ELKOMIKA | ISSN (p): 2338-8323 | ISSN (e): 2459-9638 | Vol. 6 | No. 2 | Halaman 180 - 193
DOI : http://dx.doi.org/10.26760/elkomika.v6i2.180 Mei 2018

Analisis Performansi Layanan *FTP* dan *Video Streaming* berbasis *Network Function Virtualization* menggunakan *Docker Containers*

MANZILA IZNIARDI DJOMI, RENDY MUNADI, RIDHA MULDINA NEGARA

S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
Email: manzilaardi@gmail.com

Received 23 April 2018 **|** *Revised* 14 Mei 2018 **|** *Accepted* 25 Mei 2018

ABSTRAK

Infrastruktur jaringan seperti router, secara tradisional menggunakan hardware yang bersifat proprietary. Teknologi virtualisasi pada fungsi jaringan atau NFV (Network Function Virtualization) membuat layanan ini dapat diimplementasikan sebagai aplikasi perangkat lunak yang dapat dijalankan di lingkungan virtual atau Virtualized Network Functions (VNFs). Selain menggunakan hypervisor (hardware-level virtualization), teknologi virtualisasi memiliki alternatif pengimplementasian dengan menggunakan teknologi containers (Operating system -level virtualization), salah satunya menggunakan Docker. Penelitian ini mengimplementasikan layanan FTP dan video streaming pada jaringan NFV di Docker Containers. Tanpa backgound traffic, layanan menunjukkan performansi QoS yang memenuhi standarisasi ITU-T G.1010 dengan delay FTP 0,12 ms dan delay video streaming 6,21 ms serta nilai packet loss kedua layanan sebesar 0%. Penggunaan CPU pada Docker ketika layanan dijalankan dibawah 1 %.

***Kata kunci**: Virtualisasi, Containers, Docker, Network Function Virtualization, QoS*

ABSTRACT

Network infrastructure such as routers, traditionally using proprietary hardware. Virtualization technology on network function or NFV (Network Function Virtualization) makes this service can be implemented as a software application that can run in virtual environment or Virtualized Network Functions (VNFs). In addition to using hypervisor (hardware-level virtualization), virtualization technology has an alternative implementation using containers technology (Operating system-level virtualization), one of them using Docker. This research implements FTP and video streaming services on NFV networks in Docker Containers. Without background traffic, the service demonstrates QoS performance that meets the ITU-T G.1010 standardization with 0.12 ms FTP delay and 6.21 ms video streaming delay and with packet loss value of both services at 0%. CPU usage on Docker when service runs below 1%.

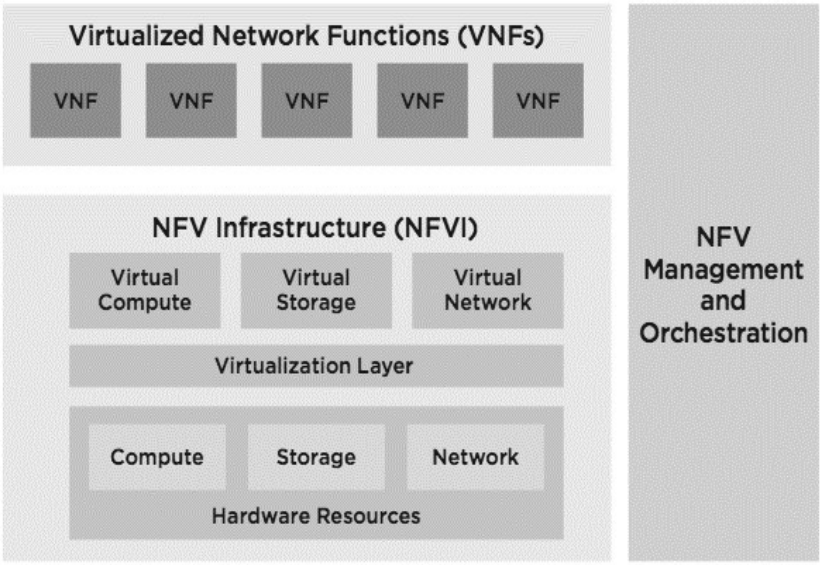
***Keywords**: Virtualization, Containers, Docker, Network Function Virtualization, QoS*

1. PENDAHULUAN

Pada jaringan telekomunikasi tradisional, layanan seperti *routing* dan *firewall* umumnya di implementasikan menggunakan perangkat keras (*hardware)* khusus yang bersifat *proprietary*. Dengan NFV *(Network Function Virtualization)*, layanan ini dapat diimplementasikan sebagai aplikasi perangkat lunak (*software*) yang dapat dijalankan di lingkungan virtual yang disebut sebagai *Virtualized Network Functions (VNFs)* **(Falkner, 2016)**. NFV dipercaya mampu mengurangi beban biaya operasional, serta memudahkan dalam mengelola fungsi dari suatu jaringan. Konsep NFV hadir membawa paradigma baru yang diaplikasikan untuk mengelola dan mengimplementasikan layanan pada jaringan dengan memisahkan fungsi pada jaringan dari peralatan perangkat keras *(hardware)* yang dapat dijalankan di dalam lingkungan *hypervisor* maupun *platform /* teknologi virtualisasi lainnya sebagai infarastuktur yang membangun NFV atau *Network Function Virtualization Infrastucture (NFVI)*. Badan standarisasi *European Telecommunications Standards Institute (ETSI)* mendefinisikan NFVI sebagai kumpulan komponen *hardware* dan *software* yang dapat membangun lingkungan virtual untuk menjalankan layanan VNFs **(European Telecommunications Standards Institute , 2012)**. Layanan infrastruktur *multi-tenant* yang disediakan NFVI dapat mendukung dan memaksimalkan kinerja *hardware* serta dapat menjalankan aplikasi secara bersamaan dengan cara memanfaatkan teknologi virtualisasi.

Network Functions Virtualization (NFV) merupakan sebuah konsep dari arsitektur jaringan yang dapat memvirtualisasikan fungsi setiap *node-node* pada jaringan dengan menggunakan teknologi virtualisasi perangkat *IT (Information Technology)* agar dapat saling terhubung sehingga tercipta sebuah layanan komunikasi. Konsep NFV (*Network Functions Virtualization*) muncul dari para operator telekomunikasi dalam mendukung strategi bisnis dan pertumbuhan pendapatan mereka dengan mencari solusi agar implementasi layanan baru jaringan dapat dilakukan dengan cepat. Salah satu hambatannya adalah ketergantungan terhadap *hardware-based appliance.* NFV bertujuan untuk mengubah cara operator jaringan dalam merancang jaringan dengan mengembangkan standar teknologi virtualisasi untuk mengakomodasi berbagai jenis peralatan jaringan ke standar industri volume tinggi seperti *server, switch dan storage* yang bisa ditemukan di *data centres* serta *node-node* jaringan hingga ke *end user.* Pada Gambar 1 menunjukkan arsitektur dasar dari jaringan NFV.

