ANALISIS PERFORMANSI NETWORK FUNCTION VIRTUALIZATION PADA *CONTAINERS* MENGUNAKAN *DOCKER*

PERFORMANCE ANALYSIS OF NETWORK FUNCTION VIRTUALIZATION ON CONTAINERS USING DOCKER

Manzila Izniardi Djomi[1], Ir. Rendy Munadi, M.T[2], Ridha Muldina Negara, S.T, M.T[3]

[1,2,3]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[2,3]Prodi S2 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[1]manzilaizniardidjomi@student.telkomuniversity.ac.id,
[2]rendymunadi@telkomuniversity.ac.id, [3]ridhanegara@telkomuniversity.ac.id

Abstrak

Perkembangan layanan internet harus didukung oleh infrasruktur jaringan yang memadai. 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 di implementasikan sebagai aplikasi perangkat lunak (*software*) yang dapat dijalankan di lingkungan virtual yang disebut sebagai *Virtualized Network Functions (VNFs)*. Pada umumnya layanan ini mengunakan *hypevisor (hardware-level virtualization)* untuk membuat lingkungan virtualisasi, namun dewasa ini teknologi virtualisasi memiliki alternatif pengimplementasian dengan menggunakan teknologi *containers (Operating system -level virtualization)*. Containers pada perkembangannya menjadi salah satu teknologi yang memiliki performansi yang baik dalam mengisolasi dan menjalankan aplikasi serta memiliki *delay latancy* yang rendah. Sehingga dapat mereduksi waktu pengimplementasian dan pengoperasian aplikasi. Containers juga dapat di implemetasikan pada NFV yang dapat membangun lingkungan virtual untuk menjalankan fungsi layanan *networking*.

Pada tugas akhir ini akan dilakukan pengujian yang menghasilkan analisis suatu *containers* dengan menggunakan *platform Docker* yang akan menjalankan VNF berupa *virtual router*. Penelitian ini bertujuan untuk mengetahui performansi Docker ketika menjalankan *virtual router* yang dilewatkan oleh layanan FTP dan *video streaming* menggunakan parameter *delay, packet loss, jitter, throughput.*

Dari hasil pengujian dan analisis, dapat disimpulkan bahwa kedua layanan berjalan dengan baik pada jaringan NFV pada Docker Container. Berdasarkan pengujian pada kedua layanan, dapat dibuktikan bahwa jaringan yang dibangun memiliki performansi yang baik, hal ini dapat dibuktikan pada hasil pengukuran dimana pada pengujian menggunakan layanan video streaming dan FTP memenuhi kriteria standarisasi ITU-T G.1010.

Kata Kunci : Containers, Docker, *Network Function Virtualization*, *QoS*

Abstract

The development of internet services is supported by adequate network infrastructure. Network infrastructure services such as routing, 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 called Virtualized Network Functions (VNFs). In general, this service uses hypevisor (hardware-level virtualization) to create a virtualization environment, but now virtualization technology has an alternative implementation using the technology containers (Operating system-level virtualization). Containers in its development become one of the technologies that have a good performance in isolating and running applications and have a low latancy delay. So it can reduce the application and operation of the application. Containers can also be implemented on an NFV that can build virtual environments to perform networking service functions.

In this final task will be tested that resulted analysis of a containers by using platform Docker that will run VNF in the form of virtual router. This study aims to determine the performance of Docker

when running a virtual router that is passed by FTP service and streaming video using delay, packet loss, jitter, throughput parameters.

From the results of testing and analysis, it can be concluded that both services run well on the NFV network on the Docker Container. Based on testing on both services, it can be proven that the built network has good performance, it can be proved in the measurement results where the test using video streaming service and FTP meet the criteria of ITU-T G.1010 standardization.

