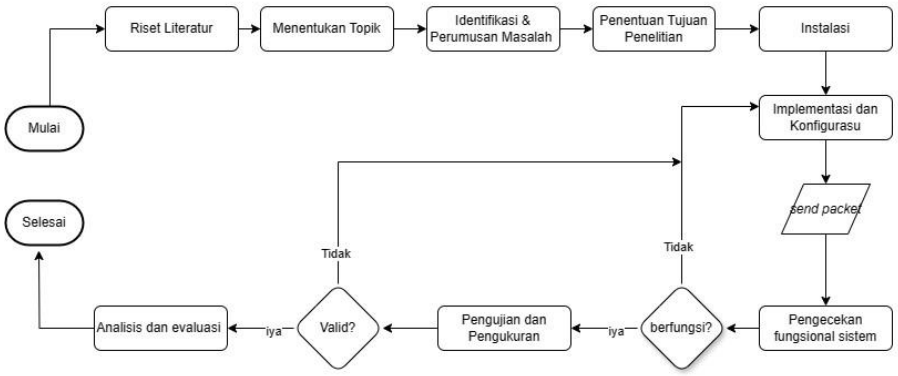
BAB III

METODE PENELITIAN

3.1 Alur Penelitian

Alur penelitian dijelaskan berdasarkan Gambar 3.1 seperti berikut.



Gambar 3. 1 Diagram Alur Penelitian

3.1.1 Riset Literatur

Tahap awal penelitian yaitu melakukan riset literatur untuk mendapatkan pemahaman, mengidentifikasi kesenjangan penelitian dan memahami perkembangan teknologi terkini. Pendekatan riset literatur pada jurnal, buku serta tesis pada penelitian terdahulu.

3.1.2 Menentukan Topik Penelitian

Topik penelitian diawali dengan mengidentifikasi isu-isu terkini dalam bidang jaringan. Berdasarkan riset literatur seperti jurnal ilmiah, buku, artikel untuk menemukan kesinambungan pengetahuan dan isu-isu yang belum terselesaikan pada bidang jaringan SDN. Sehingga topik penelitian ini yaitu keamanan siber pada jaringan SDN dari serangan.

3.1.3 Identifikasi dan Rumusan Masalah

Mengidentifikasi masalah penelitian serta merumuskan pertanyaan penelitian dalam memberikan kontribusi terhadap penelitian. Adapun rumusan masalah penelitian ini mencakup; pengaruh arsitektur *multi-controller* berbasis *load balancer* dalam mengurangi beban dan *response time* pada *controller*. Bagaimana

multi-controller berbasis *load balancer* dalam menjaga *high availability*. Serta, bagaimana analisis kinerja responsivitas dan *high availability multi-controller* berbasis *load balancer* dibandingkan dengan jaringan *single controller* pada jaringan SDN?

3.1.4 **Tujuan penelitian**

Tujuan penelitian untuk mengidentifikasi dalam menjawab rumusan masalah penelitian. Tujuan penelitian adalah untuk menjawab pertanyaan dari rumusan masalah dalam mempermudah pengembangan terhadap strategi rancangan sistem. Adapun tujuan penelitian untuk mengukur pengaruh arsitektur *multi-controller* menggunakan *load balancing* dalam responsivitas dari *controller*. Serta untuk mengukur kualitas jaringan (QoS) SDN pada saat serangan DDoS. Dan menganalisis responsivitas dan *high availability controller* berbasis *multi controller* dan *load balancer* dibandingkan dengan *single controller.*

3.1.5 **Instalasi**

Tahap instalasi melibatkan beberapa langkah yang diperlukan untuk mengkonfigurasi menggunakan kebutuhan penunjang seperti pada Tabel 3.1 yang diintegrasikan pada Virtual Machine (VM) ubuntu 22.0.1.

3.1.6 **Implementasi dan Konfigurasi**

Tahap implementasi digunakan pada *controller* ryu yang terintegrasi pada ubuntu. Proses implementasi menggunakan metode PPDIOO. Metode PPDIOO menghasilkan sebuah formula siklus hidup perencanaan jaringan yang terdiri dari 6 Fase (Saputra dkk., t.t.) yaitu *prepare, plan, desain, implement, operate,* dan *optimize* seperti desain penelitian pada Gambar 3.5. Proses konfigurasi mencakup perangkat *controller,* topologi dan TCP *dump* sebagai wadah monitoring *traffic.* Konfigurasi melibatkan penyesuaian *controller main* dan *backup* dalam menangani lalu lintas jaringan yang tinggi. Tahap ini mencakup konfigurasi dengan melakukan pengaturan peralihan *traffic* yang terintegrasi pada *controller*.

