ANALISIS PERFORMANSI PENGGUNAAN ONOS SONA-CNI DI JARINGAN KUBERNETES

PERFORMANCE ANALYSIS OF THE USE OF ONOS SONA-CNI IN THE KUBERNETES NETWORK

Bachril Qirom[1], Sofia Naning Hertiana[2], Ridha Muldina Negara[3]
[1,2,3]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
bachrilq@students.telkomuniversity.ac.id,[2]sofiananing@telkomuniversity.ad,
ridhanegara@telkomuniversity.ac.id

Abstrak

Container merupakan salah satu cara untuk membangun sebuah layanan di internet. Dengan menggunakan Container kita bisa membangun layanan lebih ringan dan cepat. Menjalankan sebuah layanan membutuhkan banyak container, sehingga dibutuhkannya *Container Orchestration* seperti Kubernetes. Namun ada beberapa masalah performa jaringan di Kubernetes. Disebabkan pengaturan jaringan yang kurang optimal karena menggunakan interface yang sama untuk komunikasi internal dan eksternal, pemilihan protokol *tunneling* yang kurang tepat, serta kurangnya fitur untuk memantau trafik paket.

Pada tugas akhir ini akan membahas tentang performansi jaringan Kubernetes menggunakan *Software Defined Networking* (SDN) dengan menggunakan SONA-CNI. SONA-CNI menggunakan pengaturan jaringan terpusat sehingga dapat memantau trafik lebih mudah dan memisahkan *network interface* yang digunakan untuk manajemen dan pengaksesan dari luar. Ada 3 protokol *tunneling* yang dapat digunakan SONA-CNI yaitu GRE, VxLAN dan GENEVE. Pengujian performansi dengan cara membandingkan dengan CNI Flannel dan Calico, serta membandingkan protokol *tunneling* yang digunakan.

Dari hasil pengujian dapat disimpulkan, besar MTU pada *network interface* berpengaruh pada protokol tunneling yang akan digunakan. Hanya SONA-CNI yang menyediakan monitoring trafik paket di kubernetes tanpa memasang aplikasi tambahan. SONA-CNI menjadi paling baik dalam pengujian pada parameter *latency* dengan nilai sekitar 29.63% - 48.46% lebih rendah dibandingkan Flannel dan Calico. Untuk parameter *throughput* SONA-CNI sedikit lebih rendah dibandingkan Flannel dan Calico sekitar 4.46% - 31.04%. Namun pada parameter *packet loss* SONA-CNI mendapatkan persentase paling buruk sekitar 23.64%-290.48% lebih tinggi dibandingkan Flannel dan Calico.

Kata Kunci: ***Container, CNI, SDN, Kubernetes, Controller, Tunneling.***

Abstract

Containers are one way to make services on the internet. By using Container, we can make services lighter and fast. Running a service requires many containers. So it needs Container Orchestration like Kubernetes. But in Kubernetes have several Network performance issues. Due to network are not optimal, because use the same interface for internal and external communication, the selection of tunneling protocols that are less precise, as well as features for monitoring traffic packets on the Kubernetes network.

In this final Thesis will explain about the performance of the Kubernetes network using Software Defined Networking (SDN), using SONA-CNI. SONA-CNI uses centralized network settings so that it can monitor traffic more easily, and separate the network interface used for management and access from the outside. There are 3 tunneling protocols that can be used by SONA-CNI is GRE, VxLAN, and GENEVE. Performance testing by comparing with CNI Flannel and Calico, and comparing the tunneling protocol used.

The test results can be concluded, the large MTU on the network interface affects the tunneling protocol to be used. Only SONA-CNI that provides packet traffic monitoring without installing additional applications. SONA-CNI is the best in testing on latency parameters with values around 29.63% - 48.46% smaller than Flannel and Calico. The SONA-CNI throughput parameter a little lower than Flannel and Calico, although it is lower

around 4.46% - 31.04%. However, the packet loss parameter SONA-CNI get the worst percentage of about 23.64%-290.48% higher than Flannel and Calico.

Keywords: Container, CNI, SDN, Kubernetes, Controller, Tunneling.

1. Pendahuluan

Pada dewasa ini teknologi *cloud computing* dan internet berkembang sangat cepat. Ditandai dengan penggunaan layanan internet menjadi sangat banyak, sehingga tidak cukup hanya menggunakan server fisik sebagai penyedia layanan. Untuk mengatasi hal tersebut maka dikembangkan teknologi baru seperti virtualisasi dan *container* [1].

