Analisis Penggunaan Metode *Adaptive 3 in 1 Message exchange* pada Arsitektur *Active-Active Distributed Controller* di Jaringan *Software Defined Networks*

1st Nadia Vebbi Febrianti
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
nadiavebbif@students.telkomuniversity.ac.id

2nd Vera Suryani
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
verasuryani@telkomuniversity.ac.id

3rd Muhammad Arief Nugroho
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
arif.nugroho@telkomuniversity.ac.id

Abstrak

Penggunaan arsitektur *Distributed Controller (active-active)* dengan menggunakan *asynchronous message exchange* dengan metode *3 in 1 Heartbeat* yaitu mengirimkan tiga pesan dan dibalas satu *acknowledgment* dengan menggunakan dua *controller* dimana *controller* A bertugas mengirimkan pesan dan *controller* B bertugas untuk membalas pesan. Dengan begitu, kedua *controller* tidak dapat saling bertukar peran dan memperoleh waktu *failover (migration time)* yang lama. Oleh karena itu, dalam penelitian ini dilakukan pengembangan untuk mengurangi waktu *failover (migration time)* dan mengurangi beban sumber daya pesan dalam melakukan pengiriman informasi pesan antara *controller*. Penelitian ini mengusulkan untuk menggunakan metode *adaptive 3in 1 message exchange*. Arsitektur tersebut memungkinkan semua *controller* bersifat *master-master* dan aktif bekerja sama untuk mengelola suatu jaringan secara bersamaan. Hasil pengujian yang dilakukan dengan metode *adaptive 3 in 1* memperoleh waktu *failover* lebih rendah dengan selisih satu detik dan nilai CPU *usage* dengan selisih sebesar 1.50% lebih rendah dibandingkan dengan metode *3 in 1 heartbeat.* Begitupula dengan pengujian nilai *timeout* tiga detik, metode *adaptive 3 in 1* memperoleh selisih CPU *Usage* sebesar 2.15% lebih rendah dibandingkan dengan metode *3 in 1 heartbeat.* Hal tersebut terjadi karena berkurangnya sumber daya pesan yang dilakukan ketika melakukan pengiriman pesan sehingga pada proses waktu*failover (migration time)* akan berjalan lebih cepat. Oleh karena itu, dengan menggunakan metode *adaptive 3 in 1* beban kerja pada masing-masing *controller* berkurang karena kedua *controller* bersifat *main to main.*

Kata kunci : asynchronous, distributed controller, message exchange

Abstract

The use of Distributed Controller architecture (active-active) using asynchronous message exchange with the 3 in1 Heartbeat method, which is to send three messages and receive an acknowledgment by using two controllers where controller A is in charge of sending messages and controller B is in charge of replying messages. That way, the two controllers can't swap roles and get a long failover (migration time). Therefore, in this research, development is carried out to reduce the failover time (migration time) and reduce the message resource load in sending message information between controllers. This study proposes to use the adaptive 3 in 1 message exchange method. This architecture allows all controllers to be master-master and actively work together to manage a network simultaneously. The results of the tests carried out with the adaptive 3 in 1 method obtained a lower failover time with a difference of one second and the CPU usage value by a difference of 1.50% lower than the 3 in 1 heartbeat method. Likewise, by testing the timeout value of three seconds, the adaptive 3 in 1 method obtains a CPU Usage difference of 2.15% lower than the 3 in 1 heartbeat method. This happens because of the reduced message resources that are carried out when sending messages so that the failover time (migration time) process will run faster. Therefore, by using the adaptive 3 in 1 method, the workload on each controller is reduced because both controllers are main to main.

Keywords: asynchronous, distributed controller, message exchange, software defined network

I. PENDAHULUAN

A. Latar Belakang

Software-Defined Network (SDN) memungkinkan realistis dan pengontrolan manajemen *control plane* (*controller*) dan *data plane* untuk mendukung jaringan berskala besar. SDN merupakan paradigma jaringan baru yang memungkinkan fleksibel manajemen untuk jaringan [1]. Ketika ukuran jaringan meningkat, pengontrol terpusat tunggal tidak dapat memenuhi peningkatan permintaan untuk pemrosesan aliran. Jadi, solusi yang menjanjikan untuk SDN dengan jaringan skala besar adalah *multiple controller* [1]. *Multiple controller* merupakan bagian dari *Distributed Controller* yang digunakan untuk meningkatkan skalabilitas dalam jaringan. Skalabilitas merupakan salah satu hal penting dalam suatu sistem dan jaringan untuk menangani jumlah peningkatan kinerja yang berpotensi besar. Pada *Distributed Controller* terdapat dua arsitektur dasar yaitu *flat architecture* dan *hierarchical architecture*. Dimana *flat architecture* membagi area jaringan dan *hierarchical architecture* mengelola setiap area [2]. *Distributed Controller* (*active-active*) dengan menggunakan *flat architecture* membuat semua *controller* aktif dan bekerja sama untuk mengelola suatu jaringan secara bersamaan serta dapat memperluas kapabilitas dari *control plane*.

Pada penelitian sebelumnya dilakukan penelitian *Distributed Controller (active-active)* yang menggunakan *asynchronous message exchange* [3]. Pada penelitian tersebut memiliki kekurangan yaitu jika *controller* A mengalami *down* maka semua jaringan perangkat yang dikelola oleh *controller* A akan dimigrasikan ke *controller* B dan menunggu *controller* A hidup kembali untuk melakukan pengiriman pesan. Jika *controller* A kembali pulihmaka *controller* A yang tetap melakukan pengiriman pesan dan *controller* B sebagai penerima pesan sehingga kedua *controller* tidak dapat saling bertukar peran dan menghasilkan waktu *failover (migration time)* yang lama. Penelitian ini merupakan pengembangan dari penelitian sebelumnya [3], dimana pada penelitian kali ini akan menggunakan metode *adaptive 3 in 1 message exchange* dengan menjadikan *controller* B sebagai *controller* utama, sehingga jika *controller* A terjadi *down* kemudian *controller* tersebut hidup kembali, maka *controller* B akan menjadi *controller* utama yang bertugas untuk mengirimkan tiga pesan dan *controller* A yang akan membalaspesan. Dengan metode tersebut kedua *controller* akan bersifat *master-master* yang akan saling menggantikan peran jika salah satu dari *controller* mengalami *down* dan kemudian hidup kembali.