3.1.7 **Pengecekan Fungsional sistem**

Tahap pengecekan fungsional sistem dilakukan *testing* dengan pengujian pengiriman *packet* normal dan serangan pada arsitektur *single* dan *multi controller.* Tahap ini dilakukan dengan pengiriman *packet* dari masing-masing *host*

menggunakan *tools scapy* dan menggunakan *ping,* untuk mengetahui *host* yang terhubung dari hasil topologi yang dibuat.

3.1.8 **Pengukuran**

Tahap pengukuran dalam penelitian ini menggunakan skenario pengujian *traffic* normal dan serangan seperti pada Tabel 3.2. Parameter pengukuran untuk menganalisis respons sejauh mana sistem dapat merespons permintaan. Pengukuran *Availability* bertujuan mengukur sejauh mana sistem tersedia untuk pengguna sah dalam situasi *downtime* yang dapat berdampak negatif pada pengguna. Parameter pengukuran *availability* terdiri dari *latency, jitter* dan *throughput. Latency* bertujuan untuk mengukur lamanya waktu yang diperlukan untuk sistem merespons permintaan berdasarkan skenario pengujian. Sedangkan *throughput* untuk mengukur seberapa efisien sistem dalam mentransfer data dengan memantau dan menjaga kualitas layanan.

Efisiensi hasil pengukuran penggunaan sumber daya tersebut untuk menganalisis dan evaluasi kelayakan sistem keamanan jaringan SDN pada arsitektur *multi-controller* berbasis *load balancer.* Pengukuran menggunakan skenario pengujian berdasarkan Tabel 3.2 yang digunakan untuk melakukan analisis dan evaluasi parameter *responsivitas,* dan *high availability controller* dalam menangani paket yang masuk berdasarkan arsitektur jaringan SDN yang telah dibuat pada *emulator* mininet. Adapun rumus pengukuran parameter pengujian responsivitas *controller* seperti berikut.

RMC_nA = d1_nA – Avg (d2_nA,d3_nA, d4_nA, d5_nA)……..(3.1)

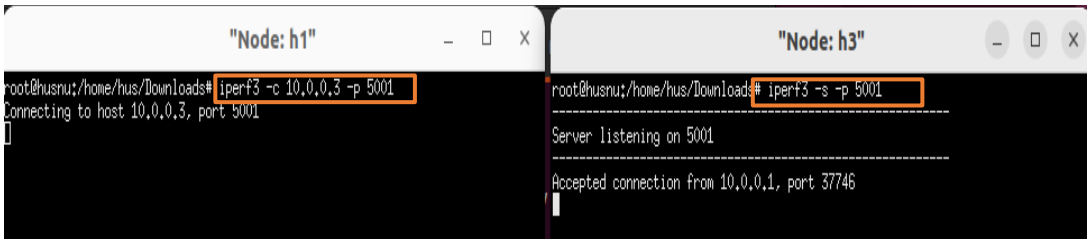
RSC_nA = d1_nA – Avg (d2_nA,d3_nA, d4_nA, d5_nA)……...(3.2)

RMC_wA=d1_wA-Avg (d2_wA,d3_wA, d4_wA,d5_wA)……..(3.3)

RSC_wA=d1_wA-Avg (d2_wA,d3_wA,d4_wA,d5_wA)……....(3.4)

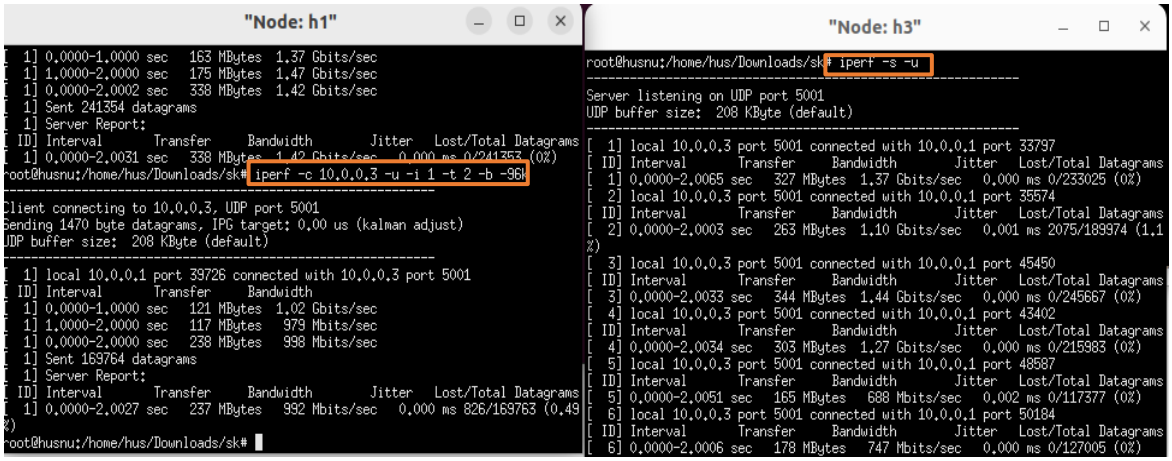
Keterangan : Wa = *With Attack*
nA = *Not Attack*
Avg = Rata- rata
d = Delta time
RMC = Responsivitas *Multi Controller*
RSC = Responsivitas *Single Controller*