Virtualisasi merupakan teknologi yang memungkinkan satu perangkat komputer dapat menjalankan lebih dari satu *operating system* secara bersamaan[2]. Tujuan dari virtualisasi agar dapat memaksimalkan sumber daya pada komputer. Perkembangan dari teknologi virtualisasi adalah teknologi *container*. *Container* secara garis besar mirip seperti virtualisasi. Namun pada container menggunakan *kernel host* dibawahnya atau yang di sebut *shared kernel*[3]. Ini menye babkan container menjadi lebih ringan dalam penggunaan sumber daya dibandingkan virtualisasi. Karena penggunaan sumber daya pada container lebih kecil dibandingkan virtualisasi, maka jumlah *container* akan lebih banyak dibandingkan virtualisasi[3]. *Container* yang berjumlah banyak membutuhkan aplikasi tambahan atau biasa disebut *container orchestration*. Terdapat beberapa aplikasi *container orchestration* yaitu Docker Swarm, Apache Mesos, dan yang terkenal adalah Kubernetes[4].

Pada kubernetes setidaknya membutuhkan dua *node server* yang digunakan sebagai *Master Node* dan *Worker Node*. Kubernetes tidak menyediakan komponen jaringan didalamnya, namun hanya menyediakan model jaringannya saja. Sehingga konfigurasi jaringan pada Kubernetes diatur oleh aplikasi tambahan yang disebut *Container Network Interface* (CNI). Beberapa CNI yang direkomendasikan oleh *Cloud Native Computing Foundation* (CNCF) adalah flannel dan calico [5]. Namun pada flannel dan calico belum dapat memantau trafik paket secara jelas. Serta performa yang dihasilkan masih belum bisa maksimal, karena pemilihan protokol *tunneling* yang kurang tepat[1], dan menggunakan *interface* yang sama untuk komunikasi *internal* dan *external*[4]. Sehingga diharapkan dengan menggunakan konsep SDN pada CNI dapat memberikan performa yang maksimal pada jaringan. Karena terdapat *controller* yang mengatur keseluruhan jaringan. Serta dengan dipisahnya *interface* yang digunakan untuk akses dari luar diharapkan juga akan meningkatkan performa jaringan pada kubernetes.

Dalam tugas akhir ini akan menggunakan SDN *Controller* ONOS dengan proyek SONA-CNI kubernetes. Karena pada SONA-CNI kubernetes terdapat pemisahan permrosesan jaringan secara *logical*, serta menggunakan konsep SDN pada pengaturan jaringannya. Dan juga memisahkan *network interface* yang digunakan untuk manajemen dan pengaksesan dari luar. Serta memiliki beberapa kelebihan yaitu protokol *tunneling* yang bervariasi serta dapat diubah, dan dapat memantau trafik paket pada jaringan.

2. Dasar Teori /Material dan Metodologi/perancangan

2.1 *Software Defined Network*

Software Defined Networking (SDN) merupakan arsitektur jaringan yang memisahkan antara kontrol jaringan dan forwarding plane yang dapat langsung terprogram[6]. Dengan hal ini proses pengontrolan jaringan menjadi terpusat, serta dapat melihat keseluruhan topologi pada suatu jaringan. SDN membuat sebuah jaringan menjadi lebih sederhana dalam mendesain serta mengontrol. Karena tidak memerlukan banyak standar protokol hanya menerima perintah yang diberikan oleh kontroler saja.

2.2 Container

Container merupakan pengembangan dari *Virtual Machine* (VM). Container mirip seperti VM namun memiliki perbedaan pada sisi virtualisasi. Container berjalan dengan menggunakan k*ernel host* dan tidak membutuhkan *hypervisor* seperti VM [7].

2.3 Kubernetes

Kubernetes merupakan aplikasi *open source* yang digunakan untuk memanajemen *container* dalam jumlah banyak pada *server* yang berbeda-beda (*Container Orchestrator*)[8]. Pada arsitektur Kubernertes dibagi menjadi dua bagian. *Master node* dan *worker node*. *Master Node* sebagai tempat mengatur container. *Worker Node* sebagai *host* dari *container* dijalankan.

2.4 *Container Network Interface* (CNI)

Container Network Interface (CNI) merupakan spesifikasi jaringan container yang diusulkan oleh CoreOS[1]. Plugin jaringan pada CNI hanya menyediakan dua perintah. Pertama untuk menambahkan antarmuka pada Container dan kedua untuk menghapusnya.