Keywords : Containers, Docker, Network Function Virtualization, QoS

1. Pendahuluan

Pengembangan Pada jaringan telekomunikasi tradisional, layanan seperti *routing* dan *firewall* biasanya 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)*[1]. NFV mampu mengurangi biaya operasi, serta meningkatkan dalam pengelolaan dan inovasi dalam ruang lingkup fungsi jaringan. Teknologi NFV mengenalkan konsep baru dalam merancang, menyebarkan dan mengelola layanan jaringan dengan cara memisahkan fungsi jaringan dari peralatan *hardware proprietary* sehingga dapat berjalan pada suatu *hypervisor* atau *platform* virtualisasi sebagai salah satu infarastuktur yang membangun NFV (NFVI)[2]. Badan standarisasi *European Telecommunications Standards Institute* (ETSI) mendefinisikan NFVI sebagai kumpulan dari komponen *hardware* dan *software* yang membangun lingkungan untuk VNFs[3]. NFVI menyediakan layanan infrastruktur *multi-tenant* dengan memanfaatkan teknologi virtualisasi yang dapat mendukung beberapa kasus penggunaan dan bidang aplikasi secara bersamaan.

Seiring perkembangannya, 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*)[4]. 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 cendreung memvirtualisasikan dalam level abstraksi *hardware*, sedangkan containers adalah emulasi pada level abstraksi Operating System (OS)[5]. Salah satu *platform* populer yang menerapkan teknologi *containers* yaitu Docker.

Beberapa percobaan telah dilakukan dalam menganalis perfomansi ketika menjalankan aplikasi pada *containers* yang menunjukan perfomansi yang lebih baik dibandingkan teknologi *hypervisor*[6]. *Containers* dapat memberikan *latency* dan *delay* variasi yang rendah, dan dapat dimanfaatkan untuk teknologi jaringan kinerja tinggi yang sebelumnya hanya digunakan pada virtualisasi *hardware*[1]. Oleh karena itu, dalam tugas akhir ini diusulkan suatu analisis *containers* yang dapat menjalankan VNFs diatasnya untuk melihat dan menguji perfomansi yang dihasilkannya.

.

2. Landasan Teori

2.1 Virtualisasi

Virtualisasi adalah sebuah konsep pendekatan teknologi penyatuan dan berbagi sumber daya untuk meningkatkan penggunaan aset dan menyederhanakan manajemen sehingga sumber daya TI dapat lebih mudah memenuhi permintaan bisnis. Teknologi virtualisasi mengemulasi sumber daya fisik komputasi, seperti komputer desktop, server, prosesor dan memori, sistem penyimpanan, jaringan, dan aplikasi individu yang membentuk sebuah "lingkungan virtual". Dalam server atau jaringan, virtualisasi digunakan untuk mengambil sumber daya fisik tunggal lalu membuatnya dapat beroperasi seolah-olah sumber daya fisik tunggal tersebut memiliki lebih dari satu sumber daya. Hal ini dapat meningkatkan pemanfaatan dan efisiensi sumber daya aset, serta mengurangi biaya dengan mengurangi kebutuhan untuk aset fisik.

2.2 Network Function Virtualization (NFV)

Network functions virtualization (NFV) adalah sebuah konsep dari arsitektur jaringan yang menggunakan teknologi virtualisasi IT untuk memvirtualisasikan fungsi setiap *node-node* pada jaringan yang dapat saling terhubung sehingga tercipta sebuah layanan komunikasi. Konsep NFV (*Network Functions Virtualization*) muncul dari para operator/telco yang mencari solusi untuk mempercepat implementasi layanan baru jaringan untuk mendukung strategi bisnis dan pertumbuhan pendapatan mereka.

Salah satu hambatan signifikan yang mereka rasakan adalah ketergantungan terhadap *hardware-based appliance*

2.3 Container-Based Virtualization

Container-Based Virtualization adalah alternatif yang lebih ringan dari *hypervisors* juga dikenal sebagai *Operating System Level virtualization*. *Container-Based Virtualization* adalah metode virtualisasi yang bekerja pada tingkat sistem operasi dengan berbagi *host* kernel untuk membuat satu atau lebih wadah *(instance).* Virtualisasi berbasis *hypervisor* memberikan abstraksi untuk *full guest* (satu per mesin virtual), virtualisasi berbasis kontainer bekerja pada tingkat sistem operasi, memberikan abstraksi langsung untuk setiap guest. Dalam prakteknya, hypervisors bekerja pada tingkat abstraksi hardware dan kontainer di lapisan system call / ABI.[10]

2.4 Docker

Docker adalah sebuah project open-source yang menyediakan platform terbuka untuk developer maupun sysadmin untuk dapat membangun, mengemas, dan menjalankan aplikasi dimanapun sebagai sebuah wadah (container) yang ringan. Docker awal mulanya dikembangkan oleh Solomon Hykes sebagai project internal di dotCloud, sebuah perusahaan PaaS (platform as a service). Docker memungkinkan kita untuk memisahkan aplikasi dari infrastruktur, sehingga kita dapat membangun perangkat lunak dengan cepat

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.[7]..

