Perancangan dan Implementasi *Network Functions Virtualization* (NFV) Berbasis *Cloud Computing* dengan OpenStack

Muhammad Fadhil[#1], Eddy Prasetyo Nugroho[#2], Yudi Wibisono[#3], Ibrahim Zein Abdillah[*4]

[#]*Departemen Pendidikan Ilmu Komputer, Universitas Pendidikan Indonesia*
Bandung, Indonesia
[#1] muhammad.fadhil94@student.upi.edu
[#2,3] {eddypn, yudi}@upi.edu

[*]*Divisi Digital Service, PT. Telekomunikasi Indonesia, Tbk.*
Bandung, Indonesia
[*4] ibrahim.zein@telkom.co.id

***Abstrak* – *Network Functions Virtualization* (NFV) adalah sebuah inisiatif yang digagas oleh sejumlah *service provider* (SP) oleh European Telecommunications Standards Institute (ETSI) yang bertujuan untuk mentransformasikan cara suatu penyedia layanan jaringan dengan mengembangkan teknologi virtualisasi untuk menggabungkan berbagai tipe perangkat jaringan ke perangkat server dalam bentuk Software Defined Networking (SDN). OpenStack menyediakan dasar untuk arsitektur NFV dalam bentuk Infrastructure-as-a-Service (IaaS), yang dapat digunakan untuk melakukan *deploy*, *orchestration*, dan mengelola Virtual Network Functions (VNF) dengan OpenStack Tacker. Penelitian ini bertujuan merancang dan mengimplementasikan NFV dengan OpenStack, serta membangun perangkat lunak pengelolaan NFV berbasis web dengan bahasa pemrograman PHP dan *Application Programming Interface* (API) OpenStack Tacker. Eksperimen yang dilakukan menunjukkan OpenStack dapat diimplementasikan sebagai NFV Infrastructure (NFVI) dan implementasi layanan *Virtual Router* dan *Virtual Firewall* dapat dilakukan pada VNF OpenWRT. Hasil eksperimen yang dilakukan didapat nilai rata-rata kinerja VNF dengan throughput tertinggi 938,37 Mbps dan CPU *Utilization* terendah 30,44%.**

Kata Kunci – ***Cloud Computing, Virtualisasi, NFV, OpenStack.***

I. PENDAHULUAN

Cloud computing telah berdampak besar dalam konsep komputasi selama satu dekade terakhir. Paradigma mengenai virtualisasi komputasi telah dikembangkan sekitar tahun 1990, menjelaskan sebuah server fisik dapat menampung banyak Virtual Machine (VM) serta menampung Operating System (OS). Layanan cloud telah dikembangkan dalam Infrastruktur, Platform, dan Software sebagai layanan. *Service Provider* (SP) telah berperan sebagai penyedia pengiriman *bitstream*, sementara *Application Service Provider* (ASP) mengembangkan layanan yang bersifat *Over the Top* (OTT) secara terpusat pada jaringan SP. SP mengetahui bahwa mereka tidak dapat bersaing dengan ASP yang berskala global dengan aplikasi yang mereka tawarkan. Melihat dampak dari *cloud computing*, ASP seperti WhatsApp dan Skype menjual layanan OTT yang secara langsung bersaing dengan layanan suara atau telepon dan pesan singkat atau *Short Message Service* (SMS) yang umumnya disediakan oleh SP. Akibatnya, SP mengalami penurunan besar dalam layanan *roaming* internasional mereka yang sangat menguntungkan [1]. Setiap SP telah banyak berinvestasi dalam infrastruktur jaringan mereka, memasang jaringan berbasis *fiber*, sementara ASP berhasil memanfaatkan *cloud computing* tanpa perlu biaya untuk berinvestasi pada infrastruktur jaringan. SP tidak dapat terus menerus membeli perangkat jaringan, sementara ASP mendapat keuntungan dari *Cloud Computing*. Sejumlah SP (AT&T, BT, Deutsche Telekom, Orange, Telecom Italia, Telefonica, dan Verizon) yang terpilih oleh European Telecommunications Standards Institute (ETSI) membentuk inisiatif untuk mengembangkan *Network Functions Virtualization* (NFV) [2].

