p2-fast 4 100 in_port=2 fast=7
	fvctl -n add-flowspace dpid4-p3-slow 4 1 in_port=3 slow=7

4. RESULTS AND DISCUSSION

This section discusses the results and analysis of the system design that has been made. The tests carried out in this research were connectivity and functionality tests.

4.1. Connectivity Test Results

The results of the topology connectivity test that has been added to the network slicing configuration are shown in Fig. 8. Based on the figure, it can be seen that each host is successfully connected and all hosts.

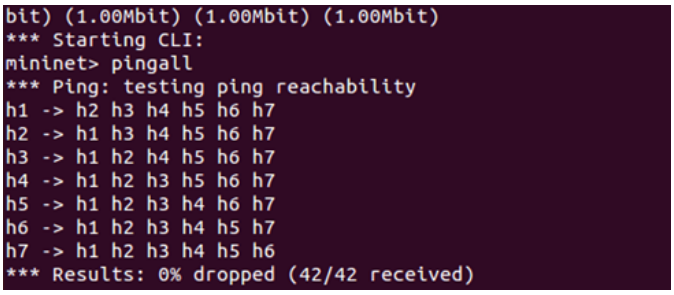


Fig. 8. Connectivity test

4.2. Functionality Testing Results

Based on the FlowVisor configuration that has been made, the research network topology consists of 2 slices, namely slow slice and fast slice. Every host communicating via TCP port 80 will enter flow space in a fast slice configuration with a link bandwidth of 10 Mbps. Meanwhile, every host communicating through ports other than port 80, such as port 22 or port 443, will enter flow space in a slow slice with a link bandwidth of 1 Mbps.

The results of functionality testing between node h1 and node h6, which communicate via TCP port 80, are shown in Fig. 9. Based on a report from iperf, bandwidth results were obtained in the 7-12 Mbps range. This shows that the flow space configuration for the fast slice is successful because when h1 communicates with h6 via TCP port 80, iperf reports that the bandwidth is in the range of 10 Mbps, according to the configuration applied to the fast slice flow space.

Fig. 10 results from functionality testing between node h1 and node h4 communicating via TCP port 22. Based on a report from iperf, bandwidth results are obtained in the 1-5 Mbps range. This shows that the flow space configuration for slow slices is successful because when h1 communicates with h4 via TCP port 22 or not port 80, iperf reports that the bandwidth is in the range of 1 Mbps, according to the configuration applied to slow slice flow space. In other words, the measured bandwidth does not reach 10 Mbps, indicating that the communication does not use fast slices of flow space.

Based on the results above, testing the FlowVisor functionality for flow space isolation on each slice was successful. This can be seen from the fact that hosts that communicate via port 80 succeed through a link

bandwidth of 80 Mbps, while hosts that communicate via ports other than 80 (in testing using port 22) succeed through a link bandwidth of 1 Mbps.

