Docker-based Network Functions Virtualization as a Learning Tool in Computer Network Course

Bagus Jati Santoso, Royyana Muslim Ijtihadie, Muhammad Al Fatih Abil FIda
Department of Informatics,
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
{bagus, roy}@if.its.ac.id, alfatih.14@mhs.if.its.ac.id

***Abstract*— Recently, a network device virtualization concept called Network Functions Virtualization (NFV) has been developed and utilized in industries. Several companies, especially the telecommunications sector, have played a role in developing and implementing various software options. Virtualization on NFV can be done using Docker Software, which continues to experience updates. Therefore, this article offers the implementation of NFV to the present alternative learning tool for computer network courses. Over the conducted experiments, the intended implementation is able to fulfill the practical requirement of the Computer Network course. Moreover, the experimental results show that shows that the performance parameters are increasing along with the increment of the number of users in a reasonable manner.**

Keywords—Docker, Network Function Virtualization, Learning Tool, Computer Networks

I. INTRODUCTION

Computer network is one of the courses studied in lectures in the area of computer science. In lectures on computer networks, lab work is one of the learning methods that are useful to ensure students have competence in implementing theory into practice.

In most computer science lectures, lab work from computer networking courses utilizes User Mode Linux (UML) as a medium for simulating virtual computer networks. The UML installation is done on a Linux operating system. Unfortunately, over time, the development of computer operating systems was not accompanied by the development of UML. When the operating system of the lab server is about to be updated to a more up-to-date version, UML is not fully functional. This is due to the system daemon transition from init.d to systemd. So, with the fact that Debian 7 entered the end of life date in May 2018, an alternative learning tool is needed for lab work from computer networking courses.

On the other hand, the European Telecommunications Standard Institute (ETSI) formulated a concept called Network Functions Virtualization (NFV) recently. The concept has become popular among companies because of the need for virtualization from network device functions. The concept theoretically has a broad compatibility with various software to compile NFV components.

Therefore, to present an alternative Computer Network practicum tool, the implementation of NFV is done with the docker as a virtualization platform, replacing UML-based virtualization. The decision to choose a docker in implementing NFV was based on the fact that the software would continue to be developed and appropriately maintained. In addition, the application of container technology to the docker, in its concept, has a lighter burden on computer resources compared to the UML, which uses a virtual machine.

This paper discusses the implementation of NFV with the docker virtualization platform to produce learning tools for lab work from Computer Network courses. Some related works are discussed in the Section II. Section III discusses the problem background of this work. In Section IV, the design and implementation of the proposed work are discussed. The performance evaluation and its results are intensively discussed in Section V. Finally, Section VI summarizes all the discussion over this work.

II. RELATED WORKS

Recently, Docker has been intensely being used and utilized in supporting the research based on virtualization on computer and network devices. Boettiger in [1] examines the Docker in supporting research execution by combining cross-platform portability, re-usable modular elements, versioning, virtualization, and philosophy of 'DevOps'. In [2], Ismail et al evaluate the performance of Docker as a platform for the Edge Computing. In their work, evaluation and experiment show that Docker provides a good performance, fast deployment, and small footprint to support Edge Computing as a platform.

There also exist some works that employ Docker as a platform for the implementation of microservices architecture. Stubbs et al review the challenge over microservice architecture, review the container technology and introduce Serf, a non-intrusive Docker image, in [3]. Jaramillo et al in [4] discusses the effectiveness of Docker in leveraging microservices architecture with a real working model by minimizing the complexity over development.

Moreover, Xingtao et al in [5] utilizes Docker to develop the Software-Defined Networking (SDN) controller in presenting network virtualization. Their conducted experiment shows that the mobility, the extensibility and the speed of deployment over network virtualization are improved.

III. PROBLEM BACKGROUND

A. Lab Work of Computer Networks Course

The lab work of computer networks course mainly discusses and practices the basic networking skills, for example, subnetting, routing, Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP), proxies, and firewalls. To practice the setup over subnetting and routing, the network functions that needed are routers. Whereas DNS,

DHCP, proxy, and Linux firewalls are carried out on end-point networks, which means relying on hosts.

