

Sistem Pengamanan Jaringan SDN dari Serangan DDoS Berbasis Multi Controller dan Load Balancer

DDoS Protection System for SDN Network Based on Multi Controller and Load Balancer

¹Husnul Ulfa*, ²Akbari Indra Basuki, ³Galura Muhammad Suranegara, ⁴Ahmad Fauzi

^{1,3,4} Sistem Telekomunikasi, Kamda Purwakarta, Universitas Pendidikan Indonesia, ² Research Center For Artificial Intelligence and Cyber Security BRIN

^{1,3,4} Jl.Veteran No.8, Nagri Kaler, Kec.Purwakarta, Kabupaten Purwakarta, 41115, ²Bandung cisitu, Jawa Barat

*e-mail: husnululfa3@upi.edu

(received: 30 December 2023, revised: 17 January 2024, accepted: 24 January 2024)

Abstrak

Serangan DDoS terhadap jaringan SDN dapat menciptakan satu titik kegagalan yang berpotensi merusak kinerja keseluruhan jaringan. Pada skema *single controller*, terdapat potensi untuk mengalami *buffer overload*, yang dapat menyebabkan kemacetan lalu lintas karena *switch* harus menunggu respon dari *controller* sebelum meneruskan paket jaringan. Untuk mengatasi tantangan ini, penelitian ini mengimplementasikan langkah-langkah keamanan dengan menggunakan pendekatan *multi controller* dan *load balancer*, bertujuan untuk meningkatkan ketahanan jaringan SDN terhadap serangan DDoS. Sistem ini bekerja dengan cara mendistribusikan beban kerja dari *controller* utama ke *controller* cadangan (*back-up*) melalui *load balancer* ketika terdeteksi adanya indikasi serangan DDoS. Indikasi serangan ini ditentukan berdasarkan nilai *miss rate* dari *unique forwarding request* yang melebihi batas tertentu (*threshold*). Hasil dari pendekatan ini terbukti mampu meningkatkan kehandalan, *responsivitas*, dan kualitas trafik jaringan SDN saat menghadapi serangan DDoS. Parameter pengujian yang dilibatkan dalam penelitian ini mencakup jeda respon *controller* serta kualitas lalu lintas jaringan, yang terdiri dari *latency*, *throughput*, dan *jitter*. Berdasarkan hasil pengujian, pendekatan berbasis *multi controller* dan *load balancer* berhasil meningkatkan kualitas jaringan serta *respons controller* sebesar 66,51% jika dibandingkan dengan skenario *single controller* yang lebih lama yaitu 202,49 % ketika terjadi serangan DDoS. Pada *responsivitas controller latency* terdapat peningkatan yang sangat kecil sekitar 0,01%, antara keduanya, *Multi Controller* berhasil menunjukkan peningkatan *throughput* yang mencolok sebesar 43,21% dibandingkan dengan *single controller*. sedangkan peningkatan *throughput* ini juga disertai dengan peningkatan *jitter* yang signifikan, mencapai 204%.

Kata kunci: DDoS, *load balancer*, *multi-controller*, *availability*, *overload*

Abstract

DDoS attacks on SDN networks can create a single point of failure that has the potential to disrupt the overall network performance. In a single controller scheme, there is a potential risk of experiencing buffer overload, leading to traffic congestion as switches must wait for responses from the controller before forwarding network packets. To address this challenge, this research implements security measures using a multi-controller and load balancer approach, aiming to enhance SDN network resilience against DDoS attacks. The system operates by distributing the workload from the main controller to a backup controller through a load balancer when indications of a DDoS attack are detected. These attack indications are determined based on the miss rate value of unique forwarding requests exceeding a specific threshold. The results of this approach have proven effective in improving the reliability, responsiveness, and quality of SDN network traffic during DDoS attacks. The testing parameters involved in this research include controller response time and network traffic quality, comprising latency, bandwidth, throughput, and jitter. Based on the test results, the multi-controller and load balancer-based approach successfully enhanced network quality and controller responsiveness by 66.51% compared to the longer single controller scenario, specifically 202.49%

<http://sistemasi.ftik.unisi.ac.id>

during DDoS attacks. In terms of controller responsiveness, there is a very slight increase of around 0.01% in latency between the two. While Multi Controller demonstrated a remarkable 43.21% increase in throughput compared to Single Controller, this improvement in throughput is accompanied by a significant 204% increase in jitter

Keywords: DDoS, load balancer, multi-controller, availability, overload

1 Pendahuluan

Perkembangan teknologi yang sangat pesat menjadikan jaringan internet kebutuhan manusia di era modern. Banyaknya pendekatan untuk menyelesaikan kinerja jaringan internet agar lebih efisien dan optimasi untuk menunjang kebutuhan komunikasi saat ini. Kebutuhan akan jaringan internet yang diprogramkan telah menarik minat dan industri serta akademisi untuk mengembangkan paradigma jaringan. Hal ini terjadi karena arsitektur internet tradisional yang cukup kompleks dan tertutup menyulitkan administrator jaringan untuk mengoperasikan serta mengelola dinamika jaringan dari berbagai kebutuhan aplikasi [1]. Arsitektur jaringan tradisional yang terbatas tidak cukup memenuhi permintaan aplikasi heterogen pada kebutuhan komunikasi yang terus meningkat. Oleh karena itu arsitektur *Software Defined Networking* (SDN) telah diusulkan sebagai paradigma jaringan yang dapat memecahkan masalah tersebut dalam mempercepat inovasi untuk mengatasi tantangan infrastruktur jaringan dan komunikasi saat ini.

Arsitektur jaringan SDN memisahkan bidang *control* dan bidang data dalam *control* yang terpusat. Pemisahan bidang *control* dan bidang data digabungkan menjadi satu entitas bernama pengontrol. Pengontrol SDN menyediakan API untuk mengambil keputusan dan memungkinkan operator menerapkan berbagai kebijakan jaringan secara fleksibel dan dinamis. API pada SDN mampu mengetahui statistik perangkat dan memberikan kebebasan *programmer* dalam mengembangkan suatu aplikasi perangkat [2]. Pemisahan *control plane* dan *data plane* yang dapat diprogramkan secara langsung untuk mempermudah operator membangun, mengelola dan mengoptimalkan konfigurasi jaringan dengan fleksibel dalam memperluas fungsi sumber daya jaringan yang cepat. Hal ini memberikan kemudahan termasuk manajemen, dinamika dan efektivitas biaya. Akan tetapi, pengontrolan yang terpusat memberikan tantangan besar bagi keamanan jaringan SDN.

Arsitektur jaringan SDN yang terpusat, terbuka dan dapat diprogramkan mengakibatkan celah kerentanan keamanan dalam manajemen jaringan. Tantangan keamanan SDN menurut struktur intrinsik SDN terdiri dari dua aspek yaitu tantangan berbasis perangkat keras dan tantangan berbasis protokol. Pada tantangan berbasis perangkat keras, *controller* menjadi pilihan utama bagi penyerang karena *controller* adalah perangkat yang secara logis terpusat. Pemisahan lapisan *control* dari lapisan *forwarding* menyebabkan ketergantungan pada *control* yang terpusat sebagai otak dari suatu jaringan. Oleh karena itu banyaknya pendekatan serangan yang memilih *controller* sebagai target serangan. Hal ini menjadi celah bagi penyerang untuk dapat meluncurkan serangan DDoS (*Distributed Denial of Service*)[3]. Serangan DDoS dilakukan penyerang dengan menghasilkan sejumlah besar paket palsu dan mengirimkannya melalui *switch* pada saat yang bersamaan. Pada *switch* semua paket palsu baru yang tidak sesuai dengan *flow rules*, dikirimkan ke *controller* yang menghasilkan banyaknya jumlah permintaan. Dengan hal ini sumber daya komputasi *controller* akan terkuras dalam waktu yang singkat. Pada tantangan berbasis protokol mekanisme *verifikasi* pada versi *OpenFlow* menjadikan autentikasi timbal balik opsional antara pengontrol dan *switch*, sehingga kurangnya adopsi TLS yang menyebabkan penyerang dapat dengan cepat mendapatkan informasi dalam penggabungan satu entitas pada *controller protocol OpenFlow*.

Controller sebagai komponen utama memegang peranan penting dalam arsitektur SDN. Peranan *controller* menjadi target para penyerang yang berpotensi untuk meluncurkan serangan. Semua lalu lintas yang tidak diketahui harus dikirimkan ke *controller* untuk diselidiki. Sehingga, lalu lintas berbahaya dapat menyebabkan terjadinya serangan DDoS (*Distributed Denial of Service*) pada jaringan SDN. Serangan DDoS ditunjukkan untuk membuat sumber daya sistem tidak tersedia bagi pengguna yang sah. Sistem *controller* yang terkena serangan DDoS menyebabkan kelebihan beban pada lalu lintas sehingga mengakibatkan *server down* [4]. Penyerang dapat meluncurkan serangan DDoS antara komunikasi pengontrol dan *switch* untuk membuat jaringan lumpuh. Setelah *switch* SDN diserang, *flow rule* dapat dengan mudah dimodifikasi. Serangan DDoS menyerang menggunakan IP *addres* yang dipalsukan untuk menyembunyikan identitas penyerang. Serangan DDoS dilakukan

<http://sistemasi.ftik.unisi.ac.id>

dengan pengiriman layanan yang membanjiri *control* dan *entri flow*. Hal ini terjadi dengan mengirimkan paket unik yang berbahaya pada lalu lintas *traffic*. Sehingga memudahkan aplikasi berbahaya dan akses aplikasi yang tidak sah muncul di bidang aplikasi. Timbulnya celah keamanan ini menyebabkan perubahan data, pemblokiran akses ke jaringan serta pelanggaran data.