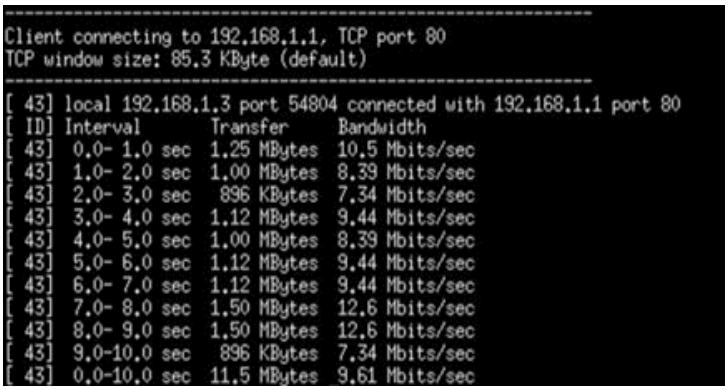


Fig. 9. Fast slice functionality testing

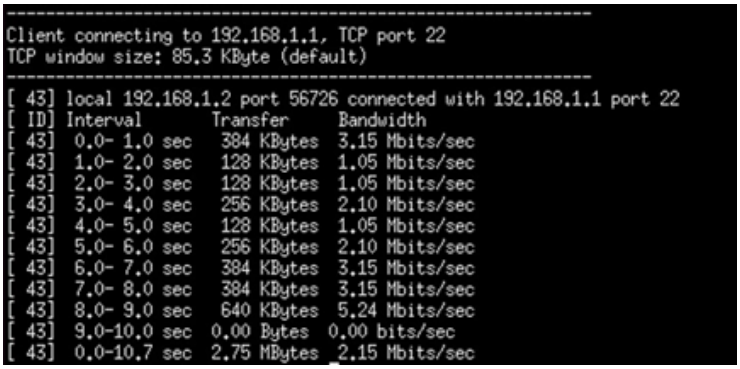


Fig. 10. Slow slice functionality testing

4.3. Resource Utilities Testing Results

Resource utility testing is carried out to determine how much FlowVisor consumes memory from the CPU, and testing is carried out with FlowVisor on and off. The test is repeated ten times to find the average range value 10, 20, 30, 40, 50, and 60 with seconds of time units. Based on the results of CPU performance testing, as shown in Fig. 11, CPU usage on the SDN network that does not use FlowVisor gets an average value of 17.16%. In contrast, with FlowVisor, it gets an average value of 22.83%, this is due to the slicing process in FlowVisor. Based on the results of testing computer memory performance, as shown in Fig. 12, it tends to be higher when using FlowVisor with an average result of 33.33%, while not using FlowVisor with a predetermined timeframe getting an average result of 54.67%, this is because FlowVisor requires more memory than the another because FlowVisor does slicing because FlowVisor does slicing.

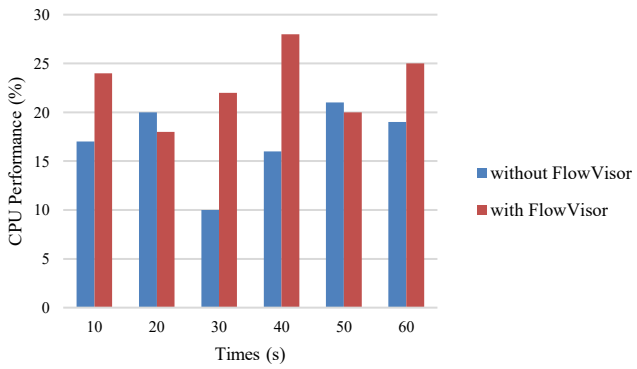


Fig. 11. CPU performance testing

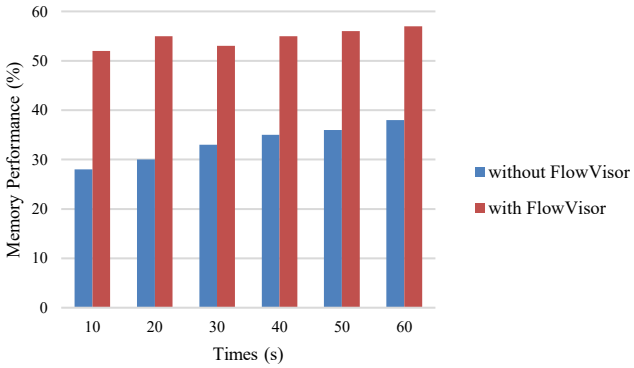


Fig. 12. Memory performance testing

4.4. Strong Isolation Bandwidth Testing Results

This strong isolation test aims to ascertain whether the slice has strong isolation by implementing FlowVisor. This is done by limiting the existing bandwidth on the network created by comparing FlowVisor before and after isolation, whether the slice is disrupted or not when initiating different bandwidths. This strong isolation test is carried out by sending different bandwidth determination packages with a range of 0 Mbps, 10 Mbps, 20 Mbps, 30 Mbps, 40 Mbps, and 50 Mbps and testing both slices. The test was carried out when the slice-1 condition obtained an average yield of 25.73 Mbps. The slice-2 condition obtained an average yield of 25.26 Mbps, so the results are shown in Fig. 13. Based on the results of the tests that have been carried out, the slices do not affect each other in the existing strong isolation and bandwidth tests are not too different from each other. Hence, the isolation by FlowVisor is strong.