B. Network Functions Virtualization

Network functions generally refer to network infrastructure components that provide specific functions such as intrusion detection, intrusion prevention, routing, and others. These network functions are usually manifested in physical form and require manual installation into a computer network.

In the telecommunications industry, virtualization technology was developed and applied to complement physical infrastructure. One of the developments of virtualization technology is the presence of the concept of Network Functions Virtualization (NFV). The concept initiated by the European Telecommunications Standard Institute (ETSI) has an architecture depicted in Figure 1 [6].

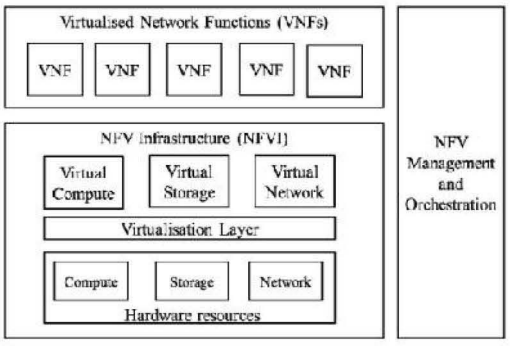


Fig. 1. NFV Architecture. Source: [6]

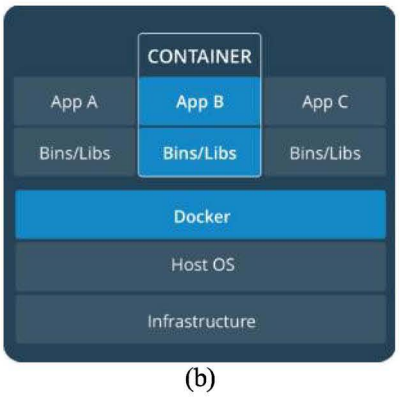
Broadly speaking, the parts in NFV and their functions are: Virtual Network Functions (VNF) as a network function virtualized by NFVI; Network Functions Virtualization Infrastructure (NFVI) as a hardware and software component that provides an environment where VNF is built and performs virtualization; and Management and Orchestration (MANO) as a regulator of the interaction between VNF and resources on NFVI, that also perform operations such as initialization and termination on VNF.

NFV components can be implemented with a variety of software. One of them is the virtualization process, there are several options that can be used. For example hypervisor (hypervisor-based), container or mixed-based virtualization (mixed technologies) [7].

C. Docker

Docker is a software that is developed to package applications in a special form called containers. This container technology bears similarities to the hypervisor technology that uses virtual machines such as VMware, VirtualBox, etc. in terms of establishing an isolated environment. The most distinguishing thing is that virtual machines require an operating system in each virtual unit, so there is always an operating system above the operating system and kernel above the kernel. While containers run like other applications that use the kernel on their host operating system. This difference makes the docker lighter than virtual machines [8]. The illustration of the hypervisor architecture can be seen in Figure 2(a), while the illustration of container architecture can be seen in Figure 2(b) [9].

(a)



(b)

Fig. 2. Hypervisor and Docker Architectures. Source: [9]

IV. Design and Implementation

A. Use Cases

In order for the system to be built according to the requirement of the Computer Network lab work, a use case is formulated to be implemented, depicted in Table 1. In the system, there are only two actors: Admin and User. The admin role is more towards doing account management and facilitating users. While the user is the student who uses the system as the learning tool.

TABLE I. Use Cases Of The System

Use Case	Description
Creating a user account	Admin is able to create a new user account
User management	Admin is able to change details and delete user accounts
Log in	Users are able to enter the system with the specified username and password
Network topology workspace	The user is able to see the network topology that he built
Deploying host	Users are able to make VNF hosted through the system
Deploying router	Users are able to create VNF routers through the system
Deploying switch	Users are able to make VNF switches through the system
Remote access	Users are able to do remote access to enter the bash terminal for each VNF host or router to make further configuration as needed. This use case depends on creating a VNF host or router.
Ping test	Users are able to test the connectivity between VNFs by pinging the provided system
VNF management	Users are able to turn on, turn off, delete and rename VNF

B. System Design

Figure 3 shows the architecture of the proposed system, that was built as a learning tool of computer networks course. The proposed system is a system that is capable of managing VNF, which regulates the life cycle (making and deleting), turning on, configuring, and turning off VNF independently for system users. All the VNF settings in the figure are translated by the flask web server into a command to be executed by the docker on the server provided.