Selain itu, implementasi arsitektur *single controller* pada jaringan SDN tidak dapat memenuhi kebutuhan pengendalian dan pengelolaan dalam skala besar karena kemampuan yang terbatas. Sehingga menyebabkan kemacetan pada satu titik *controller* yang dapat mengakibatkan penurunan *fleksibilitas* [5]. Penurunan *fleksibilitas* dapat mempengaruhi kemampuan manajemen lalu lintas dalam jaringan SDN yang menyebabkan lumpuhnya sistem pada jaringan SDN. Keterbatasan arsitektur *single-controller* dalam menangani *packet* dengan jumlah besar yang terindikasi sebagai serangan mengakibatkan beban kerja yang tinggi pada *controller*. Hal ini berpotensi sebagai *buffer overload* pada *openflow switch* karena menunggu *response* dari *controller*. *Response* arsitektur *single controller* yang berurutan terhadap latensi aplikasi yang sensitif menyebabkan kelebihan beban dan meningkatkan latensi *responsivitas* pada *controller* [6]. Dalam ketersediaan layanan yang tinggi memerlukan *responsivitas* dalam situasi *downtime* yang terindikasi serangan. Hal ini tidak dapat diterima dalam pengendalian arsitektur *single controller* karena tidak mampu memenuhi kebutuhan jaringan yang luas.

Penelitian ini bertujuan untuk mengevaluasi efektivitas arsitektur multi-controller dan load balancer dalam menjaga responsivitas controller utama dan kualitas keseluruhan jaringan SDN saat menghadapi serangan DDoS. Untuk mengukur kemampuan kinerja arsitektur yang diusulkan, penelitian ini mengamati nilai jeda waktu respon dari controller dan parameter kualitas jaringan, termasuk latency, throughput, dan jitter. Evaluasi peningkatan kinerja dilakukan dengan membandingkan hasil pengukuran dari parameter tersebut dengan hasil pengukuran pada skenario menggunakan single controller.

2 Tinjauan Literatur

Jaringan SDN (*Software Defined networking*) adalah paradigma pengelolaan jaringan yang dapat memberikan fleksibilitas yang dinamis pada *control sentral* terhadap infrastruktur jaringan. SDN memisahkan integrasi vertikal pada *control plane* dari *data plane*. Pemisahan *data plane* dan *control plane* pada arsitektur jaringan SDN mempermudah administrator jaringan dalam mengontrol kerja pada perangkat melalui *controller* tanpa mengkonfigurasi setiap perangkat. Dengan pemisahan arsitektur SDN tersebut, *switch* menjadi perangkat *forwarding* dan logika *control* yang diimplementasikan dalam pengontrol terpusat.

Arsitektur SDN terdiri dari tiga lapisan yaitu, lapisan infrastruktur, lapisan kontrol dan lapisan aplikasi. SDN juga terdiri dari *interface* yaitu *northbound* dan *southbound* API. Pada lapisan infrastruktur menyediakan informasi *forwarding* aliran ke *controller* melalui *interface data plane* [1]. *Data plane* terdiri dari *switch OpenFlow* yang menerima perintah dari *controller* serta memperlakukan sebagai *rules* yang dapat di program. Pada lapisan *control* memberikan tautan untuk interaksi antara aplikasi dan lapisan kontrol. *Control* pada dasarnya bertanggung jawab untuk mengirimkan aturan *forwarding* ke *switch*. Pada bidang aplikasi layer berisi layanan SDN untuk memenuhi kebutuhan pengguna dalam akses dan mengontrol peralihan di *data plane* melalui layer *control*. API pada SDN terdiri dari *northbound* dan *southbound* API. *Southbound API (Application Programming Interface)* adalah komunikasi yang dapat menghubungkan lapisan infrastruktur dan lapisan *control* [7]. Sedangkan *northbound API interface* komunikasi yang dapat menghubungkan lapisan *control* dan lapisan aplikasi. Penempatan *control* pada arsitektur jaringan SDN didasarkan pada metrik latensi rata-rata *controller* peralihan maksimum dan latensi antar *controller* [8].

Konsep *multi-controller* mengacu pada penggunaan beberapa *controller* SDN yang bekerja bersama untuk mengelola jaringan. Jaringan SDN memungkinkan pemisahan antara lapisan *control* dan lapisan data (*switch* atau *router*). Hal ini pada mengacu pada jaringan berkemampuan SDN yang memiliki lebih dari satu *controller* untuk menghasilkan kinerja tinggi, *skalabilitas*, serta ketersediaan. Arsitektur *multi-controller* diimplementasikan dalam bidang *control* yang terdistribusi, terpusat secara logis tetapi terdistribusi secara fisik. Sulitnya *single controller* untuk mengelola jaringan yang berskala besar dengan *multi domain*, karena fitur yang berbeda di beberapa *controller*. Kolaborasi dari beberapa pengontrolan untuk dapat meningkatkan efisien dalam manajemen jaringan *multi domain*

menggunakan arsitektur *multi-controller* [1]. Sehingga dengan arsitektur *multi-controller* menyebabkan permintaan menjadi seimbang antara pengontrol yang berbeda ke sebagian pengontrol yang dapat mengalokasikan untuk mengurangi latensi pada proses lalu lintas trafik. Hal ini terjadi karena banyaknya aplikasi sensitif terhadap latensi pada *controller*. Latensi *response* permintaan dapat dikurangi melalui *load balancer* untuk menghindari *controller* yang kelebihan beban dengan menambahkan secara dinamis untuk mencapai kinerja yang lebih tinggi.

Load balancer pada jaringan SDN mendistribusikan beban kerja dinamis secara merata. Mekanisme pendistribusian terjadi ketika *controller* yang kelebihan beban dapat mentransfer beban ke *controller* lain, sehingga dapat meningkatkan kemampuan dalam bidang kendali secara *respon* cepat terhadap permintaan *switch* [1]. Pendistribusian *load balancer* pada SDN secara merata diklasifikasi menjadi dua kategori yaitu; *load balancer* terpusat dan *load balancer terdistribusi*. Dalam *load balancer* terpusat bertanggung jawab dalam menyeimbangkan beban. Melakukan pengumpulan beban dari *controller* lain, menginformasikan *controller* yang kelebihan beban untuk melakukan transfer sebagian bebannya ke *controller* yang memiliki beban ringan. *Load balancer* pada pendekatan terdistribusi ditentukan pada ambang batas untuk setiap *controller* berdasarkan kapasitas perangkat keras pengontrol [9]. *Load balancer* tidak diperlukan hingga beban lalu lintas melewati nilai tersebut. Ketika beban pada *controller* melewati ambang batasnya, *load balancer* mengumpulkan dari *controller* lain kemudian *controller* dapat memulai proses *load balancer* [10]. *Load balancer* diperlukan untuk efisiensi, skalabilitas dan ketersediaan jaringan dalam meningkatkan kinerja *controller* untuk meminimalisir *downtime* dengan meningkatkan skalabilitas dan fleksibilitas dari serangan DDoS [1].

Serangan DDoS bersifat terdistribusi yang dapat mengakibatkan lumpuhnya layanan jaringan dengan menghancurkan perangkat sistem jaringan. Serangan DDoS membanjiri target dengan mengirimkan paket unik yang berbahaya [11]. Tujuan utama serangan DDoS untuk mengkonsumsi memori, pemrosesan, dan sumber daya *throughput* dalam komponen yang akhirnya mengarahkan pada kinerja jaringan degradasi [12]. Serangan DDoS pada jaringan SDN dibagi menjadi tiga kategori yaitu serangan DDoS pada lapisan aplikasi, serangan DDoS pada lapisan *control* dan serangan DDoS pada lapisan infrastruktur. Serangan DDoS pada lapisan aplikasi dengan melakukan penyerangan langsung aplikasi dan melakukan serangan pada *outbound* API [13]. Serangan DDoS pada lapisan *controller* dapat dilakukan dengan penyerangan *controller* melalui *northbound* API, *southbound* API, dan *westbound* API. Serangan DDoS pada lapisan infrastruktur dapat dilakukan dengan melakukan serangan secara langsung terhadap *switch* melalui *southbound* API. Serangan ini mengakibatkan elemen memori pada *node* mengalami *bottleneck* dikarenakan beban *traffic* yang tinggi [13].

Beberapa solusi terhadap penanganan serangan DDoS pada jaringan SDN diusulkan dalam literatur untuk jaringan SDN. Serangan DDoS pada *controller* dan *switch* dapat melumpuhkan jaringan yang menyebabkan *flow table* dengan mudah dimodifikasi yang berdampak buruk pada keseluruhan jaringan. Dalam mengatasi masalah ini, solusi yang diusulkan penulis [14] dengan peningkatan mekanisme dalam deteksi serangan DDoS berbasis grafik otomatis menggunakan *fuzzy cognitive map* di jaringan SDN. Pencegahan serangan DDoS pada layanan DHCP yang menargetkan pada *controller* yang dapat mengakibatkan pelambatan jaringan dan membebani *controller* secara berlebihan. Peneliti [15] menggunakan modul keamanan DHCP guard, memanfaatkan fungsi pengintaian DHCP, pembatasan laju, dan pemulihan kumpulan IP untuk memblokir pesan DHCP berbahaya untuk mengurangi dampak negatif serangan terkait DHCP pada jaringan tanpa *overhead* yang signifikan.