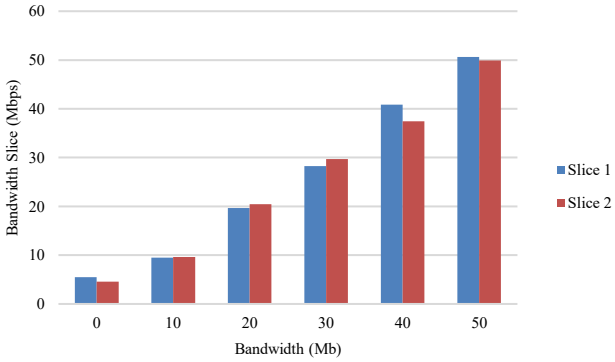


Fig. 13. Strong Isolation Bandwidth Testing

5. CONCLUSIONS AND SUGGESTIONS

Based on the test results and discussion, it can be concluded that FlowVisor succeeded in isolating the flow space for each slice on the SDN network based on the TCP Port through which communication occurs between hosts. Implementing the TCP port slicing method effectively flows data traffic according to the link bandwidth capacity available in the configuration of each slice. Even though there was a slight decrease in resource utilities and strong isolation bandwidth of SDN networks with and without network slicing, these differences were not practically significant.

Suggestions that can be applied in future research are to use a network topology that is more complex and representative of the existing network conditions in the field. In addition, you can add the number of TCP ports that can be traversed in the flow space of each slice for the TCP port slicing method. In addition to the TCP port slicing method, other network slicing methods, such as IP address slicing or MAC address slicing, can be used to verify their performance and effectiveness.

REFERENCES

[1] S. Ahmad and A. H. Mir, "Scalability, Consistency, Reliability and Security in SDN Controllers: A Survey of Diverse SDN Controllers," *Journal of Network and Systems Management*, vol. 29, no. 1, 2021, https://doi.org/10.1007/s10922- 020-09575-4.

[2] T. A. Assegie and P. S. Nair, "A review on software defined network security risks and challenges," *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 17, no. 6, 2019, https://doi.org/10.12928/TELKOMNIKA.v17i6.13119.

[3] G. Di Lena, A. Tomassilli, D. Saucez, F. Giroire, T. Turletti, and C. Lac, "Distrinet: A mininet implementation for the cloud," *Computer Communication Review*, vol. 51, no. 1, 2021, https://doi.org/10.1145/3457175.3457177.

[4] Y. Kim, S. Kim, and H. Lim, "Reinforcement learning based resource management for network slicing," *Applied Sciences (Switzerland)*, vol. 9, no. 11, 2019, https://doi.org/10.3390/app9112361.

[5] D. Scano, L. Valcarenghi, K. Kondepu, P. Castoldi, and A. Giorgetti, Network Slicing in SDN networks. *International Conference on Transparent Optical Networks*, pp. 1-4, 2020, https://doi.org/10.1109/ICTON51198.2020.9203184.

[6] P. Subedi, A. Alsadoon, P. W. C. Prasad, S. Rehman, N. Giweli, M. Imran, and S. Arif, Network Slicing: A Next Generation 5G Perspective. *Eurasip Journal on Wireless Communications and Networking*, vol. 1, p. 102, 2021, https://doi.org/10.1186/S13638-021-01983-7.

[7] R. Casellas, A. Giorgetti, R. Morro, R. Martínez, R. Vilalta and R. Muñoz, Enabling Network Slicing Across a Disaggregated Optical Transport Network. *Optical Fiber Communications Conference and Exhibition (OFC)*, pp. Tu3H-3, 2019, https://doi.org/10.1364/OFC.2019.Tu3H.3.

[8] R. Casellas, A. Giorgetti, R. Morro, R. Martinez, R. Vilalta and R. Munoz, Virtualization Of Disaggregated Optical Networks With Open Data Models In Support Of Network Slicing. in *IEEE/OSA Journal of Optical Communications and Networking*, vol. 12, no. 2, pp. A144-A154, 2020, https://doi.org/10.1364/JOCN.12.00A144.