Pada umumnya, teknologi *hypervisor* yang banyak digunakan dalam virtualisasi, termasuk pada penelitian Aswariza **(Aswariza, 2017)** yang membandingkan performansi terbaik dari beberapa jenis *hypervisor*. Teknologi virtualisasi memiliki alternatif selain *hypervisor* dengan hadirnya teknologi *containers*. Teknologi *containers* pada lingkungan *Linux (Linux Envirovment*) memberikan sistem yang terisolasi (*isolated environment*) pada level OS yang dijalankan pada satu induk *linux kernel* (*host*). Pada penelitian Minh Thanh Chung disebutkan fungsional *containers* ketika bekerja pada *cloud computing* memiliki banyak keuntungan dalam mengurangi *overhead* dan karakteristiknya yang ringan dibandingkan dengan teknologi *hypervisor* pada *Virtual Machines (VMs)* yang memvirtualisasikan dalam level abstraksi *hardware*, sedangkan *containers* adalah emulasi pada level abstraksi *Operating System* (OS) **(Chung, 2016)**. Salah satu *platform* populer yang menerapkan teknologi *containers* yaitu *Docker*. *Docker Containers* memiliki keuntungan besar yang memungkinkan aplikasi dijalankan secara terpisah dari infrastruktur *host* dan memperlakukan infrastruktur seperti aplikasi yang dapat dikelola **(Preeth E N, 2015).**



Gambar 1. Arsitektur NFV

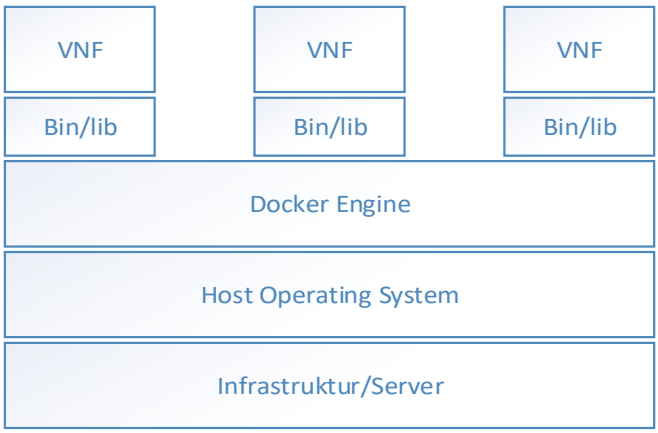
Docker merupakan proyek terbuka *(open-source project)* yang mengembangkan *platform open-source)* untuk para *developer*, *sysadmin,* maupun *devops* agar dapat membangun, mengemas, dan menjalankan aplikasi sebagai wadah *(containers)* dengan ringan **(Docker, 2018).** *Docker* menyediakan kemampuan untuk menjalankan aplikasi dalam lingkungan yang terisolasi. Isolasi dan keamanan memungkinkan untuk menjalankan banyak wadah secara bersamaan pada *host* yang diberikan. Karena sifat ringan dari *containers* yang berjalan tanpa beban tambahan *hypervisor*, kita dapat menjalankan wadah lebih pada kombinasi *hardware* tertentu daripada jika menggunakan mesin virtual. *Docker* merupakan *tool* berbasis *command line* yang memberikan layanan dalam membuat, melakukan manajemen dan mengembangkan *containers*. *Docker* mengijinkan pengguna untuk melakukan akses ke *repository* publik pada internet dimana di dalamnya terdapat beberapa *image* yang siap digunakan.

Beberapa percobaan telah dilakukan dalam menganalisis performansi ketika menjalankan aplikasi pada *containers* yang menunjukan performansi yang lebih baik dibandingkan teknologi *hypervisor*. Salah satunya penelitian yang dilakukan oleh Jason Anderson didapatkan *containers* dapat memberikan *latency* dan *delay* variasi yang rendah, dan dapat dimanfaatkan untuk teknologi jaringan kinerja tinggi yang sebelumnya hanya digunakan pada virtualisasi *hardware* **(Anderson, 2016)**. Lalu pada penelitian **(Felter, 2015)** menyimpulkan bahwa *docker containers* memiliki performansi yang seimbang atau lebih baik dari *hypervisor* berdasarkan parameter *CPU, memory, storage, and networking resources* . Sedangkan pada tahun 2017, **(Aravinthan, 2017)** meneliti implementasi *Radio Access Network (RAN)* untuk pengembangan teknologi 5G menunjukkan *Docker Containers* lebih unggul dibanding VM. Oleh karena itu, dalam penelitian ini menyajikan suatu analisis *containers* yang menjalankan VNFs di atasnya untuk melihat dan menguji performansi yang dihasilkannya. Penelitian ini bertujuan untuk mengukur kehandalan dari *Network Function Virtualization (NFV)* yang dibangun dan dijalankan di atas *Docker Container* yang menjalankan *virtual router* dan dapat terhubung dengan jaringan yang *real*. Dari hasil analisis *Quality of Service* dari *router* ketika dijalankan layanan *video streaming* dan FTP dengan parameter yang meliputi *delay, throughput, packet loss* dan *jitter* tersebut dapat disimpulkan seberapa besar performansi *Quality of Service* yang dimilikinya.

2. METODOLOGI

Pada bagian ini akan dijelaskan metodologi penelitian yang dilakukan meliputi arsitektur sistem yang dibangun untuk menjalankan *virtual router,* topologi pengujian sistem dan topologi jaringan virtual yang dirancang sehingga saling terkoneksi. Bagian ini juga menjelaskan spesifikasi sistem dan alat yang digunakan pada penelitian ini. Selain itu, dijelaskan pula bagaimana skenario pengujian serta metode pengujian yang digunakan.