Paket yang tidak sesuai dengan *flow table* pada *switch* akan diteruskan ke *controller*. Hal ini menyebabkan penyerang mengirimkan paket dengan jumlah yang besar dari beberapa alamat IP yang tidak diketahui pada *switch* yang akan diteruskan ke *controller*. Serangan DDoS dapat melumpuhkan sistem yang menyebabkan hubungan antar *switch* dan *controller* menjadi tidak tersedia karena kemacetan lalu lintas berbahaya. Dalam permasalahan ini peneliti [16] Mengevaluasi sistem keamanan berbasis entropi dengan model JESS untuk meningkatkan keamanan SDN dengan memperkuat arsitektur SDN terhadap serangan DDoS. Manajemen jaringan SDN yang terpusat rentan terhadap kegagalan dalam satu titik *controller* yang dapat mengakibatkan penurunan fleksibilitas dalam ketahanan terhadap serangan yang menyebabkan penyadapan dan manipulasi lalu lintas jaringan. Penulis [17] mengimplementasikan metode *clustering* skema otentikasi untuk *multi-controller*. Hierarki pengontrol yang terpusat menimbulkan satu titik kegagalan dan resiko terjadinya serangan

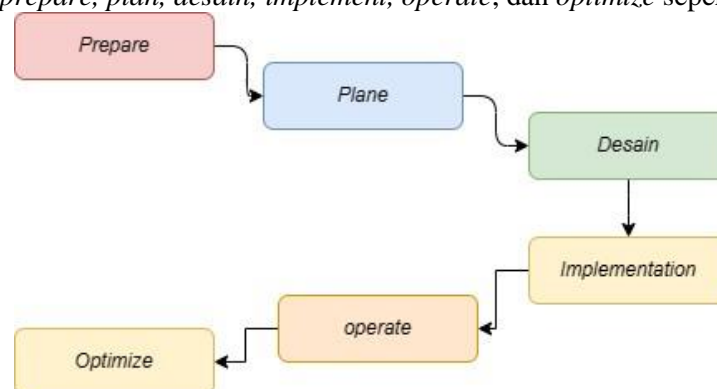
<http://sistemasi.ftik.unisi.ac.id>

DDoS yang memberikan kemampuan program jaringan untuk menarik penyerangan. Oleh karena itu penulis [18] mengimplementasikan solusi yang memanfaatkan kemampuan dalam kontrol yang terpusat dengan dua subsistem, pertama mencegah pengguna yang tidak sah dengan memverifikasi identitas *host*. Kedua, keamanan terpusat pada aplikasi firewall diatas *controller* untuk menyajikan pos pemeriksaan terdistribusi yang dapat memberikan pertahanan.

Arsitektur *multi-controller* yang terpusat dapat terdistribusi diusulkan dalam penelitian ini untuk meningkatkan pertahanan terhadap satu titik kegagalan dan skalabilitas pengontrol. Dalam menghindari satu titik kegagalan dan keterbatasan untuk menangani beban kerja yang tinggi. Serta meningkatkan kemampuan pemrosesan jaringan dalam skala besar pada jaringan SDN [19]. Diimplementasikan Arsitektur *multi-controller* berbasis *load balancer* untuk mengurangi latensi *response* yang tinggi terhadap permintaan. Implementasi *load balancer* berperan pada jaringan SDN dalam situasi kelebihan beban pada *controller* dengan pendistribusian trafik dalam mempertahankan *controller* terhadap satu titik kegagalan [20]. Selain itu, *overload* pada *OpenFlow switch* menyebabkan kemacetan dalam *response* dari *controller*. Sehingga perlunya implementasi *load balancer* dan *multi controller* yang digunakan untuk pertahanan dalam menjaga ketersediaan yang tinggi pada beban dan *responsivitas* dari *controller*. Fitur *extract* serangan yang digunakan untuk mendeteksi serangan DDoS pada jaringan SDN yang digunakan berdasarkan lonjakan *source port*. Ketika SDN dalam kondisi normal, jumlah *port* sumber mempunyai nilai yang relatif tetap. Akan tetapi ketika terjadi serangan, *port* sumber akan berubah lebih tinggi ketika SDN di serangan [4].

3 Metode Penelitian

Penelitian ini menggunakan jenis penelitian pengujian empiris (*experiment*) dengan pengembangan menggunakan metode PPDIIO. Menurut CCDS 640-864 *official cert guide cisco* metode PPDIIO telah menghasilkan sebuah formula siklus hidup perencanaan jaringan yang terdiri dari 6 Fase [21] yaitu *prepare, plan, desain, implement, operate, dan optimize* seperti pada Gambar 1.



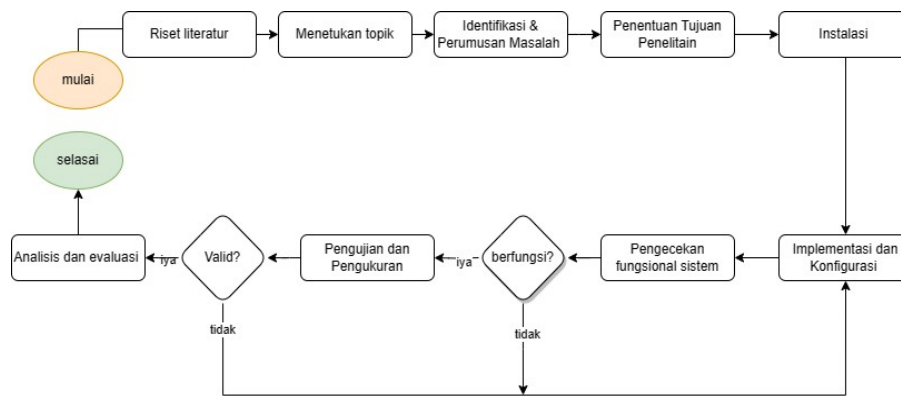
Gambar 1. Alur pengembangan sistem

a. Prepare plane

Pada tahap persiapan yang dilakukan adalah melakukan riset literatur seperti jurnal ilmiah, buku, artikel untuk menemukan kesinambungan pengetahuan dan isu-isu yang belum terselesaikan pada bidang jaringan. Kemudian penentuan topik penelitian agar menjadi lebih spesifik. Topik penelitian yang diangkat yaitu keamanan jaringan SDN.

b. Plane phase

Tahap ini melakukan perencanaan penelitian pembuatan rancangan sistem keamanan jaringan SDN yang mencakup perencanaan alur penelitian, keputusan *hardware* dan *software* yang digunakan pada penelitian dan perencanaan skenario pengujian sistem. Adapun tahapan alur penelitian seperti pada Gambar 2.



Gambar 2. Diagram alur penelitian

Gambar 2 menjelaskan diagram alur penelitian yang dilakukan untuk mengimplementasikan sistem pengamanan jaringan SDN berbasis *multi-controller* dan *load balancer*. Tahap awal penelitian yaitu melakukan riset literatur. Setelah itu, melakukan identifikasi dan rumusan masalah yang menjadi dasar penelitian. Dari rumusan masalah ini diimplementasikan metode penelitian dalam mencapai tujuan penelitian. Tujuan penelitian adalah untuk menjawab pertanyaan dari rumusan masalah dan untuk mempermudah pengembangan terhadap strategi rancangan sistem. Kemudian melakukan tahap persiapan *hardware* dan *software* yang dilanjutkan dengan tahap instalasi. Setelah berhasil diinstal dilanjutkan dengan implementasi dan konfigurasi terhadap rancangan sistem pengamanan jaringan SDN. Tahap selanjutnya melakukan pengecekan fungsional sistem yang dibangun. Hal ini untuk melakukan pengecekan fungsional sistem dalam menjalankan fungsi dan tugasnya. Skenario pengujian sistem dilakukan dalam penelitian pada keadaan trafik normal dan serangan seperti pada Tabel 1 berikut :

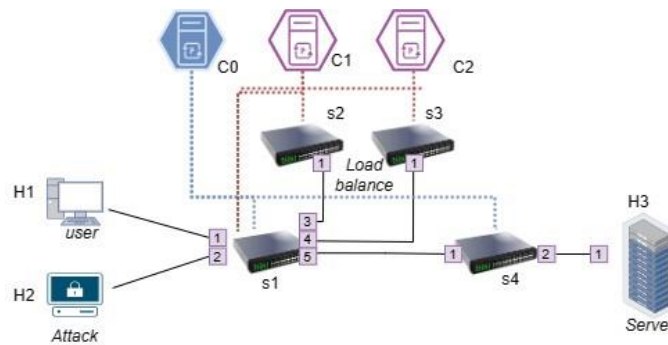
Tabel 1. Skenario Pengujian Sistem

No.	Pengujian skenario	Skenario ke
1.	Trafik normal <i>tanpa multicontroller & load balancer</i>	pertama
2.	Trafik serangan tanpa <i>multicontroller load balancer</i>	kedua
3.	Trafik normal dengan <i>multi controller load balancer</i>	ketiga
4.	Trafik serangan dengan <i>multi controller load balancer</i>	keempat

Berdasarkan Tabel 1 dilakukan perancangan skenario pengujian sistem untuk melakukan pengukuran parameter *responsivitas* dan *availability* pada 4 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 paket. Pengukuran parameter *response* untuk menganalisis sejauh mana sistem dapat merespons permintaan. Sedangkan pengukuran *Availability* bertujuan untuk mengukur sejauh mana sistem tersedia dalam mencegah *downtime* yang dapat berdampak negatif pada pengguna, yang mana pengukurannya terdiri dari *latency*, *jitter*, dan *throughput* [22]. *latency* bertujuan untuk mengukur lamanya waktu yang diperlukan untuk sistem merespons permintaan berdasarkan 4 skenario pengujian. Sedangkan *throughput* untuk mengukur seberapa efisien sistem dalam mentransfer data untuk memantau dan menjaga kualitas layanan. *Jitter* bertujuan variasi atau fluktuasi kecil dalam waktu antara dua kejadian atau sinyal. Efisiensi hasil pengukuran penggunaan sumber daya tersebut untuk menganalisis dan evaluasi kelayakan sistem keamanan jaringan SDN pada arsitektur *multi-controller* berbasis *load balancer*.