[9] A. Giorgetti, A. Sgambelluri, R. Casellas, R. Morro, A. Campanella and P. Castoldi, Control Of Open And Disaggregated Transport Networks Using The Open Network Operating System (ONOS), in *IEEE/OSA Journal of Optical Communications and Networking*, vol. 12, no. 2, pp. A171-A181, 2020, https://doi.org/10.1364/JOCN.12.00A171.

[10] E. Leao Fernandes, E. Rojas, H. Alvarez, Z. Kis, D. Sanvito, N. Bonelli, C. Cascone, R. Esteve C. Rothenberg, The Road to BOFUSS: The Basic OpenFlow User-space Software Switch, *Journal of Network and Computer Applications,* vol. 165, p. 102685, 2019, https://doi.org/10.48550/arXiv.1901.06699.

[11] D. Scano, L. Valcarenghi, K. Kondepu, P. Castoldi and A. Giorgetti, Network Slicing in SDN Networks. *22nd International Conference on Transparent Optical Networks (ICTON)*, pp. 1-4, 2020, https://doi.org/10.1109/ICTON51198.2020.9203184.

[12] H. Babbar, S. Rani, A. A. AlZubi, A. Singh, N. Nasser and A. Al, Role of Network Slicing in Software Defined Networking for 5G: Use Cases and Future Directions, *IEEE Wireless Communications,* vol. 29, no. 1, pp. 112-118, 2022, https://doi.org/10.1109/MWC.001.2100318.

[13] A. Daifallah, T. Vijey, Y. Javad, The 5G network slicing using SDN based technology for managing network traffic, *Procedia Computer Science,* vol. 194, pp. 114-121, 2021, https://doi.org/10.1016/j.procs.2021.10.064.

[14] J. -J. Chen et al., Realizing Dynamic Network Slice Resource Management Based on SDN Networks, in *International Conference on Intelligent Computing and its Emerging Applications (ICEA),* pp. 120-125, 2019, https://doi.org/10.1109/ICEA.2019.8858288.

[15] M. T. Kurniawan, I. Moszardo and A. Almaarif, Network Slicing On Software Defined Network Using Flowvisor and POX Controller To Flowspace Isolation Enforcement, in *10th International Conference on Smart Grid (icSmartGrid)*, pp. 29-34, 2022, https://doi.org/10.1109/icSmartGrid55722.2022.9848585.

[16] H. S. Hamza and M. E. Manaa, The Importance of Network Slicing and Optimization in Virtualized Cloud Radio Access Network, in *1st Babylon International Conference on Information Technology and Science (BICITS)*, pp. 245-250, 2021, https://doi.org/10.1109/BICITS51482.2021.9509925.

[17] M. T. Kurniawan, M. Fathinuddin, H. A. Widiyanti and G. R. Simanjuntak. Network Slicing on SDN using FlowVisor and POX Controller to Traffic Isolation Enforcement, in *International Conference on Engineering and Emerging Technologies (ICEET)*, pp. 1-6, 2021, https://doi.org/10.1109/ICEET53442.2021.9659765.

[18] E. Coronado, B. Gomez and R. Riggio. (2020). Demo: A Network Slicing Solution for Flexible Resource Allocation in SDN-Based WLANs, in *IEEE Wireless Communications and Networking Conference Workshop (WCNCW),* pp. 1-2, 2020, https://doi.org/10.1109/WCNCW48565.2020.9124869.

[19] A. O. Nyanteh, M. Li, M. F. Abbod and H. Al-Raweshidy. CloudSimHypervisor: Modeling and Simulating Network Slicing in Software-Defined Cloud Networks, *IEEE Access,* vol. 9, pp. 72484-72498, 2021, https://doi.org/10.1109/ACCESS.2021.3079501.

[20] C-I. Fan, Y.-T. Shih, J.-J. Huang, and W.-R. Chiu. Cross-network-slice authentication scheme for the 5th generation mobile communication system, *IEEE Transactions on Network and Service Management,* vol. 18, no. 1, pp. 701-712, 2021, https://doi.org/10.1109/TNSM.2021.3052208.