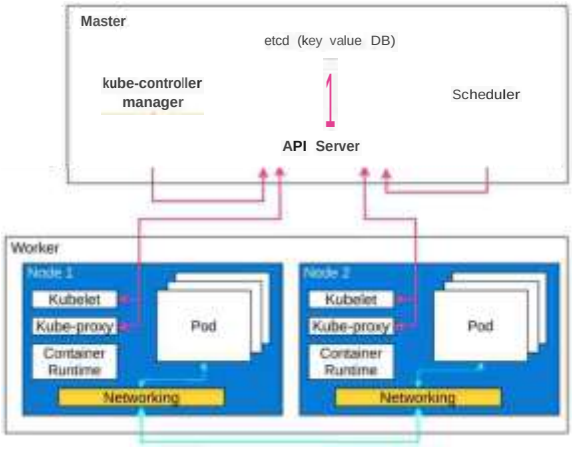
2.5 *Tunneling*

Tunneling protocol merupakan salah satu protokol jaringan yang memungkinkan kita dapat berkomunikasi dengan jaringan lain menggunakan ip *private* dengan melewati *ip public*. Konsep dari *tunneling protocol* adalah dengan mengenkapsulasi paket dengan dua *header*[9]. *Header* pertama digunakan untuk menentukan protokol *tunneling* yang digunakan. *Header* kedua digunakan untuk berkomunikasi dengan jaringan *public*.

3. Pembahasan

3.1. Desain Sistem

Pada bab ini, akan dibahas mengenai gambaran system secara umum yang dapat dilihat pada Gambar 3.1:

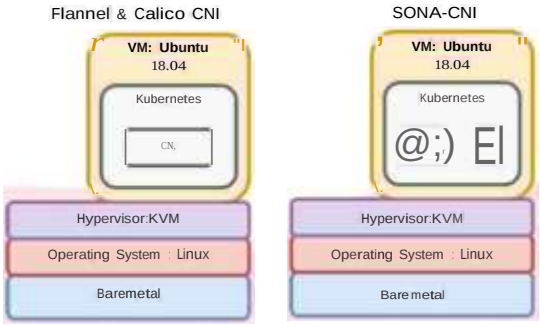


Gambar 3.1 Gambaran Sistem Secara Umum

Pada Gambar 3.1 menjelaskan gambaran sistem secara garis besar, Komponen yang diperlukan pada sistem adalah *master*, *worker*, dan *networking* yang berupa CNI. Pengerjaan Tugas Akhir ini dilakukan secara simulasi sehingga membutuhkan *hypervisor* yang digunakan untuk menjalankan VM. *Node master* disini digunakan sebagai pengatur pod yang dijalankan dan memelihara kondisi pod yang telah dijalankan. *Node worker* berfungsi sebagai tempat pod dijalankan. *Networking m*erupakan komponen yang tidak disediakan secara langsung oleh kubernetes, namun kubernetes menyediakan model jaringannya saja yang di sebut CNI. Sehingga dapat dengan bebas memilih CNI yang akan digunakan.

3.2. Perancangan Kebutuhan Sistem

Pada bagian ini akan dijelaskan mengenai perancangan sistem yang diuji. Sistem akan dibangun diatas *Virtual Machine* (VM) dengan menggunakan *hypervisor* KVM diatas *host*. Arsitektur Kebutuhan sistem yang menggunakan CNI flannel atau calico, dan yang menggunakan SONA-CNI dapat dilihat pada Gambar 3.2



Gambar 3.2 Perancangan Kebutuhan Sistem

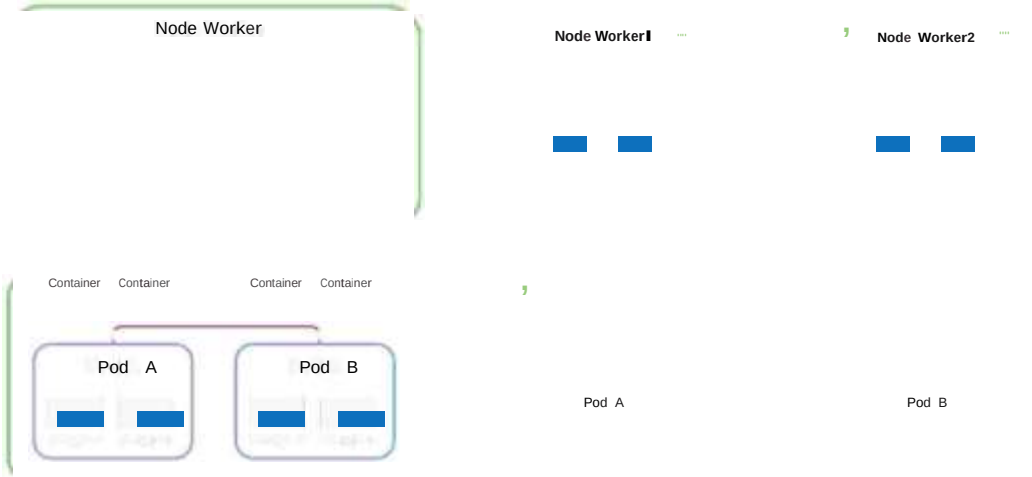
Seperti yang tertera di Gambar 3.2 sistem operasi yang akan digunakan adalah ubuntu server 18.04. Dimana Kubernetes dan perangkat lunak pendukungnya akan dipasang didalam ubuntu 18.04. VM akan terdiri dari satu VM kubernetes *master*, dua VM kubernetes *worker*. Dengan spesifikasi VM seperti yang tertera pada Tabel 3.1

