Designing Software Define Network Prototypes with Open vSwitch as Monitoring Traffic Police on The Raspberry Pi

Aldy Mohamad*, Purnawarman Musa
Department of Information Technology, Gunadarma University, Jakarta,
* email: aldymohamad12@gmail.com

Abstract

Technology is growing from year to year even day to day, this has made the increasing need for infrastructure that supports especially in aspects of computer networks. The increasing number of traffic that is burdening the router or switch encourages the increasing number of nodes to network devices with the aim of reducing and dividing the burden on network traffic. The need for traffic management and control is very important because with the increasing number of network devices and the higher traffic, making a network administrator need more time to handle if there are problems in the network. This research is trying to implement open vSwitch technology on low-cost raspberry pi devices. And by applying the traffic shaping and traffic rate methods by utilizing the traffic control feature on Linux, and then try to divide the amount of traffic received by network devices so that the traffic load becomes controlled. The results of this study, show the results of successful implementation and traffic management work well.

Keyword: Open vSwitch, Raspberry, Traffic Policing, Traffic Shaping, Openflow.

1. Introduction

The growing demand for internet services is due to the increasing number of social media users such as Instagram, Twitter, Facebook. Internet-based mobile messaging users are also a major cause of increased internet traffic such as WhatsApp and Telegram and not forgetting also streaming applications such as YouTube and other video streaming applications that make increasing the burden of traffic to the internet become heavier, and make the traffic profiles much more complex [1].

With increasing traffic, of course, requires more and more devices and new technology that is able to handle traffic loads that reach millions of requests per second. With a lot of traffic load, this certainly makes the technology and devices that have long been increasingly burdened, so many of which cause a slowdown (Congnestion) on the internet network. The main cause of this slowdown (cognition) is that more and more types of traffic are passed on the router or switch.

It can be imagined if the slowdown occurs on many switch nodes or routers that exist on the entire office network or cellular network. How many switches or routers need to be checked and troubleshooted to overcome one slowdown problem. Of course this job requires a long time and energy that is not small.

In recent years, a network paradigm was developed and began to be applied in many aspects of the network. The paradigm is the development of "Software Defined Network" (SDN) technology. SDN technology applies centralized control, meaning that this technology separates data traffic (data plane) and device management (control plane). With this method it is possible to do one configuration only through one central device and can be applied to all switch nodes or selected nodes. With SDN technology, network administrators can apply various existing technologies or new technologies on network devices in order to maintain the quality of traffic that will be passed on the router or switch [2].

Manuscript submitted 6 July 2020; revised 7 August 2020; accepted 13 August 2020.
Published 19 August 2020; http://dx.doi.org/10.28989/avitec.v2i2.712

2. Background

2.1 Cloud Computing

In terms of service provider side, Cloud Computing architecture cannot be separated from the use of virtualization technologies such as vmware or kvm. Both of these technologies have functions to create virtualization from a computing resource, so that only one physical resource can run multiple virtual machines at once and run different applications [3].

To connect many virtual machines, this certainly requires a virtual network, this virtual network is used as a network function such as existing physical devices, namely routing, switching, nat and others. With this virtual network, one virtual machine can communicate with each other in accordance with a previously implemented configuration [3].

Cloud computing is a virtual system with scalability elasticity that can be quickly propagated with a flexible pricing model. There are several delivery models in cloud computing, including Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS) as shown in Figure 1.[4] SaaS is a service to use applications that have been provided by providers, service providers manage the platform and infrastructure that runs the application. Furthermore PaaS is a service to use the platform that has been provided, the developer only focuses on applications that are made without thinking about platform maintenance. Set IaaS is a service to use the infrastructure that has been provided [5].