B. Topik dan Batasannya

Dalam penelitian sebelumnya dengan menggunakan *asynchronous message* sebagai *message exchange* pada *Distributed Controller (active-active).* Proses ini dilakukan menggunakan metode *3 in 1* yang bekerja dengan cara*controller* utama (*controller* A) mengirimkan tiga pesan kemudian akan direspon oleh *receiver* (*controller* B) dengan satu *acknowledgment*. Jika *controller* utama mengalami *down* maka proses pengiriman pesan akan terhentidan menunggu *controller* utama hidup kembali agar proses kembali berjalan. Hal ini mengakibatkan *controller* memiliki peran masing-masing serta tidak dapat bertukar peran dan menghasilkan waktu *failover (migration time)* yang lama. Oleh karena itu, penelitian ini mengusulkan untuk menggunakan metode *adaptive 3 in 1* untuk mengurangi waktu *failover (migration time)* dan meningkatkan mekanisme *message exchange* serta mengurangi beban sumber daya pesan dalam melakukan pengiriman informasi pesan antara *controller*. Metode ini dilakukan dengan cara mengirimkan tiga pesan kemudian di balas dengan satu *acknowledgment* dengan kedua *controller* (*controller* A dan *controller* B) bersifat *master-master*. Arsitektur tersebut memungkinkan semua *controller* aktifdan bekerja sama untuk mengelola suatu jaringan secara bersamaan sehingga pengiriman pesan tidak hanya dilakukan oleh satu *controller* serta dapat bertukar peran ketika salah satu *controller* mengalami *down*.

Penelitian ini melakukan pengujian metode *adaptive 3 in 1* dengan menggunakan parameter CPU *Usage* untuk menganalisa beban kerja pada kedua *controller* serta menggunakan parameter dengan mekanisme *failover time* untuk melihat waktu *failover* yang terjadi ketika melakukan *switch migration* terhadap setiap *controller*. *Controller* yang digunakan pada penelitian ini adalah *POX controller.* Kemudian dalam melakukan simulasi menggunakan mininet dan *virtualbox* yang digunakan sebagai *controller* A dan *controller* B.

C. Tujuan

Tujuan dilakukannya penelitian ini adalah untuk menganalisis waktu *failover (migration time)* kinerja *workload* terhadap *controller* dengan menggunakan metode *adaptive 3 in 1 message exchange* pada *distributed controller* (*active- active*). Parameter yang digunakan dalam penelitian ini meliputi nilai CPU *usage* dan *failover time* untuk menganalisa beban kerja terhadap *controller* serta waktu *failover* yang terjadi.

D. Organisasi Tulisan

Bagian selanjutnya terdiri dari studi terkait, sistem yang di bangun, evaluasi, dan kesimpulan. Pada bagian studi terkait dijelaskan mengenai studi literatur dan teori yang menjadi dasar pelaksanaan dan pengembangan penelitian ini. Bagian sistem yang dibangun akan lebih lanjut dijelaskan mengenai rancangan dan implementasi. Dari implementasi tersebut akan dilaksanakan beberapa skenario untuk pengujian terhadap sistem dan bagian kesimpulan akan dijelaskan hasil dan analisa berdasarkan uraian pada bagian evaluasi.

II. KAJIAN TEORI

A. Penelitian Terkait

Pada jurnal atau paper sebelumnya telah dibahas mengenai penelitian terkait dengan penelitian yang akan dilakukan diantaranya seperti pada tabel 1.

Tabel 1. *Related Works*

Judul paper	Tahun	Metode	Isi
An Overview on SDN Architecture swith Multiple *Controller*s	2016	Flat Architecture	Flat Architecture memiliki ketahanan yang kuat terhadap kegagalan dalam jaringan karena memiliki layer dengantanggung jawab yang berbeda [2].
Desain Distributed Controller dengan Metode Active-Active pada Jaringan Software Define Network	*2019*	*Active - active*	Dengan metode active-active controller akan aktif bekerja sama untuk mengelola jaringan secara bersamaan dan dapat menjamin konsistensi suatu jaringan [5].
Analisis Penggunaan Metode 3 in 1 Heartbeat Pada Arsitektur Active-Active Distributed Controller di Jaringan Software Defined Networks	*2020*	*Asynchronous message exchange*	Metode 3 in 1 lebih unggul dibandingkan metode satu pesan satu acknowledgment. Metode 3 in 1 memiliki peningkatan kinerja serta beban kerja pada controller yang lebih rendah. Selain itu, proses failover menghasilkan waktu yang lebih cepat [3].

B. Software Defined Network

Software-Defined Network (SDN) merupakan suatu paradigma arsitektur baru dalam bidang jaringankomputer yang diprogram serta dipisahkan antara sistem kontrol (*control plane*) serta sistem forwarding (*data plane*) pada fitur jaringan. *Control Plane* merupakan komponen pada jaringan yang berperan buat mengendalikan jaringan, ialah konfigurasi sistem, manajemen jaringan, memastikan data *routing table* serta *forwarding table*. *Data Plane* merupakan bagian yang bertanggung jawab meneruskan paket, tidak hanya itu *data plane* juga bertanggung jawab untuk menguraikan header paket, mengendalikan QoS, serta enkapsulasi paket. [4].