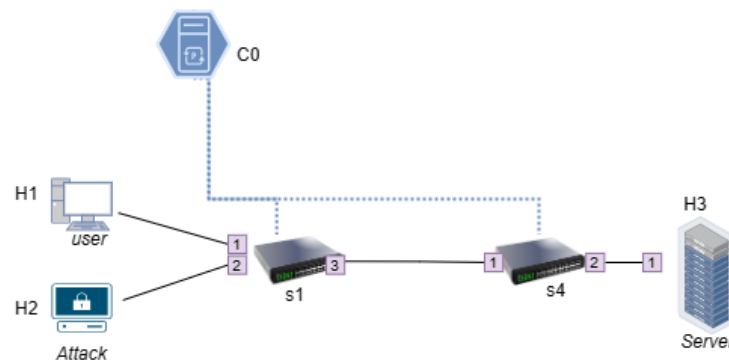
c. Desain phase

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* yang melebihi *threshold* yang berpotensi sebagai *traffic* serangan. Berikut desain arsitektur perancangan sistem keamanan jaringan SDN seperti Gambar 3.



Gambar 3. Desain arsitektur perancangan sistem multi-controller berbasis load balancer

Seperti yang terlihat pada Gambar 3 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*. Selain itu desain sistem terdiri dari 4 *switch*, dan 3 *controller*. Dalam arsitektur, *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 main* yang Fungsinya sebagai memforward *traffic* ke *server*, dan mendeteksi serangan berdasarkan nilai *miss rate*. S2 dan S3 berperan sebagai koneksi *in-band switch* pada implementasi *load balancer* untuk menghubungkan ke *controller back-up*. Ketika C0 mendeteksi nilai *miss-rate* melebihi *threshold* maka layanan trafik yang ada akan di alihkan ke C1 dan C2 (*controller back-up*). Peralihan *traffic* C1 dan C2 menggunakan kode cookies khusus. Implementasi *load balancer* berfungsi sebagai pengalihan *traffic* yang tinggi sebagai bentuk *drop packet* ketika trafik yang masuk ke *controller* melebihi batas *threshold* yang terindikasi sebagai serangan. Ketika paket telah di *drop*, *traffic* akan dialihkan ke C0 dengan menghapus aturan *flow table* dengan kode cookie khusus tersebut. Sedangkan arsitektur *single controller* seperti Gambar 4.



Gambar 4 . Desain arsitektur perancangan sistem single-controller

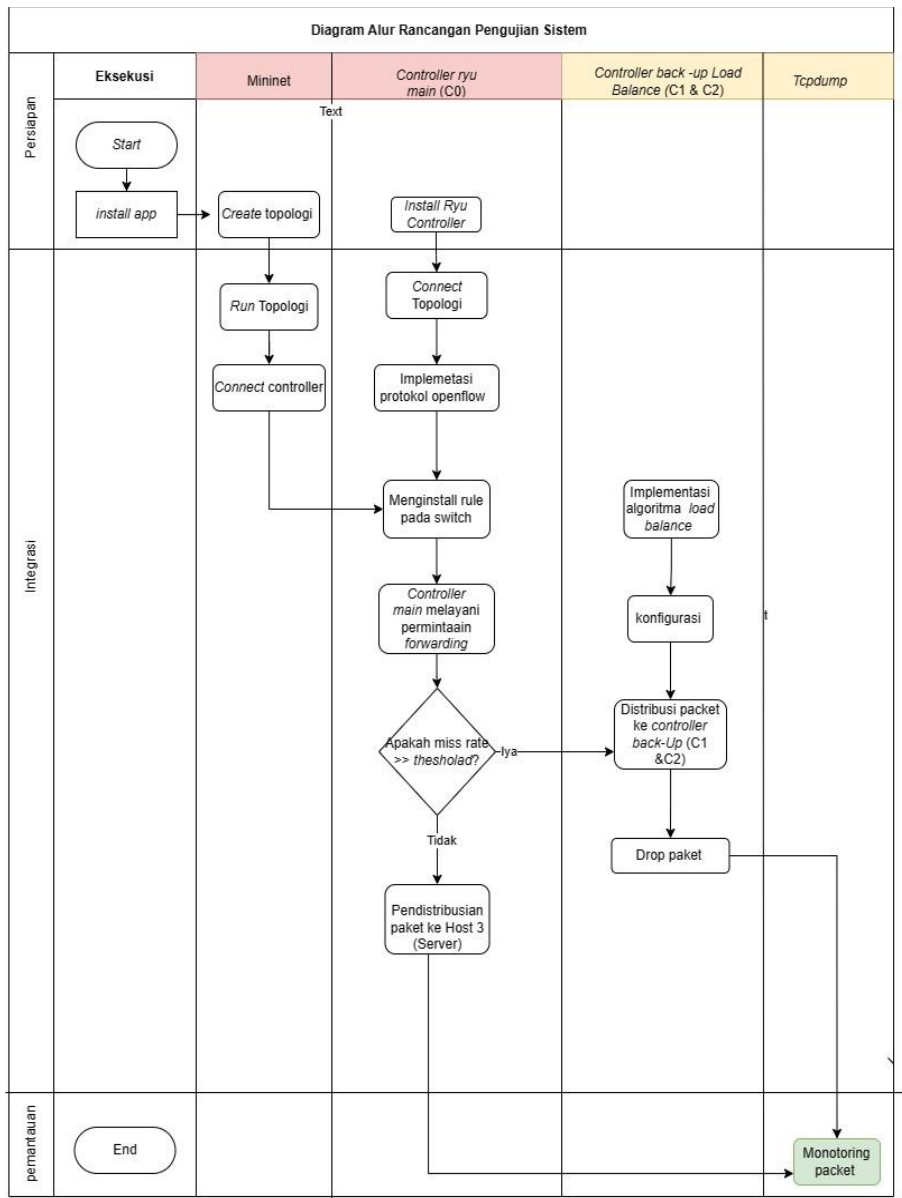
Berdasarkan Gambar 4 desain topologi *single controller* dengan menggunakan 2 *switch*, dengan 1 *controller* dan 3 *host*. C0 pada arsitektur *single controller* pada perangkat lunak berperan dalam mengelola dan mengkoordinasi seluruh sistem. Hal ini menjadi satu entitas utama yang bertanggung jawab atas pengambilan keputusan, pemrosesan 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* sebagai berikut yaitu *host 1* sebagai *user* dengan pengiriman paket normal. *Host 2* sebagai penyerang untuk pengiriman paket serangan, dan *host 3* sebagai *server*.

d. Implement Phase

Tahap Implementasi menggunakan *software* dan *hardware* yang diinstall pada sistem operasi ubuntu 22.0.1 untuk melakukan konfigurasi yang sesuai dengan rancangan sistem penelitian seperti pada Gambar 2 Pada tahap ini sistem yang telah dirancang akan diterapkan untuk mengetahui kelayakan perancangan sistem penelitian. Implementasi dimulai setelah dilakukan instalasi penunjang kebutuhan penelitian. Kemudian dilanjut dengan membuat konfigurasi topologi pada emulator

<http://sistemasi.ftik.unisi.ac.id>

mininet sesuai dengan desain topologi seperti pada Gambar 3 dan 4. Adapun diagram alur implementasi untuk melakukan pengujian sistem *multi-controller* berbasis *load balancer* seperti Gambar 5.



Gambar 5. Diagram alur rancangan pengujian sistem multi-controller berbasis load balancer

Gambar 5 Menjelaskan diagram implementasi untuk melakukan pengujian sistem pada arsitektur *multi-controller* berbasis *load balancer*. Proses eksperimen serangan dilakukan pada saat paket yang masuk nilai *miss-rate* terdeteksi melebihi *threshold*. Ketika paket yang masuk *miss-rate* nya melebihi dan mengakibatkan *traffic* yang tinggi. Maka, *controller main* mengalihkan packet pada 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 bentuk pengalihan layanan. Ketika packet yang masuk serangan maka yang mengalami *down* ialah *controller backup* bukan *controller main*. Sedangkan pengujian sistem *single controller* dengan menggunakan 1 *controller*, 2 *switch*, dan 3 *host*. Ketika paket yang masuk melebihi *threshold* yang terindikasi serangan maka, *controller main* yang berperan sebagai *controller* utama menangani paket tersebut yang bertanggung jawab mengambil keputusan dengan mendrop atau meneruskan paket ke tujuan. Ketika paket yang masuk

terindikasi sebagai serangan maka, *controller* C0 yang berperan sebagai *controller* utama dapat mengalami *down* karena terindikasi sebagai satu titik kegagalan dalam jaringan SDN.