2.1. Arsitektur Sistem

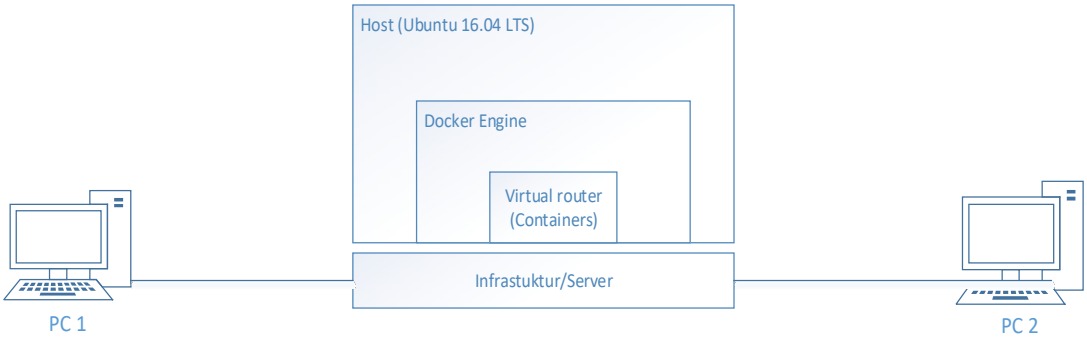


Gambar 2. Arsitektur Sistem

Pada Gambar 2 menunjukkan arsitektur sistem pada *containers* dimana *server* akan menjalankan operasi sistem sebagai *host*, lalu di atasnya dijalankan *platform containers* yaitu *Docker* yang akan mengisolasi setiap aplikasi atau *software* yang ingin dijalankan. Untuk menjalankan VNF *virtual router*, dibutukan *library* khusus untuk membuat *image* yang didapatkan melalui *repository docker*. Sistem operasi yang digunakan pada penelitian ini Ubuntu 16.04 dengan spesifikasi infrastruktur/*server* yang dijelaskan pada subbab topologi sistem. *Virtual router* yang digunakan *VyOS (VyattaOS).* Sistem ini seluruhnya berada dalam satu *server* yang menjalankan *virtual router* untuk dihubungkan ke jaringan fisik.

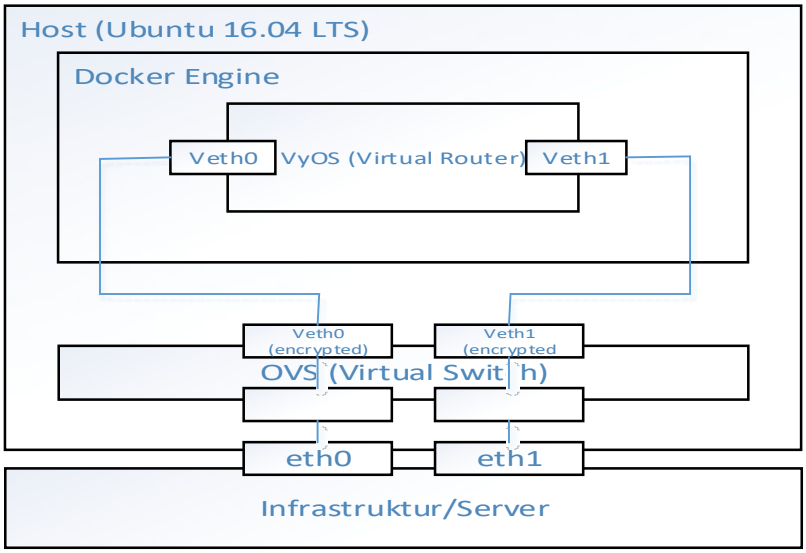
2.2. Topologi Sistem

Topologi pengujian yang akan dibangun pada penelitian ini terdiri dari *server* yang mengimplementasikan teknologi *containers. Containers* berjalan diatas *Operating System (OS)* dengan *Docker* sebagai *platform* untuk mengimplementasikan teknologi yang akan menjalankan *virtual router*. Lalu topologi tersebut diuji menggunakan layanan FTP dan *video streaming* untuk pengujian kualitas layanan membangun lingkungan yang terisolasi (*isolated Envirovment*). PC 1 bertindak sebagai *server* yang akan melayani PC 2 sebagai *client* dengan yang dihubungkan oleh jaringan virtual. Pada Gambar 3 menunjukan topologi jaringan yang dibangun untuk menguji performansi sistem.



Gambar 3. Topologi Jaringan Sistem

Virtual router memiliki dua *virtual interface* yaitu *Veth0* dan *Veth1* dimana kedua *virtual interface* tersebut dihubungkan ke *OpenVSwitch (OVS).* OVS adalah *virtual switch* yang digunakan pada pungujian ini dengan asumsi agar dapat terintegrasi dan terbuka dengan *platform* lain. Lalu, OVS akan menghubungkannya ke *pyshical* NIC pada server sehingga jaringan virtual dapat tehubung ke jaringan *real* dan dapat diintergrasikan dengan perangkat fisik. Pada Gambar 4 menunjukkan topologi jaringan virtual yang akan dibangun di dalam *server*.

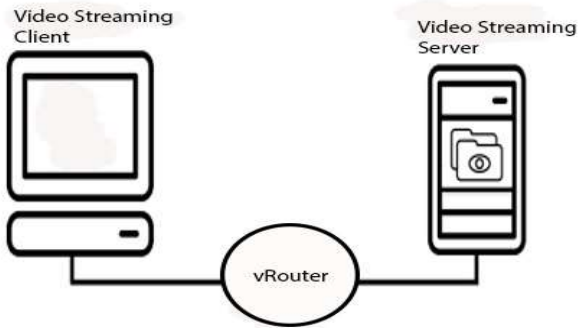


Gambar 4. Topologi Jaringan Virtual Sistem

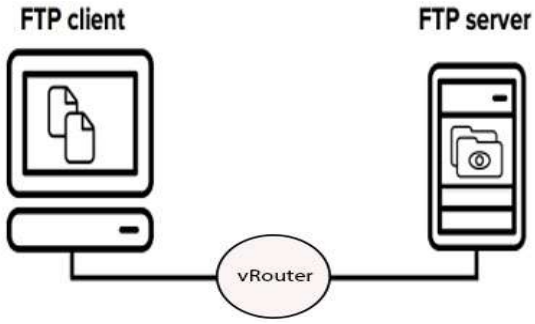
Perangkat keras pada penelitian ini terdiri dari satu buah *server* dan dua buah *PC User*. *Server* menggunakan Ubuntu sebagai *Host OS* yang menjalankan *Docker*. Lalu di dalam *Docker* dijalankan layanan fungsi *virtual router*. Spesifikasi *server* yang digunakan yaitu HP Proliant DL180 Gen 9 Intel Xeon E5-2609 v3, dengan CPU 1.90GHz (6 Cores) dan RAM sebesar 24 GB tipe DDR4-1600MHz. *Harddisk* yang digunakan sebesar 1000GB serta dua *port* NIC 1 GB. Untuk spesifikasi *user* menggunakan dua buah *user* yang bertindak sebagai *server* dan *client* dalam menjalankan layanan dengan spesifikasi ASUS All in one PC *Dekstop*, Processor Intel® Core™ i5, RAM 4 GB, HDD 2000GB, NIC satu port 1 GB NIC. Lalu untuk menghubungkan ketiga *node* tersebut digunakan Kabel UTP CAT5e yang menghubungkan dari *user* 1 ke *user* 2 yang melewati *virtual router* pada *Docker Container*.