Rumus perhitungan *responsivitas controller* berdasarkan desain arsitektur menggunakan *multi-controller* dan *load balancer* dan tidak menggunakan *multi-controller* dan *load balancer (single controller).* Pengukuran *responsivitas controller* terjadi dua titik pada arsitektur *multi controller* yaitu *traffic* paket yang masuk dari S1 menggunakan s1-eth1 dan trafik yang keluar dari S1 dengan menggunakan s1-eth5, sedangkan pada *single controller* trafik yang masuk dari S1 menggunakan s1-eth1 dan trafik yang keluar dari S1 dengan menggunakan S1-eth3. Perhitungan *high availability* yang mencakup *throughput* dan *jitter* pada skenario berdasarkan desain arsitektur pada Gambar 3.6 dan Gambar 3.7. Perhitungan parameter *throughput* dan *latency* menggunakan iperf seperti pada Gambar 3.2.



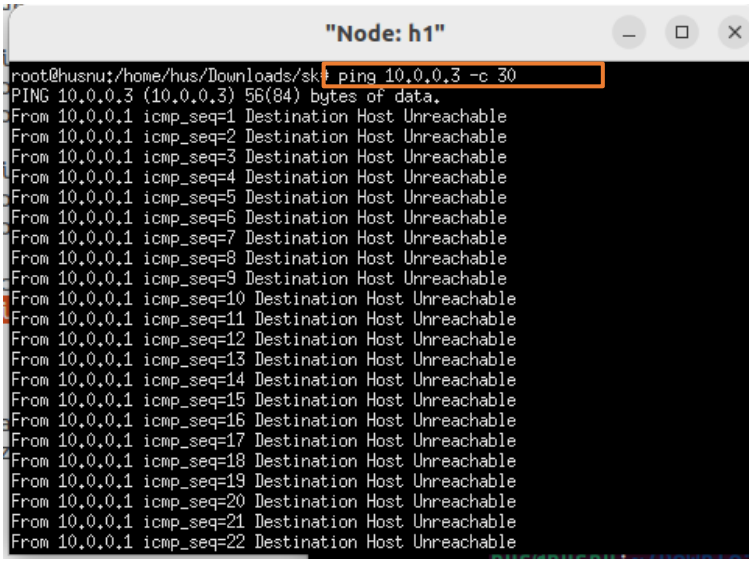
Gambar 3. 2 Perintah Pengukuran Parameter Throughput

Gambar 3.2 Menjelaskan perintah pengukuran *availability* menggunakan *iperf*. *Iperf* merupakan *software* yang digunakan untuk melakukan evaluasi kinerja jaringan melalui pengukuran *throughput, jitter,* antara dua perangkat yang terhubung dalam satu jaringan. Pengukuran *throughput* dan *jitter* menggunakan iperf3 pada jaringan, yang mana terdapat dua host yang terlibat, yaitu H1 dan H3. H1 berperan sebagai pengirim sedangkan H3 berperan sebagai server. Pengukuran *jitter* menggunakan iperf seperti Gambar 3.3. berikut.



Gambar 3. 3 Pengukuran Parameter Jitter

Pada Gambar 3.3 H1 berperan sebagai pengirim iperf3 dalam mengukur kinerja jaringan. Sedangkan "-c 10.0.0.3" menjelaskan pengaturan H1 sebagai *client* dan melakukan koneksi ke IP *address* 10.0.0.3 sebagai IP *server*"-p 5001" pada H1 berperan sebagai port 5001 yang tujuan untuk koneksi pada *host* 1. Sedangkan H3 pada "-s" berperan sebagai *server* yang artinya H3 akan mendengarkan koneksi dari *host* lainnya. Sedangkan "-p 5001" menjelaskan *port* 5001 sebagai *port* untuk menerima koneksi dari H1. *Software ipert* perlu tersedia pada sisi *server* dan *client* agar pengukuran dapat dilakukan. Pengukuran *throughput* menggunakan iperf dengan mengetikan perintah seperti pada Gambar 3.2 dengan H3 dari sisi *server* dan H1 dari sisi *client. Iperf* digunakan untuk mengukur *performance link* dari sisi TCP maupun UDP. Sedangkan pengukuran *latency* seperti pada Gambar 3.4 berikut.