[21] M.M. Raikar, S.M. Meena, Network Slicing in Software-Defined Networks for Resource Optimization. In *Intelligent Sustainable Systems: Selected Papers of WorldS4,* vol. 2, pp. 555-564, 2021, https://doi.org/10.1007/978-981-16-6369-7_51.

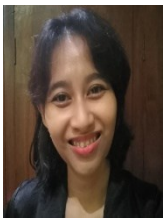
[22] S. D. A. Shah, M. A. Gregory and S. Li, Cloud-Native Network Slicing Using Software Defined Networking Based Multi-Access Edge Computing: A Survey. in *IEEE Access*, vol. 9, pp. 10903-10924, 2021, https://doi.org/10.1109/ACCESS.2021.3050155.

[23] W. Jiang, Y. Zhan, G. Zeng and J. Lu, Probabilistic-Forecasting-Based Admission Control for Network Slicing in Software-Defined Networks. in *IEEE Internet of Things Journal*, vol. 9, no. 15, pp. 14030-14047, 2022, https://doi.org/10.1109/JIOT.2022.3145475.
[24] T. Taleb, I. Afolabi, K. Samdanis, and F. Z. Yousaf, On multi-domain network slicing orchestration architecture and federated resource control, *IEEE Network*, vol. 33, no. 5, pp. 242-252. 2019, https://doi.org/10.1109/MNET.2018.1800267.
[25] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines. 5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges, *Computer Network*, vol. 167, p. 106984, 2020, https://doi.org/10.1016/j.comnet.2019.106984.
[26] Y-J Wu, W-S Hwang, C-Y Shen, Y-Y Chen, Network Slicing for mMTC and URLLC Using Software-Defined Networking with P4 Switches, *Electronics,* vol. 11, no. 14, p. 2111, 2022, https://doi.org/10.3390/electronics11142111.
[27] Y. Gyeongsik, Y. Yeonho, K. Minkoo, J. Heesang, Y. Chuck, Bandwidth Isolation Guarantee for SDN Virtual Networks, *IEEE INFOCOM 2021-IEEE Conference on Computer Communications,* pp. 1-10, 2021, https://doi.org/10.1109/INFOCOM42981.2021.9488797.
[28] L. U. Khan, I. Yaqoob, N. H. Tran, Z. Han, and C. S. Hong, Network slicing: Recent advances, taxonomy, requirements, and open research challenges, *IEEE Access,* vol. 8, pp. 36009-36028, 2020, https://doi.org/10.1109/ACCESS.2020.2975072.
[29] J. Alvarez-Horcajo, I. Martínez-Yelmo, D. Lopez-Pajares, J. A. Carral and M. Savi, A Hybrid SDN Switch Based on Standard P4 Code, *IEEE Communications Letters,* vol. 25, no. 5, pp. 1482-1485, 2021, https://doi.org/10.1109/LCOMM.2021.3049570.
[30] L. Cominardi, T. Deiss, M. Filippou, V. Sciancalepore, F. Giust, and D. Sabella, MEC Support For Network Slicing: Status And Limitations From A Standardization Viewpoint, *IEEE Communications Standards Magazine,* vol. 4, no.2, pp. 22-30, 2020, https://doi.org/10.1109/MCOMSTD.001.1900046.
[31] M. Chahbar, G. Diaz, A. Dandoush, C. Cérin and K. Ghoumid. A Comprehensive Survey on the E2E 5G Network Slicing Model, *IEEE Transactions on Network and Service Management,* vol. 18, no. 1, pp. 49-62, 2020, https://doi.org/10.1109/TNSM.2020.3044626.
[32] J. A. -Horcajo, I. M. -Yelmo, D. L. -Pajares, J. A. Carral, and M. Savi, A Hybrid SDN Switch Based on Standard P4 Code, *IEEE Communications Letters,* vol. 25, no. 5, pp. 1482-1485, 2021, https://doi.org/10.1109/LCOMM.2021.3049570.
[33] F. Paolucci, F. Cugini, P. Castoldi, and T. Osiński, Enhancing 5G SDN/NFV Edge with P4 Data Plane Programmability, *IEEE Network,* vol. 35, no. 3, pp. 154-160, 2021, https://doi.org/10.1109/MNET.021.1900599.
[34] M. Chahbar, G. Diaz, A. Dandoush, C. Cérin and K. Ghoumid, A Comprehensive Survey on the E2E 5G Network Slicing Model, in IEEE Transactions on Network and Service Management, *IEEE Transactions on Network and Service Management,* vol. 18, no. 1, pp. 49-62, 2020, https://doi.org/10.1109/TNSM.2020.3044626.
[35] L. X. Li, K. Jiao, F. Jiang, J. Wang, and M. Pan, A Service-Oriented Spectrum-Aware RAN-Slicing Trading Scheme Under Spectrum Sharing, *IEEE Internet of Things Journal,* vol. 7, no. 11, pp. 11303-11317, 2020 https://doi.org/10.1109/JIOT.2020.2997342.
[36] F. Fossati, S. Moretti, P. Perny, and S. Secci, Multi-resource Allocation For Network Slicing. *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1311-1324, 2020, https://doi.org/10.1109/TNET.2020.2979667.
[37] S. O. Oladejo and O. E. Falowo, Latency-Aware Dynamic Resource Allocation Scheme For Multi-Tier 5G Network: A Network Slicing-Multitenancy Scenario, *IEEE Access,* vol. 8, pp. 74834-74852, 2020, https://doi.org/10.1109/ACCESS.2020.2988710.
[38] S. Zhang, H. Luo, J. Li, W. Shi, and X. Shen, Hierarchical soft slicing to meet multi-dimensional QoS demand in cache-enabled vehicular networks, *IEEE Transactions on Wireless Communications,* vol. 19, no. 3, pp. 2150-2162, 2020, https://doi.org/10.1109/TWC.2019.2962798.
[39] P. Caballero, A. Banchs, G. De Veciana, and X. Costa-Perez, Network Slicing Games: Enabling Customization In Multi-Tenant Mobile Networks, *IEEE/ACM Transactions on Networking,* vol. 27, no. 2, pp. 662-675, 2019, https://doi.org/10.1109/TNET.2019.2895378.
[40] Q. Sun, L. Tian, Y. Zhou, J. Shi, and Z. Zhang, Incentive Scheme For Slice Cooperation Based On D2D Communication In 5G Networks, *China Communications,* vol. 17, no. 1, pp. 28-41, 2020, https://doi.org/10.23919/JCC.2020.01.002.

