JOIN (Jurnal Online Informatika)
Volume 5 No.1 | June 2020: 115-120
DOI: 10.15575/join.v5i1.382

p-ISSN: 2528-1682
e-ISSN: 2527-9165

Reactive Forwarding and Proactive Forwarding Performance Comparison on SDN-Based Network

Galura Muhammad Suranegara [1], Ichwan Nul Ichsan [2], Endah Setyowati[3]
[1,2,3] Telecommunication System Department, Universitas Pendidikan Indonesia, Bandung, Indonesia

Article Info

Article history:

Received October 08, 2019
Revised July 10, 2020
Accepted July 14, 2020
Published July 15, 2020

Keywords:

Open network operating system
Proactive forwarding
Reactive forwarding
Software-defined networking

ABSTRACT

Software-defined networking (SDN) technology is one key technology in telecommunications networks that are currently widely studied. In SDN-based networks, the controller holds the key to the reliability of a network because all control functions are held by the controller as one of the open-source controllers, the Open Network Operating System (ONOS) has two forwarding mechanisms, namely reactive forwarding, and proactive forwarding. This study compares performance between reactive forwarding and proactive forwarding on ONOS with VM migration as traffic. The parameter measured is the total migration time using simple network topology. From the test results, the proactive forwarding scenario can optimize the fastest potential of the topology tested by using the path that has the largest bandwidth available on the network topology. In comparison, reactive forwarding can only pass through the smallest hops of the tested topology. From the measurement results, the average migration time performance using a proactive forwarding scenario is 36.16% faster than the reactive forwarding scenario.

Corresponding Author:

Galura Muhammad Suranegara
Telecommunication System Department,
Universitas Pendidikan Indonesia,
Bandung, Indonesia
Email: galurams@upi.edu

1. INTRODUCTION

The use of cloud computing technology is becoming more and more popular in the information technology industry because of its flexibility and scalability. The flexibility and scalability offered by cloud computing can be delivered because of abstraction and virtualization technology that already can be used for various functions in the field of computing and networking.

One of the technologies used to support flexibility and scalability in cloud computing technology is virtual machine (VM) technology. One of the main features of a VM is its ability to be transferred from a physical machine (PM) to another PM [1]. In this study, the VM is used to replace the role of the physical server (PS), and we use the Kernel-based Virtual Machine (KVM) as our main migration handler because of it is the ability to customize multiple migration scheme [2].

Besides VM technology, software-defined networking (SDN) technology is also widely used as one of the supporting technologies to provide service flexibility and to facilitate network management. The main difference between SDN and conventional networks is the separation of the control plane from the data plane [3]. With this approach, the user or network administrator has the full ability to control the network by programming. Currently, SDN has separated the control function from the device into a single entity to control all SDN devices connected, named controller. All forwarding mechanisms that were initially done by forwarding devices are converted into an application that runs on the SDN controller.

In this study, Open Network Operating System (ONOS) is used as an SDN network controller, and Open vSwitch (OVS) is used as a virtual switch on the SDN network.

To do the forwarding task, ONOS was equipped with reactive forwarding as a default forwarding mechanism. This mechanism installs all forwarding entries to all OVS that connected to the network. The entry can be installed by request after the sender starts sending the first packet. Another alternative for forwarding is proactive forwarding. In this mechanism, the installation of forwarding entries is installed before the sender sends the packet [4].

In this study, OVS is used as a virtual switch that runs OpenFlow as a protocol on SDN networks. OVS is used as a form of virtualization from conventional switches. The network topology in this study built by Mininet. Mininet is an application that can create a realistic virtual network, running real kernel, switch, and application code, on a single machine (VM, cloud, or native), in seconds, with a single command [5].

This study used multiple VM as virtualization from PS and nested VM as host for simulating personal computer (PC). For doing the migration task, this study uses a warm migration scenario. This scenario moves the entire VM from one PS to another PS persistently and identically, then removes the VM that has been moved to the original PS [6]. While VM migration task executed, there is some service interrupt occurred because the VM is paused.

The measurement of total migration time is done by using a bash script that starts when the VM migration is executed and ends when the entire migration process is complete. VM migration task is accomplished twice using different forwarding schemes, reactive forwarding, and proactive forwarding schemes. Both forwarding schemes are using the same environment design and the same network topology.

This study focused on comparing two forwarding mechanisms, reactive forwarding mechanism, and proactive forwarding mechanism on the ONOS controller to determine the best performance for migrating VMs on the SDN-based network. From the measurement results, at average proactive forwarding mechanism is 36.16% faster than the reactive forwarding scenario.