C. Distributed Controller

Distributed Controller adalah *multiple-controller* pada suatu jaringan *Software Defined Network* (SDN). Pada *controller* jaringan SDN terdapat dua atau lebih *controller* yang masing-masing *controller* memiliki tanggung jawab untuk menyampaikan pesan pada *controller* lainnya. *Multiple-controller* digunakan untuk mengatasi masalah dalam peningkatan skalabilitas pada jaringan [6].

D. Mekanisme *Failover*

Mekanisme *failover* merupakan metode jaringan dengan membagi dua jalur koneksi ataupun lebih, dimana jika salah satu jalur mati, maka koneksi masih berjalan dengan mengalihkannya ke jalur yang lain. Mekanisme *failover* dapat dirancang sehingga dapat segera bertindak setelah terjadi kendala atau gangguan [8].

E. Reliabilitas

Reliabilitas adalah kemampuan jaringan yang dapat diandalkan. Reliabilitas dapat memberikan layanan secara lancar tanpa gangguan. Dapat dikatakan bahwa reliabilitas memiliki ketersediaan layanan jaringan yang tinggi [6]. Pada reliabilitas terdapat suatu teknik jaringan yang disebut mekanisme *failover* [7].

F. Skalabilitas

Skalabilitas adalah kemampuan suatu sistem, jaringan atau proses yang menangani jumlah penambahan beban atau peningkatan kinerja. Skalabilitas merupakan hal yang harus diperhatikan karena memiliki peran yang penting. Skalabilitas pada SDN yang mengalami kelebihan beban pada *control plane* dan *data plane* dapat dikurangi dengan CPU, RAM, dan fungsi file *I/O* [9].

G. Asynchronous Message Exchange

Serangkaian teknik yang memungkinkan objek untuk berkomunikasi dalam mengirim dan menerima pesan melalui jaringan atau yang disebut sistem pertukaran data (*message exchange*) terbagi menjadi dua model komunikasi yaitu, *asynchronous message* dan *synchronous message.* Pada *asynchronous message*, beberapa pesan dapat dikirimkan secara langsung kemudian dieksekusi oleh *receiver*. Pendekatan ini memiliki keuntungan yang banyak dibandingkan model *synchronous message.* Dengan menunjukkan performansi yang tinggi dalam komunikasi *controller-to-controller* menyebabkan *asynchronous message* (*nonblocking i/o*) lebih unggul dibandingkan dengan *synchronous message* (*blocking i/o*) [10].

H. Metode *Adaptive 3 in 1*

Metode *Adaptive 3 in 1* merupakan metode yang akan digunakan pada penelitian ini dengan menggunakan arsitektur *Distributed Controller active-active.* Pada penelitian ini, masing-masing *controller* dapat menyesuaikandiri dengan keadaan *controller* lainnya atau disebut *adaptive.* Setelah masing-masing *controller* sudah saling mendeteksi bahwa status *controller* sudah saling aktif, selanjutnya proses pertukaran pesan akan dilakukan. Proses pertukaran pesan antar *controller* menggunakan metode *3 in 1* yang bekerja dengan cara *controller* A sebagai *sender* mengirimkan tiga pesan dan *controller* B sebagai *receiver* yang akan merespon dengan satu *acknowledgment.* Begitu pula sebaliknya, jika salah satu *controller* mengalami *down*, maka *controller* yang lainnya akan menggantikan peran dari *controller* yang mengalami *down*.

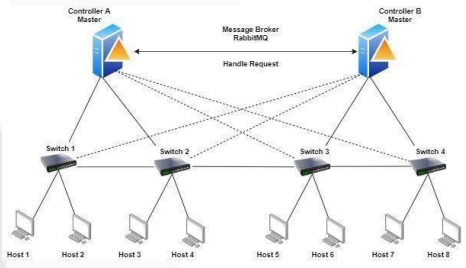
III. METODE

A. Arsitektur Sistem

Perancangan sistem pada penelitian ini merupakan pengembangan dari penelitian sebelumnya yaitu menggunakan rancangan topologi *Distributed Controller* (active-active) [3]. Pada penelitian ini menggunakan *POX controller* sebagai *controller* dan model komunikasi *message exchange* dengan metode *adaptive 3 in 1*.

___: Main link

------ : Backup link



Gambar 1. Topologi *Distributed Controller (active-active)*

Pada gambar 1 terdapat dua buah *controller* yaitu *controller* A dan *controller* B yang saling aktif dan bersifat *master-master* untuk bekerja sama. Masing-masing *controller* mengelola dua buah *switch*. Dimana *controller* A mengelola *switch* 1 dan *switch* 2 serta terhubung pada *controller* B dengan *backup link* yang aktif. Sama hal nya dengan *controller* B mengelola *switch* 3 dan *switch* 4 serta terhubung pada *controller* A. Ketika *controller* A *down*, *switch* akan berpindah ke *controller* B dan begitu pula sebaliknya. Ketika *controller* A *down* dan hidup kembali, maka *controller* B yang akan menggantikan peran dari

controller A untuk mengirimkan pesan dan *controller* A yang akan membalas pesan.

Tabel 2. *IP Address*

Device	IP Address
Controller A	192.168.1.10
Controller B	192.168.1.14
Mininet	192.168.1.11
Host 1	192.168.25.1
Host 2	192.168.25.2
Host 3	192.168.25.3
Host 4	192.168.25.4
Host 5	192.168.25.5
Host 6	192.168.25.6
Host 7	192.168.25.7
Host 8	192.168.25.8

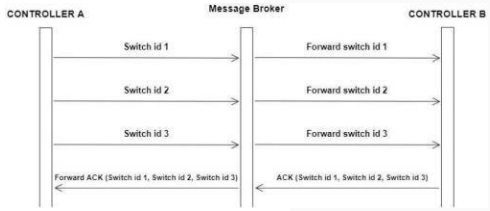
IP Address pada tabel 2 merepresentasikan topologi pada gambar 1. Tabel tersebut juga menjelaskan *IP Address* pada *controller* dan *host* yang digunakan dalam penelitian ini.