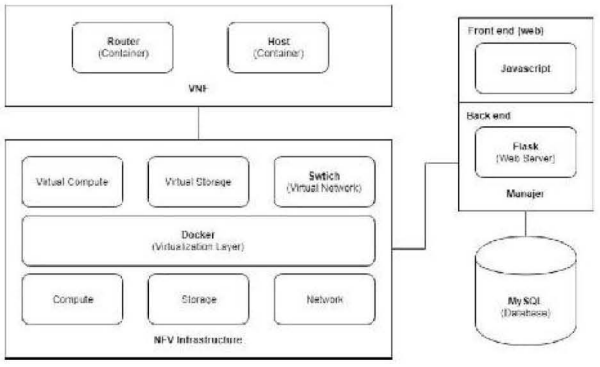


Fig. 3. The architecture of the proposed system.

Due to network function requirements that practiced in the computer networks course are limited to hosts, routers, and switches, then in this system, the host or end-point network will be implemented as a container that based on Debian docker images. Further, switches are implemented in the form of network dockers (docker network) from a VNF router. Finally, the router is implemented in the form of a container that is able to create networks and other containers internally with a docker image base called Docker-in-Docker.

The system also provides a web interface display to show the simulation of network topology made by the user visually. The visual simulation is performed over HTML canvas and animated with JavaScript. Each host and switch connected to the router is described as a single device. Management of VNF by users is done through the web interface. Users are also able to test the continuity between VNF hosts, between VNF routers, also VNF hosts and routers by performing a ping Test which is also available on the system. Remote access (remote access) into VNF hosts and routers, in the form of bash terminals, is also possible to perform further configuration so that lab work requirement can be met, such as installing web server applications, DHCP servers, and other tasks.

To support the visual simulation of computer network topology, the system uses the MySQL database to store the information about the details of VNF that has been created, including the last coordinates of the location of each VNF on the HTML canvas. There exist three tables to support the system, namely containers, networks, and users. Details of the attributes of each table are depicted in Tables 2, 3 and 4.

TABLE II. TABLE STRUCTURE OF CONTAINERS

Attribute	Description
id	The ID of the containers
name	Containers name
parent_id	The ID of docker in which switch exist
user_id	The ID of users that create the networks
type	The type of container
network_id	The network id of networks that contained by the containers
axis	A point in the x-axis of HTML canvas that points to the last position of the switch
ordinate	A point in the y-axis of HTML canvas that points to the last position of the switch

TABLE III. TABLE STRUCTURE OF NETWORKS

Attribute	Description
id	The ID switch that is generated by the docker
name	Switch name
parent_id	The ID of docker in which switch exist
user_id	The ID of users that create the networks
subnet	The subnet of networks that covered by the switch
gateway	The address of the IP interface of the gateway router or server
axis	A point in the x-axis of HTML canvas that points to the last position of the switch
ordinate	A point in the y-axis of HTML canvas that points to the last position of the switch

TABLE IV. TABLE STRUCTURE OF USERS

Attribute	Description
id	The ID of the user
name	User name
password	The password of the user

C. Design of Virtualization

To run a computer network simulation with a docker, the VNF router uses a special docker image called Docker-in-Docker (DinD). Docker image DinD allows containers to run docker functions such as creating their own containers, creating their own docker network and other tasks. In other words, the container docker functions as a docker engine.

VNF routers require this capability so that the built-in virtual network is completely isolated but still allows the routing task. Since the virtual network created by the docker basically does not open the opportunity for routing, one container is not able to be connected directly to another container. This means that the network gateway docker is going to point to the parent of the container, namely the machine that runs the docker (docker engine) itself. As a result, there exist inability to perform network simulation as well as building a computer network physically which always has network levels, as long as it only relies on one docker engine on the server. The specific designs regarding virtualization architecture are depicted in Figure 4.