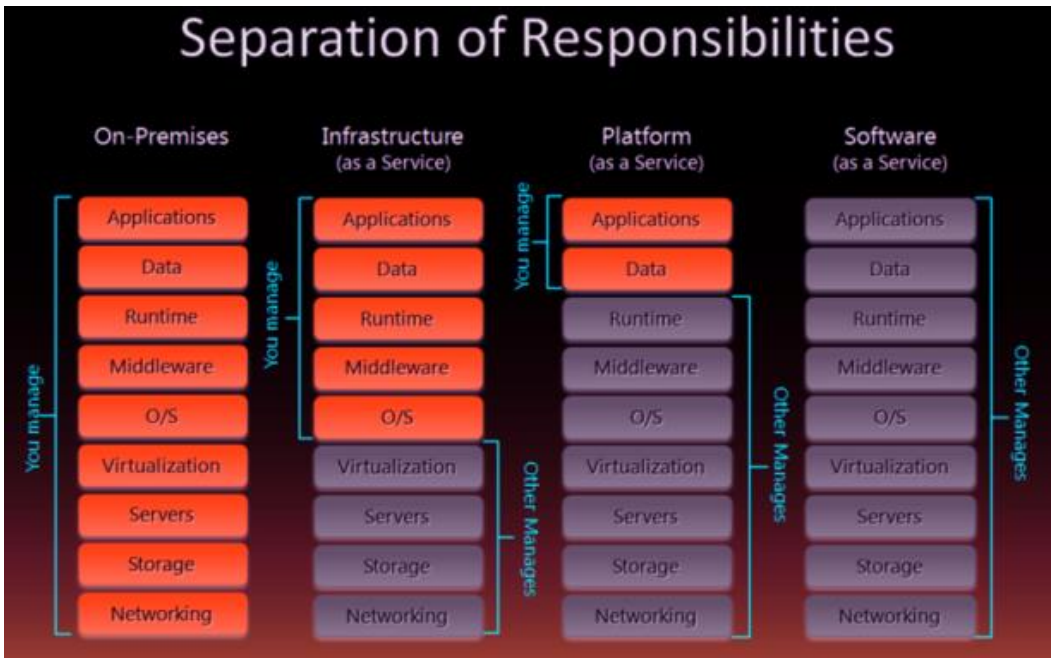

Figure 1. Cloud Computing Separation of Responsibilites

2.2 Software Defined Network

Software Defined Network (SDN) is a term that refers to new concepts / paradigms in designing, managing and implementing networks, especially to support the needs and innovations in this field which are increasingly complex. The basic concept of SDN is to make an explicit separation between control and forwarding plane, and then to do a system abstraction and isolate the complexity that exists in the components or sub-systems by defining a standard interface [6].

Some important aspects of SDN are: (a) There is a physical / explicit separation between forwarding / data-plane and control-plane. (b) A standard (vendor-agnosic) interface for programming network devices. (c) Centralized control plane (logically) or the existence of a network operating system capable of forming logical maps of the entire network and then representing it through (a type of) API (Application Programming Interface). (d) Virtualization where multiple network operating systems can control parts (slices or substrates) from the same device [6].