Tabel 3 Jumlah *Node*

Jumlah *Switch*	4	8	16	32	64
Jumlah *Host*	8	16	32	64	128

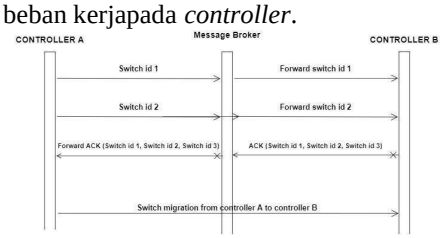
Jumlah *node* yang berisi jumlah *switch* dan jumlah *host* pada tabel 3 akan digunakan untuk pengujian pada penelitian ini.

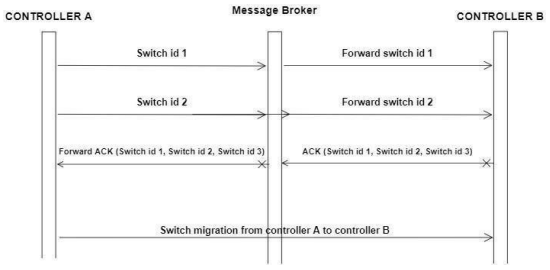
B. Arsitektur *Message Exchange*



Gambar 2. Kondisi awal arsitektur *Message exchange* tanpa masalah

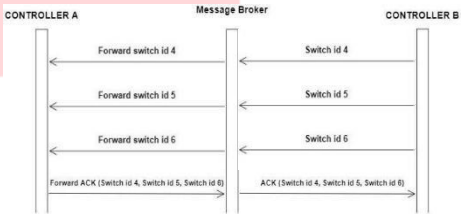
Gambar 2 merupakan *message exchange* antar *controller* menggunakan *message broker (RabbitMQ)*. Pada gambar tersebut *controller* A mengirim tiga pesan melalui *message broker*. Kemudian *message broker* meneruskanpesan tersebut pada *controller* B. Begitu pula *controller* B menyampaikan pesan melalui *message broker*. Kemudian *message broker* meneruskan pesan tersebut dengan satu *acknowledgment* (ACK) yang berisikan balasan tiga pesan kepada *controller* A. Pada pengujian ini menggunakan parameter CPU *usage* untuk melihat beban kerjapada *controller*.





Gambar 3. Arsitektur *Message exchange* ketika terjadi masalah *(Controller A down)*

Gambar 3 merupakan *message exchange* yang mengalami masalah atau *controller A* mengalami *down*. Hal ini disebabkan karena dalam waktu satu detik, pesan yang dikirimkan kurang dari tiga pesan. Oleh karena itu dilakukan proses *switch migration* dari *controller* A ke *controller* B sehingga *controller* B menjadi *controller* utama. Untuk skenario ini akan dilakukan pengujian dengan menggunakan mekanisme *failover*.



Gambar 4. Arsitektur *Message exchange* dengan *controller B* sebagai *controller* utama

Gambar 4 merupakan proses *message exchange* dengan *controller* B menjadi *controller* utama. Pada gambartersebut menunjukkan *controller* B yang menjadi *controller* utama setelah terjadi masalah pada *controller* A yangmengalami *down* kemudian hidup kembali. Dengan begitu pada gambar tersebut *controller* B yang selanjutnya akan mengirimkan pesan melalui *message broker*, kemudian *message broker* akan meneruskan pesan tersebut kepada *controller* A dan setelah *controller* A menerima pesan, *controller* A akan mengirimkan balasan pesan melalui *message broker*. Selanjutnya *message broker* akan meneruskan pesan yang berisi tiga pesan dalam satu *acknowledgment* (ACK).

C. Spesifikasi Kebutuhan

Untuk pengujian sistem yang dibangun, kebutuhan penelitian dideskripsikan pada tabel 4 dan tabel 5, simulasi pada penelitian ini menggunakan mininet serta *virtualbox* dan menggunakan *POX controller* sebagai *controller*.

Tabel 4. Spesifikasi Kebutuhan Perangkat Lunak

No	Perangkat Lunak	Detail
1.	Sistem Operasi	Ubuntu 18.04
2.	Tools	Mininet
		Pox *Controller*
		RabbitMQ

Tabel 5. Spesifikasi Kebutuhan Perangkat Keras

No	Perangkat Keras	Keterangan
1	CPU	Intel Core i7-7500U 2.70GHz

2	RAM	8.00 GB
3	GPU	NVIDIA GeForce 940MX
4	Storage	HDD Toshiba MQ01ABD100M

D. Skenario Pengujian

Pada penelitian ini dilakukan skenario pengujian sebagai berikut:

1. Pengujian *Message exchange*

Pengujian proses *message exchange* pada penelitian ini akan dilakukan dengan tiga skenario. Skenario pertama akan dilakukan dengan menjalankan sistem sebelum terjadi masalah terhadap *controller*, skenario kedua dilakukan dengan mematikan *controller* A sehingga *controller* A mengalami *down*, dan skenario ketiga dilakukan dengan status *controller* A hidup kembali, kemudian *controller* B yang akan menjadi *controller* utama dan melakukan pengiriman pesan. Proses *message exchange* pada penelitian inidilakukan dengan cara *controller* A mengirimkan tiga pesan kemudian di balas oleh *controller* B dengan satu *acknowledgmen*t. Proses pengiriman pesan dilakukan maksimal selama satu detik, sehingga waktu yang dibutuhkan untuk mengirimkan pesan kurang dari satu detik. Jika waktu pengiriman pesan lebih dari satu detik dan pengiriman pesan yang dilakukan kurang dari tiga pesan, maka *controller* dinyatakan *down* dan dilakukan *switch migration* dari *controller* A ke *controller* B. Jika *controller* A hidup kembali, maka*controller* B yang akan melakukan pengiriman pesan. *Controller* akan bekerja sama dengan sifat *master- master*. Waktu satu detik diambil sebagai batasan maksimum waktu pengiriman pesan karena waktu satudetik merupakan angka yang cukup optimal sehingga proses pengiriman pesan tidak melebihi waktu yangdapat menyebabkan proses pertukaran pesan menjadi lebih lambat.