2.3. Skenario Pengujian

Pada skema pengujian ini, *User* 1 dan *User* 2 dijalankan secara fisik pada jaringan lokal melewati dengan *virtual router* yang dibangun secara vrtual di atas *platform docker containers*. Pengujian yang dilakukan ini bertujuan untuk mengetahui performansi *Network Function Virtualization* yang dibangun di atas *Containers Docker*. *User* 1 sebagai server layanan dan *User* 2 sebagai klien dengan topologi pengujian pada Gambar 3 dengan topologi jaringan virtual seperti pada Gambar 4.



Gambar 5. Pengujian *Video Streaming*



Gambar 6. Pengujian FTP

Pengujian dilakukan dengan dua cara seperti pada Gambar 5 dan Gambar 6. Pengujian pertama dengan skema analisis pertama dilakukan dengan cara melewatkan *video streaming* tanpa *background traffic*, dilanjutkan skema analisis kedua melewatkan layanan *video streaming* dengan diberikan *background traffic* yang bertujuan untuk mengetahui performansi *Quality of Service* layanan *video streaming* berdasarkan parameter uji *delay system*, *jitter, throughput* dan *packet loss*. Pengujian kedua menggunakan layanan FTP dengan skema analisis pertama dilakukan dengan cara melewatkan layanan FTP tanpa *background traffic*, dilanjutkan skema analisis kedua melewatkan layanan FTP dengan diberikan *background traffic* yang bertujuan untuk mengetahui *Quality of Service* layanan FTP berdasarkan parameter uji *delay system*, *throughput* dan *packet loss.*

3. PENGUJIAN DAN HASIL

Pengujian dilakukan berdasarkan skenario pengujian yang telah dijelaskan sebelumnya dimana pengujian sistem menggunakan dua jenis pengujian dengan topologi pengujian seperti pada Gambar 3. Setiap layanan akan diuji dengan tambahan *background traffic* sebesar 100, 200, 400, 600 dan 800 MBps karena pada pengujian ini *bandwidth* yang tersedia sebesar 1 Gbps

sehingga dipilihlah nilai-nilai tersebut untuk melihat seberapa pengaruhnya terhadap layanan yang dijalankan. Penggunaan *link bandwidth* sebesar 1 Gbps karena untuk memaksimalkan pengujian pada lingkup *bandwidth* yang yang besar. Untuk mendapatkan nilai dari parameter-parameter QoS didapatkan dengan menggunakan *Wireshark.* Kemudian hasil yang diperoleh akan mengacu kepada standarisasi ITU-T G.1010 **(ITU-T, 2001).** Pada Tabel 1 menunjukan kriteria standarisasi QoS menurut ITU-T G.1010 sebagai acuan untuk mendapatkan kesimpulan dari hasil pengujian.

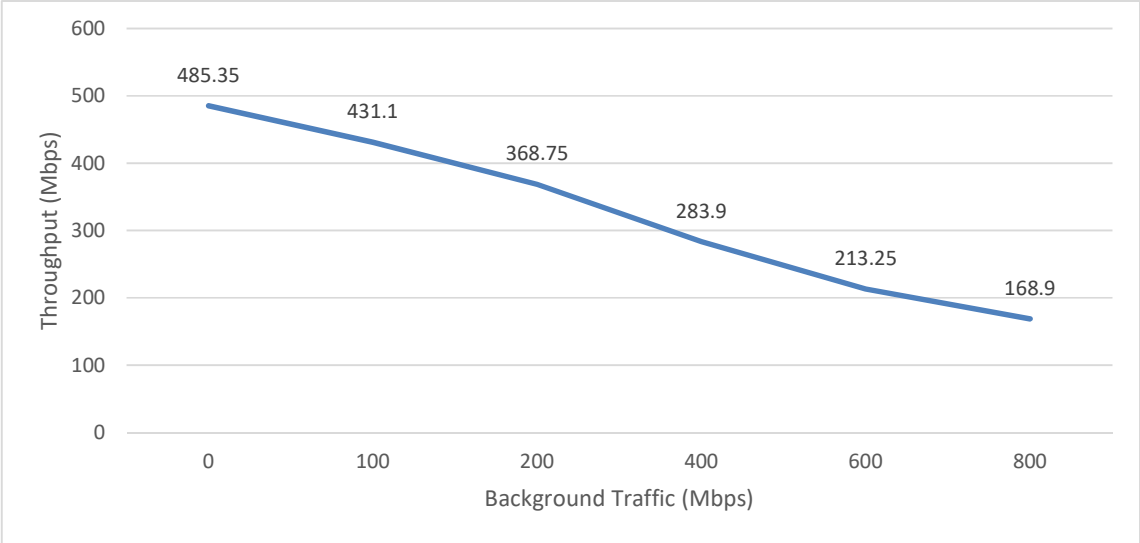
Tabel 1. Kriteria Standarisasi QoS (ITU- T G.1010)

Parameter	Standarisasi ITU-T G.1010	***Video Streaming***
Packet Loss	0	0
Delay	*Preferred* < 15 s *Acceptable* < 60 s	< 1% PLR
Jitter	N/A	< 10 s

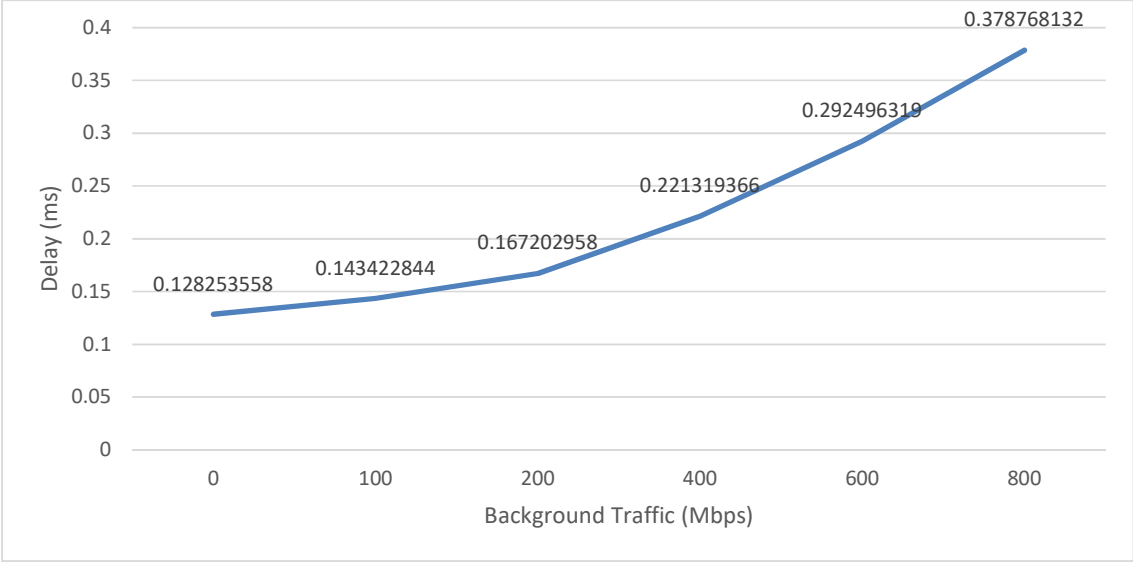
Pengujian pertama menggunakan layanan FTP yang bertujuan untuk mengetahui performansi sistem yang dibangun berdasarkan parameter-parameter *Quality of Service (QoS)* ketika sistem melewatkan layanan *non real-time* menggunakan *File Transfer Protokol (FTP)* pada jaringan lokal serta menguji pengaruhnya ketika diberikan variasi beban *background traffic.* Skenario pengujian ini dilakukan dengan cara mengirimkan data sebesar 1079076943 *bytes* dengan *bandwidth* total sistem *up to* 1 Gbps menggunakan layanan FTP dari *server (host 1)* ke *client (host 2)* melewati *virtual router Vyos* yang dijalankan di dalam lingkungan virtualisasi *Docker Containers*. Lalu diberikan variasi beban trafik UDP menggunakan *tools iperf* yang akan memberikan *background traffic* saat transfer data melalui FTP sebesar 100, 200, 400, 600 dan 800 Mbits untuk mengetahui seberapa besar pengaruhnya tehadap parameter QoS. Aplikasi FTP *server* yang digunakan proftpd lalu di sisi *client* digunakan *tool filezilla*. Ketika layanan dijalankan, paket yang dikirim dianalisis menggunakan *wireshark* untuk mengambil nilai parameter pengujian yang dibutuhkan. Pengujian dilakukan sebanyak 20 kali pada setiap parameter pengujian untuk memberikan nilai yang lebih akurat. Hasil pengujian FTP tanpa *background traffic* memenuhi kriteria standarisasi ITU-T G.1010 seperti pada Tabel 2.