Gambar 3. 4 Perintah Pengukuran Latency

Gambar 3.4 Menjelaskan perintah pengukuran *latency* pada sisi *client* dengan menggunakan perintah *ping* untuk mengukur keandalan dan kinerja suatu jaringan dengan mengirimkan paket data untuk mencatat pengiriman paket kembali. *Destination* IP menggunakan 10.0.0.3 sebagai IP dasar *host* tujuan dengan pengiriman 30 *packet.*

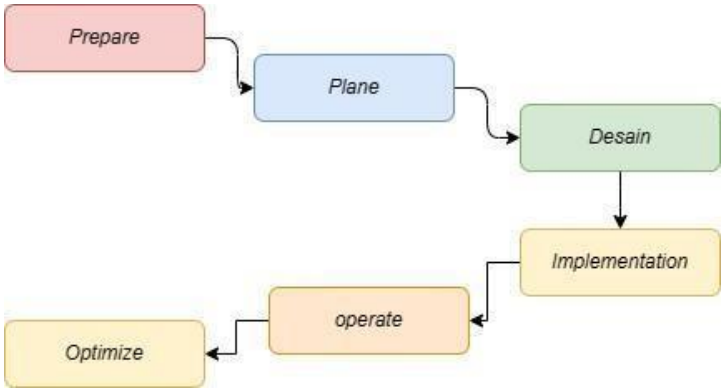
3.1.9 **Analisis dan Evaluasi**

Analisis pengukuran parameter pengujian dilakukan untuk menilai sejauh mana sistem dapat bekerja dalam mempertahankan *controller* dari serangan DDoS. Tahap analisis data yang dilakukan berdasarkan *miss rate* paket yang masuk berdasarkan selisih nilai turunan dari *matching rule* yang diinstall pada *switch* 1 dalam skenario pengujian. Hasil analisis berpedoman terhadap teori, jurnal, buku serta penelitian terdahulu untuk membandingkannya dalam mendapatkan hasil yang optimal.

Hasil pengukuran valid untuk dievaluasi berdasarkan analisis statistik dalam memastikan keandalan dan signifikansi hasil yang berpedoman terhadap teori dari sumber literatur jurnal, buku dan artikel berdasarkan *interval confidence* yang didapatkan dengan melakukan 10x pengujian terhadap sistem. Akan tetapi jika tidak valid, dilakukan pengecekan kembali pada tahap implementasi dan konfigurasi, sampai dengan tahap pengukuran.

3.2 Desain Penelitian

Penelitian ini menggunakan pengujian empiris dengan pengembangan menggunakan metode PPDIOO. Menurut CCDS 640-864 *official cert guide cisco* metode PPDIOO telah menghasilkan sebuah formula siklus hidup perencanaan jaringan yang terdiri dari 6 Fase (Saputra dkk., t.t.) yaitu *prepare, plan, desain, implement, operate,* dan *optimize.* CSL adalah pendekatan sistematis yang digunakan oleh Cisco untuk dapat merancang, mengimplementasikan serta mengoptimalkan solusi jaringan. Penjelasan detail metode sistem yang diajukan sebagai berikut dari rancangan pengamanan jaringan SDN menggunakan arsitektur *multi-controller* berbasis *load balancer* seperti Gambar 3.5 berikut.



Gambar 3. 5 Desain Penelitian

3.2.1 *Prepare plane*

Pada tahap persiapan kebutuhan penelitian *hardware* dan *software* yang digunakan untuk menunjang proses penelitian. Adapun kebutuhan penelitian rancangan sistem pengamanan jaringan SDN berbasis *multi controller* dan *load balancer* seperti pada Tabel 3.1.

Tabel 3. 1 Kebutuhan Penelitian

No	**ALAT**	**Keterangan**
Hardware		
1.	Laptop Asus X441UV Processor Intel Core I3 6006	Wadah pembuatan sistem perancangan keamanan jaringan SDN

	RAM 12 GB	
Software		
1.	Scapy 2.5.0	Tools pengujian paket serangan
2.	Ubuntu 22.0.1	Operation sistem untuk membangun sistem jaringan
3.	PHP 8.1	Mengelola *database* pada *web server*
4.	Python 9.10	Pengembangan kode pada *web* dalam membuat visualisasi data
5.	Ryu *controller*	*Framework Controller* SDN
6.	Mininet 2.2.2	*Emulator* membangun topologi
7.	OpenVSwitch 1.5	*Switch* virtual *protocol OpenFlow*
8.	Apache2	Jembatan *web server* dan *web browser*

Tabel 3.1 merupakan kebutuhan penunjang penelitian yang terdiri dari *hardware* dan *software* yang akan digunakan dalam penelitian. Perangkat *software* digunakan untuk simulasi dan alat pengembangan penelitian.