2. Pengujian *Failover/Migration Time*

Pengujian ini dilakukan untuk menganalisa kinerja *workload* terhadap *controller* dengan menggunakan mekanisme *failover time* sebagai parameter untuk melihat waktu *failover* yang terjadi ketika melakukan *switch migration* terhadap *controller*.

3. Pengujian parameter CPU Usage

Pengujian ini bertujuan untuk merepresentasikan skalabilitas jaringan yang berpengaruh terhadap beban kerja pada *controller.*

IV. HASIL DAN PEMBAHASAN

Bab ini menjelaskan hasil pengujian simulasi yang dilakukan. Pengujian simulasi dilakukan berdasarkan tiga skenario, yaitu skenario sebelum *controller down*, skenario ketika *controller down*, dan skenario ketika *controller* lainnya menjadi *controller* utama yang berperan sebagai pengirim pesan. Parameter yang dilakukan adalah CPU *Usage* dan *failover time.*

A. Pengujian *Message Exchange*

Pengujian proses *message exchange* akan dilakukan dengan tiga skenario. Skenario pertama akan dilakukan dengan menjalankan sistem sebelum terjadi masalah terhadap *controller*, skenario kedua dilakukan dengan mematikan *controller* A sehingga *controller* A mengalami *down*, dan skenario ketiga dilakukan dengan status *controller* A hidup kembali, kemudian *controller* B yang akan menjadi *controller* utama dan melakukan pengiriman pesan.

1. Pengujian Skenario Sebelum Terjadi *Down*

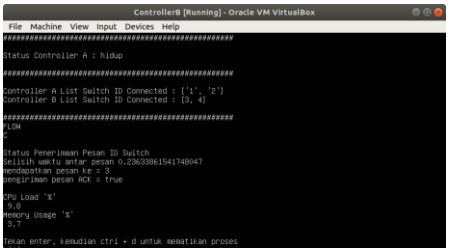
Pengujian pada skenario pertama sebelum *controller* terjadi *down* dilakukan pengujian fungsionalitas proses mekanisme *message exchange* pada *controller* A dan *controller* B. Kemudian, pada pengujian selanjutnya meliputi nilai skalabilitas dengan CPU *Usage.* Pengujian ini dilakukan dengan metode *adaptive 3 in 1* dan metode*3 in 1 heartbeat* dan kedua metode tersebut menggunakan *distributed controller (active-active).*

Gambar 5. Proses *message exchange*

Gambar 5 merupakan proses *message exchange* sebelum terjadi *down* pada *controller* A "msgController1". Pengiriman pesan dilakukan dari *controller* A sebanyak tiga pesan kemudian akan dibalas oleh *controller* B dengan satu *acknowledgment* dari gambar dapat dilihat dari "ACK from Contoller B". Proses pengiriman pesan dilakukan selama satu detik, jika melebihi waktu tersebut maka akan terjadi *down* dan akan terjadi *switch migration* terhadap *controller* B.

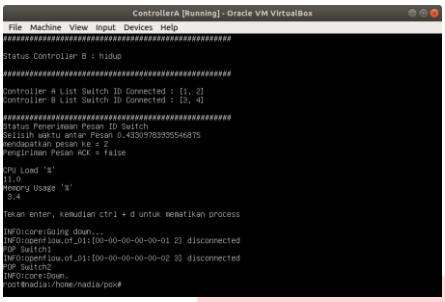
2. Pengujian Skenario Ketika Terjadi *Down*

Pengujian skenario kedua yaitu ketika terjadinya *down* pada salah satu *controller.* Penelitian ini dilakukan menggunakan metode *adaptive 3 in 1 message exchange* dan menggunakan *distributed controller(active-active)* sehingga kedua *controller* akan saling aktif bekerja sama dan bersifat *master-master* yang dapat saling bertukarperan jika salah satu *controller down* kemudian hidup kembali. Pengujian pada skenario ini dilakukan dengan cara mematikan *controller* A. Ketika *controller* A *down* maka *RabbitMQ* yang berperan sebagai *message broker*akan menerima pesan bahwa *controller* A *down* dan melakukan *forward* kepada *controller* B, kemudian akan dilakukan *switch migration* dari *controller* A ke *controller* B sehingga *switch* yang terhubung pada *controller* Aakan berpindah ke *controller* B.



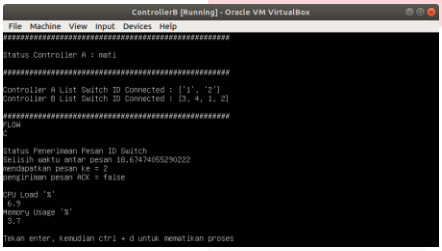
Gambar 6. Kondisi *controller* B sebelum *controller* A terjadi *down*

Gambar 6 merupakan kondisi awal *controller* B sebelum *controller* A terjadi *down* dengan *switch* 1 dan 2 masih terhubung pada *controller* A dan *controller* B mendeteksi status *controller* hidup.



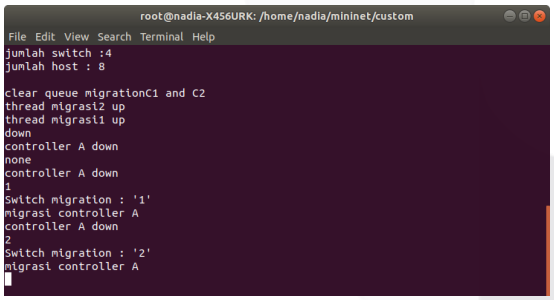
Gambar 7. Kondisi *controller* A *down*

Gambar 7 merupakan kondisi *controller* A ketika terjadi *down* dengan *switch* 1 dan 2 sudah tidak terhubungdengan *controller* A.