2.5 Quality of Service

Terdapat beberapa factor yang mempengaruhi kualitas *Real Time Video Streaming*, yaitu waktu tunda *(delay),* paket loss dan pemilihan jenis codec. Ukuran dan pengalokasian kapasitas jaringan juga mempengaruhi kualitas *Real Time Video Streaming* secara keseluruhan. Berikut penjelasan dari beberapa factor tersebut.

1. Delay

   Delay adalah salah waktu yang dibutuhka oleh sebuah pake yang dikirim dari pengirim ke penerima dan salah satu parameter pada QOS. Delay juga memiliki Batasan-batasan atau kriteria Delay berdasarkan waktu, berikut tabel kriteria Delay.

   $$\boldsymbol{Delay}\ (\boldsymbol{t}) = (\boldsymbol{Tr} - \boldsymbol{Ts})\boldsymbol{detik}$$

   dimana:

   Tr = Waktu penerimaan paket (detik)

   Ts = Waktu pengiriman paket (detik)

2. Packet loss

   Packet loss pada jaringan IP telephony sangat besar pengaruhnya, dimana jika terjadi packet loss dalam jumlah tertentu akan menyebabkan terjadinya interkoneksi TCP melambat. Packet loss maksimum yang masih bisa ditolerir adalah kurang dari 10 %. Pendekatan yang digunakan untuk mengkompensasi packet loss meliputi interpolasi suara dengan pengulangan paket terakhir, pengiriman informasi redundan, menggunakan metode pembagi bandwidth.

   $$\boldsymbol{Packet\ Loss} = (\frac{Paket\ dikirim}{Paket\ diterima}\ x\ 100\%)$$

3. Throughput

   Throughput merupakan jumlah rata-rata byte/bit data yang berhasil diterima per satuan detik melalui sebuah sistem komunikasi. Throughput diukur ketika selesai mentransmisikan data pada suatu host atau client. Bagian yang terpenting pada throughput yaitu terdapat pada bandwidth yang cukup. Rumus yang digunakan untuk menghitung throughput rata-rata adalah sebagai berikut:

   $$\boldsymbol{Throughput} = \frac{Jumlah\ data\ yang\ berhasil\ lewat\ (bps)}{Lama\ waktu\ pengamatan\ (s)}$$

4. Jitter

   Jitter merupakan merupakan variasi dari delay atau selisih antara delay pertama dengan delay selanjutnya. Jitter merupakan masalah khas dari connectionless network atau packet switched network serta slow speed links. Besarnya nilai jitter akan sangat dipengaruhi oleh variasi beban trafik dan besarnya tumbukan antar paket (congestion) yang ada dalam jaringan IP. Semakin besar beban trafik di dalam jaringan akan menyebabkan semakin besar pula peluang terjadinya congestion

dengan demikian nilai jitter-nya akan semakin besar. Semakin besar nilai jitter akan mengakibatkan nilai QoS akan semakin turun. Untuk mendapatkan nilai QoS jaringan yang baik, nilai jitter harus dijaga seminimum mungkin

$$\boldsymbol{Jitter} = \boldsymbol{delay}\ (\boldsymbol{A}) - \boldsymbol{delay}\ (\boldsymbol{B})$$

-

2.6 Penelitian Terkait

Pada penelitian sebelumnya dilakukan penelitian mengenai performansi bare-metal hypervisor (XEN, VMware ESXi) dan hosted hypervisor (Kernel-based Virtual Machine atau KVM). Dari penelitian tersebut diperoleh hasil metrik overall performance, XEN memiliki nilai tertinggi. Untuk parameter throughput, KVM memiliki performansi kecepatan tertinggi. Sedangkan untuk parameter skalabilitas, VMware memiliki skalabilitas yang sangat baik yang ditunjukkan dengan kecilnya degradasi performansi pada throughput saat menjalankan banyak VNF[11]. Sedangkan pada penelitian [6] dilakukan analisis pada Networking Technologies pada Docker menunjukan perfomansi yang mendekati native daripada menggunakan hypervisor.