3.2.2 *Plane*

Tahap ini mencakup perencanaan skenario pengujian sistem. Adapun skenario pengujian sistem dilakukan dalam penelitian pada keadaan trafik normal dan serangan seperti pada Tabel 3.2.

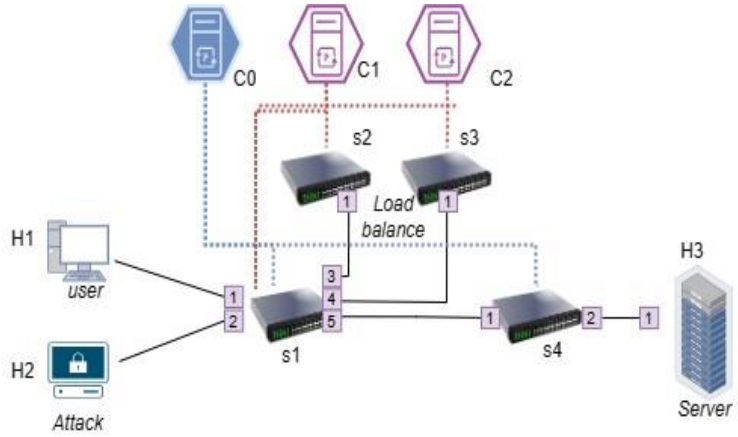
Tabel 3. 2 Skenario Pengujian Sistem

Pengujian	**Status *traffic***	***Single controller***	***multi controller***	***Load balancer***	**Parameter**
I	Tanpa serangan	–	√	√	*Responsivitas, latency, jitter* dan *throughput*
II.	Dengan serangan	–	√	√	
III.	Tanpa serangan	√	–	–	
IV	Dengan serangan	√	–	–	

Berdasarkan Tabel 3.2 dilakukan perancangan skenario pengujian sistem untuk melakukan pengukuran parameter *responsivitas* dan *availability* berdasarkan skenario pengujian. Pengamatan untuk pengukuran masing-masing parameter terdiri dari 2 titik yaitu *switch* 1 sebagai penerima *traffic* paket yang masuk dan sisi *server* sebagai *destination* tujuan *packet.*

3.2.3 *Desain*

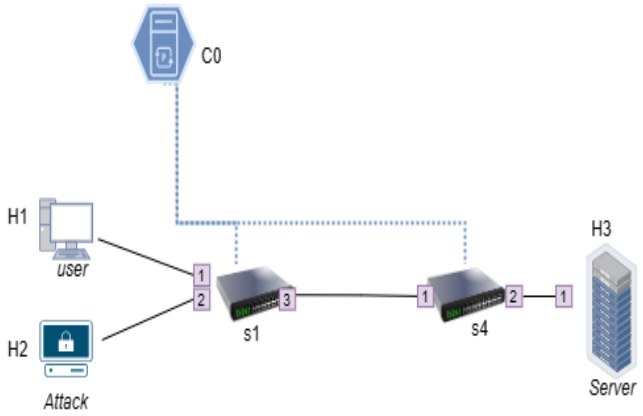
Tahap desain arsitektur topologi jaringan menggunakan 2 model yaitu arsitektur *multi-controller* berbasis *load balancer* dan *single controller*. Arsitektur pengamanan jaringan SDN *multi-controller* berbasis *load balancer* menggunakan 3 *controller* yang mana C0 berperan sebagai *production controller*, dan C1 serta C2 berperan sebagai *back-up controller* dalam menangani *traffic* melebihi *threshold* yang berpotensi sebagai *traffic* serangan. Berikut desain arsitektur skema *main-back up controller* berbasis *load balancer* seperti Gambar 3.6.