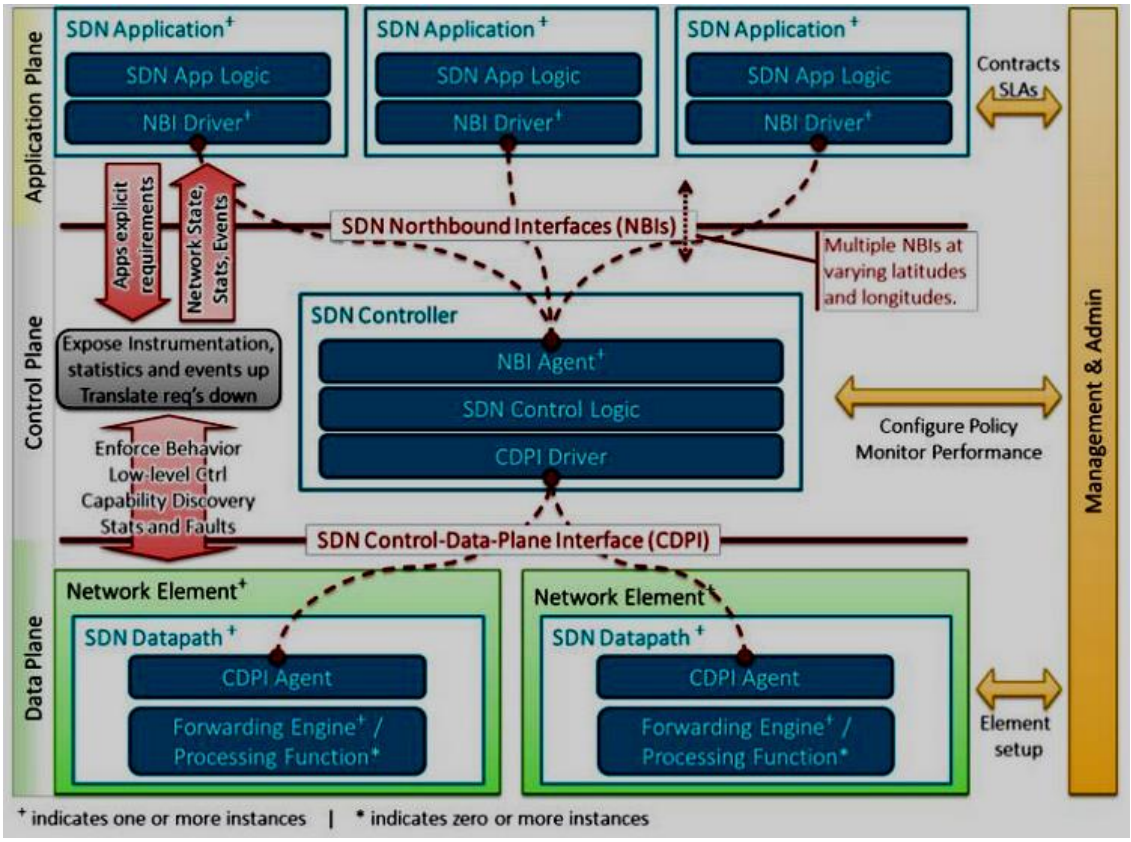

Figure 2. SDN Architecture

From Figure 2, SDN Architecture can be seen as 3 layers / fields: (a) **infrastructure** (data-plane / infrastructure layer): consists of network elements that can manage the SDN Datapath in accordance with the instructions given via the Control-Data-Plane Interface (CDPI). (b) **control** (control plane / layer): the control entity (SDN Controller) translates application needs with infrastructure by providing instructions that are appropriate for the SDN Datapath and provides information relevant and needed by the SDN Application. (c) **application** (application plane / layer): located at the top layer, communicating with the system via the NorthBound Interface (NBI) [6].

2.3 Open vSwitch

Open vSwitch (OVS) is a multilayer software switch that is licensed under the Apache 2 open source license. Open vSwitch is perfect for functioning as a virtual switch in a VM environment. In addition to exposing the control interface and visibility standards to the virtual network layer, open vswitch is designed to support distribution on many physical servers. Open vSwitch supports several Linux-based virtualization technologies including Xen / XenServer, KVM, and VirtualBox [7].

There are three main components in Open vSwitch, according to the documentation provided by the developer: database server (ovsdb-server), daemon (ovs-vswitchd), and kernel modules [7].