NFV bertujuan untuk mentransformasikan cara suatu penyedia layanan jaringan dengan mengembangkan teknologi virtualisasi untuk menggabungkan berbagai tipe perangkat jaringan ke dalam server, *switch*, dan *storage*, yang dapat ditemukan pada *Data Center*, *node* jaringan, dan perangkat yang berada pada sisi pelanggan. Hal tersebut melibatkan implementasi fungsi-fungsi jaringan dalam bentuk perangkat lunak (*software*) yang dapat dijalankan pada perangkat *server*, tanpa memerlukan instalasi perangkat baru [3]. Ide utama NFV adalah mengembangkan *Elastic Network* dan melakukan migrasi dari hardware dengan cara yang serupa seperti pada Data Center yang menggunakan *Software Defined Networking* (SDN).

Konsep NFV berawal dari penyedia layanan atau perusahaan telekomunikasi yang mencari solusi untuk mempercepat implementasi layanan baru untuk

mendukung strategi bisnis dan pertumbuhan pendapatan mereka. Salah satu hambatan signifikan yang mereka rasakan adalah ketergantungan terhadap perangkat keras. Dengan konsep tersebut, sebuah layanan dapat didistribusikan ke SP sebagai *software*.

Penyediaan layanan jaringan umumnya menggunakan perangkat-perangkat fisik dan peralatan lainnya dari setiap fungsi pada jaringan yang menjadi bagian dari layanan hingga dapat tersedia untuk pelanggan. Namun, kebutuhan pelanggan untuk beragam layanan-layanan baru dengan penggunaan yang tinggi terus meningkat. Oleh karena itu, para penyedia layanan harus terus membeli, menyimpan dan mengoperasikan perangkat baru [4].

Dengan permintaan dan jumlah pelanggan yang terus bertambah, meningkatnya biaya modal dan biaya operasional tidak dapat diwujudkan dalam biaya berlangganan yang lebih tinggi. Dengan naiknya biaya berlangganan, pelanggan akan mulai mempertimbangkan untuk meninggalkan penyedia layanan jaringan tersebut. Oleh karena itu, SP mencari cara untuk membangun jaringan yang lebih dinamis dengan tujuan mengurangi siklus produk, pengoperasian dan pengeluaran modal serta meningkatkan kecepatan layanan [5].

Ada tiga proyek yang terkait dengan NFV: Open NFV (OPNFV), digagas oleh Linux Foundation yang menggunakan OpenStack sebagai Virtual Infrastructure Manager (VIM). Dua proyek lainnya diluncurkan pada tahun 2016, yang pertama berasal dari laboratorium Telefonica di Spanyol bersama dengan ETSI yaitu Open Source MANO (OSM), dan yang kedua diinisiasi oleh perusahaan dari Tiongkok, China Mobile dan Huawei, digagas bersama dengan Linux Foundation yang awalnya diberi nama Open Orchestrator Project (OPEN-O) namun baru saja bergabung dengan proyek AT&T yaitu Enhanced Control, Orchestration, Management and Policy (ECOMP) pada tanggal 23 Februari 2017 untuk membuat Open Network Automation Platform (ONAP) [2].

OpenStack menyediakan dasar untuk arsitektur NFV dalam bentuk *Infrastructure-as-a-Service* (IaaS), yang secara esensial dapat digunakan untuk melakukan *deploy*, orkestrasi (*orchestration*), dan mengelola *Virtual Network Functions* (VNF). OpenStack bersifat terbuka (*open*), modular, dan interoperabilitasnya memudahkan SP untuk merancang sistem NFV yang sesuai dengan keinginan mereka. OpenStack dapat diimplementasikan untuk merealisasikan NFV berdasarkan standar ETSI NFV sebagai *NFV Infrastructure* (NFVI), dengan melakukan virtualisasi *resources* berupa *compute*, *storage*, dan *networking* pada perangkat *server*, melalui *virtualization layer* berupa Operating System dan Hypervisor.

Pada bulan April 2016, OpenStack memperkenalkan layanan (service) baru yang berfungsi sebagai NFV Orchestration, yaitu Tacker. Tacker memungkinkan para pengembang dapat mengimplementasikan NFVI, membangun VNF Manager (VNFM) dan NFV Orchestrator (NFVO) untuk menyediakan elemen Management & Orchestration (MANO) agar dapat menerapkan dan mengoperasikan layanan-layanan jaringan dan VNF pada infrastruktur OpenStack. Selain itu, Tacker juga memiliki dukungan Application Programming Interface (API) dan menyediakan endpoint yang berdasar pada ETSI NFV MANO yang memungkinkan pengembang dapat membuat program untuk pengelolaan VNF pada OpenStack.