Pada penelitian [17] mengembangkan skema dari [11] dengan menggunakan containers Docker sebagai lingungannya. Pada penelitian kali ini akan menggunakan skema penelitian [11] dengan menggunakan layanan FTP dan video streaming untuk menguji jaringan yang dibangun.

3. Pembahasan

Pada bab ini dibahas mengenai pengujian dan analisis hasil implementasi *Network Function Virtualization* yang telah dibangun. Pengujian dan analisis ini bertujuan untuk mengetahui performansi *Network Function Virtualization* pada *virtual router* yang dibangun di lingkungan *Docker Containers* ketika menjalankan dan meneruskan layanan *real-time (video streaming)* maupun layanan *non-realtime* (FTP) berdasarkan parameter – parameter *Quality of Service (QoS).*

Pengujian dilakukan berdasarkan skenario pengujian yang telah dijelaskan sebelumnya pada subbab 3.5 dimana pengujian sistem mengunakan dua jenis pengujian dengan topologi pengujian seperti pada Gambar 6. 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 dipililah 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. Sedangkan untuk mendapatkan nilai-nilai parameter QoS didapatkan dengan menggunakan Wireshark sebagai *network protocol analyzer*. Kemudian hasil yang diperoleh akan mengacu kepada standarisasi ITU-T G.1010 untuk mendapatkan kesimpulan dari hasil Pengujian tersebut.

Parameter	Data Transfer (FTP)	Video Streaming
Packet Loss	0	< 1% PLR
Delay	Preferred < 15 s Acceptable < 60 s	< 10 s
Jitter	N/A	N/A

Tabel 1 Kriteria Standarisasi QoS Menurut ITU-T G.1010

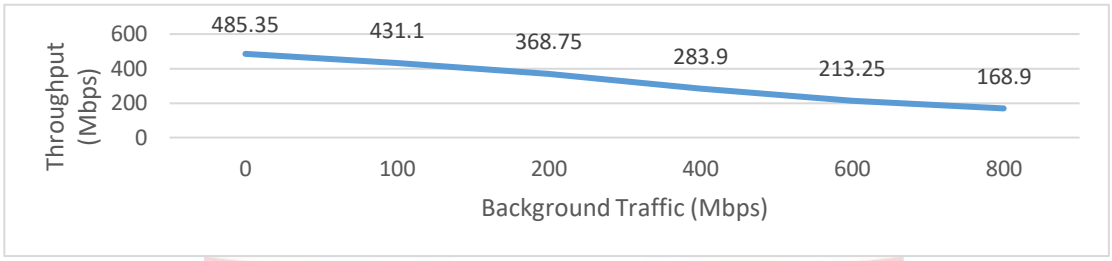
3.1 Pengujian FTP

Pengujian pertama ini 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*. Lalu, ketika layanan dijalankan, paket yang dikirim di analisis menggunakan

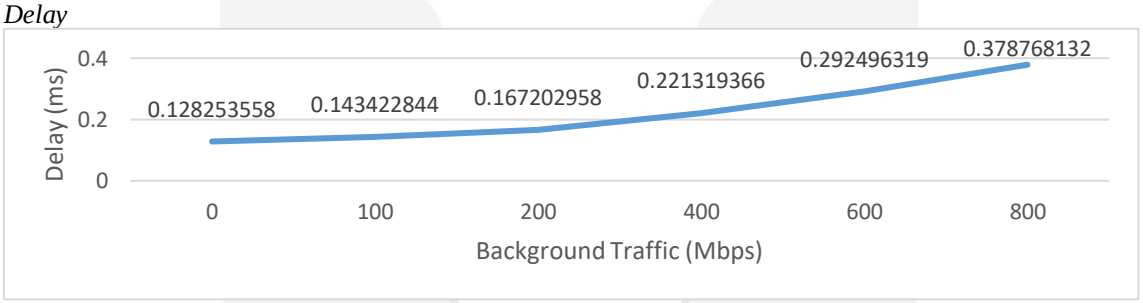
wireshark untuk mengambil nilai parameter pengujian yang dibutuhkan. Pengujian dilakukan sebanyak 20 kali pada setiap parameter pengujian untuk memberikan nilai yang lebih akurat.