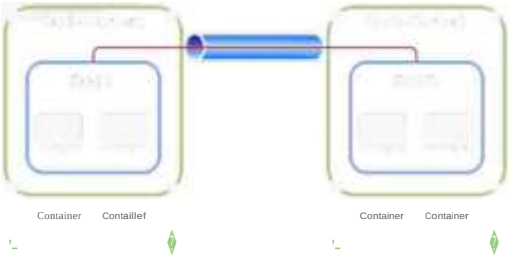
Tabel 3.1 Spesifikasi VM

Spesifikasi	*Node Master* dan *Node Worker*	
	Flannel dan Calico	SONA-CNI
Processor	4 *Core*	
Memory	4 GB	
Harddisk	10 GB	
Network Interface	1 *Network*	2 *Network*

3.4. Skenario Pengujian

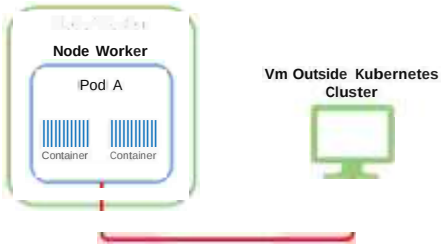
Pengujian berupa melakukan pengecekan fitur monitoring trafik paket pada CNI dan pengujian terhadap konektifitas Pod yang berada pada Kubernetes. Skenario dilakukan dengan menggunakan 3 sistem yang berbeda pada CNI nya, yaitu menggunakan Flannel,Calico dan SONA-CNI. Menggunakan beberapa parameter performansi jaringan seperti *throughput*, *latency* , *packet loss* dan *cpu usage.* Dengan pengambilan data pada setiap skenario akan dilakukan sebanyak 30 kali. Skenario pertama adalah antar pod satu *host* ,pengujian dilakukan dengan melakukan pengiriman paket antar Pod yang berada dalam satu *host* seperti ditunjukkan pada gambar 3.3. Skenario kedua adalah antar pod berbeda *host*, pengujian dilakukan dengan melakukan pengiriman paket antar Pod yang berbeda *host* seperti ditunjukkan pada gambar 3.4.





Gambar 3.3 Pengujian Antar Pod Satu Host **Gambar 3.4** Pengujian Antar Pod Berbeda Host

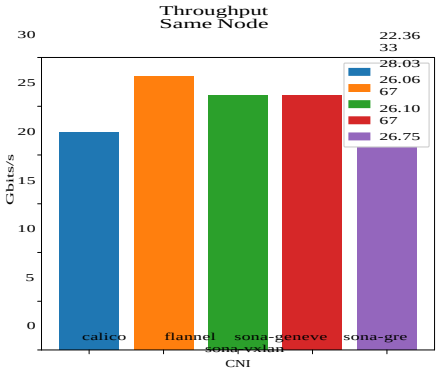
Skenario ketiga adalah pod dengan eksternal *host*, Pengujian dilakukan dengan melakukan pengiriman paket dari Pod menuju jaringan di luar kubernetes *cluster* seperti ditunjukkan pada gambar 3.5.



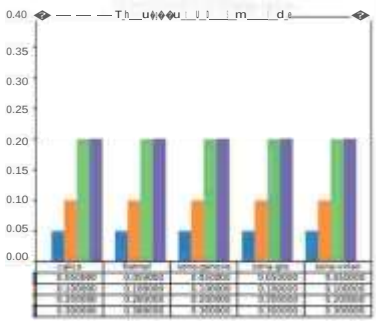
Gambar 3.5 Pengujian Eksternal *Host*

3.4.1. Pengujian *Throughput*

Pengujian ini bertujuan untuk mengetahui berapa besar paket yang dapat dikirimkan antar pod, atau pod dengan host, sesuai dengan skenario yang dilakukan. Pengujian ini dilakukan menggunakan iperf3 pada mode TCP dan UDP. Pada UDP *bandwidth* yang digunakan adalah 50 Mbits, 100 Mbits, 200 Mbits, dan 300 Mbits.