BIOGRAPHY OF AUTHORS

Vivi Monita, graduated (cum laude) from School of Electrical Engineering (Telecommunications), Telkom University, Bandung, Indonesia in 2021 and 2023 for her Bachelor degree (S.T.) and Master Degree (M.T.), respectively. She is currently a full-time lecturer and researcher at Pignatelli Triputra University, where she continues to mentor and instruct the upcoming generation of informatics experts. Artificial intelligence, machine learning, and networking. Email: vivimonita@upitra.ac.id.

Wisnu Wendanto, graduated with a Bachelor of Computer Engineering at STMIK AUB Surakarta, where he developed a solid background in computer science. 2012 he graduated from Diponegoro University with a master's degree in information systems. At Pignatelli Triputra University, where he is a full-time instructor and researcher, he continues to mentor and instruct new generations of professionals. His areas of interest in research cover networking. Email: wwendanto9@gmail.com.



Endang Anggiratih, who has a solid computer science and information technology background, earned her Bachelor of Informatics Engineering degree at UPN Veteran Yogyakarta in 2010. 2018 saw her graduate with a master's in computer science from Universitas Gajah Mada. At Pignatelli Triputra University, where she works as a full-time lecturer and researcher, she continues to mentor and instruct the upcoming generation of informatics specialists. Her areas of interest are artificial intelligence, machine learning, and computer vision. Email: endang_anggiratih@upitra.ac.id.