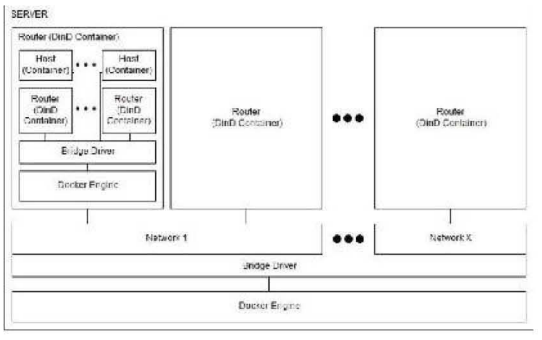


Fig. 4. The architecture of the virtualization.

V. Experimental Evaluation

The experimental evaluation is conducted out on the servers that running the Xubuntu Desktop 16.04 operating system with the hardware specifications of the Intel (R) Core (TM) i3 CPU @ 2.40GHz - 4 Logical Processors, 6GB of RAM, 6.5GB of swap and hard disk space at 91GB. Moreover, the system supporting software installed is MySQL 5.7, Flask 1.0, Python 2.7 and Docker 18.06.1.

The experiments are carried out in using performance tests. Performance tests are intended to measure the ability of a testing server to handle requests from client systems based on the use of computer resources (RAM, hard drive and processor) and the response time needed by the server.

The experiment is conducted by employing the python programming script that sends the requests to the server to create a network topology, as shown in Figure 5. The intended topology consists of five VNF switches, three VNF routers, and two VNF hosts. This experiment is carried out by varying the number of clients set using the python script. The variants chosen are 1, 2, 3, 5, and 8 clients.

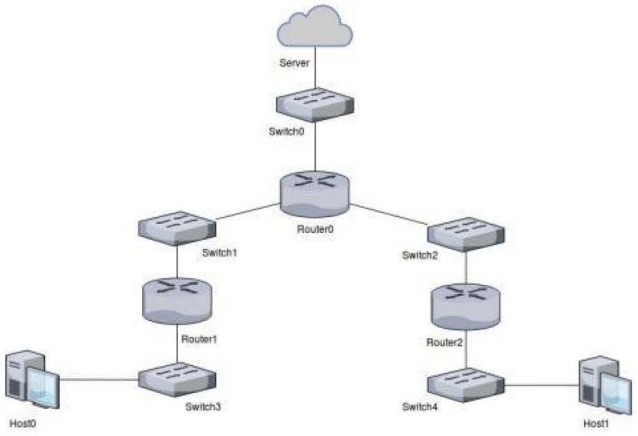


Fig. 5. Network topology for the experiment.

The experimental result for this part is summarized in Figures 6, 7, 8, and Table 5. Figure 6 shows the memory usage versus the number of users. Figure 7 shows the usage over the hard drive versus the number of users. While Figure 8 shows the usage of the swap resources. Finally, Table 5 shows the time duration of the script execution until the topology completed.

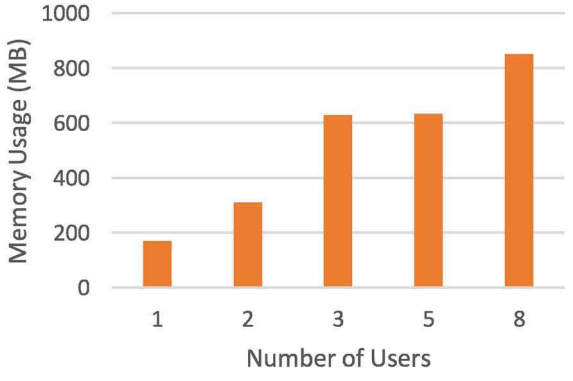


Fig. 6. Number of users vs. memory usage

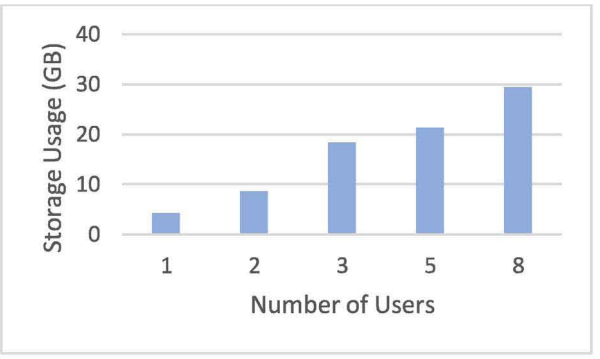


Fig. 7. Number of users vs. storage usage.