II. PENELITIAN TERKAIT

Sebagian besar penelitian yang membahas NFV berawal dari konsep *cloud computing* yang memiliki peluang untuk melakukan virtualisasi suatu *hardware* yang mana layanan berbasis cloud dapat dikembangkan menjadi sebuah infrastruktur, *platform*, dan *software*. Namun, NFV masih dalam tahap pengembangan awal. ETSI telah menyediakan standar spesifikasi untuk arsitektur NFV dan sejumlah proyek yang berkembang untuk memenuhi standar tersebut. Proyek OPNFV yang digagas oleh Linux Foundation melihat potensi dari proyek-proyek virtualisasi dan platform cloud computing seperti OpenStack , Kernel Virtual Machine (KVM), dan Open vSwitch (OvS) dapat dikembangkan sebagai komponen NFV dengan ekosistem *open source* [2]. Paradigma NFV menggabungkan fleksibilitas dan kendali terhadap setiap perangkat jaringan yang ditawarkan oleh SDN, yang memungkinkan untuk melakukan virtualisasi terhadap jaringan yang akan membentuk masa depan arsitektur jaringan, SP tradisional akan digantikan dengan dengan data *center cloud*. [6].

OpenStack bekerja secara paralel dengan komponen-komponen lainnya yang mampu menyertakan elemen NFVO dan VNFM pada ekosistem OpenStack. OpenStack Tacker telah dirilis sebagai bagian dari OpenStack versi Mitaka pada Maret 2016. Tacker menggabungkan komponen-komponen OpenStack, yang kemudian dapat digunakan untuk pengujian *deployment* NFV [2].

Penelitian sebelumnya [7] membahas mengenai analisis *throughput* dan skalabilitas VNF VyOS di berbagai [2] *hypervisor*. Uji skalabilitas pada penelitian ini menguji *throughput* ketika jumlah VNF ditingkatkan. Hasil penelitian menyatakan KVM memiliki nilai *throughput* paling baik.

Walaupun sudah ada penelitian mengenai NFV [6] [7], tetapi belum ada penelitian yang menggunakan OpenStack dengan Tacker sebagai NFV *Orchestration* pada infrastruktur NFV berbasis OpenStack. Dalam penelitian ini akan dirancang sebuah konfigurasi sistem infrastruktur NFV dengan OpenStack menggunakan OpenStack Tacker sebagai NFV *Orchestration*, serta membuat perangkat lunak pengelolaan NFV dengan menggunakan API OpenStack Tacker untuk keperluan *deployment* VNF.

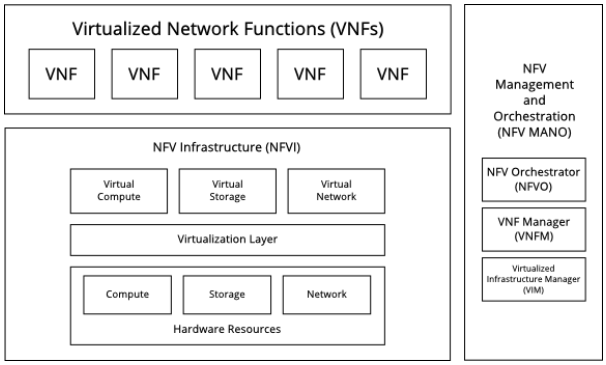
III. NETWORK FUNCTIONS VIRTUALIZATION

Network Functions Virtualization berhubungan dengan transisi dari sekumpulan perangkat *proprietary* dalam bentuk perangkat keras ke dalam bentuk perangkat lunak (*software-defined*) yang dijalankan pada perangkat keras berstandar industri. Pada SP tradisional, masing-masing

fungsi yang berbeda diimplementasikan sebagai peralatan yang khusus berdasarkan perangkat keras yang bersifat *proprietary*. Contoh elemen perangkat jaringan berbasis perangkat keras yang *proprietary* adalah beragam jenis *router*, perangkat *deep packet inspection*, *mobile base station controllers*, *mobile packet gateways* dan lain-lain.