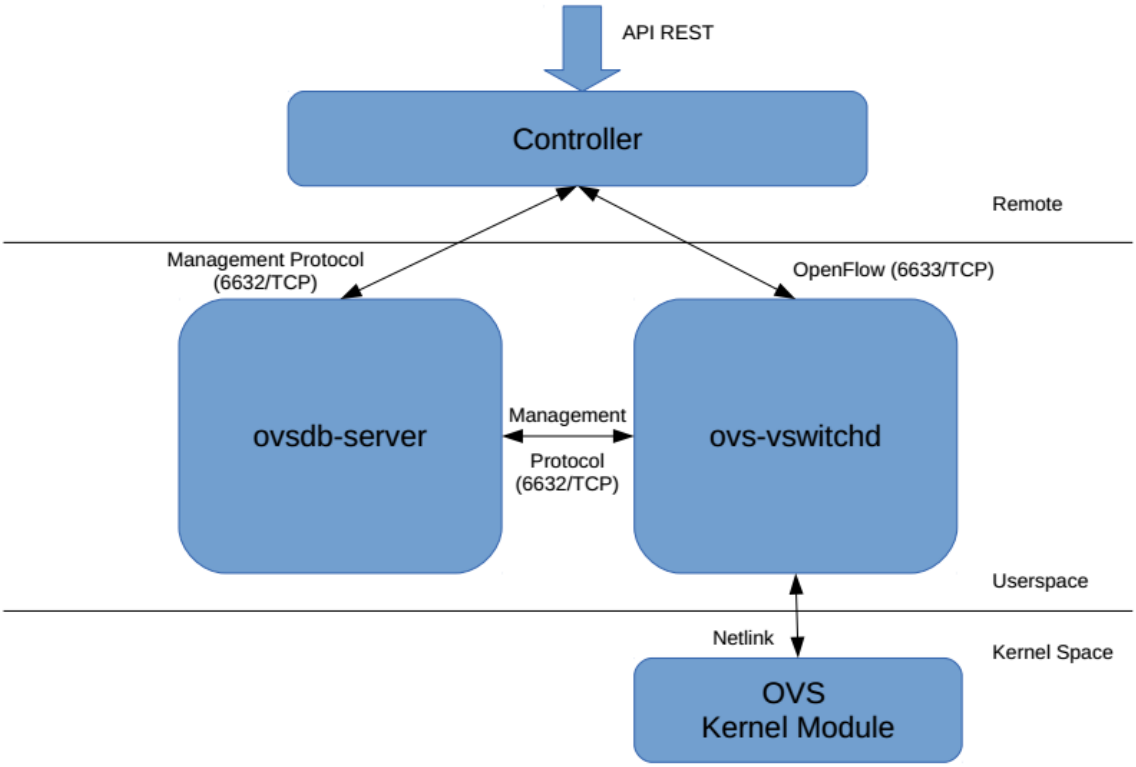

Figure 3. Open vSwitch architecture

In Figure 3 Controller can be an OpenFlow Controller such as Ryu, or an OVS database manager. All Open vSwitch components can be configured remotely. Ovsdb-server is a component that stores configuration databases, this component stores information that will not be lost even if the machine is rebooted, examples of configuration stored on ovsdb-server are configurations for bridges and interfaces. And Ovs-vswitchd is the most core part in open vswitch, the ovs-vswitchd section functions to handle all flow settings defined by the administrator. While the function of the kernel module is to help improve the performance of OVS itself [7].

2.4 Traffic Shaping and Policing

To implement traffic shaping using the traffic filter method using ip tables [8]. Ip tables is an administration tool on linux for IPv4 that functions to perform packet filtering and NAT on incoming or outgoing traffic [8]. As for the traffic policing method, using the Hierachical Token Bucket (HTB) method, HTB is a method for controlling outbound bandwidth [9].

3. Implementation and Testing

3.1 Experiment Setup

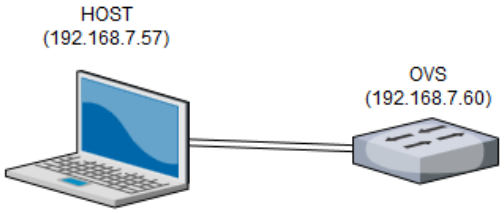

Figure 4. Experiment Topology

Figure 4 is a form of design topology for implementing a defined network software, there are three devices, namely host, ovs and internet (modem). The three devices are connected using a LAN cable. (a) **Host**, in this implementation uses a laptop to access the internet, the service that will be accessed by the host is video (port 41995), voip (port 5001) and web (port 80). Video access will be simulated by sending traffic using iperf with destination port 41995. Voip access will be done by sending traffic using iperf with destination port 5001. Whereas for web access, hosts will be done by sending iperf traffic using ports. For host specifications, configure IP address 192.168.7.57 and connect peer to peer using LAN cables to eth1 OVS ports. (b) **OVS**, this device uses raspberry pi type 2b which is configured to be an SDN switch using open vswitch. There are two LAN ports eth0 and eth1. Eth0 uses an onboard LAN port while eth1 uses a USB to LAN converter because the Raspberry Pi 2B only has 1 LAN port. For eth0 LAN port connections are configured with IP address 192.168.7.58 and connected using a LAN cable to an internet modem while eth1 port is connected to the host using a LAN cable.

3.2 Design Method

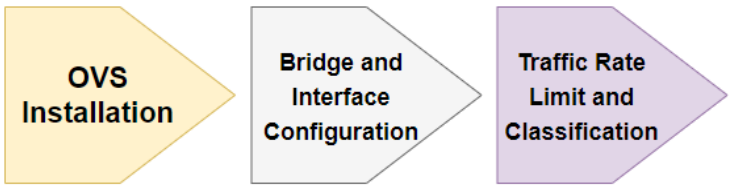

Figure 5. Design Method

Figure 5 shows the OVS design process in this implementation, there are several stages to be carried out, the design stages are OVS installation, bridge and interface configuration and Traffic rate limit and classification.

Open vSwitch installation is tested on Raspberry Pi. OVS installation is done on OS Raspbian Jessie with kernel 4.9.67+ and the OVS version used is 2.8.1. Installing OVS requires several additional steps such as updating the package and installing several libraries and supporting applications so that OVS can run well on the Raspberry Pi. Bridges in OVS function as a link between hosts that are virtual. The implementation in this study is used as a bridge connecting three types of traffic that will be passed on the OVS switch. bridges configured on OVS are, input bridges, output bridges and traffic bridges.

Traffic limit is a method used to limit the traffic rate or bandwidth that passes through each bridge in OVS. Each bridge has a different traffic limit configuration. While traffic classification is a method used to classify traffic that passes through OVS switches based on

the type of traffic. There are three types of traffic classification based on port numbers, namely video type traffic (41995), voip traffic (5001) or voice and web traffic (80). Traffic limit configuration is done on veth1 traffic because the veth1 interface is an interface that will receive traffic packets before being forwarded to OVS. While the traffic classification configuration uses the iptables method by applying rules to the eth1 interface which is the peer interface of the host. The rule set on the eth1 interface is to change the packet traffic using the 'mangle' method.