Dari grafik pada Gambar 7 menunjukkan nilai *throughput* cenderung menurun seiring dengan penambahan *background traffic* secara bertahap dengan variasi trafik UDP yang ditambahkan sebesar 100, 200, 400, 600, dan 800 Mbps. Nilai *throughput* yang didapatkan ketika tidak diberikan *background traffic* menunjukan hasil yang lebih baik yaitu sebesar 485 Mbps dari total *bandwidth* sistem yang tersedia sebesar up to 1 Gbps. Namun ketika diberikan *background traffic* sebesar 100 Mbps, nilai *throughput* yang dihasilkan menurun sebesar 54 Mbps menjadi 431.1 Mbps. Lalu ketika diberikan trafik 200 Mbps menurun sebesar 116.6 menjadi 368.75 Mbps.

Saat diberikan beban trafik 400 nilai *throughput* menurun sebesar 201.45 Mbps, dan juga saat diberikan beban trafik 600 menurun sebesar 272.1 Mbps hingga saat diberikan beban trafik 800 Mbps *throughput* menurun sebesar 316.45 Mbps menjadi 168.9 Mbps. Sehingga jika *background traffic* yang diberikan semakin besar, penurunan *throughput* yang terjadi saat pengiriman data layanan FTP juga semakin besar yang menyebabkan pengiriman data akan menurun kecepatannya.

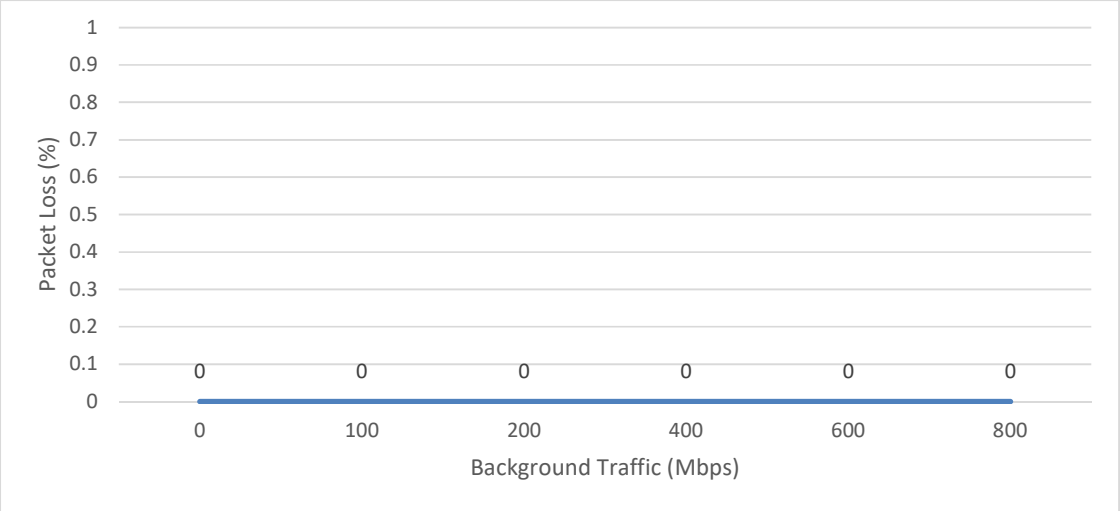


Gambar 7. Grafik *Throughput* FTP



Gambar 8. Grafik *Delay* FTP

Dari grafik pada Gambar 8 menunjukkan waktu *delay* mengalami peningkatan seiring dengan penambahan beban *background traffic*. Jika *background traffic* yang dibebankan pada layanan semakin besar, maka peningkatan *delay* yang terjadi juga semakin besar sehingga penambahan beban *background traffic* ini menyebabkan waktu *delay* atau lama waktu yang dibutuhkan paket untuk sampai ke *client* juga meningkat. Nilai *delay* maksimum didapatkan ketika diberi beban trafik sebesar 800 Mbps yaitu 0.378 ms yang memenuhi standarisasi ITU-T G.1010.