NFV adalah konsep yang dimunculkan untuk membuat *network function* yang dapat diimplementasikan seluruhnya yang berbentuk *software-defined* yang berjalan pada perangkat keras berstandar industri. Secara umum, perangkat keras berstandar industri merupakan *server* komersial yang berbasis arsitektur Intel x86, bersama dengan perangkat *switch Ethernet*. Dengan konsep ini, sebuah *network function* dapat didistribusikan ke SP hanya dalam bentuk *software*. Yang perlu SP lakukan hanyalah instalasi *software* tersebut pada infrastruktur *Data Center* mereka, beserta perangkat seperti perangkat *server* yang saling terhubung dengan *switch ethernet*.

ETSI memiliki standar rancangan arsitektur NFV yang terbagi atas tiga elemen, yaitu NFVI, NFV MANO, dan VNF [8] yang dapat dilihat pada Gambar 1 sebagai berikut:



Gambar 1. Arsitektur NFV (ETSI GS NFV 002)

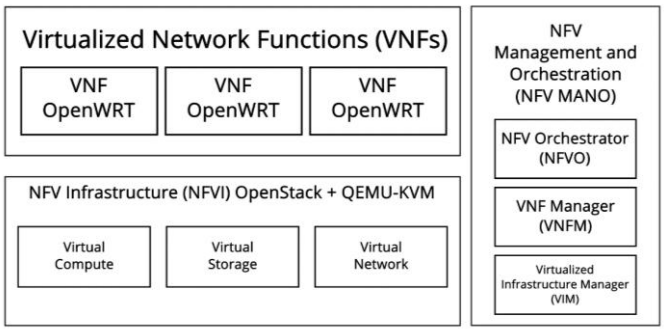
1. NFVI menyediakan sumber daya virtual yang dibutuhkan untuk mendukung eksekusi dari *Virtualized Network Functions* (VNF). Pada NFVI termasuk didalamnya perangkat keras dengan *Commercial-off-the-shelf* (COTS), komponen akselerator ketika dibutuhkan, dan lapisan *software* yang divirtualisasikan secara abstrak terletak pada perangkat keras tersebut.
2. VNF merupakan implementasi *software* dari sebuah fungsi jaringan yang mampu berjalan pada NFVI. VNF dapat dijalankan bersamaan dengan *Element Management System* (EMS), sepanjang berlaku untuk fungsi yang khusus, yang memahami dan mengatur sebuah VNF yang individual dan tidak lazim. VNF merupakan entitas yang berhubungan dengan *node* jaringan saat ini, yang mana diharapkan dihadirkan dalam bentuk *software* murni terlepas dari ketergantungan perangkat keras.
3. NFV MANO meliputi orkestrasi (*orchestration)* dan manajemen siklus hidup dari sumber daya fisik atau *software* yang mendukung infrastruktur virtualisasi, dan manajemen siklus hidup dari VNF. NFV MANO berfokus ke tugas pengelolaan virtualisasi yang khusus pada *framework* NFV. NFV MANO juga berinteraksi dengan NFV eksternal seperti OSS/BSS, yang memungkinkan NFV diintegrasikan dengan jaringan yang sudah ada secara luas.

IV. HASIL DAN PEMBAHASAN

Pada penelitian ini terdapat empat pembahasan dalam perancangan dan implementasi NFV berbasis OpenStack, yaitu melakukan perancangan dan konfigurasi infrastruktur sistem, dan melakukan analisis terkait hasil eksperimen kinerja VNF dengan parameter *throughput* dan CPU *Utilization*.

A. *Konfigurasi Sistem*

Dalam tahap konfigurasi sistem, dilakukan konfigurasi infrastruktur dengan standar yang diadaptasi dari publikasi ETSI GS NFV 002: Network Functions Virtualization; Architectural Framework, yang meliputi ketiga elemen dalam NFV. Rancangan dan konfigurasi infrastruktur sistem dapat dilihat pada Gambar 2.