2. METHOD

, the network must be built using a full SDN environment and uses the OpenFlow protocol. This method is implemented to find out the best performance between reactive forwarding and proactive forwarding. This study uses open source software for all required software in which the source code can be accessed and modified freely by all users.

The first step done in this research is creating an SDN environment. The environment is fully made by utilizing virtualization functions. The environment is separated into five sections to make networking easier. It is based on its functions, namely Host 1 and Host 2, Switch VM, Controller VM, and Management VM.

After creating the SDN environment, then the topology is configured by utilizing the Mininet application. The topology created is only a simple topology consisted of three virtual switches with OpenFlow as a protocol. After the topology ready to be connected to the ONOS controller, reactive and proactive forwarding mechanisms are used interchangeably after the total migration time has been measured.

Total migration time is measured by a timer built through a bash script. The timer starts when the migration task is executed and ends when the whole migration task completed. For the migration task, this study uses a warm migration scenario. The warm migration scenario actually transfers the entire VM from a PS to another, and the migrated VM is persistent. This means that the migrated VM can be used as a normal VM and stored in the destination virtualization PS storage. All VMs that have been stored on destination PS storage is removed from the original PS virtualization. This study also uses three different VM size to be migrated.

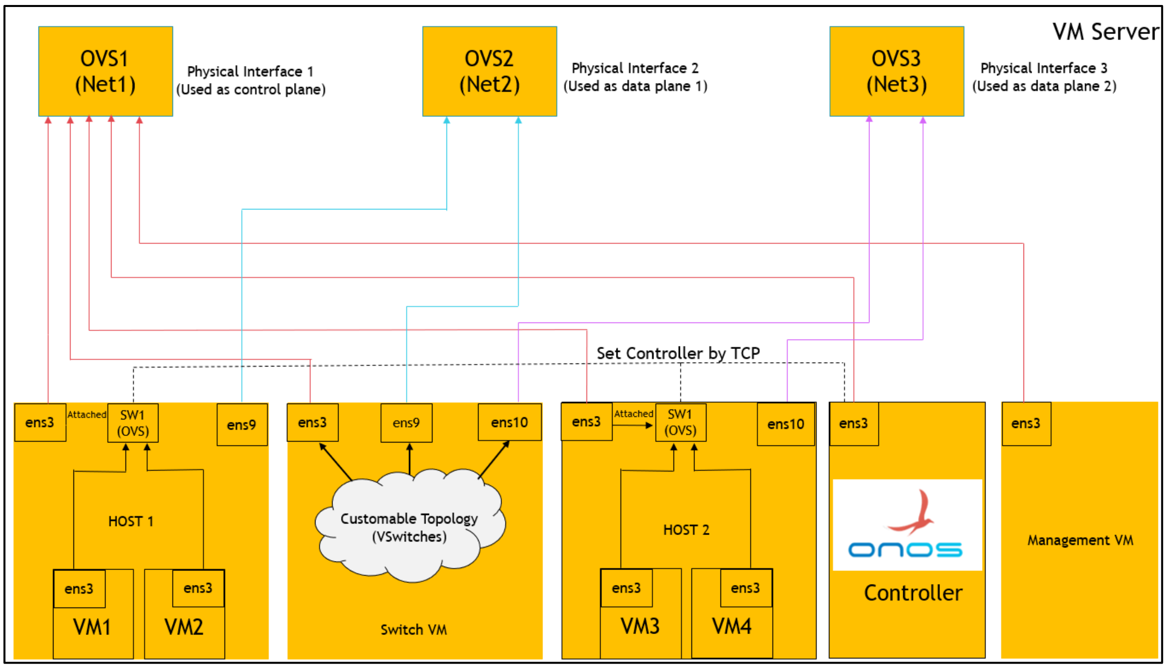
2.1 ENVIRONMENT DESIGN

The entire environment design in this study was built by using SDN-based environments and open-source platforms. Software-defined networking (SDN) is a new paradigm that performs abstraction by separating the control plane from the data plane. With this approach, the network developer or network administrator can get the full ability to manage the network by programming. Then the separated control plane is made into a stand-alone entity, commonly known as a controller.

ONOS is one of the opensource-based controllers available and can be used on SDN-based networks. ONOS can be used to serve communication among hosts to perform SDN-based network routing, management, and monitoring functions.