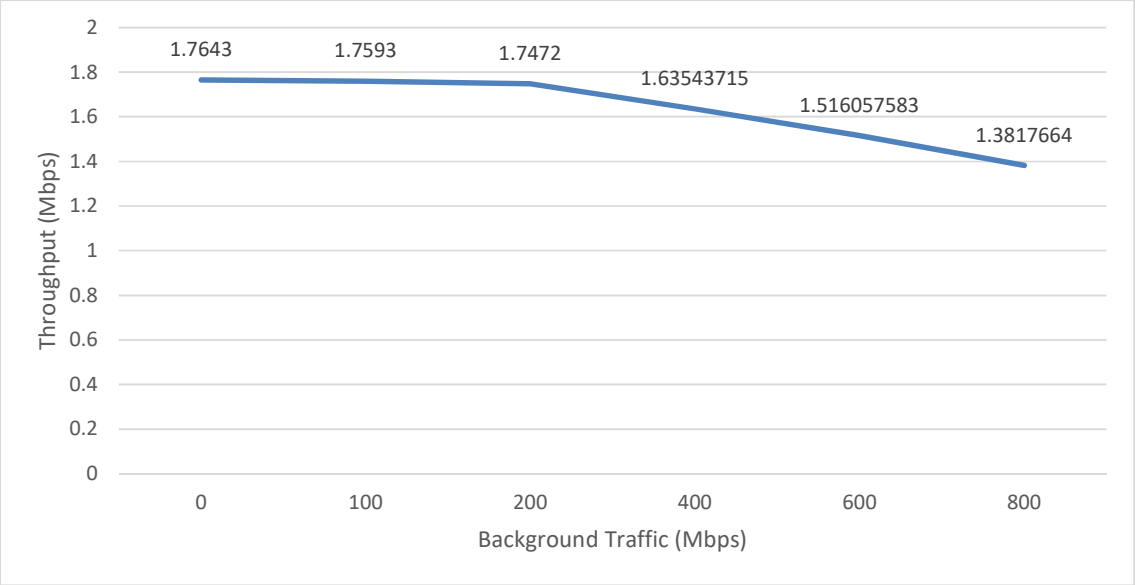
Gambar 9. Grafik *Packet Loss* FTP

Dari grafik pada Gambar 9 menunjukkan nilai *packet loss* pada layanan FTP adalah 0% di setiap pengujian menggunakan *background traffic*. Layanan FTP menggunakan TCP sebagai *protocol* transportnya. Untuk mengukur paremeter *packet loss* pada TCP yang menjamin paket yang dikirim akan diterima, dapat dilihat dari paket yang mengalami *retransmission* atau paket yang dikirim tidak sampai lalu dikirim kembali melalui mekanisme TCP. Lalu dapat dilihat bahwa tidak ada paket yang mengalami *retransmission* ketika paket dikirim. Hal tersebut menunjukkan layanan FTP yang dijalankan pada jaringan yang dibangun memenuhi kriteria ITU-T G.1010 yang mensyaratkan nilai *packet loss zero.*

Tabel 2. Hasil Pengujian FTP tanpa *Background Traffic*

Parameter	Standarisasi ITU-T G.1010	Hasil Pengujian
Packet Loss	0	0
Delay	*Preferred* < 15 s *Acceptable* < 60 s	0.12 ms
Jitter	N/A	N/A

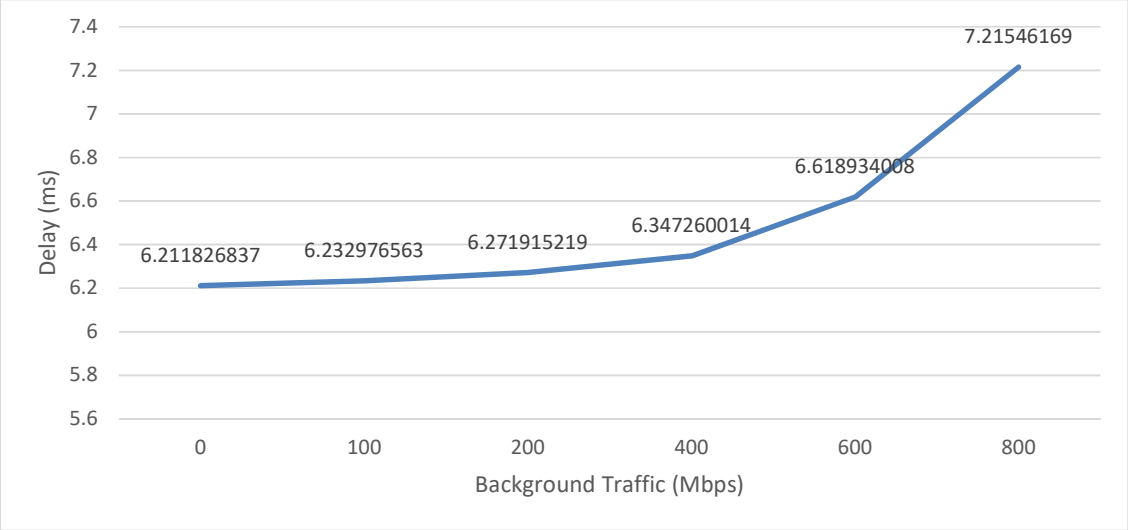
Lalu pada pengujian selanjutnya menggunakan layanan *video streaming* yang bertujuan untuk mengetahui performansi sistem yang dibangun berdasarkan parameter QoS ketika sistem melewatkan layanan *real-time video streaming* menggunakan aplikasi VLC dimana *client (host 2)* mengakses layanan *video streaming* dari *server (host 1)* pada jaringan lokal yang dibangun menggunakan layanan *virtual router* yang dijalankan di dalam lingkungan *virtual containers* menggunakan *Docker*. Sistematika pengujian ini dilakukan dengan cara menjalankan layanan *video streaming* selama satu menit dengan *bandwidth* total sistem 1 Gbps menggunakan aplikasi VLC dari *server (host 1)* ke *client (host 2)* melewati *virtual router Vyos* yang dijalankan di dalam lingkungan virtualisasi *Docker Containers. Bitrate* video 1550 kbps dan bitrate audio 125 kbps Lalu diberikan variasi beban trafik UDP menggunakan *tools iperf* yang akan memberikan *background traffic* saat layanan *video streaming* dijalankan sebesar 100, 200, 400, 600 dan 800 Mbits untuk mengetahui seberapa besar pengaruhnya tehadap parameter QoS. Aplikasi *video streaming* yang digunakan VLC yang di-*install* di sisi *client* maupun *server*. Lalu, ketika layanan dijalankan, paket yang dikirim *dianalisis* menggunakan *wireshark* untuk mengambil nilai parameter pengujian yang dibutuhkan. Pengujian dilakukan selama satu menit dengan mengunakan video yang sama sebanyak 20 kali pengujian untuk memberikan nilai yang lebih akurat. Hasil pengujian *video streaming* tanpa *background traffic* memenuhi kriteria standarisasi ITU-T G.1010 seperti pada Tabel 3.