Throughput



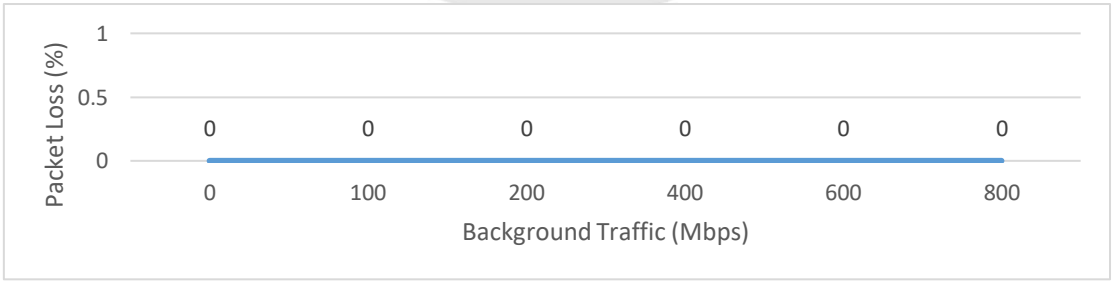
Dari grafik diatas menunjukan nilai *throughput* cenderung menurun seiring dengan penambahan *background traffic* secara bertahap dengan variasi trafik UDP yang ditambahkan sebesar 100, 200, 400, 600, dan 800 Mbps. Nilai throughput yang didapatkan ketika tidak diberikan *background traffic* menunjukan 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 semakin besar *background traffic* yang diberikan semakin besar pula penurunan *throughput* yang terjadi saat pengiriman data layanan FTP yang memyebabkan pengiriman data akan menurun kecepatannya.

Delay



Dari grafik diatas terlihat bahwa waktu delay mengalami peningkatan seiring dengan penambahan *beban background* traffic. Semakin besar background traffic yang di bebankan pada layanan semakin besar pula 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. Nilai delay maksimum didapatkan ketika diberi beban trafik sebesar 800 Mbps yaitu 0.378 ms yang memenuhi standar ITU-T G.1010.

Packet Loss



Dari Grafik diatas menunjukan 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, kita dapat

melihatnya dari paket yang mengalami *retransmission* atau paket yang dikirim tidak sampat lalu dikirim kembali melalui mekanisme TCP. Lalu dapat kita lihat bahwa tidak ada paket yang mengalami retransmission ketika paket dikirim. Hal tersebut menunjukan layanan FTP yang dijalankan pada jaringan yang dibangun memenuhi kriteria ITU-T G.1010 yang mensyaratkan nilai *packet loss zero*.

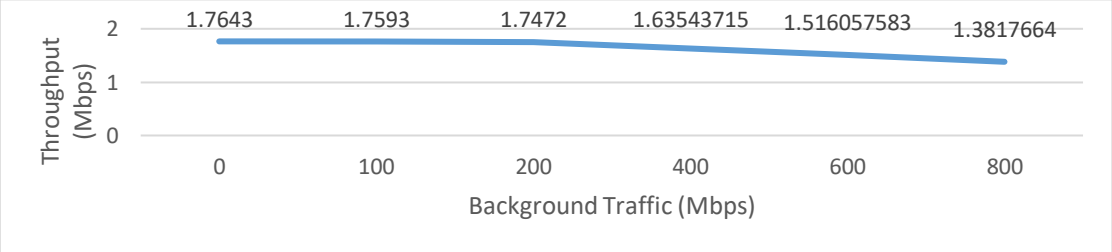
3.2 Pengujian Video Streaming

Pengujian kedua ini bertujian 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 dilakukan dengan cara menjalankan layanan video streaming selama satu menit dengan *bandwidth* total sistem 941 Mbps menggunakan aplikasi VLC 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 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 diinstall di sisi *client* maupun *server*. Lalu, ketika layanan dijalankan, paket yang dikirim di analisis 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.