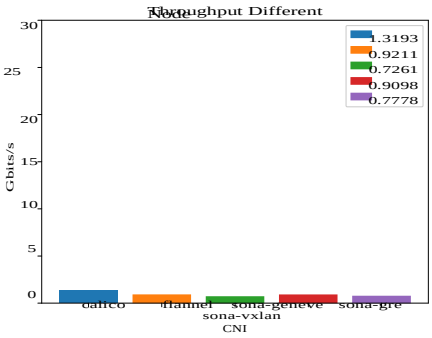
Gambar 3.6 Pengujian Antar Pod Satu Host



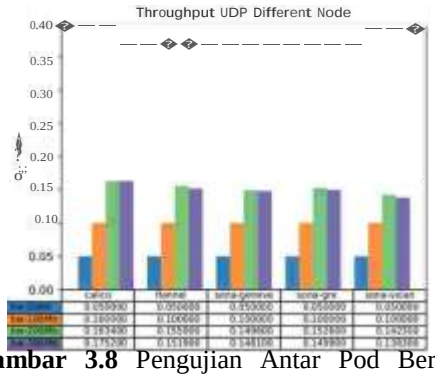
Gambar 3.7 Pengujian Antar Pod Satu Host Protokol UDP

Dari Hasil Pengujian throughput Antar Pod Satu Host pada gambar 3.6. Nilai rata-rata *throughput* setiap CNI mencapai sekitar 20 Gbits/sec karena transfer paket melalui *virtual bridge* yang dibatasi oleh CPU. Nilai *throughput* Flannel dan SONA-CNI lebih tinggi dibandingkan Calico karena pada Flannel dan SONA-CNI menggunakan *virtual bridge* sedangkan Calico menggunakan *iptables.*

Dari hasil pengujian menggunakan protokol UDP pada gambar 3.7, *throughput* pada setiap CNI stabil dan terus meningkat sesuai dengan *bandwidth* yang diberikan.



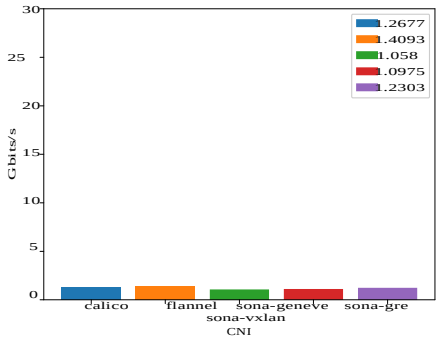
Gambar 3.7 Pengujian Antar Pod Berbeda Host



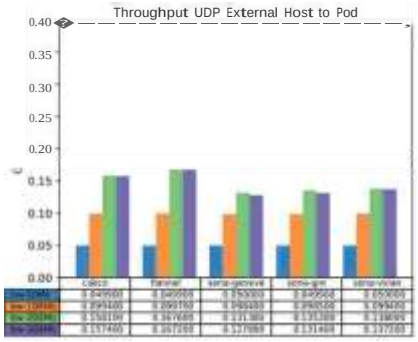
Gambar 3.8 Pengujian Antar Pod Berbeda Host Protokol UDP

Dari hasil pengujian Antar Pod Berbeda Host pada gambar 3.7. Performansi yang berbeda disebabkan besar total *header* pada setiap protokol *tunneling* bergantung dengan MTU *interface* yang digunakan, pada pengujian ini MTU yang digunakan 1500 bytes. Semakin besar total *header* maka semakin besar fragmentasi pada paket yang terjadi, sehingga menyebabkan nilai *throughput* semakin kecil. Maka nilai *throughput* paling tinggi adalah Calico yang menggunakan protokol IP-IP untuk *tunneling* dengan besar total *header* 20 bytes. Kemudian yang kedua adalah Flannel menggunakan protokol *tunneling* VxLAN. Flannel dapat menyesuaikan secara otomatis MTU yang digunakan, sehingga nilai *throughput* lebih besar dibandingkan SONA-CNI.

Dari hasil pengujian menggunakan protokol UDP pada gambar 3.8 *throghput* disetiap CNI meningkat sesuai dengan *bandwidth* yang diberikan. Performansi nilai *throughput* pada protokol UDP sama dengan yang menggunakan protokol TCP dengan nilai *throughput* paling tinggi adalah calico, kemudian flannel, lalu SONA-CNI.