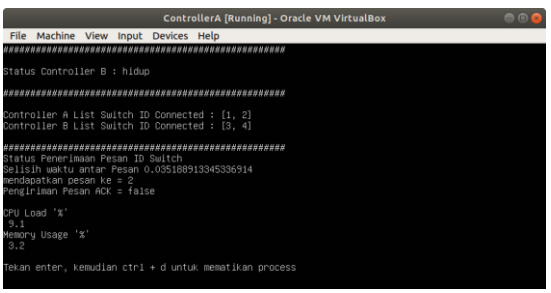
Gambar 8. Kondisi *controller* B ketika *controller* A *down*

Gambar 8 merupakan kondisi *controller* B yang mendeteksi bahwa telah terjadi *down* pada *controller* A.Kemudian terjadi *switch migration* dari *controller* A ke *controller* B.



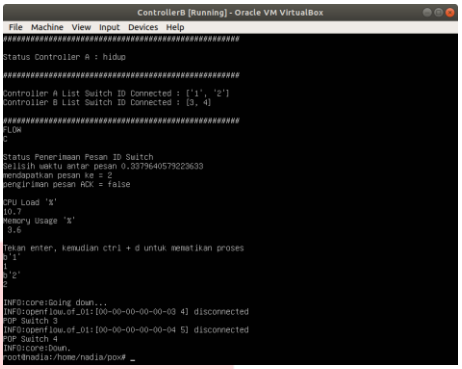
Gambar 9. Kondisi mininet ketika *controller* A terjadi *down*

Gambar 9 merupakan kondisi pada mininet ketika *controller* A terjadi *down* dan melakukan *switch migration* dari *controller* A ke *controller* B.



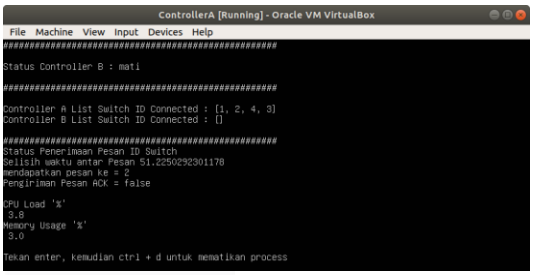
Gambar 10. Kondisi *controller* A sebelum *controller* B terjadi *down*

Gambar 10 merupakan kondisi awal *controller* A sebelum *controller* B terjadi *down* dengan *switch* 3 dan 4masih terhubung pada *controller* B dan *controller* A mendeteksi status *controller* B hidup.



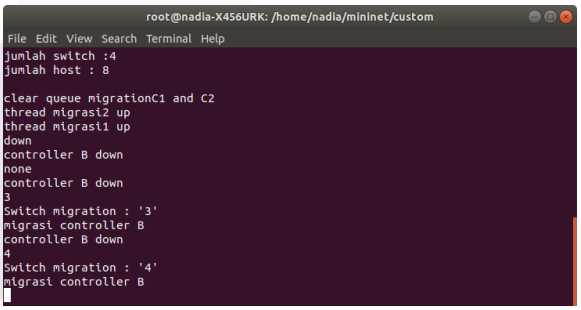
Gambar 11. Kondisi *controller* B *down*

Gambar 11 merupakan kondisi *controller* B ketika terjadi *down* dengan *switch* 3 dan 4 sudah tidak terhubungdengan *controller* B.



Gambar 12. Kondisi *controller* A ketika *controller* B terjadi *down*

Gambar 12. merupakan kondisi *controller* A yang mendeteksi bahwa telah terjadi *down* pada *controller* B. Kemudian terjadi *switch migration* dari *controller* B ke *controller* A.



Gambar 13. Kondisi mininet ketika *controller B* terjadi *down*

Gambar 13 merupakan kondisi pada mininet ketika *controller* B terjadi *down* dan melakukan *switch migration* dari *controller* B ke *controller* A.

3. Skenario ketika Controller B menjadi Controller Utama

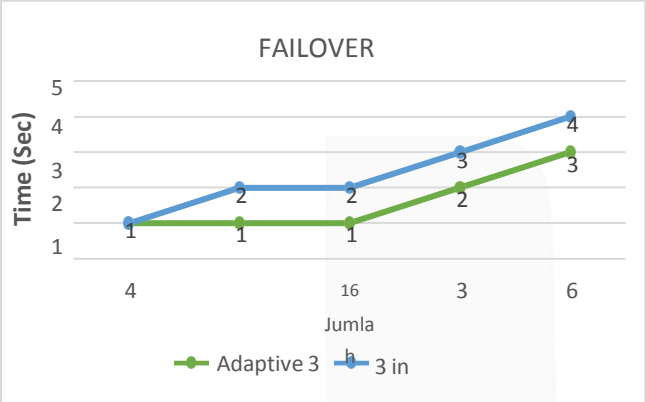
Skenario ketiga yaitu dengan *controller* B menjadi *controller* utama yang bertugas mengirim pesan ketika *controller* A *down* lalu terjadi *switch migration* dan *controller* A hidup Kembali. Dengan hal

tersebut proses akankembali berjalan normal dengan *controller* B yang menjadi *controller* utama.