Setelah melakukan implementasi dengan melakukan konfigurasi pada *hardware* dan *software*. Selanjutnya dilakukan pengujian untuk mendapatkan hasil pengukuran berdasarkan parameter penelitian. Pengukuran menggunakan skenario eksekusi pengujian berdasarkan Tabel 1 digunakan untuk melakukan analisis dan evaluasi parameter *responsivitas*, dan *high availability controller* dalam menangani paket pada jaringan SDN yang telah dibuat pada emulator mininet.

e. Operate Phase

Setelah tahap Pengukuran berdasarkan skenario pengujian maka, dilakukan tahap validasi sistem. Hasil validasi pengukuran didapatkan menggunakan metode analisis statistik menggunakan interval confident. Hasil validasi pengukuran menggunakan metode statistik dengan melakukan perbandingan terhadap data hasil pengukuran dengan 5 kali pengukuran confidence interval. Hal ini bertujuan untuk mendapatkan rerata hasil validasi *response controller* dan *high availability* pada efektivitas strategis manajemen terhadap ancaman serangan DDoS pada jaringan. Validasi dilakukan untuk menilai sejauh mana sistem dapat efektif dan efisiensi dalam menangani serangan DDoS. Hal ini melibatkan uji coba skenario berdasarkan Tabel 1 untuk mengukur parameter. Validasi hasil pengukuran terhadap perancangan sistem keamanan jaringan SDN pada parameter *responsivitas* dan *availability* berdasarkan perbandingan teoritis dari referensi peneliti sebelumnya, literatur, buku, dan jurnal terkait teori pengamanan sistem jaringan yang mengacu pada standar internasional.

f. Optimize Phase

Hasil pengukuran berdasarkan skenario Tabel 1 dilakukan analisis dan evaluasi untuk mengetahui pengaruh arsitektur *load balancer* dan *multi-controller* pada pengurangan beban pada *controller* dalam mengamankan jaringan dari serangan DDoS. Analisis pengukuran parameter pengujian dilakukan untuk menilai sejauh mana sistem dapat bekerja dalam mempertahankan *controller* pada jaringan SDN 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 dari jurnal, buku serta penelitian terdahulu untuk membandingkannya dalam mendapatkan hasil yang optimal. Evaluasi dilakukan untuk mengoptimalkan identifikasi terhadap kinerja sistem yang telah dibangun dalam memaksimalkan terhadap penanganan serangan DDoS untuk penelitian yang akan datang.

Teknik pengumpulan data menggunakan akuisisi dari hasil pengujian *experiment* lalu lintas trafik secara *real time* pada sistem. Sistem yang dibuat dilakukan pengujian secara langsung berdasarkan parameter penunjang penelitian untuk dilakukan analisis dan evaluasi kelayakan sistem penelitian. Metode pengumpulan data yang digunakan menggunakan metode observasi skenario. Yaitu hasil pengujian dari skenario penelitian yang dibuat pada emulator mininet dengan arsitektur *multi-controller* berbasis *load balancer*. Hasil lalu lintas trafik dari pengujian di *capture* untuk akuisisi data pada *tcpdump* dalam monitoring *traffic* pada lalu lintas sistem penelitian. Kemudian hasil pengujian dilakukan pengukuran berdasarkan parameter pengujian yaitu *responsivitas* dan *high availability*. Hasil pengukuran dilakukan analisis dan evaluasi kelayakan sistem berdasarkan acuan dari referensi dari buku, jurnal yang berkaitan dengan penelitian terdahulu. Data hasil pengukuran yang dianalisis dan evaluasi dengan 5 kali percobaan mengambil data untuk mendapatkan *confidence interval*. Hasil pengukuran dalam evaluasi menggunakan metode statistik data akuisisi dari *experiment* penelitian.

4 Hasil dan Pembahasan

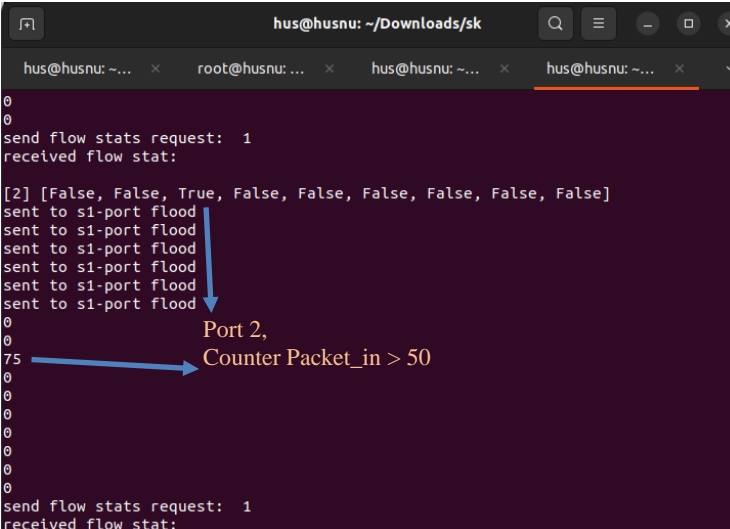
Rancangan sistem penelitian ini mencangkup tiga aspek utama pertama, implementasi *multi-controller* dan *load balancer* dilakukan untuk pengalihan trafik serangan DDoS. Kedua, dilakukan analisis *responsivitas* dari *controller* utama, membuktikan kemampuannya dalam mengatasi tekanan serangan dan menjaga ketersediaan jaringan. Ketiga, analisis kualitas jaringan SDN juga dilakukan, memberikan gambaran lengkap tentang bagaimana arsitektur tersebut mampu menjaga performa jaringan saat menghadapi serangan DDoS. Hasil penelitian ini membawa pemahaman yang mendalam tentang efektivitas dan keandalan arsitektur *multi-controller* dan *load balancer* dalam konteks perlindungan terhadap serangan DDoS.

4.1 Implementasi pengalihan trafik serangan DDoS

Implementasi pengalihan trafik serangan DDoS melibatkan mekanis yang dijalankan oleh controller utama. Mekanisme ini diawali dengan deteksi pada parameter yaitu melalui packet in yang diterima dari switch. Jika nilai packet_in dari openflow switch mencapai Tingkat yang cukup tinggi, hal ini menunjukkan adanya potensi serangan DDoS dari switch. Deteksi packet dilakukan berdasarkan nomor port, sehingga mitigasi serangan DDoS dapat diimplementasikan dengan fokus pada basis port, dalam menjaga agar pengalihan trafik tidak mengganggu trafik normal di port lainnya.

Parameter pengalihan trafik serangan menjadi kunci implementasi dalam mendrop packet serangan. Dalam skenario pengujian, background trafik ditetapkan pada 1 paket per detik(pps), dan threshold peralihan dianggap setara dengan 10 kali lipat trafik background, yaitu 10 pps. Deteksi di controller dilakukan setiap 5 detik, sehingga nilai counter packet in menjadi 10 kali lipat trafik normal atau setara dengan 50. Jika nilai counter packet in dari suatu port melebihi 10 kali lipat trafik normal yaitu 50, maka seluruh trafik dari port tersebut akan dialihkan ke controller cadangan. Dengan penerapan ini controller utama dapat terhindar dari potensi serangan DDoS, sehingga tetap mampu melayani trafik dari port dan switch yang lainnya tanpa terganggu dari serangan yang membahayakan stabilitas dan kinerja jaringan.

Skenario pengujian ini memungkinkan pengujian efektivitas dari implementasi pengalihan trafik serangan DDoS. Dengan background trafik yang diatur pada 1 pps, controller dapat mengidentifikasi potensi serangan dan merespon secara cepat. Threshold peralihan yang ditetapkan pada 10 kali lipat trafik normal memberikan ruang untuk mengakomodasi fluktuasi trafik. Dalam implementasi pengalihan trafik serangan DDoS menggunakan multi controller berbasis load balancer, mekanisme utama yang menjadi fokus adalah peran controller utama dalam mengenali dan mengatasi potensi serangan DDoS. Mekanisme ini berkaitan erat dengan parameter deteksi, dimana packet in dari switch menjadi indikator utama untuk mengidentifikasi perubahan trafik yang mencurigakan. Apabila nilai packet in dari openflow switch mencapai tingkat yang mencukupi, dapat diasumsikan bahwa potensi serangan DDoS berasal dari switch tersebut. Berikut bentuk pengalihan layanan pada controller main seperti pada Gambar 6.



```
hus@husnu: ~/Downloads/sk
0
0
send flow stats request: 1
received flow stat:
[2] [False, False, True, False, False, False, False, False, False]
sent to s1-port flood
sent to s1-port flood
sent to s1-port flood
sent to s1-port flood
sent to s1-port flood
sent to s1-port flood
0
0
Port 2,
Counter Packet_in > 50
75
0
0
0
0
0
0
0
0
0
0
send flow stats request: 1
received flow stat:
```

Gambar 6. Pengalihan layanan ke controller main

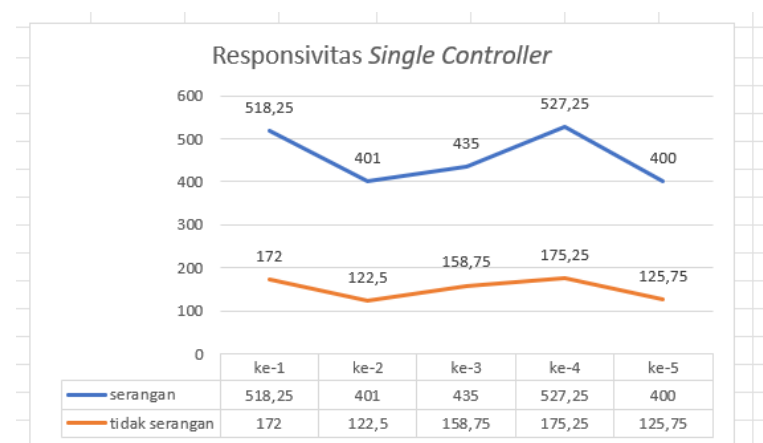
Gambar 6 menunjukkan pengalihan lalu lintas dari controller main ke controller backup dengan port aktif 2 dan nilai counter packet_in yang lebih besar dari 50 merupakan strategi untuk meningkatkan efisiensi dan ketersediaan sistem SDN. Ketika controller main mendeteksi bahwa jumlah paket masuk (packet_in) melebihi ambang batas 50, sistem memutuskan untuk mengalihkan lalu lintas ke controller backup. Pada controller backup, terdapat implementasi multi-controller berbasis load balancer yang bertugas mendistribusikan lalu lintas yang diterima secara merata ke beberapa controller backup yang ada dalam arsitektur SDN.