Gambar 3.9 Pengujian *Throughput* dari *External Host* ke Pod



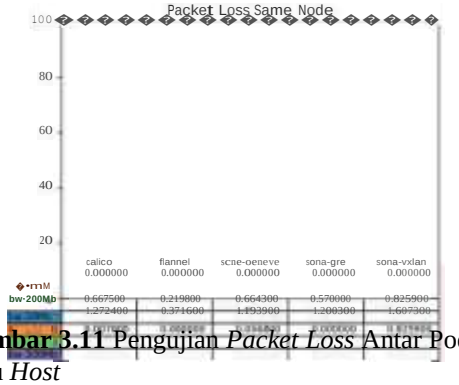
Gambar 3.10 Pengujian *Throughput* dari *External Host* ke Pod Protokol UDP

Dari hasil pengujian Pod Dengan External Host pada gambar 3.9. Performansi yang berbeda disebabkan pada Calico dan Flannel menggunakan kube-proxy yang menggunakan iptables untuk merutekan paket, sedangkan SONA-CNI menggunakan OVS untuk merutekan paket. Sehingga nilai *throughput* pada Calico dan Flannel lebih tinggi dibandingkan SONA-CNI.

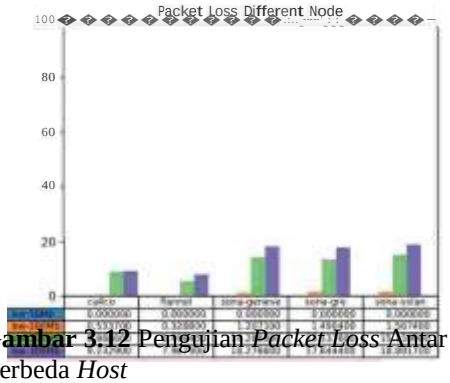
Throughput meningkat sesuai dengan *bandwidth* yang diberikan, namun pada *bandwidth* 200Mb dan 300Mb nilai *throughput* hampir sama, ini menunjukan maksimum *bandwidth* untuk pengaksesan eksternal *host* ada di 200Mb. Hasil pada protokol udp mirip dengan protokol tcp dengan nilai throughput paling tinggi adalah, flannel, calico lalu SONA-CNI.

3.4.2. Pengujian *Packet Loss*

Pengujian ini dilakukan untuk mengetahui seberapa besar paket yang hilang ketika dikirimkan dengan menggunakan iperf3 pada mode UDP dengan menggunakan *bandwidth* yang berbeda-beda yaitu 50 Mbits, 100 Mbits, 200 Mbits, dan 300 Mbits.



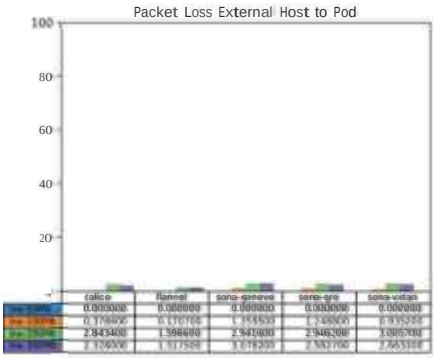
Gambar 3.11 Pengujian *Packet Loss* Antar Pod Satu *Host*



Gambar 3.12 Pengujian *Packet Loss* Antar Pod Berbeda *Host*

Hasil dari pengujian *packet loss* antar pod satu Host pada gambar 3.11 menunjukan semakin tinggi *bandwith* akan semakin tinggi paket yang hilang karena semakin banyak juga paket yang dikirimkan. CNI Flannel mendapat persentase yang paling kecil dibandingkan Calico dan SONA-CNI. Namun secara keseluruhan *packet loss* setiap CNI sangat kecil.

Hasil dari pengujian *packet loss* antar pod berbeda *host* pada gambar 3.12 menunjukan semakin tinggi *bandwith* akan semakin tinggi paket yang hilang karena semakin banyak juga paket yang dikirimkan. Flannel mendapatkan nilai persentase *packet loss* yang paling rendah dibandingkan Calico dan SONA-CNI. Pada *bandwidth* 200Mb dan 300Mb setiap CNI menunjukan kenaikan *packet loss* yang tinggi sebesar 808,33% - 1584,2%, sehingga untuk *bandwidth* 200Mb keatas performa *packet loss* setiap CNI akan menjadi sangat buruk.