Gambar 14. Proses *message exchange*

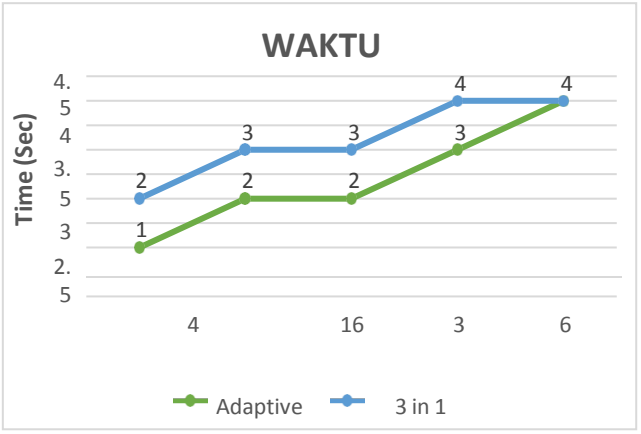
Gambar 14 merupakan proses *message exchange* setelah *controller* B menjadi *controller* utama yang bertugas mengirim pesan. Pengiriman pesan dilakukan dari *controller* B sebanyak tiga pesan kemudian akan dibalas oleh *controller* A dengan satu *acknowledgment.* Proses pengiriman pesan dilakukan selama 1 detik, jika melebihi waktu tersebut maka akan terjadi *down* dan akan terjadi *switch migration* terhadap *controller* A.

B. Pengujian Waktu *Failover*

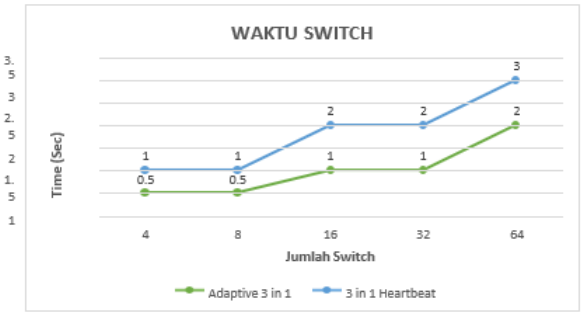


Gambar 15. Perbandingan waktu *failover*

Pada gambar 15 menunjukkan grafik dari selisih waktu *failover* untuk melakukan *switch migration* ketika *controller* terjadi *down.* Selisih tersebut merupakan hasil yang didapatkan dari kondisi *switch* yang sudah tidak terhubung pada *controller* hingga waktu ketika seluruh *switch* tersebut terhubung pada *controller B.* Pada grafik menjelaskan bahwa semakin banyaknya jumlah *switch,* maka waktu *failover* akan semakin tinggi. Hal ini dapat disimpulkan bahwa waktu *failover* akan berbanding lurus dengan jumlah *switch*. Untuk melihat selisih waktu ketika *switch-switch* yang sudah tidak terhubung dengan *controller* yang mengalami *down* hingga adanya informasi *switch* berpindah ke *controller* yang masih hidup adalah dengan menjalankan perintah *tail -f/var/log/openvswitch/ovs-vswitchd.log.*



Gambar 16. Waktu yang diperlukan *switch* ketika terjadinya *switch migration* saat proses *failover*

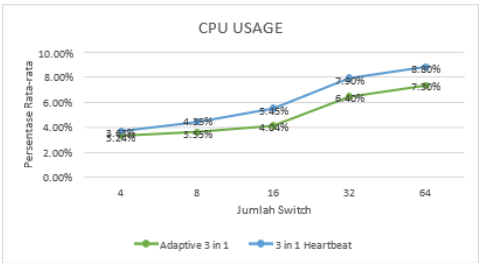


Gambar 17. Waktu yang diperlukan *switch* ketika proses kembali normal

Gambar 17 menunjukkan waktu yang diperlukan *switch* ketika *controller* kembali hidup atau proses pengiriman pesan kembali berjalan. Hasil menunjukkan bahwa pada metode *adaptive 3 in 1* dengan 4 *switch* dan 8 *switch* masing-masing membutuhkan waktu selama 0.5 detik, untuk 16 *switch* dan 32 *switch* membutuhkan waktu masing-masing 1 detik dan untuk 64 *switch* membutuhkan waktu selama 2 detik. Sedangkan pada metode *3 in 1 Hearbeat* dengan 4 *switch* dan 8 *switch* membutuhkan waktu selama 1 detik, untuk 16 *switch* dan 32 *switch*membutuhkan waktu selama 2 detik dan untuk 64 *switch* membutuhkan waktu selama 3 detik. Dari grafik tersebut dapat disimpulkan bahwa waktu yang dibutuhkan *switch* ketika proses kembali normal pada metode *adaptive 3 in1* lebih rendah.

C. Pengujian CPU *User*

Pengujian CPU *Usage* bertujuan untuk merepresentasikan skalabilitas jaringan yang berpengaruh terhadap beban kerja pada *controller.* Pada pengujian ini dilakukan perbandingan antara metode *adaptive 3 in 1* dan metode *3 in 1 heartbeat.*

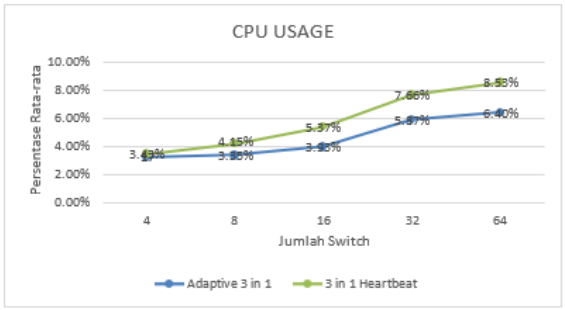


Gambar 18. Perbandingan CPU *Usage*

Gambar 18 merupakan grafik persentase rata-rata perbandingan dari nilai CPU *Usage* pada metode *adaptive 3 in 1* dan metode *3 in 1 heartbeat.* Untuk mendapatkan nilai tersebut dilakukan pengujian sebanyak 10 kali dengan waktu pengujian selama 10 detik untuk masing-masing *switch*. Pengujian dilakukan ketika salah satu*controller* terjadi *down* dengan melihat CPU load pada *controller* yang masih hidup.