Penggunaan load balancer pada controller backup membantu mencegah terjadinya titik kegagalan tunggal dan meningkatkan ketersediaan layanan. load balancer yang diatur untuk mengarahkan lalu lintas melalui jalur yang telah diisolasi saat serangan terdeteksi mengalami

kenaikan *traffic* dalam menjaga sumber daya jaringan pada *controller* utama tetap efisien. Peningkatan pertahanan terhadap satu titik kegagalan dan skalabilitas tinggi terhadap pengontrolan diperlukan dalam menunjang keterbatasan dalam pemrosesan jaringan berskala besar. Latensi respon permintaan dapat dikurangi melalui *load balancer* dalam menghindari kelebihan beban pada jaringan. *Load balancer* berfungsi dalam meningkatkan kinerja *controller* dalam meminimalisir *downtime* untuk meningkatkan skalabilitas dan fleksibilitas.

4.2 Responsivitas controller utama

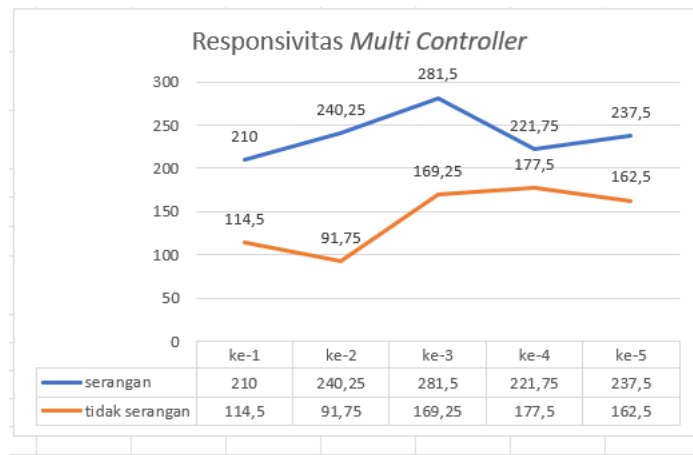
Responsivitas *controller* pada suatu sistem sangat penting untuk memastikan ketersediaan (*availability*) dan keamanan dari sistem tersebut. Dalam skenario *Single controller* Normal, hasil pengukuran responsivitas dilakukan melalui serangkaian percobaan dengan mengamati nilai respons keluar (*Response_out*) dan response masuk (*Response_int*) pada beberapa *port* yang diukur bersama dengan selisihnya (*Delta*). Hasil percobaan dilakukan menggunakan 5x percobaan yang menunjukkan bahwa nilai rata-rata delta (*RC*) pada *Single controller* Normal berkisar antara 172 hingga 125,75 seperti pada Gambar 7. Hal ini mengindikasikan bahwa sistem mampu memberikan respons yang konsisten dalam kondisi normal, dengan variasi yang relatif antara beberapa kali percobaan.



Gambar 7. Grafik responsivitas *single controller*

Di sisi lain, pada skenario *Single controller Attack*, responsivitas *controller* diuji ketika sistem menghadapi serangan. Hasil percobaan menunjukkan bahwa nilai rata-rata delta (*RC*) pada *Single Attack* berkisar antara 518,25 hingga 400 seperti pada Gambar 7. Dapat diamati bahwa responsivitas sistem mengalami peningkatan delta dibandingkan dengan skenario paket normal, yang dapat diartikan sebagai adanya gangguan atau peningkatan beban pada sistem dalam skenario serangan. Nilai *RC* yang lebih tinggi pada *Single controller* pada skenario *attack* menunjukkan adanya potensi serangan yang dapat mempengaruhi *responsivitas* sistem. Secara lebih spesifik, pada *Single controller* kondisi normal, variasi nilai delta yang relatif kecil menunjukkan bahwa sistem mampu menjaga responsivitasnya. Namun, pada *Single controller Attack*, terlihat peningkatan delta dibandingkan skenario paket normal. Hal ini menandakan bahwa *respons* sistem dapat terpengaruh oleh serangan, yang dapat berdampak pada ketersediaan dan keamanan.

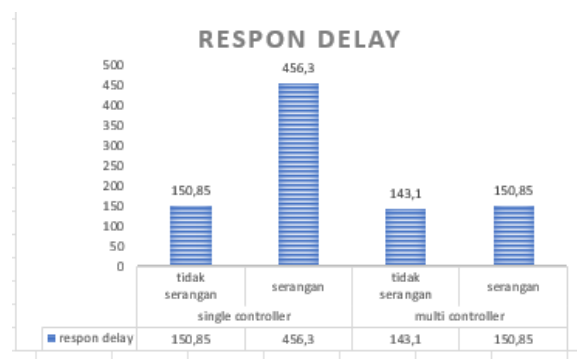
Perbedaan responsivitas pada skenario normal dan serangan pada *controller* dapat dijelaskan oleh sejumlah faktor yang mempengaruhi performa sistem dalam dua kondisi tersebut. Skenario normal, sistem beroperasi dalam keadaan yang stabil dan tidak mengalami gangguan yang signifikan. Oleh karena itu, responsivitas *controller* cenderung konsisten dengan variabilitas delta yang relatif kecil antar percobaan. Peningkatan delta pada skenario serangan dapat menyebabkan peningkatan beban pada sistem. Serangan dapat menciptakan lalu lintas yang lebih tinggi dan meningkatkan permintaan terhadap sumber daya, sehingga *controller* harus bekerja lebih keras untuk memproses permintaan tersebut. Peningkatan beban ini dapat tercermin dalam peningkatan delta, yang mencerminkan perbedaan antara respons keluar dan respons internal pada sistem. Sedangkan *responsivitas multi controller* seperti Gambar 8.



Gambar 8. Grafik responsivitas *multi controller*

Gambar 8 menjelaskan hasil pengukuran *responsivitas* pada arsitektur *multi controller*. Berdasarkan skenario tidak serangan *multi controller* menunjukkan variabilitas delta yang relatif kecil pada beberapa kali percobaan. Hal ini terlihat pada percobaan 1 nilai delta sebesar 114,5 yang mengindikasikan sistem mampu memberikan respons yang stabil pada *controller*. Rata-rata delta berkisar antara 114,5 hingga 162,5. Hal ini menunjukkan bahwa *multi-controller* mampu mengelola lalu lintas dengan baik dalam menjaga ketersediaan jaringan dalam merespons perubahan dengan efisien.

Pada skenario *multi controller* dengan *traffic* serangan hasil percobaan menunjukkan peningkatan yang signifikan pada nilai delta dalam beberapa kali percobaan. Peningkatan yang signifikan pada nilai delta menunjukkan penurunan responsivitas yang nyata dengan nilai rata rata delta berkisar antara 210 hingga 237,5. Serangan mampu membebani sistem dan mengganggu distribusi lalu lintas antara *controller*. Serangan pada *controller* membuat sistem menjadi lambat dalam merespons dan lalu lintas yang lebih tinggi. Berikut perbandingan nilai responsivitas *controller* pada *single* dan *multi controller* seperti pada Gambar 9.



Gambar 9. Hasil perbandingan *responsivitas controller*

Gambar 9 menunjukkan perbandingan *responsivitas controller*. Dalam skenario *Single Controller*, terlihat bahwa nilai rata-rata delta (RC) pada *Single Normal* berkisar antara 150,85, sementara pada *Single Attack* mencapai 456,3. Ini menunjukkan adanya peningkatan yang signifikan dalam responsivitas sistem ketika dihadapkan pada serangan DDoS. Sebaliknya, pada skenario *Multi Controller*, responsivitas pada *Multi controller* pada kondisi Normal relatif lebih baik, dengan nilai RC berkisar 143. Namun, saat terjadi serangan pada *Multi Controller*, responsivitas jaringan meningkat dengan nilai RC mencapai 150,85.

Response delay controller melakukan pengukuran waktu yang dibutuhkan oleh sistem kontrol dalam merespons terhadap kondisi serangan dan tidak serangan. Pada kondisi serangan, *multi controller* memiliki *response delay* sebesar 238,2 sementara *single controller* memiliki *response delay* yang lebih tinggi yaitu 456,3. Hal ini mengindikasikan bahwa *multi controller* lebih baik dalam

<http://sistemasi.ftik.unisi.ac.id>

menanggapi serangan dengan menjaga *response delay* yang lebih rendah dibandingkan dengan *single controller*. sedangkan dalam kondisi tidak serangan, *multi controller* tetap mempertahankan perform yang lebih baik dengan *respons delay* sebesar 143,1 semestara *single controller* memiliki *respons* sebesar 150,85. Meskipun *response delay* antara keduanya tidak sebesar kondisi serangan, *multi controller* masih menunjukkan keunggulan dalam menjaga *response delay* tetap rendah dalam kondisi normal.

Peningkatan *respons delay* pada *single controller* yang lebih besar dalam kondisi serangan menunjukkan bahwa sistem lebih rentan terhadap gangguan. Sebaliknya *multi controller* menunjukkan ketangguhan yang lebih baik dalam mengatasi serangan dengan mempertahankan *response delay* yang relatif baik. *Respon delay* yang rendah menunjukkan kemampuan sistem dalam merespons perubahan dengan cepat. Berdasarkan hasil pengukuran, *multi controller* memenuhi kriteria lebih baik dari pada *single controller* baik dalam kategori paket serangan atau normal, yang menunjukkan kemampuan adaptasi yang lebih baik terhadap gangguan. pada *Multi Controller*, distribusi beban yang merata memberikan ketersediaan yang baik pada *multi controller*. Hal ini terjadi karena *multi controller* mampu melakukan distribusi *traffic* jika terindikasi sebagai serangan. Sehingga Ketika paket serangan yang masuk pada jaringan *controller* yang terindikasi down ialah *controller backup* bukan *controller main*. Sedangkan hasil pengukuran berdasarkan *Availability single controller* dan *multi controller* seperti pada Tabel 2.