Gambar 3. 6 Desain Arsitektur Perancangan Sistem Multi-Controller

Pada Gambar 3.6 desain arsitektur topologi *multi-controller* berbasis *load balancer* terdiri dari 3 *host*, yang mana *host* 1 sebagai *user* dengan pengiriman paket normal. *Host* 2 sebagai penyerang untuk pengiriman paket serangan, dan *host* 3 sebagai *server* sebagai *host destination.* Selain itu, desain sistem terdiri dari 4 *switch*, dan 3 *controller.* Dalam arsitektur *multi-controller switch* 1 terkoneksi pada C0, C1 dan C2. Sedangkan S2 terkoneksi ke C1 dan S3 terkoneksi ke C2, S4 terkoneksi C0. Peranan C0 sebagai *controller* utama berfungsi memforward *traffic* ke *server* dan mendeteksi serangan berdasarkan nilai *miss rate*. Pada implementasi

load balancer S2 dan S3 berperan sebagai koneksi *in-band switch* untuk menghubungkan ke *controller back-up.* Ketika C0 mendeteksi nilai *miss-rate* melebihi *threshold* maka layanan trafik akan dialihkan ke C1 dan C2 berbasis *load balancer* (*controller back-up*). Peralihan *traffic* C1 dan C2 menggunakan kode *cookies* khusus yang dikonfigurasikan pada *switch. Load balancer* aktif ketika terjadinya peralihan *traffic* yang tinggi dari *main controller ke backup controller* sebagai bentuk *drop packet* ketika trafik yang masuk ke *controller* melebihi batas *threshold* yang terindikasi sebagai serangan. *Controller backup* akan melakukan pengalihan *traffic* ke C0 ketika *packet* telah berhasil di *drop*. Sedangkan arsitektur *single controller* seperti Gambar 3.7.

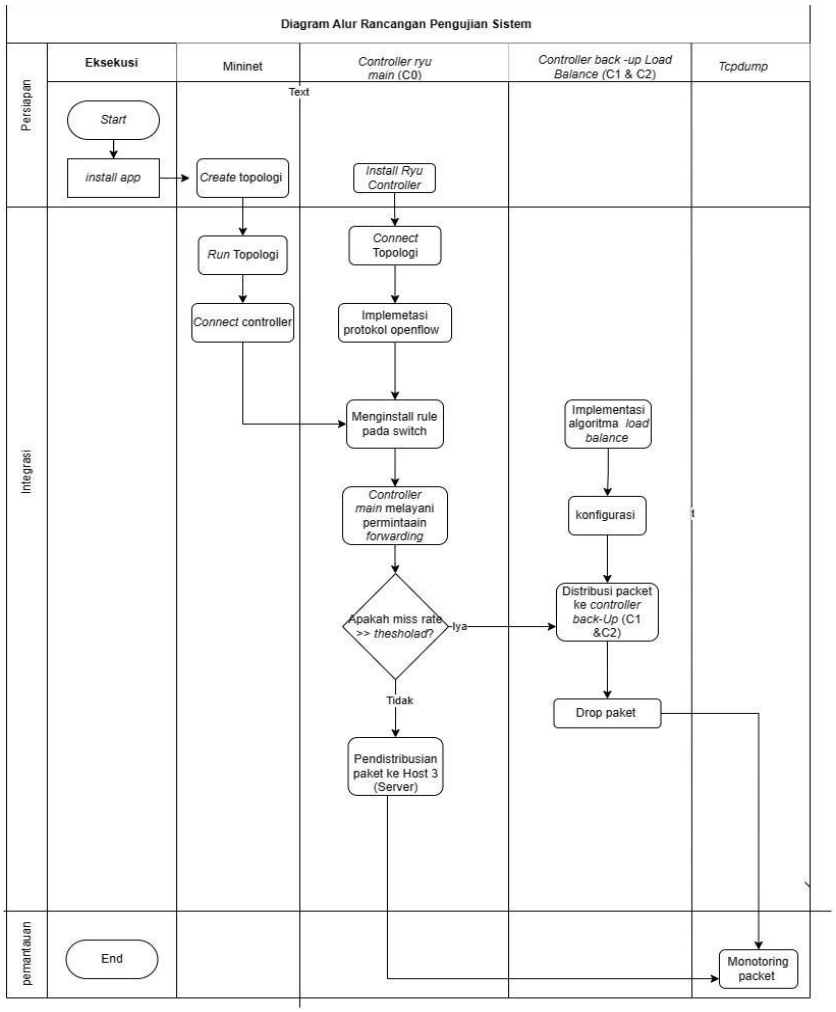


Gambar 3. 7 Desain Arsitektur perancangan Sistem Single-Controller

Berdasarkan Gambar 3.7 desain arsitektur topologi *single controller* dengan menggunakan 2 *switch*, dengan 1 *controller* dan 3 *host*. C0 pada arsitektur *single controller* berperan dalam mengelola dan mengkoordinasi seluruh sistem. Hal ini menjadikan arsitektur *single controller* satu entitas utama yang bertanggung jawab atas pengambilan keputusan, pemprosesan data dan mengelola interaksi antara komponen sistem sekaligus berperan dalam mendrop *packet* jika terindikasi sebagai serangan. Dalam desain topologi *single controller*, peranan masing-masing *host* yaitu *host* 1 sebagai *user* dengan pengiriman paket normal. *Host* 2 sebagai penyerang untuk pengiriman paket serangan, dan *host* 3 sebagai *server* sebagai *destination.*