Open vSwitch (OVS) is a production quality, multilayer virtual switch licensed under the open-source Apache 2.0 license [7]. OVS is designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols (e.g., NetFlow, sFlow, IPFIX, RSPAN, CLI, LACP, 802.1ag). In addition, it is designed to support distribution across multiple physical servers similar to VM ware's vNetwork distributed switch or Cisco's Nexus 1000V.

This study used Kernel Virtual Machine (KVM) for virtualizing PC. Since KVM is an open-source, the migration method can be configured freely by following the general QEMU development guide [8][9]. The design environment used in this study is as follows:


VM Server
OVS1 (Net1)
Physical Interface 1 (Used as control plane)
OVS2 (Net2)
Physical Interface 2 (Used as data plane 1)
OVS3 (Net3)
Physical Interface 3 (Used as data plane 2)
Set Controller by TCP
ens3
Attached
SW1 (OVS)
ens9
HOST 1
VM1
VM2
Customable Topology (VSwitches)
Switch VM
ens10
HOST 2
VM3
VM4
ONOS
Controller
Management VM
Figure 1. Environment Design

Based on figure 1, it can be seen that the environment in this study was built following the main principle of SDN, which is the separation of the control plane and the data plane. All control plane traffic is only allowed to pass physical interface one while data plane traffic is only allowed to pass physical interface 2.

Host 1 is used as the original physical server (PS), while Host 2 is used as the destination PS. Mininet installed inside the VM Switch performs all functions related to routing, switching, and all other matters relating to network topology settings. The Controller VM is used to be a controller that regulates the use of reactive forwarding or proactive forwarding scenarios. VM management is used to provide migration execution commands and measure the total migration time of a VM. Migration commands are done by python commands [10] and built into three migration scenarios, cold migration [11], warm migration [12], and live migration [13].

2.2 NETWORK TOPOLOGY IMPLEMENTATION

This study uses a Mininet application to build network topology. Mininet creates some OVS and also create a link among OVS created. Each OVS created, communicate using the OpenFlow protocol. Mininet can manage bandwidth link size through programming but can not exceed its physical network interface [14]. This is the simple network topology used in this study:

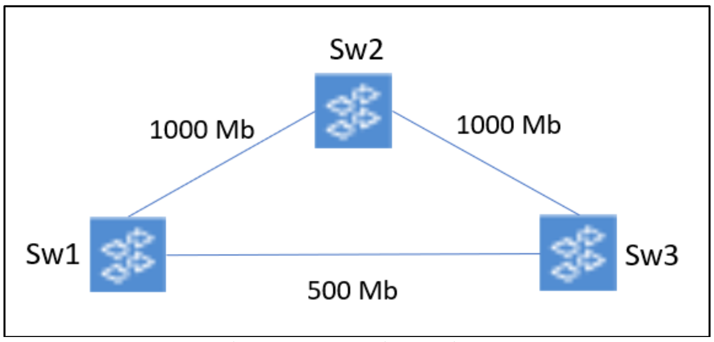


Figure 2. Network Topology

Based on figure 2, the topology used in this study only uses 3 OVS; each link has a different bandwidth limit. A-B and B-C links have 1 Gb bandwidth, while A-C links have 500 Kb bandwidth.

A. PROACTIVE FORWARDING IMPLEMENTATION

Beside implementing reactive forwarding, this research also implementing a proactive forwarding mechanism for all virtual switches connected to ONOS. The execution between reactive forwarding and proactive forwarding in this study are done one by one. This is the ONOS's view on the implementation of proactive forwarding:

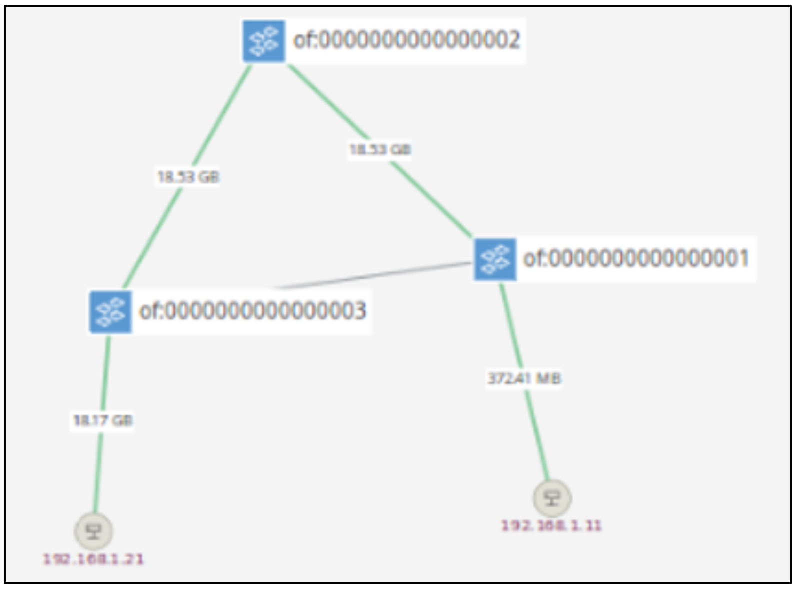


Figure 3. Proactive Forwarding Implementation

The proactive forwarding mechanism is implemented to take advantage of the maximum benefits of the network topology. Unlike reactive forwarding mechanism that always passes through the smallest hop without considering the maximum benefits of network users can have.

2.3 REACTIVE FORWARDING IMPLEMENTATION

Reactive forwarding implementation is done at the ONOS controller. By default, the ONOS controller already has a reactive forwarding mechanism but must be activated manually [15]. The reactive forwarding mechanism can not be configured. It will forward any packet through connected OVS. Following is the picture when reactive forwarding is implemented:

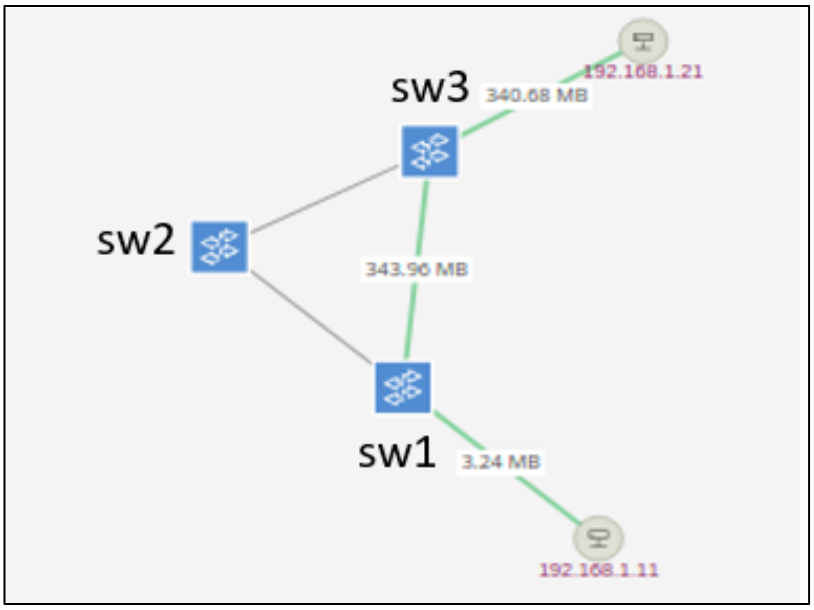


Figure 4. Reactive Forwarding Implementation

Figure 4 shows the implementation of reactive forwarding has been successfully implemented. It also can be seen by observing the data flow, as shown above. By default, if there is no forwarding mechanism activated, there will be no data flow on the SDN network. The reactive forwarding mechanism can be activated by turning it on at the ONOS controller.

3. RESULTS AND DISCUSSION

This study uses total migration time as the main parameter of measurement. After all of the VM samples measured, then the average migration time is used as a measurement result. The total migration time is obtained from a timer created using a bash script when the migration is executed. VM size samples that are migrated are 1GB, 2GB, and 5GB. Following is the comparison of the results from the measurement result from both reactive forwarding and proactive forwarding mechanism conducted in this study:

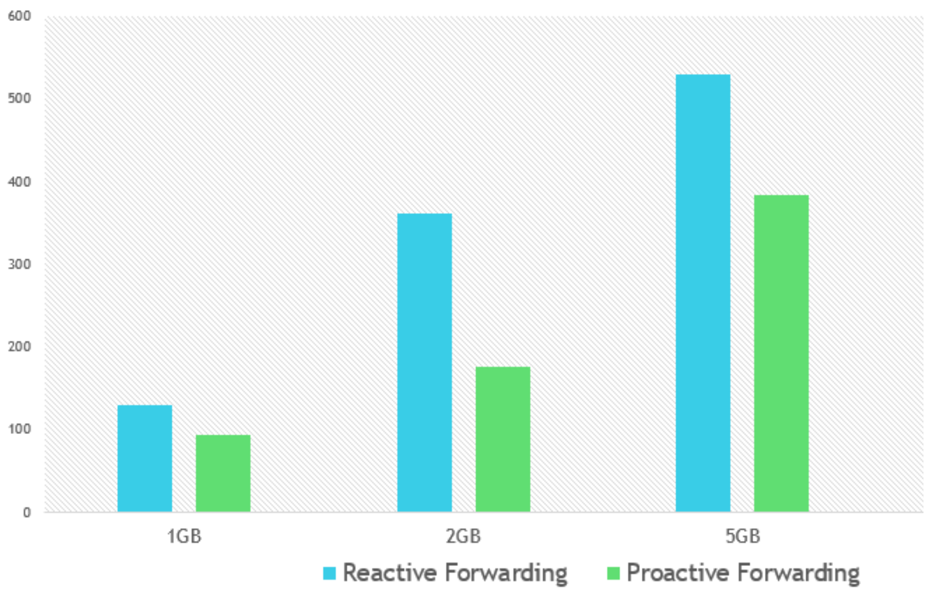


Figure 5. Migration Time Measurement Result

From figure 5, it can be seen that the migration time required by the reactive forwarding mechanism is greater than proactive forwarding at any different VM size sample.

The migration time with a proactive forwarding mechanism on a 1GB VM is 27.83% faster than the reactive forwarding mechanism, for a 2GB VM, the migration time with proactive forwarding mechanism is 51.39% faster than the reactive forwarding mechanism. And for 5GB VM, the migration time with a proactive forwarding mechanism is 27.79% faster than the reactive forwarding mechanism.

4. CONCLUSION

From the measurement results from this study, it can be concluded that the VM migration process on the SDN-based network that uses a proactive forwarding mechanism is faster than the reactive forwarding mechanism. By average measurement, the proactive forwarding mechanism is 36.16% faster than the reactive forwarding mechanism in terms of migrating 1Gb, 2GB, and 5GB VMs on the SDN-based network.

5. REFERENCES

[1] Kernel Virtual Machine Documentation, URL: https://www.linux-kvm.org/page/Documents (January 2020)
[2] Suranegara, G. M., Marenda, D. A., Hakimi, R., Risdianto, A. C., & Mulyana, E. (2018). Design and Implementation of VM Migration Application on SDN-Based Network. Proceeding of 2018 4th International Conference on Wireless and Telematics, ICWT 2018, 1–6. https://doi.org/10.1109/ICWT.2018.8527782
[3] SDN Architecture Overview White Paper, ONF. URL: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-521_SDN_Architecture_issue_1.1.pdf, (January 2020)
[4] ONOS, User Documentation Guide, ONOS. URL: https://wiki.onosproject.org, (January 2020)
[5] Mininet Documentation, Mininet. URL: http://docs.mininet.org, (January 2020)
[6] Singh, H., Next-Gen Virtualization, VMWare, URL: https://www.vmware.com/topics/glossary/content/virtual-machine, (January 2020)
[7] Finucane, S. "Why OpenvSwitch?". URL: https://github.com/openvswitch/ovs/blob/master/Documentation/intro/why-ovs.rst, (January 2020)
[8] "QEMU version 2.12.0 User Documentation," URL: https://qemu.weilnetz.de/doc/qemu-doc.html, (January 2020)
[9] Shah, Amit. "Kernel-based virtualization with KVM". Deep Virtue, issue 86, pp 37-39. January 2008.
[10] W.D. Ashley, D. Berrange, C. Lalancette, et.al. "Libvirt Application Development Guide Using Python". Version 1.1, 29 May 2001.

[11] VMWare, “Cold Migration,” vSphere 5 Documentation Center. URL: https://pubs.vmware.com/vsphere-50/index.jsp, (January 2020)
[12] Veritas, “Overview of a Warm Migration,” Oracle VM Server for SPARC Guest Domain Migration. URL: https://sort.veritas.com/ public/documents, (January 2020)
[13] U. Lublin, Q.A. Liguori, IBM. “KVM Live Migration,” KVM Forum, 2007
[14] Mininet Pythion API Reference Manual, URL: http://mininet.org/api/classmininet_1_1link_1_1TCIntf.html, (January 2020)
[15] ONOS SDN-API Reactive Routing, URL: https://wiki.onosproject.org/display/ONOS/SDN-IP+Reactive+Routing, (January 2020)