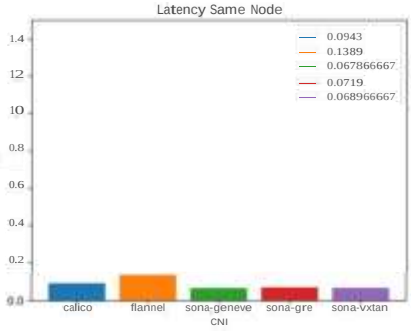


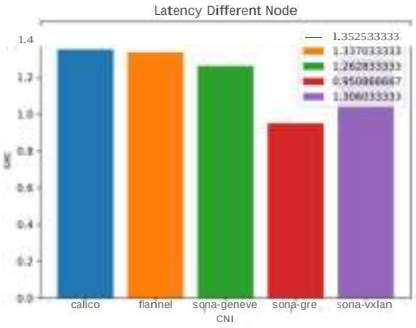
Gambar 3.13 Pengujian *Packet Loss* dari Eksternal *Host* ke Pod

Hasil dari pengujian *packet loss* antar pod berbeda *host* pada gambar 3.13 menujukan Flannel mendapatkan persentase paling rendah. Pada SONA-CNI yang mendapatkan persentase paling rendah adalah protokol *tunneling* GRE.

3.4.3. Pengujian *Latency*

Pengujian ini dilakukan untuk mengetahui *latency* pada saat pengiriman paket menggunakan ping dan sockperf.



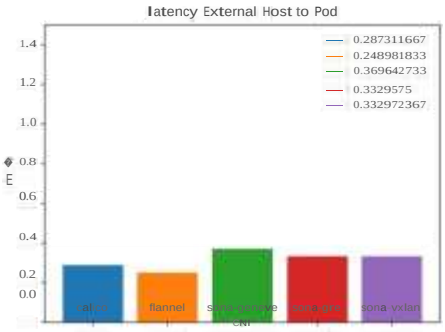


Gambar 3.14 Pengujian *Latency* Antar Pod Satu *Host*

Gambar 3.15 Pengujian *Latency* Antar Pod Berbeda *Host*

Hasil pengujian *latency* antar pod satu host pada gambar 3.14 menunjukan nilai *latency* yang didapatkan pada SONA-CNI lebih rendah dibandingkan menggunakan Flannel dan Calico.

Hasil pengujian *latency* antar pod berbeda *host* pada gambar 3.15 menunjukan SONA-CNI dengan protokol GRE mendapatkan nilai *latency* paling rendah. Serta nilai *latency* pada SONA-CNI secara keseluruhan lebih rendah dibandingkan Flannel dan Calico

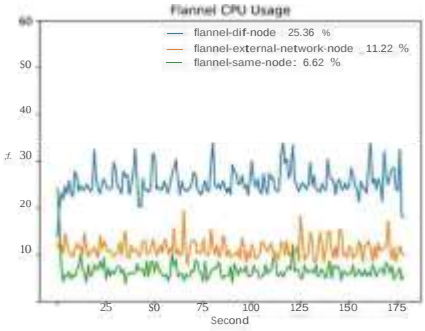


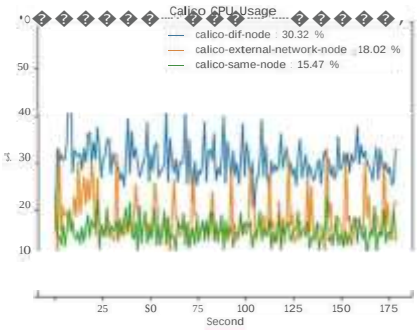
Gambar 3.16 Pengujian *Packet Loss* dari Eksternal *Host* ke Pod

Hasil pengujian *latency* dari *external host* Pod pada gambar 3.16 diuji menggunakan sockperf karena pada SONA-CNI memblok protokol ICMP. Flannel mendapatkan nilai *latency* paling rendah. Pada SONA-CNI nilai *latency* pada setiap protokol tunneling hampir sama, dengan yang paling rendah pada protokol *tunneling* GRE.

3.4.4 CPU *Usage*

Pengujian dilakukan untuk mengetahui CPU *usage* pada setiap CNI ketika dilakukan pengujian. Nilai CPU *usage* merupakan rata-rata CPU *usage* dari setiap *host.*

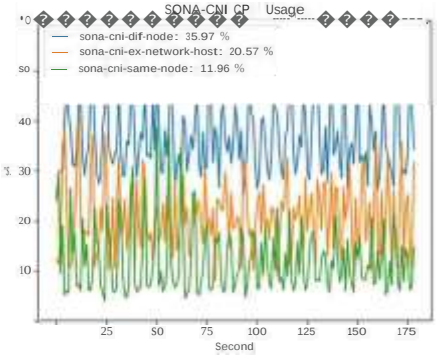




Gambar 3.17 CPU *usage* CNI Flannel

Gambar 3.18 CPU *usage* CNI Calico

Hasil rata-rata persentase CPU *usage* setiap host pada CNI flannel dan calico, menunjukan persentase CPU *usage* tertinggi pada saat pengujian antar pod berbeda *host* dan yang terendah pengujian antar pod satu *host*.