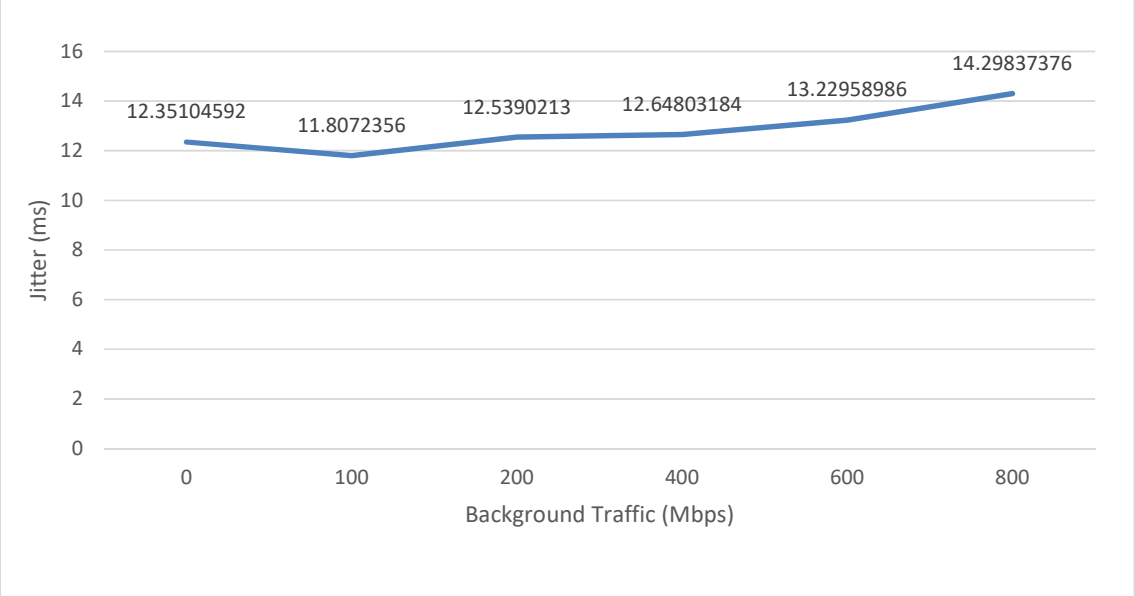
Gambar 10. Grafik *Throughput Video Streaming*

Dari grafik pada Gambar 10 menunjukan nilai *throughput* cenderung menurun dengan penambahan *background traffic* dengan variasi trafik UDP yang ditambahkan sebesar 100, 200, 400, 600, dan 800 Mbps. Nilai *throughput* yang didapatkan ketika tidak diberikan *background traffic* menunjukan nilai yang lebih baik. Throughput maksimal didapatkan ketika layanan tidak dibebankan dengan *background traffic* sedangakan *throughput* paling kecil didapatkan ketika diberikan beban sebesar 800mb atau 80% dari total *bandwidth* maksimal. Selain background traffic, menurut penelitian yang dilakukan Kurniawan nilai *throughput* juga dipengaruhi oleh nilai fps dari video yang dijalankan dan *bandwidth* yang digunakan **(Kurniawan, 2016)**. Untuk meningkatkan nilai throughput dapat dilakukan dengan cara menurunkan fps video yang dijalankan atau meminimalkan *background traffic*.

Dari grafik pada Gambar 11 terlihat bahwa waktu *delay* mengalami peningkatan seiring dengan penambahan beban *background traffic*. Jika *background traffic* yang di bebankan pada layanan semakin besar, maka semakin besar juga peningkatan *delay* yang terjadi sehingga penambahan beban *background traffic* ini menyebabkan waktu *delay* atau lama waktu yang dibutuhkan paket untuk sampai ke *client* juga meningkat. Selain *background traffic*, besarnya *delay* dipengaruhi oleh *bandwidth* yang tersedia dan nilai fps video. Nillai *delay* maksimum didapatkan ketika diberikan beban trafik UDP sebesar 800 Mbps yaitu sebesar 7.215 ms yang masih memenuhi kriteria standarisasi ITU-T G.1010 yang menstandarkan *delay* dibawah 10 *second*

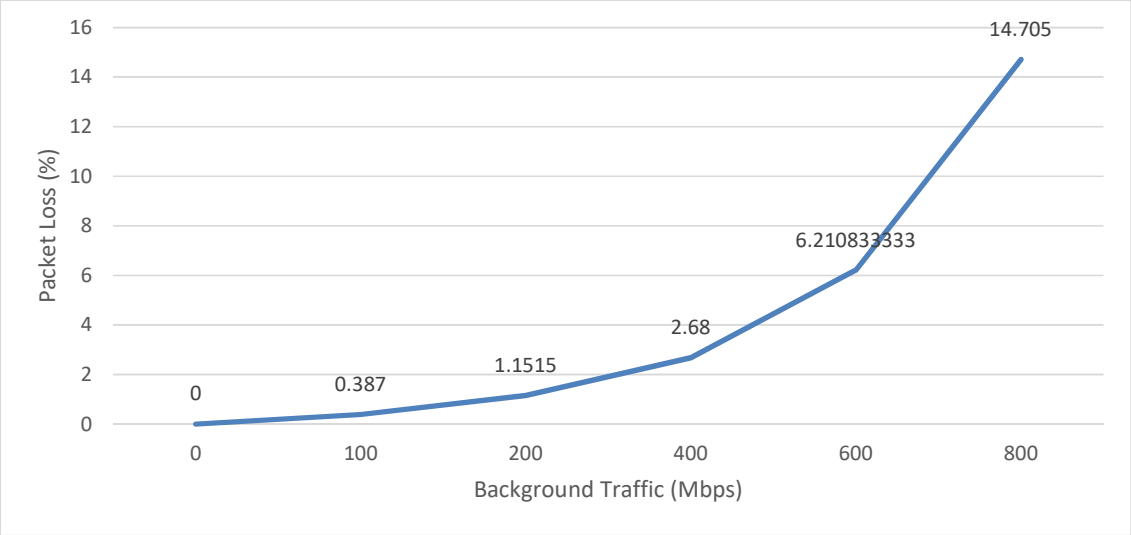


Gambar 11. Grafik *Delay Video Streaming*



Gambar 12. Grafik *Jitter Video Streaming*

Dari grafik pada Gambar 12 menunjukkan nilai *jitter video streaming* pada grafik diatas menunjukan nilai yang cenderung meningkat dengan penambahan *background traffic* sebesar 100, 200, 400, 600 dan 800 Mbps. Maka dapat dikatakan bahwa *background traffic* mempengaruhi nilai *jitter*. Semakin besar *background taffic* semakin meningkat nilai *jitter*. Nilai *jitter* terbesar yang didapatkan sebesar 14.528 ms. Untuk layanan *video streaming* tidak ada kriteria atau standar khusus untuk parameter *jitter* menurut ITU-T G.1010.