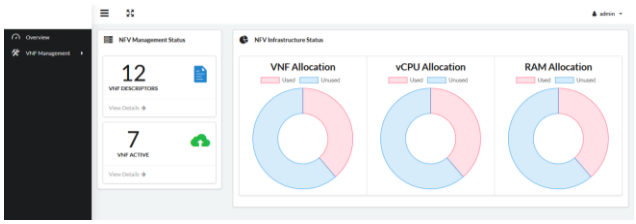
Gambar 2. Rancangan dan konfigurasi sistem

1. VNF yang digunakan dalam perancangan ini adalah *image* OpenWRT, yang akan difungsikan sebagai *Virtual Router* dan *Virtual Firewall*.
2. NFVI dalam perancangan ini menggunakan OpenStack RDO versi Newton. Dengan konfigurasi Compute, Storage, dan Network pada satu buah *node* yang sama. *Node* yang digunakan adalah sebuah PC Server HP ProLiant DL380 Gen 8. *Hypervisor* pada *Virtualization Layer* yang digunakan adalah QEMU-KVM.
3. NFV MANO yang digunakan dalam perancangan ini adalah Tacker. Tacker akan menjadikan OpenStack sebagai *Virtualized Infrastructure Manager*.

B. *Pengembangan Perangkat Lunak*

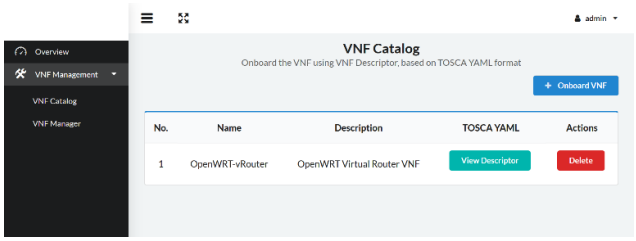
Pembangunan perangkat lunak dilakukan untuk menerapkan pengelolaan NFV (NFV *Management*) dengan API OpenStack Tacker dalam sistem menggunakan bahasa pemrograman PHP dengan *framework* CodeIgniter dan php-opencloud. Perangkat

lunak yang dibangun memiliki fungsi sebagai VNF Management pada infrastruktur *cloud* berbasis OpenStack. Terdapat menu VNF *Management* yang memiliki fungsi untuk melakukan *deployment* VNF, meliputi *Onboard* VNF, dan mengelola VNF yang berjalan pada arsitektur sistem yang dirancang. Perangkat lunak ini ditujukan untuk *administrator* pada SP agar dapat mengelola VNF yang dibuat pada infrastruktur *cloud* OpenStack dengan Tacker. Terdapat tiga antarmuka utama pada perangkat lunak Generic VNF Manager, yaitu *Overview Monitoring*, VNF *Catalog*, dan VNF *Manager*. Tampilan antarmuka *Overview Monitoring* dapat dilihat pada Gambar 3.



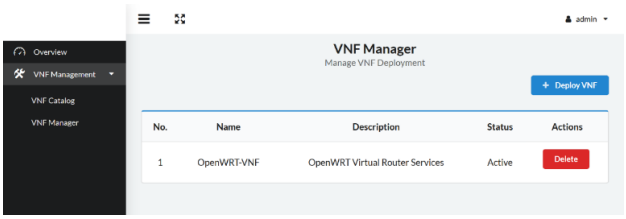
Gambar 3. Tampilan Antarmuka *Overview Monitoring*

Pada halaman *Overview Monitoring*, terdapat informasi jumlah VNF *Descriptor* yang tersedia, dan jumlah VNF yang aktif yang ditampilkan pada bagian NFV *Management Status*, dan NFV *Infrastructure Status* menampilkan jumlah alokasi VNF, vCPU dan RAM yang telah digunakan dan yang belum digunakan pada infrastruktur *cloud* OpenStack. Selanjutnya, halaman VNF *Catalog administrator* dapat melakukan Onboard VNF dengan melakukan input template TOSCA YAML, menampilkan daftar VNF *Descriptor* yang tersedia, melihat isi dari VNF *Descriptor*, dan menghapus VNF *Descriptor*. Tampilan antarmuka VNF *Catalog* dapat dilihat pada Gambar 4.



Gambar 4. Tampilan Antarmuka VNF *Catalog*

Selanjutnya, pada halaman VNF *Manager* admin dapat mengelola VNF yang meliputi *deploy* VNF menggunakan *template* VNF dan menghapus VNF yang aktif. Tampilan antarmuka VNF *Manager* dapat dilihat pada Gambar 5.