Gambar 3.19 CPU *usage* SONA-CNI

Hasil rata-rata persentase CPU *usage* setiap host pada SONA-CNI, menunjukan persentase CPU *usage* tertinggi pada saat pengujian antar pod berbeda *host* dan yang terendah pengujian antar pod satu *host*

3.4.5 *Monitoring* Trafik Paket

CNI yang dapat memonitoring paket tanpa memerlukan memasang aplikasi tambahan hanya SONA-CNI dengan adanya ONOS controller yang menyediakan GUI dan CLI yang dapat digunakan untuk memonitoring.

Gambar 3.20 *Monitoring* Topologi Kubernetes

Gambar 3.21 *Monitoring* Grafik Diagram Kubernetes

Pada SONA-CNI menyediakan dua tampilan untuk memonitoring jaringan yaitu berupa gambaran topologi jaringan pada gambar 3.17 dan grafik diagram batang yang memperlihat trafik di kubernetes pada gambar 3.18.

4. Kesimpulan dan Saran

4.1. Kesimpulan

Berdasarkan pengujian dari percobaan yang telah dilakukan dapat disimpulkan Penggunaan protokol *tunneling* yang tepat bergantung pada besar MTU pada *network interface* yang digunakan. Jika membutuhkan *throughput* yang cepat dapat memilih CNI Flannel atau Calico. Namun jika membutuhkan *packet loss* yang rendah dapat menggunakan CNI Flannel, tapi jika membutuhkan *latency* yang rendah dapat menggunakan SONA-CNI. Kemudian CNI yang menyediakan memonitoring trafik paket tanpa memasang aplikasi hanya SONA-CNI dengan menggunakan ONOS controller.

4.2 Saran

Saran yang dapat diberikan pada penelitian berikutnya, dapat menggunakan *network interface* dengan MTU yang besar atau biasa disebut *jumbo frame*, dan menyesuaikan MTU CNI dengan MTU yang digunakan pada *network interface*. Pada SONA-CNI karena menggunakan OVS sehingga dapat dicoba bagaimana performansi jaringan jika fitur DPDK diaktifkan.

Daftar Pustaka:

[1] H. Zeng, B. Wang, W. Deng, and W. Zhang, "Measurement and evaluation for docker container networking," Proc. - 2017 Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discov. CyberC 2017, vol. 2018-Janua, pp. 105–108, 2018.

[2] S. J. Vaughan-Nichols, "New approach to virtualization is a lightweight," *Computer (Long. Beach. Calif).*, vol. 39, no. 11, pp. 12–14, 2006.

[3] C. C. Chang, S. R. Yang, E. H. Yeh, P. Lin, and J. Y. Jeng, "A Kubernetes-Based Monitoring Platform for Dynamic Cloud Resource Provisioning," *2017 IEEE Glob. Commun. Conf. GLOBECOM 2017 - Proc.*, vol. 2018-Janua, pp. 1–6, 2018.

[4] K. Suo, Y. Zhao, W. Chen, and J. Rao, "An Analysis and Empirical Study of Container Networks," *Proc. - IEEE INFOCOM*, vol. 2018-April, pp. 189–197, 2018.

[5] Narunas Kapocius. Performance Studies of Kubernetes Network Solutions. 2020 IEEE Open Conference of Electrical, Electronic and Information Sci- ences, eStream 2020 - Proceedings, 2020.

[6] O. N. F. 2012, "Software-Defined Networking: The New Norm for Networks [white paper]," *ONF White Pap.*, pp. 1–12, 2012.

[7] D. Bernstein, "Containers and cloud: From LXC to docker to kubernetes," *IEEE Cloud Comput.*, vol. 1, no. 3, pp. 81–84, 2014.

[8] L. Abdollahi Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, "Deploying Microservice Based Applications with Kubernetes: Experiments and Lessons Learned," *IEEE Int. Conf. Cloud Comput. CLOUD*, vol. 2018-July, pp. 970–973, 2018.

[9] A. Zhao, Y. Yuan, Y. Ji, and G. Gu, "Research on tunneling techniques in virtual private networks," *Int. Conf. Commun. Technol. Proceedings, ICCT*, vol. 1, pp. 691–697, 2000.