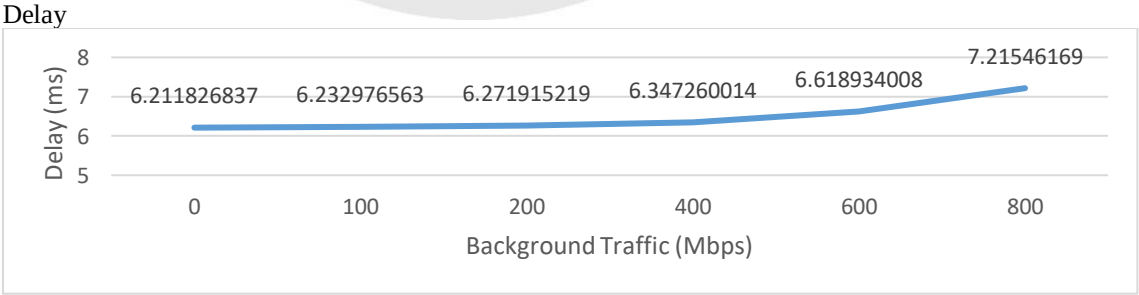
Hasil Pengujian

Throughput



Dari grafik diatas 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* makasimal. Selain background traffic, nilai *throughput* juga dipengaruhi oleh nilai fps dari video yang dijalankan[15]. Untuk meningkatkan nilai throughput dapat dilakukan dengan cara menurunkan fps video yang dijalankan atau meminimalkan *background traffic*.

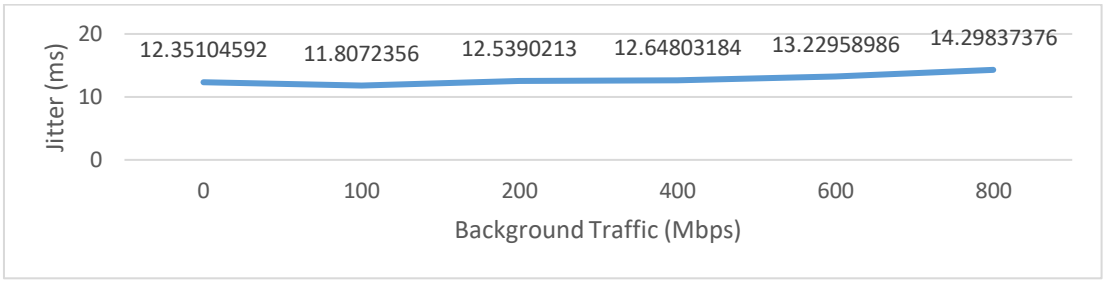
Delay



Dari grafik diatas terlihat bahwa waktu delay mengalami peningkatan seiring dengan penambahan *beban background* traffic. Semakin besar *background traffic* yang di bebankan pada layanan semakin besar pula peningkatan *delay* yang terjadi sehingga penambahan beban

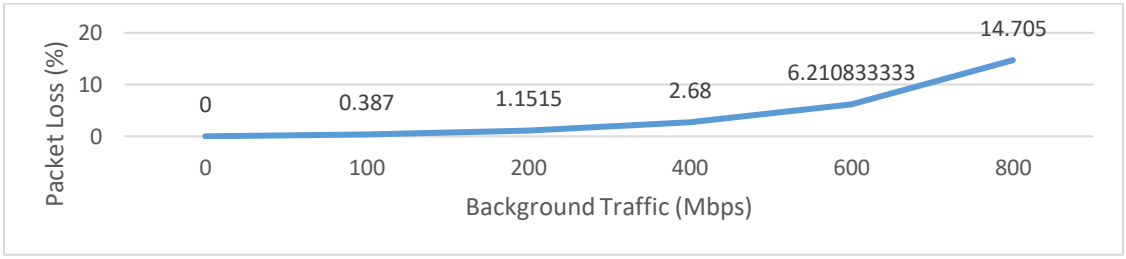
background traffic ini menyebabkan waktu delay atau lama waktu yang dibutuhkan paket untuk sampai ke *client* juga meningkat.. Selain *background traffic*, besarnya *delay* dipengaruhi oleh bandwidth yang tersedia dan nilai fps video. Nillai delay maksimum didapatkan ketika diberikan beban trafik UDP sebesar 800 mbps yaitu sebesar 7.215 ms yang masih memenuhi kriteria standar ITU-T G.1010 yang menstandarkan delay dibawah 10 second.