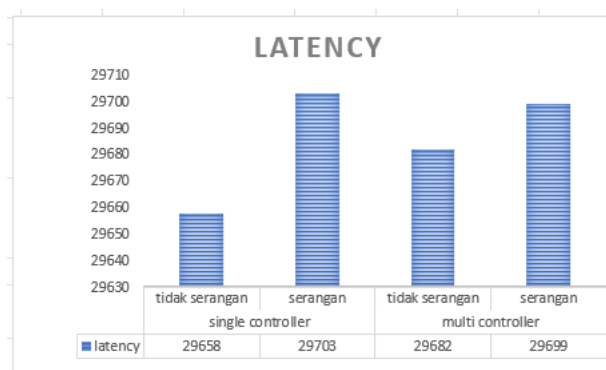
4.3 Analisis kualitas jaringan

Pengukuran performa kualitas jaringan pada skenario *single controller* dan *multi controller* befokus pada paaremtter *latency*, *throughput*, dan *jitter*. Kualitas aliran data dalam suatu jaringan yang mempengaruhi stabilitas layanan. *Jitter*, sebagai indikator fluktuasi waktu pengiriman data, memberikan gambaran tentang stabilitas transmisi. *Latency*, yang mengukur waktu yang dibutuhkan data untuk mencapai tujuan, menjadi parameter penting dalam mengevaluasi responsivitas jaringan. Selain itu, *throughput* mencerminkan sejauh mana jaringan mampu mengirim dan menerima data dengan efisien. Dalam konteks *single controller* dan *multi controller*, hasil pengukuran seperti Tabel 2 berikut:

Tabel 2. hasil pengukuran *availability single controller* dan *multi controller*

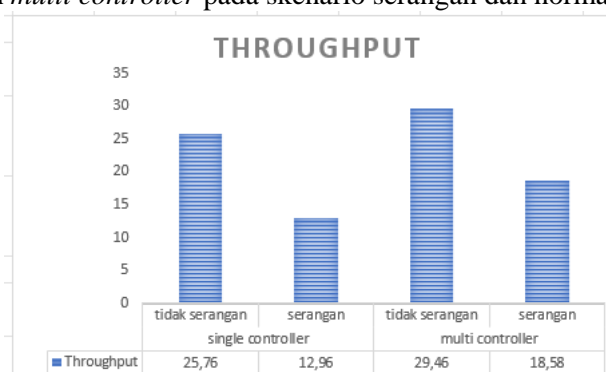
	<i>latency</i>	<i>Throughput</i>	<i>Jitter</i>
<i>Single controller</i>			
Serangan	29703 ms	12,96 Gbits/sec	0,022 ms
Normal		25,76	
	29658 ms	Gbits/sec	0,007 ms
<i>Multi controller</i>			
Serangan	29699 ms	18,58 Gbits/sec	0,0668 ms
Normal	29682 ms	29,46 Gbits/sec	0,015 ms

Tabel 2 menunjukkan nilai *availability* pada *single controller* dan *multi controller* dengan mempertimbangkan *latency*, *jitter* dan *throughput* dalam kondisi serangan dan kondisi normal. *Availability* adalah ukuran sejauh mana sistem atau layanan dapat diakses dan berfungsi sesuai dengan kebutuhan. Pada umumnya, *availability* diukur dalam persentase waktu di mana sistem beroperasi dengan baik. *Latency* pada normal pada *multi controller* adalah 29682 ms. Nilai ini mengindikasikan waktu yang dibutuhkan untuk pemrosesan dan respons sistem dalam kondisi normal. Semakin rendah nilai *latency*, semakin cepat sistem merespons permintaan. Grafik nilai perbandingan *latency* pada *single* dan *multi controller* seperti pada Gambar 10.



Gambar 10. Hasil *latency single controller*

Latency single controller seperti Gambar 10 menunjukkan nilai paket normal sebesar 29658 ms. Nilai ini menunjukkan keterlambatan saat menjalankan operasi normal dibandingkan *multi controller*. *Latency* yang rendah pada kondisi normal mengindikasikan efisiensi sistem dalam memberikan *respons* tanpa adanya ancaman. Dalam skenario serangan *Multi controller* menunjukkan *latency* sebesar 29699 ms, sedangkan *single controller* memiliki *latency* sebesar 29703 ms. Meskipun perbedaannya cukup kecil, hasil ini menggambarkan bahwa *Multi Controller* mungkin lebih efisien dalam menanggapi serangan dengan sedikit penurunan *latency* dibandingkan dengan *Single Controller*. kedua jenis model arsitektur menunjukkan sedikit keunggulan dengan mempertahankan *latency* yang sedikit lebih rendah dibandingkan dengan *single controller*. *latency* yang rendah mencerminkan kemampuan sistem untuk merespons inputan dalam kondisi yang cepat. Perbandingan *throughput* pada *single* dan *multi controller* pada skenario serangan dan normal seperti Gambar 11.



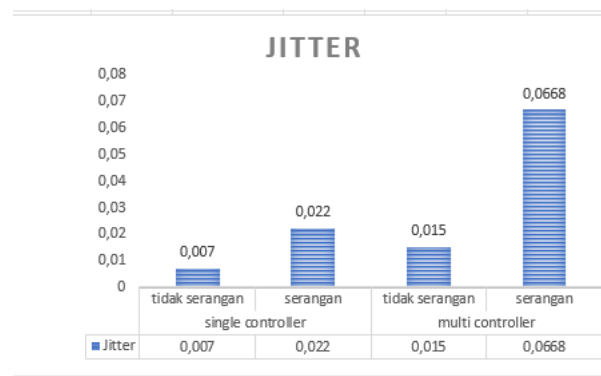
Gambar 11. *Throughput*

Nilai *throughput single controller* pada Gambar 11 dalam kondisi normal yaitu 25,76 Gbits/sec. Sedangkan pada *multi controller* nilainya 29,46 Gbits/sec. Ini menunjukkan kemampuan sistem dalam mentransfer data dalam kondisi paket normal. Nilai *throughput* yang tinggi pada kondisi normal mengindikasikan efisiensi sistem dalam mengelola lalu lintas data tanpa adanya gangguan. *Throughput* pada serangan pada *single controller* adalah 12,96 GBytes/sec. Ini mencerminkan jumlah data yang dapat ditransfer dalam satu waktu saat terjadi serangan. Penurunan *throughput* pada serangan dapat menandakan adanya penghalangan dalam distribusi data atau pemrosesan yang disebabkan oleh upaya serangan.

Pada kondisi serangan, *multi controller* menunjukkan *throughput* sebesar 18,58 Gbit/sec. sementara *single controller* menunjukkan *throughput* yang lebih rendah sebesar 12,96 Gbits/sec. Hasil ini menunjukkan bahwa *multi controller* lebih mampu menjaga *throughput* yang tinggi bahkan dalam kondisi serangan yang menandakan efisiensi dan skalabilitas dalam mengelola lalu lintas data saat terjadi gangguan. Dalam skenario tidak seorangpun *multi controller* tetap mempertahankan keunggulan yang menandakan kapabilitasnya untuk menangani beban trafik yang lebih besar dalam kondisi normal. *Multi Controller* memiliki performa *throughput* yang lebih baik daripada *Single Controller* dalam kedua kondisi, khususnya saat menghadapi serangan. Hal ini bisa disebabkan oleh efisiensi manajemen sumber daya, strategi penanganan lalu lintas, atau adaptasi sistem yang lebih baik pada *Multi Controller*. *throughput* yang tinggi diinginkan karena mencerminkan kemampuan

<http://sistemasi.ftik.unisi.ac.id>

sistem untuk mengelola dan mentransfer data dengan efisien. Hasil perbandingan grafik Jitter pada skenario *single* dan *multi controller* dalam situasi serangan dan tidak serangan seperti Gambar 12.



Gambar 12. Jitter

Gambar 12 menunjukkan kondisi serangan jitter pada arsitektur *Multi Controller* sebesar 0,0668 ms, sedangkan *Single Controller* sebesar 0,022 ms. Hasil ini menunjukkan bahwa *Multi Controller* mengalami variasi waktu yang lebih besar dalam kondisi serangan dibandingkan dengan *Single Controller*. *Jitter* yang tinggi dapat menunjukkan ketidakstabilan dalam mentransmisikan data, dan hasil ini dapat diartikan bahwa *Multi Controller* mungkin memiliki tantangan lebih besar dalam menjaga konsistensi waktu transmisi saat terjadi serangan. kondisi Tidak Serangan, *Multi Controller* memiliki *jitter* sebesar 0,015 ms, sedangkan *Single Controller* memiliki *jitter* sebesar 0,007 ms. Perbedaan *jitter* antara keduanya tetap, meskipun kurang signifikan dibandingkan dengan kondisi serangan. Hal ini menunjukkan bahwa *Multi Controller* mampu menjaga stabilitas waktu transmisi dengan baik, meskipun masih mengalami variasi yang lebih besar daripada *Single Controller* dalam kondisi normal, *jitter* yang rendah diinginkan karena mencerminkan kemampuan sistem untuk mengelola variasi waktu dengan efisien.

Pengamanan jaringan SDN dari serangan DDoS merupakan salah satu aspek penting dalam menjaga ketersediaan dan performa jaringan. Serangan DDoS menyebabkan penurunan *responsivitas controller* yang mengakibatkan gangguan layanan dan merugikan operasional jaringan SDN. Dalam rancangan ini, penggunaan *multi controller* menjadi solusi yang diajukan peneliti untuk meningkatkan ketersediaan dan ketahanan jaringan. Perananan beberapa *controller* mengakibatkan pendistribusian beban kerja yang merata untuk mengurangi *overloading* pada satu titik *controller*. Integrasi *load balancer* pada arsitektur *multi controller* membantu dalam mendistribusikan lalu lintas secara merata ke berbagai *controller*, dalam mencegah *buffer overload* *openflow switch* dalam meminimalisir kemacetan untuk menunggu respons *controller* yang berurutan.