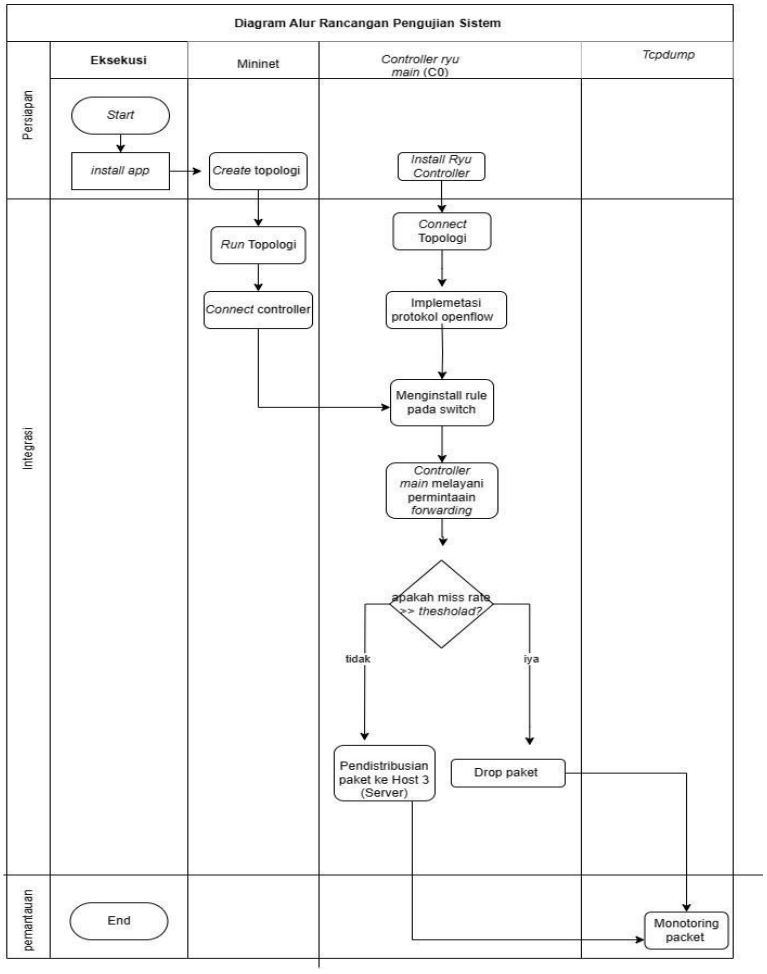
3.2.4 *Implementation*

Tahap Implementasi menggunakan *software* dan *hardware* yang diinstall pada sistem operasi ubuntu 22.0.1 untuk melakukan konfigurasi yang sesuai dengan rancangan sistem penelitian. Pada tahap ini arsitektur sistem yang telah dirancang akan diterapkan untuk mengetahui kelayakan perancangan sistem penelitian. Implementasi dimulai setelah dilakukan instalasi penunjang kebutuhan penelitian berdasarkan Tabel 3.1. Kemudian dilanjut dengan membuat konfigurasi pada *controller ryu* dan topologi pada *emulator* mininet sesuai dengan desain topologi seperti pada Gambar 3.6 dan 3.7. Adapun diagram alur implementasi untuk melakukan rancangan sistem *multi-controller* berbasis *load balancer* seperti Gambar 3.8.



Gambar 3. 8 Diagram Alur Rancangan Sistem Multi-Controller Berbasis Load Balancer

Gambar 3.8 Menjelaskan diagram implementasi sistem pada arsitektur *multi-controller* berbasis *load balancer*. Proses *experiment* serangan terjadi pada saat *packet* yang masuk nilai *miss-rate* terdeteksi melebihi *threshold*. Ketika *packet* yang masuk *miss-rate* nya melebihi dan mengakibatkan *traffic* yang tinggi. Maka, *main controller* mengalihkan *packet* ke C1 dan C2 yang berperan sebagai *controller back-up.* Pengalihan layanan dilakukan berdasarkan *port* masukan dengan mengimplementasi *load balancer* pada S2 dan S3. Setelah C1 dan C2 mendeteksi penurunan level serangan, C1 dan C2 mengalihkan layanan *traffic* ke C0 sebagai bentuk pengalihan layanan. Ketika *packet* yang masuk serangan maka yang mengalami *down* ialah *controller backup* bukan *controller main.* Sedangkan alur pengujian sistem *single controller* seperti pada Gambar 3.9.