Jitter



Nilai jitter video streaming pada grafik diatas menunjukan nilai yang cendrung 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 miningkat 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.

Packet Loss



Dari grafik diatas, 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 video streaming yang dijalankan. Nilai packet loss optimal yang memenuhi standar ITU-T G.1010 ketika diberi beban trafik sekitar 100 Mbps.

4. Kesimpulan

Berdasarkan hasil penelitian perfomansi layanan FTP dan video streaming pada jaringan NFV yang di bangun di dalam Docker Container dapat disimpulkan bahwa :

1. Layanan FTP dan *Video Streaming* dapat berjalan pada jaringan NFV yang dibangun di dalam Docker Container.
2. Layanan FTP dapat dijalankan dengan baik pada jaringan NFV Docker Container yang memenuhi standar ITU-T G.1010 yaitu Delay Preferred kurang dari 15 second dan packet loss 0% dengan delay terbesar saat bebean 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.
3. Aplikasi layanan *video streaming* pada jaringan NFV Docker Container menunjukan nilai pab cket loss yang memenuhi standar ITU-T G.1010 untuk 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 Pustaka

[1] M. Falkner, A. Leivadeas, I. Lambadaris, and G. Kesidis, “Performance Analysis of Virtualized

Network Functions on Virtualized Systems Architectures," pp. 71–76, 2016.

[2] D. Qualification, "Indian Institute of Technology Bombay," no. November, pp. 1–4, 2012.

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

[4] Linuxcontainer. URL https://linuxcontainers.org/lxc/introduction

[5] Z. Ding, S. P. Architect, S. Sanjabi, and A. S. Engineer, "Using Docker in High Performance Computing in OpenPOWER Environment," *2016 IEEE Sixth Int. Conf. Commun. Electron.*, pp. 1–17, 2016.

[6] Anderson, J., Agarwal, U., Li, H., & Software, D. (2016). Performance Considerations of Network Functions Virtualization using Containers, 1–25

[7] Docker. URL https://docs.docker.com/engine/understanding-docker/.

[8] Introduction, A., Action, C., Chiosi, M., Clarke, D., Willis, P., Reid, A., … Michel, U. (2012). Network Functions Virtualisation,An Introduction, Benefits, Enablers, Challenges & Call for Action. Citeseer, (1), 1–16. https://doi.org/10.1.1.402.8583

[9] Xavier, M. G., Neves, M. V, Rossi, F. D., Ferreto, T. C., Lange, T., & De Rose, C. A. F. (2013). Performance Evaluation of Container-Based Virtualization for High Performance Computing Environments. 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, (LXC), 233–240. https://doi.org/10.1109/PDP.2013.41

[10] Rathore, M. S. (2013). KVM vs. LXC: Comparing Performance and Isolation of Hardware-assisted Virtual Routers. American Journal of Networks and Communications, 2(4), 88. https://doi.org/10.11648/j.ajnc.20130204.11

[11] Aswariza, Revan, Perdana, Doan, Negara, Ridha. " Analisis Throughput Dan Skalabilitas Virtualized Network Function VyOS Pada Hypervisor VMWare ESXi, XEN, DAN KVM" JURNAL INFOTEL [Online], Volume 9 Number 1 (10 February 2017)

[12] "ETSI GS NFV 003 V1.2.1: Network Functions Virtualisation (NFV); Terminology for main concepts in NFV," ETSI Ind. Spec. Group (ISG) Netw. Functions Virtualisation (NFV), Sophia-Antipolis Cedex, France, April. 2017.[Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/003/01.02.01_60/gs_NFV003v010201p.pdf

[13] "ETSI GS NFV-MAN 001 V1.1.1: Network Functions Virtualisation (NFV); Management and orchestration," ETSI Ind. Spec. Group (ISG) Netw. Functions Virtualisation (NFV), Sophia-Antipolis Cedex, France, April. 2017. [Online]. Available:http://www.etsi.org/deliver/etsi_gs/NFVMAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf

[14] https://www.redhat.com/en/topics/virtualization. Diakses pada 11 Mei 2017.

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

[16] Docker, "Docker Overview." [Online]. Available: https://docs.docker.com/engine/docker-overview/#docker-architecture. [Accessed: 15-Jun-2017].