5 Kesimpulan

Penggunaan *load balancer* pada skenario *multi-controller* mampu meningkatkan ketersediaan dan responsivitas sistem SDN. Distribusi yang merata dari beban lalu lintas pada setiap controller tidak hanya mengurangi risiko titik kegagalan tunggal, tetapi juga meningkatkan ketersediaan layanan secara keseluruhan. Strategi pengalihan trafik serangan DDoS memberikan keunggulan dalam merespons ancaman, dan pengamatan terhadap *latency*, *throughput*, dan *jitter* menunjukkan performa lebih baik pada multi-controller, terutama dalam kondisi serangan. Analisis *responsivitas controller main* mengkonfirmasi bahwa *multi-controller* memberikan respons delay yang lebih rendah sebesar 66,51% jika dibandingkan dengan skenario *single controller* yang lebih lama yaitu 202,49%. *Latency* yang rendah, *throughput* tinggi, dan *jitter* yang stabil mencerminkan kemampuan arsitektur ini dalam menjaga stabilitas, kecepatan, dan konsistensi dalam mentransmisikan data. *latency* pada multi controller mengalami penurunan yang sangat kecil sebesar 0,01% dibandingkan dengan single controller. Ini menunjukkan bahwa kinerja latency keduanya hampir serupa. Peningkatan *throughput* sebesar 43,21% dibandingkan dengan *Single Controller* pada saat serangan. *Jitter* pada *Multi Controller* mengalami peningkatan sebesar 204% dibandingkan dengan *Single Controller* pada saat serangan. Hal ini menunjukkan variasi yang lebih tinggi dalam waktu transmisi data pada *multi*

<http://sistemasi.ftik.unisi.ac.id>

controller selama serangan. Dengan demikian, penggunaan *load balancer* dalam skenario *multi-controller* menjadi solusi yang terukur untuk meningkatkan ketersediaan dan responsivitas sistem SDN, dengan pemantauan terus-menerus dan penerapan tindakan adaptif sebagai kunci keberhasilan dalam menghadapi tantangan serangan.

Ucapan Terima Kasih

Ucapan terima kasih yang tulus kepada BRIN cistitu yang telah memberikan kesempatan saya untuk belajar dan menggali ilmu pada penelitian ini. Serta bapak pembimbing Akbari Indra Basuki S.Si., M.T, dan dosen pembimbing bapak Galura Muhammad Suranegara, S.Pd., M.T atas dukungan, masukan serta arahan penuh pada penelitian ini. Kontribusi mereka telah menjadi pilar utama kesuksesan proyek ini dan saya sangat berterima kasih atas bimbingan dan inspirasi yang telah diberikan.

Referensi

- [1] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, "A Survey on Software Defined Networking With Multiple Controllers," *Journal of Network and Computer Applications*, vol. 103. Academic Press, pp. 101–118, Feb. 01, 2018. doi: 10.1016/j.jnca.2017.11.015.
- [2] M. Iqbal and M. Arif Ramadhan, "Analisa Quality of Service pada Jaringan Wireless Berbasis Software-Defined Network dengan Protokol Openflow Menggunakan Floodlight Controller," 2020.
- [3] E. Taherian-Fard, T. Niknam, R. Sahebi, M. Javidsharifi, A. Kavousi-Fard, and J. Aghaei, "A Software Defined Networking Architecture for DDoS-Attack in the Storage of Multimicrogrids," *IEEE Access*, vol. 10, pp. 83802–83812, 2022, doi: 10.1109/ACCESS.2022.3197283.
- [4] X. Li, Z. Fan, Y. Xiao, Q. Xu, and W. Zhu, "Improved Automated Graph And FCM Based DDoS Attack Detection Mechanism In Software Defined Networks," *Journal of Internet Technology*, vol. 20, no. 7, pp. 2117–2127, 2019, doi: 10.3966/160792642019122007010.
- [5] H. H. Saleh, I. A. Mishkal, and D. S. Ibrahim, "Controller Placement Problem In Software Defined Networks," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 27, no. 3, pp. 1704–1711, Sep. 2022, doi: 10.11591/ijeecs.v27.i3.pp1704-1711.
- [6] D. P. Harja, A. Rakhmatsyah, and M. A. Nugroho, "Implementasi untuk Meningkatkan Keamanan Jaringan Menggunakan Deep Packet Inspection pada Software Defined Networks," *Indonesian Journal on Computing (Indo-JC)*, vol. 4, no. 1, p. 133, Mar. 2019, doi: 10.21108/indojc.2019.4.1.286.
- [7] J. Prathima Mabel, K. A. Vani, and K. N. Rama Mohan Babu, "SDN Security: Challenges and Solutions," in *Lecture Notes in Electrical Engineering*, Springer Verlag, 2019, pp. 837–848. doi: 10.1007/978-981-13-5802-9_73.
- [8] A. K. Singh, S. Maurya, and S. Srivastava, "Varna-Based Optimization: A Novel Method For Capacitated Controller Placement Problem in SDN," *Front Comput Sci*, vol. 14, no. 3, Jun. 2020, doi: 10.1007/s11704-018-7277-8.
- [9] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task Offloading in Vehicular Edge Computing Networks: A Load-Balancing Solution," *IEEE Trans Veh Technol*, vol. 69, no. 2, pp. 2092–2104, Feb. 2020, doi: 10.1109/TVT.2019.2959410.
- [10] R. R. Zebari, S. R. M. Zeebaree, A. B. Sallow, H. M. Shukur, O. M. Ahmad, and K. Jacksi, "Distributed Denial of Service Attack Mitigation using High Availability Proxy and Network Load Balancing," in *3rd International Conference on Advanced Science and Engineering, ICOASE 2020*, Institute of Electrical and Electronics Engineers Inc., Dec. 2020, pp. 174–179. doi: 10.1109/ICOASE51841.2020.9436545.
- [11] I. M. Huda and I. M. Suartana, "Implementasi Intrusion Prevention System Untuk Mencegah Serangan DDOS pada Software Defined Network," *Journal of Informatics and Computer Science*, vol. 03, 2021.
- [12] S. Kaur, K. Kumar, N. Aggarwal, and G. Singh, "A Comprehensive Survey of DDoS Defense Solutions in SDN: Taxonomy, Research Challenges, and Future Directions," *Computers and Security*, vol. 110. Elsevier Ltd, Nov. 01, 2021. doi: 10.1016/j.cose.2021.102423.
- [13] T. Akhir, M. Kuliah, K. Informasi, J. El5241, and D. Pratama, "Serangan Ddos Pada Software Defined Network."
- [14] G. Li, X. Wang, and Z. Zhang, "SDN-Based Load Balancing Scheme for Multi-Controller Deployment," *IEEE Access*, vol. 7, pp. 39612–39622, 2019, doi: 10.1109/ACCESS.2019.2906683.

- [15] M. S. Tok and M. Demirci, "Security analysis of SDN Controller-Based DHCP Services and Attack Mitigation with DHCPguard," *Comput Secur*, vol. 109, Oct. 2021, doi: 10.1016/j.cose.2021.102394.
- [16] K. Kalkan, L. Altay, G. Gür, and F. Alagöz, "JESS: Joint Entropy-Based DDoS Defense Scheme in SDN," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2358–2372, Oct. 2018, doi: 10.1109/JSAC.2018.2869997.
- [17] C. Bhatt, V. Sihag, G. Choudhary, P. V. Astillo, and I. You, "A Multi-Controller Authentication Approach for SDN," in *2021 International Conference on Electronics, Information, and Communication, ICEIC 2021*, Institute of Electrical and Electronics Engineers Inc., Jan. 2021. doi: 10.1109/ICEIC51217.2021.9369825.
- [18] P. Valizadeh and A. Taghinezhad-Niar, "DDoS Attacks Detection in Multi-Controller Based Software Defined Network," in *2022 8th International Conference on Web Research, ICWR 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 34–39. doi: 10.1109/ICWR54782.2022.9786246.
- [19] Y. C. Wang and E. J. Chang, "Cooperative Flow Management in Multi-domain SDN-based Networks with Multiple Controllers," in *HONET 2020 - IEEE 17th International Conference on Smart Communities: Improving Quality of Life using ICT, IoT and AI*, Institute of Electrical and Electronics Engineers Inc., Dec. 2020, pp. 82–86. doi: 10.1109/HONET50430.2020.9322815.
- [20] K. Mulligan, J. T. Grant, S. T. Mockabee, and J. Q. Monson, "Response Latency Methodology for Survey Research: Measurement and Modeling Strategies," *Political Analysis*, vol. 11, no. 3, pp. 289–301, 2003, doi: 10.1093/pan/mpg004.
- [21] A. Saputra, M. Akbar, I. Solikin, and M. Kom, "Pengembangan Jaringan Wireless Local Area Network (Wlan) Menggunakan Metode PPDIOO (Studi kasus : SMK N 1 Indralaya Utara)."
- [22] R. R. Zebari, S. R. M. Zeebaree, A. B. Sallow, H. M. Shukur, O. M. Ahmad, and K. Jacksi, "Distributed Denial of Service Attack Mitigation using High Availability Proxy and Network Load Balancing," in *3rd International Conference on Advanced Science and Engineering, ICOASE 2020*, Institute of Electrical and Electronics Engineers Inc., Dec. 2020, pp. 174–179. doi: 10.1109/ICOASE51841.2020.9436545.