Gambar 5. Tampilan Antarmuka VNF Catalog

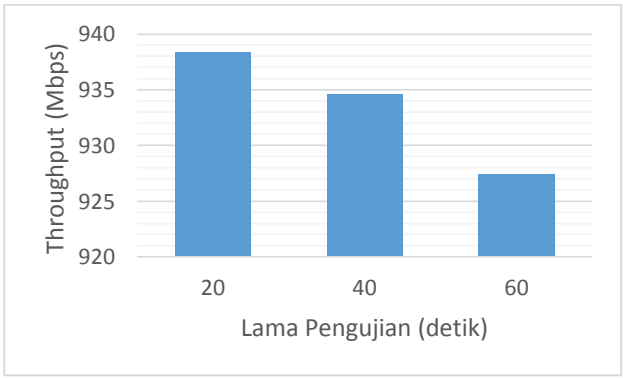
C. Perbandingan Throughput berdasarkan Lama Pengujian

Pengujian *throughput* dilakukan menggunakan aplikasi *iperf3* dengan *protocol* TCP untuk menguji kemampuan *virtual* NIC pada VNF dalam menerima *traffic* dari *client* [9]. Dalam eksperimen ini, pengujian dilakukan dengan lama waktu pengujian yaitu 20, 40, dan 60 detik. Dari eksperimen yang dilakukan, didapatkan hasil rata-rata pengujian pada parameter *throughput* dan CPU *Utilization* yang dapat dilihat pada Tabel I. Kolom *Throughput* menunjukkan jumlah maksimum *bandwidth* yang dapat dicapai oleh virtual NIC pada VNF, dengan satuan Mbits/sec (Mbps). Sedangkan CPU *Utilization* menunjukkan persentase utilisasi CPU pada VNF dalam menangani *traffic*.

TABEL 1 . HASIL RATA-RATA PENGUJIAN THROUGHPUT DAN CPU UTILIZATION

Lama Pengujian (detik)	Throughput (Mbps)	CPU Utilization (%)
20	938,37	30,44
40	934,57	32,85
60	927,43	33,8

Dari pengujian yang dilakukan, hasil yang didapat pada *throughput* berbanding terbalik dengan CPU *Utilization*. Dalam hal ini, *throughput* yang dihasilkan pada pengujian 20 detik mendapat nilai tertinggi yaitu 938,37 Mbps dengan CPU *Utilization* 30,44%. Sedangkan pada pengujian 60 detik didapatkan nilai terendah yaitu 927,43 dengan CPU *Utilization* 33,8%. Hasil perbandingan rata-rata pengujian *throughput* dapat dilihat pada grafik yang ditunjukkan pada Gambar 6.



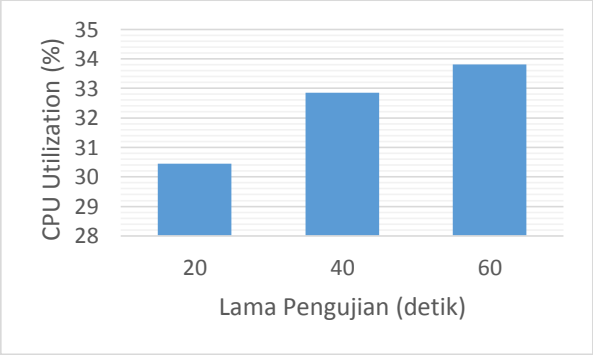
Gambar 6. Hasil rata-rata pengujian throughput

Dari Gambar 3 dapat dilihat jika semakin lama VNF menerima *traffic* secara bersamaan maka akan semakin rendah nilai *throughput* yang dihasilkan. Dari hasil tersebut, lama waktu pengujian akan mempengaruhi nilai *throughput* yang dihasilkan, dengan asumsi jumlah *traffic* yang diterima berbeda pada setiap lama pengujian.

Penelitian sebelumya [6] dengan pengujian 100 detik pada OpenStack versi Mitaka menghasilkan *throughput* di kisaran 800 Mbps. Sedangkan dalam penelitian ini yang menggunakan OpenStack versi Newton menghasilkan *throughput* di kisaran 900 Mbps. Perbandingan yang signifikan ini tentu tidak terlepas dari perbedaan kondisi jaringan dan kemampuan NIC pada *client* dan *server* yang digunakan pada 1 tahun lalu, sehingga nilai *throughput* yang dihasilkan bervariasi.