3.3. Testing

In this experiment, testing is carried out to get the value of the level of traffic that passes through OVS. Testing is done by the method, namely by using iperf application tools [10], the next is to test data transmission by sending data of various sizes, with sizes of 512KB, 1MB, 5MB and 10MB.

Table 1. Testing Results

Configuration	Port 41995	Port 5001	Port 80
Rate limit	200 Kbps	100 Kbps	200 Kbps

Table 1 shows how the rate limit configuration is applied to OVS. each port is configured with its own rate limit so that data received (ingress) by OVS will automatically be limited based on the rate limit as shown in Table 1.

Table 2. Testing Results

Iperf Testing Scenario	***Send Packet 512KB***	***Recieve Packet 512KB***	***Send Packet 1MB***	***Receive Packet 1MB***	***Send Packet 5MB***	***Recieve Packet 5MB***	***Send Packet 10MB***	***Recieve Packet 10MB***
Bandwidth	374 Kbps	190 Kbps	253 Kbps	191 Kbps	202 Kbps	192 Kbps	146 Kbps	143 Kbps
Host <> OVS port 41995	512KB	512KB	1MB	1MB	5MB	5MB	10MB	10MB
Bandwidth	184 Kbps	94.1 Kbps	124 Kbps	94.0 Kbps	101 Kbps	96.2 Kbps	92.5 Kbps	90.7 Kbps
Host <> OVS port 5001	512KB	512KB	1MB	1MB	5MB	5MB	10MB	10MB
Bandwidth	347 Kbps	183 Kbps	252 Kbps	190 Kbps	202 Kbps	192 Kbps	135 Kbps	133 Kbps
Host <> OVS port 80	512KB	512KB	1MB	1MB	5MB	5MB	10MB	10MB

Table 2 shows how the data sent from the host has different data size variations and different bandwidths, but after the data is received by OVS, it will automatically follow the configured limits as shown in Table 1.

4. Conclusion

The purpose of this paper is to implement the SDN technology on raspberry pi, and configure with traffic shaping and traffic limitation. The test is performed using a simulation of sending traffic and reading the bandwidth results using iperf. The test results show that there are no obstacles in implementation and the results of testing the delivery of traffic are in accordance with what was planned, where the traffic is based on ports (41995, 5001 and 80) having rate limits of 200kbits (port 41995), 100kbits (port 5001) and 200kbits (port 80).

References

[1] Diorio Fernando R., Timoteo Salvador V. (2016). *Multimedia Content Delivery in OpenFlos SDN: An Approach Based on a Multimedia Gateway*. CPS 978-1-5090-5510-4.

[2] Seddiki Said, M., Shahbaz, M., Donovan, S., Grover, S., Par, M., Feamster, N., and Song, YQ. (2014). *FlowQOS: QoS for the Rest of Us*. ACM 978-1-4503-2989-17/14/08.

[3] Virtualisasi dan Hypervisor. *Buku Komunitas SDN-RG*. [Online]. Available at : https://eueung.gitbooks.io/buku-komunitas-sdn-rg/content/pengantar_openstack/virtualisasi_dan_hipervisor.html. [Accessed 15 January 2020].

[4] Suryadinata, W. (2016). *Cloud Computing*. [Online]. Available at : https://sis.binus.ac.id/2016/12/16/cloud-computing/, [Accessed 5 January 2020].

[5] Pfaff B., Pettit J., Koponen., etc. (2015). *The Design and Implementation of Open vSwitch*. USENIX Symposium on Networked Systems Design and Implementation (NSDI '15). 117 – 130.

[6] Fernandez, C., Munoz, L. J., *Software Defined Networking (SDN) with OpenFlow 1.3, Open vSwtich and Ryu.*

[7] Open vSwitch. (2016). *Open vSwitch Advanced Features*. [Online]. Available at : http://docs.openvswitch.org/en/latest/tutorials/ovs-advanced/, [Accessed 2 January 2020].

[8] Ip Tables Classify. (2006). *Implementing Quality of Service with iptables CLASSIFY rules.* [Online]. Available at : http://www.telogic.com.sg/Imagestream_Tech_QOS%20with%20iptables.html. [Accessed 2 January 2020].

[9] HTB. (2002). *Hierachical Token Bucket Theory*. [Online]. Available at : http://luxik.cdi.cz/~devik/qos/htb/manual/theory.htm. [Accessed 4 January 2020].

[10] Iperf Utility. (2019). *Iperf Manual*. [Online]. Available at : http://manpages.ubuntu.com/manpages/xenial/man1/iperf.1.html. [Accessed 5 January 2020].