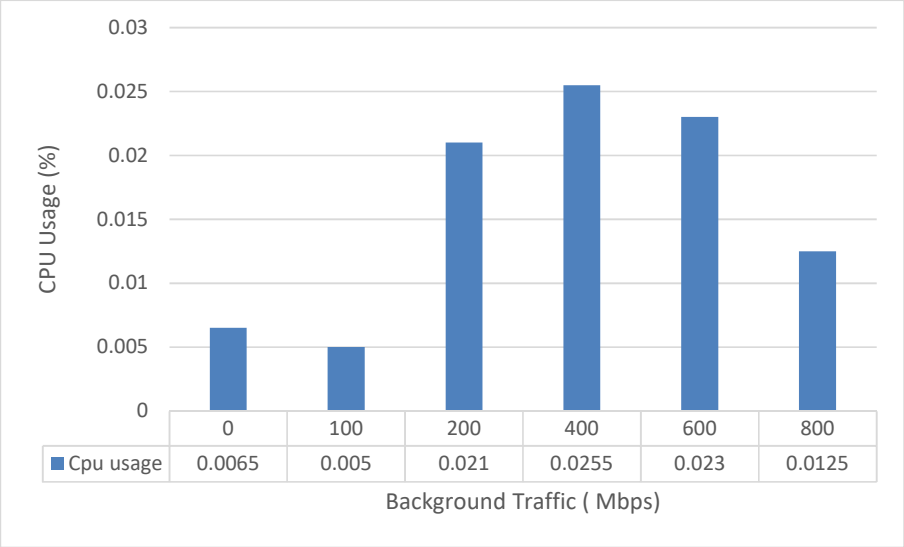
Gambar 13. Grafik *Packet Loss Video Streaming*

Dari grafik pada Gambar 13, dapat diketahui bahwa nilai *packet loss video streaming* yang dihasilkan meningkat secara eksponensial seiring dengan penambahan *background traffic.* Peningkatan nilai *packet loss* paling tinggi saat diberikan *background traffic* sebesar 800 Mbps atau 80% dari *bandwidth* total sebesar 14,705% besar *packet loss* pada layanan *video streaming* yang dijalankan. Nilai *packet loss* optimal yang memenuhi standarisasi ITU-T G.1010 ketika diberi beban trafik hingga 100 Mbps.

Tabel 3. Pengujian *Video Streaming* tanpa *Background Traffic.*

Parameter	Standarisasi ITU-T G.1010	Hasil Pengujian
Packet Loss	< 1% PLR	0 %
Delay	*Preferred* < 15 s *Acceptable* < 60 s	6.21 ms
Jitter	N/A	12.3 ms

Selain QoS, dianalisis pula dari sisi *CPU Usage router* terhadap prosesnya sebagai elemen jaringan. Dapat dilihat pada Gambar 14, *CPU Usage* pada *virtual router* yang dijalankan pada *Containers* menunjukKan hasil yang bervariasi berdasarkan *background trafficnya*. Nilai terbesar yang didapatkan berdasarkan grafik di atas yaitu sebesar 0.0255 % dari total *CPU* yang digunakan. Hal ini menunjukkan bahwa *virtual router* yang dijalankan dalam *docker containers* hanya menggunakan *resource* dari *CPU* di bawah 1% dengan asumsi *CPU* yang digunakan pada pengujian kali ini 1.90GHz.



Gambar 14. *CPU Usage*

4. KESIMPULAN

Berdasarkan hasil penelitian performansi layanan FTP dan *video streaming* pada jaringan NFV yang dibangun di dalam *Docker Containers* dapat disimpulkan bahwa Layanan FTP dan *Video Streaming* dapat berjalan pada jaringan NFV yang dibangun di dalam *Docker Containers*. Layanan FTP dapat dijalankan dengan baik pada jaringan NFV *Docker Containers* yang memenuhi standarisasi ITU-T G.1010 yaitu *delay preferred* lebih kecil dari 15 *second* dan *packet loss* 0% dengan *delay* terbesar saat beban trafik 800 Mbps sebesar 0.378 ms dan *packet loss* sebesar 0%. Nilai *throughput* maksimal sebesar 485.3 Mbps atau sekitar 48,5 % dari *bandwidth* yang tersedia. Penambahan skenario *background traffic* 100 Mbps – 800 Mbps berbanding lurus dengan perubahan parameter QoS *pada throughput, packet loss* dan *delay.* Aplikasi layanan *video streaming* pada jaringan NFV *Docker Containers* menunjukkan nilai *packet loss* yang memenuhi standarisasi ITU-T G.1010 dengan *delay* kurang dari 10 *second* dan *packet loss* kurang dari 1% ketika diberikan beban trafik sebesar 100 Mbps yaitu 0.387. Nilai *delay* terbesar yaitu 7.21546169 ms ketika diberi beban 800 Mbps. Nilai *throughput* maksimal sebesar 1764.3 Kbps ketika tidak diberikan beban trafik.

DAFTAR RUJUKAN

Falkner, M., Leivadeas, A., Lambadaris, I., & Kesidis, G. (2016). Performance analysis of virtualized network functions on virtualized systems architectures. In *IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD*(pp. 71–76). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/CAMAD.2016.7790333

European Telecommunications Standards Institute. (2012). Network Functions Virtualisation (NFV); White Paper #1.

Aswariza, R., Perdana, D., & Negara, R. (2017). Analisis Throughput Dan Skalabilitas Virtualized Network Function VyOS Pada Hypervisor VMWare ESXi, XEN, DAN KVM. *JURNAL INFOTEL, 9*(1), 70-74.

Chung, M. T., Quang-Hung, N., Nguyen, M.-T., & Thoai, N. (2016). Using Docker in high performance computing applications. In *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)* (pp. 52–57). IEEE. https://doi.org/10.1109/CCE.2016.7562612

Preeth E N, Mulerickal, F. J. P., Paul, B., & Sastri, Y. (2015). Evaluation of Docker containers based on hardware utilization. In *2015 International Conference on Control Communication & Computing India (ICCC)* (pp. 697–700). IEEE. https://doi.org/10.1109/ICCC.2015.7432984

Docker. (2018, Jan 10). *Docker Overview.* Retrieved from https://docs.docker.com/engine/docker-overview/

Anderson, Jason & Hu, Hongxin & Agarwal, Udit & Lowery, Craig & Li, Hongda & Apon, Amy. (2016). Performance Considerations of Network Functions Virtualization using Containers. *IEEE 2016 International Conference on Computing, Networking and Communications, Internet Services and Applications Performance*, 1–25. https://doi.org/10.1109/ICCNC.2016.7440668

Felter, W., Ferreira, A., Rajamony, R., & Rubio, J. (2015). An updated performance comparison of virtual machines and Linux containers. In *ISPASS 2015 - IEEE International Symposium on Performance Analysis of Systems and Software*(pp. 171–172). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ISPASS.2015.7095802

Aravinthan, G., Herculea, D., Chen, C.S., & Roullet, L. (2017). Virtualization of radio access network by Virtual Machine and Docker: Practice and performance analysis. 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 680-685.

International Telecommunication Union. (2001). ITU-T Recommendation G. 1010: End-user multimedia QoS categories (Quality of service and performance). *International Telecommunications Union*, *1010*. Retrieved from https://www.itu.int/rec/T-REC-G.1010-200111-I/en

Kurniawan, E., & Sani, A. (2014). Analisis Kualitas Real Time Video Streaming terhadap Bandwidth Jaringan yang Tersedia. *Singuda Esikom*, 9(2), 92–96.