Dari grafik diatas, hasil pengujian cukup representatif dari 4 *switch* hingga 64 *switch* yang semakin meningkat. Metode *adaptive 3 in 1* memperoleh hasil tertinggi sebesar 7.30% sedangkan nilai tertinggi yang diperoleh metode *3 in 1 heartbeat* sebesar 8.80%. Berdasarkan hasil tersebut metode *adaptive 3 in 1* memperoleh nilai selisih 1.50% lebih rendah dibandingkan metode *3 in 1 heartbeat*. Dengan begitu beban kerja pada metode *adaptive 3 in 1* lebih rendah. Hal tersebut dikarenakan pada metode *adaptive 3 in 1* kedua *controller*dapat saling bekerja sama dengan bertukar peran apabilasalah satu *controller* mengalami *down* sehingga beban kerja kedua *controller* menjadi seimbang. Kemudian, semakin banyak jumlah *switch* yang digunakan maka CPU *Usage* akan semakin tinggi dikarenakan banyaknya *switch* yang terhubung pada *controller.*

1. Pengujian CPU *Usage* dengan Nilai *Timeout* Tiga Detik



Gambar 19. Perbandingan CPU *Usage* dengan nilai *timeout* tiga detik

Gambar 19 merupakan grafik hasil persentase rata-rata perbandingan dari nilai CPU *Usage* pada metode *adaptive 3 in 1* dan metode *3 in 1 heartbeat.* Untuk mendapatkan nilai tersebut dilakukan pengujian sebanyak 10 kali dengan waktu pengujian selama 20 detik untuk masing-masing *switch*. Pengujian dilakukan ketika salah satu*controller* terjadi *down* dengan melihat CPU load pada *controller* yang masih hidup.

Pada hasil pengujian dengan nilai *timeout* tiga detik, metode *3 in 1 hearbeat* memperoleh hasil dengan persentase tertinggi dengan nilai 8.53% sedangkan metode *adaptive 3 in 1* memperoleh hasil persentase tertinggidengan nilai 6.40%. Dengan hasil tersebut dapat disimpulkan bahwa metode *adaptive 3 in 1* memperoleh nilai selisih 2.15% lebih rendah dibandingkan dengan metode *3 in 1 heartbeat.*

V. KESIMPULAN

Berdasarkan hasil pengujian yang telah dilakukan melalui metode *adaptive 3 in 1* dan metode *3 in 1 heartbeat, p*ada pengujian CPU *Usage* metode *adaptive 3 in 1* mendapatkan persentase CPU *Usage* lebih rendah dibandingkan metode *3 in 1 heartbeat.* Kemudian, hasil yang diperoleh dari pengujian dengan nilai *timeout* tiga detik dan percobaan sebanyak 10 kali dengan waktu pengujian selama 20 detik juga menghasilkan persentase CPU *Usage* yang semakin rendah. Hal ini dikarenakan kedua *controller* berperan sebagai *main to main* dan secara terus menerus aktif bekerja sama serta bertukar peran ketika salah satu *controller down* dan kemudian hidup kembali sehingga *controller* akan mengalami peningkatan kinerja. Pada pengujian waktu *failover*, penggunaan metode *adaptive 3 in 1* memperoleh waktu yang lebih cepat untuk menerima migrasi *switch*. Hal tersebut berpengaruh karena mekanisme *message exchange* yang terjadi dengan menggunakan metode*adaptive 3 in 1* menghasilkan persentase CPU *Usage* yang lebih rendah karena berkurangnya sumber daya pesan yang dilakukan ketika melakukan pengiriman pesan sehingga pada proses *failover* akan berjalan lebih cepat. Hasil pengujian menyimpulkan bahwa pengujian CPU *Usage* pada kinerja jaringan SDN menggunakan metode *adaptive 3 in 1* lebih baik. Hasil membuktikan bahwa metode *adaptive 3 in 1* memiliki peningkatan kinerja serta beban kerja pada *controller* yang lebih rendah dan *failover time* menghasilkan waktu yang lebih cepat.

REFERENSI

[1] Hu, T., Guo, Z., Yi, P., Baker, T., & Lan, J. (2018). *Multi-controller Based Software-Defined Networking: A Survey. IEEE Access, 6, 15980–15996.*

[2] O. Blial, M. Ben Mamoun, and R. Benaini, "An Overview on SDN Architectures with Multiple *Controller*s," J. Comput. Networks Commun., vol. 2016, 2016.

[3] Mutiara Ramadhani Wijaya, "Analisis penggunaan Metode 3-in-1 *Heartbeat* pada Arsitektur Active-active Distributed *Controller* di Jaringan Software Defined Networks," Program Studi Sarjana Informatika Fakultas Informatika Universitas Telkom Bandung, 2020.

[4] M. H. Hidayat and N. R. Rosyid, "Analisis Kinerja dan Karakteristik Arsitektur Software-Defined Network Berbasis OpenDaylight *Controller*," Citee, no. 2085–6350, pp. 194–200, 2017.

[5] M.N.A. Syaehoni, "Desain Distributed *Controller* dengan Metode Active-Active pada Jaringan Software Define Network," Program Studi Sarjana Informatika Fakultas Informatika Universitas Telkom Bandung, 2019.

[6] T. F. Oliveira and L. F. Q. Silveria, "Distributed SDN *controller*s optimization for energy saving," 2019 Fourth Int. Conf. Fog Mob. Edge Comput., pp. 86–89, 2019.

[7] T. Kim, T. Koo, and E. Paik, "SDN and NFV benchmarking for performance and reliability," 17th Asia- Pacific Netw. Oper. Manag. Symp. Manag. a Very Connect. World, APNOMS 2015, pp. 600–603, 2015.

[8] F. Fatturrahman, "SDN *Controller* Robustness and Distribution Framework," 2017.

[9] A. Kondel, "Evaluating System Performance for handling scalability challenge in SDN," pp. 594–597, 2015.

[10] F. Benamrane, M. Ben Mamoun, and R. Benaini, "New method for *controller*-to-*controller* communication in distributed SDN architecture," Int. J. Commun. Networks Distrib. Syst., vol. 19, no.3, pp. 357–367,2017