D. Perbandingan Kinerja VNF berdasarkan CPU Utilization

Hasil eksperimen kinerja VNF berdasarkan CPU *Utilization* pada saat pengujian *throughput* dapat memperlihatkan perbandingan utilisasi CPU pada saat menerima *traffic* dengan lama pengujian 20 detik, 40 detik, dan 60 detik. CPU *Utilization* merupakan faktor lain dalam pengujian kinerja VNF untuk melihat utilisasi CPU saat memproses jumlah *throughput* yang diterima. Peningkatan CPU *Utilization* akan terjadi ketika VNF mulai menerima *traffic* dari *client*. Dari hasil eksperimen pada Tabel I, didapatkan hasil perbandingan rata-rata CPU *Utilization* berdasarkan lama pengujian *throughput* yang telah dilakukan yang dapat dilihat pada Gambar 7.



Gambar 7. Hasil rata-rata CPU Utilization

Dari Gambar 4 terlihat persentase CPU *Utilization* pada setiap pengujian *throughput* dengan lama pengujian 20, 40, dan 60 detik. Semakin lama waktu pengujian, CPU *Utilization* akan semakin meningkat pada saat *throughput* yang diterima VNF diproses oleh CPU. Dari keseluruhan pengujian yang dilakukan, terlihat bahwa CPU *Utilization* dapat mempengaruhi hasil *throughput* yang diterima. Jika dibandingkan dengan hasil rata-rata pengujian *throughput* yang ditunjukkan pada Gambar 3, dengan CPU *Utilization* yang rendah maka *throughput* yang dihasilkan semakin tinggi. Sebaliknya, dengan CPU *Utilization* yang tinggi maka *throughput* yang dihasilkan semakin rendah. Hal ini dipengaruhi oleh jumlah atau ukuran *traffic* yang dikirimkan oleh *client* yang mempengaruhi beban kerja CPU pada VNF saat menerima 10 koneksi secara bersamaan

V. KESIMPULAN

Berdasarkan hasil dan pembahasan pada eksperimen yang telah dilakukan, kinerja VNF dengan parameter *throughput packet* berbanding terbalik dengan CPU *Utilization* jika dilihat dari lamanya waktu pengujian. Peningkatan CPU *Utilization* terjadi ketika VNF mulai menerima *traffic* dari *client*. Namun hal ini disebabkan oleh perbedaan jumlah *traffic* yang diterima oleh VNF, kondisi jaringan, dan kemampuan NIC pada *client* dan *server*, serta beban kerja CPU pada VNF saat menerima *traffic*. Selain itu, versi OpenStack yang digunakan juga dapat memberikan perbedaan hasil *throughput*. Hasil perancangan dan implementasi yang dilakukan memiliki kinerja VNF dengan nilai rata-rata *throughput* dan CPU *Utilization* secara berturut-turut 938,37 Mbps dan 30,44%.

REFERENSI

[1] E. Heinrich, "Telecom companies count $386 billion in lost revenue to Skype, WhatsApp, others," 2014. [Daring]. Tersedia pada: http://fortune.com/2014/06/23/telecom-companies-count-386-billion-in-lost-revenue-to- skype-whatsapp-others/.

[2] D. Denieffe, Y. Kavanagh, D. Denieffe, Y. Kavanagh, dan D. Okello, "A report on the status of Network Functions A report on the status of Network Functions Virtualisation," no. April, 2017.

[3] M. Chiosi *et al.*, "Network Functions Virtualisation,An Introduction, Benefits, Enablers, Challenges & Call for Action," *Citeseer*, no. 1, hal. 1–16, 2012.

[4] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck, dan R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surv. Tutorials*, vol. 18, no. 1, hal. 236–262, 2016.

[5] J. Wu, Z. Zhang, Y. Hong, dan Y. Wen, "Cloud radio access network (C-RAN): A primer," *IEEE Netw.*, vol. 29, no. 1, hal. 35–41, 2015.

[6] F. Callegati, W. Cerroni, dan C. Contoli, "Virtual Networking Performance in OpenStack Platform for Network Function Virtualization," *J. Electr. Comput. Eng.*, vol. 2016, 2016.

[7] R. F. Aswariza, D. Perdana, dan R. M. Negara, "Analisis Throughput dan Skalabilitas Virtualized Network Function VyOS Pada Hypervisor VMware ESXi , XEN , dan," vol. 9, no. 1, hal. 70–74, 2017.

[8] ETSI, "NFV-Architectural Framework," *Etsi*, vol. 1, hal. 1–21, 2013.*Urban Planning*, vol. 55, pp. 79-93, October 2001.

[9] J. Menon, "A Performance Comparison of Hypervisors," hal. 167–178, 2011.