Gambar 3. 9 Diagram Alur Rancangan Pengujian Sistem Single Controller

Berdasarkan Gambar 3.9 alur pengujian sistem pada arsitektur *single controller* menggunakan 1 *controller*, 2 *switch*, dan 3 *host.* Ketika paket masuk melebihi *threshold* yang terindikasi serangan maka, *controller main* yang berperan sekaligus sebagai *controller* utama menangani paket tersebut. *Controller* bertanggung jawab mengambil keputusan dengan mendrop atau meneruskan paket ke tujuan. Ketika paket yang masuk terindikasi sebagai serangan maka, *controller* C0 berperan sebagai *controller* utama dapat mengalami *down* karena terindikasi sebagai satu titik kegagalan dalam jaringan SDN.

3.2.5 *Operate*

Setelah dilakukan implementasi dan konfigurasi dilanjutkan dengan pengukuran berdasarkan skenario pengujian Tabel 3.2 untuk dilakukan tahap validasi sistem. Validasi dilakukan dengan melakukan pemantauan pada *controller* dalam efektivitas rancangan sistem. Implementasi pemantauan dengan monitoring *availability* dan *responsivitas controller* untuk mengukur efisiensi penggunaan sumber daya. Hasil pengukuran parameter dilakukan analisis statistik untuk mendapatkan keandalan dan signifikansi hasil *interval confident* dengan 10x pengujian untuk mendapatkan validasi hasil pengukuran. Validasi dilakukan untuk menilai sejauh mana sistem dapat efektif dan efisiensi dalam menangani serangan DDoS.

3.2.6 *Optimation*

Hasil pengukuran berdasarkan skenario Tabel 3.2 dilakukan evaluasi. Tahap evaluasi untuk mengetahui pengaruh arsitektur *load balancer* dan *multi-controller* dalam pengurangan beban *controller* pada satu titik kegagalan dari serangan DDoS. Evaluasi dilakukan untuk mengoptimalisasi identifikasi terhadap kinerja sistem yang telah dibangun dalam memaksimal terhadap penanganan serangan DDoS. Memastikan perpindahan *traffic* arsitektur *multi controller* berbasis *load balancer* ketika lalu lintas *traffic* tinggi. Evaluasi keandalan sistem mencakup perhitungan persentase kenaikan parameter pengukuran pada arsitektur *single* dan *multi controller*. Hasil pengujian memberikan pemahaman kemampuan sistem dalam kondisi pengiriman *packet* tidak serangan dan serangan. Evaluasi dapat memberikan wawasan yang mendalam tentang kelayakan sistem untuk melindungi

jaringan SDN dari serangan DDoS.

3.3 Teknik Analisis Data

Data yang digunakan dalam penelitian ini merupakan data akuisisi dari hasil pengujian. Sistem yang dibangun dilakukan pengujian secara langsung berdasarkan parameter penunjang penelitian untuk dilakukan pengumpulan data pada sistem. Metode pengumpulan data yang digunakan dalam penelitian ini antara lain:

1. Metode observasi skenario, yaitu hasil pengujian dari skenario pengujian arsitektur *single controller* dan *multi controller* berbasis *load balancer*. Hasil lalu lintas *traffic* dari pengujian di *capture* untuk akuisisi data pada *Tcpdump* dalam memonitoring *traffic* lalu lintas jaringan. Kemudian hasil *log* data pengujian disimpan untuk dilakukan pengukuran. Hal ini bertujuan untuk melakukan pengukuran parameter *responsivitas,* dan *high availability* berdasarkan skenario pengujian. Hasil pengukuran dilakukan analisis serta evaluasi pengaruh *multi-controller* berbasis *load balancer* dengan melakukan perbandingan dengan arsitektur *single controller* dalam mengurangi beban *controller* dari serangan DDoS.
2. Metode studi kepustakaan, yaitu dengan melakukan pemahaman konsep pengamanan jaringan SDN menggunakan *multi-controller* berbasis *load balancer* yang berpedoman terhadap referensi dari berbagai buku serta jurnal acuan yang berkaitan dengan penelitian. Setelah pengumpulan data hasil observasi skenario eksperimen maka dilanjutkan tahap analisis dan evaluasi informasi dari data hasil akuisisi yang bertujuan untuk mengutip, menganalisis dan evaluasi data hasil eksperimen dalam memproses informasi terhadap hasil penelitian. Data hasil pengukuran dilakukan validasi dengan analisis statistik dengan melakukan 10x pengujian untuk mendapatkan *interval confidence* hasil data akuisisi.