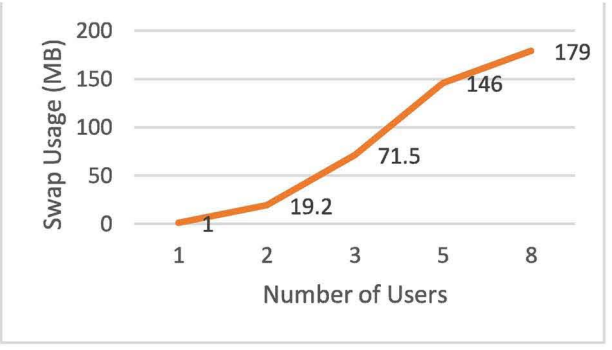


Fig. 8. Number of users vs. swap usage.

TABLE V. Execution Time of The Network Topology Script

Number of Users	Execution Time (s)
1	112.07
2	186.71
3	328.87
5	938.51
8	2065.34

It can be seen that the use of server resources is indeed increasing as the number of users increases. However, it is essential to note that the use of swap space occurs when there exists more than one user. This means that the memory is insufficient to handle client requests. Even the total memory available is 6GB, while the usage itself does not exceed 1.5GB. After tracing it, it seems that RAM is full, forcing the server to use swap space due to the large system cache generated by the docker. This ultimately affects the record time for completing client requests to the server, because when the swap has been used, the process runs slower.

As for the use of hard disk space, it is depicted that consumption for one whole topology can reach almost 5GB. This number is quite large since the containers (hosts and routers) involved only number five. This condition is obtained because of the process of loading the docker image before creating a host or router takes place on each router. While the docker image size of the DinD used to make the router itself reaches 198MB, and the size of the Debian docker image used to create hosts reaches 101MB.

VI. CONCLUSIONS

In this work, the implementation of Network Functions Virtualization (NFV) is done by using Docker as a learning tool for computer networks course. Router and host network functions are implemented in the form of Docker containers, while the network switch functions are implemented in the form of a network docker. Moreover, for routers, a special docker image must be used in which is able to perform the docker function properly, namely a Docker-in-Docker (DinD). Moreover, the NFV then implemented by using a Flask web server that acts as a manager in the NFC concept. Finally, the experimental evaluation shows that the performance parameters (running time, memory usage, storage usage, and swap usage) are increasing along with the increment number of users in a reasonable manner.

REFERENCES

[1] Carl Boettiger. 2015. An introduction to Docker for reproducible research. SIGOPS Oper. Syst. Rev. 49, 1 (January 2015), 71-79. DOI:

[2] B. I. Ismail et al., "Evaluation of Docker as Edge computing platform," 2015 IEEE Conference on Open Systems (ICOS), Bandar Melaka, 2015, pp. 130-135.

[3] J. Stubbs, W. Moreira and R. Dooley, "Distributed Systems of Microservices Using Docker and Serfnode," 2015 7th International Workshop on Science Gateways, Budapest, 2015, pp. 34-39.

[4] D. Jaramillo, D. V. Nguyen and R. Smart, "Leveraging microservices architecture by using Docker technology," SoutheastCon 2016, Norfolk, VA, 2016, pp. 1-5.

[5] L. Xingtao, G. Yantao, W. Wei, Z. Sanyou and L. Jiliang, "Network virtualization by using software-defined networking controller based Docker," 2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference, Chongqing, 2016, pp. 1112-1115.

[6] European Telecommunications Standard Institute. (2014, December). ETSI GS NFV 002 V1.2.1 (2014-12) [Online]. pp. 10. Available: http://www.etsi.org

[7] European Telecommunications Standard Institute. (2016, March). ETSI GS NFV-EVE 004 V1.1.1 (2016-03) [Online]. pp. 7-13. Available: http://www.etsi.org

[8] Docker Inc. (2018). What is a Container. [Online]. Available: https://www.docker.com/resources/what-container

[9] Docker Inc. (2018). Get Started, Part 1: Orientation and setup. [Online]. Available: https://docs.docker.com/get-started/