IJAI (Indonesian Journal of Applied Informatics)
Vol. 10 No. 1 Tahun 2025 pISSN: 2548-3846, eISSN: 2598-5981

Design and Implementation of SDN Firewall Using Pox Controller and Open vSwitch

Eka Stephani Sinambela[1]*, Jesika L. Manurung[1], Grace Agnes Kesya[1], Harli J. Sinabutar[1], Istas Pratomo Manalu[1], Gerry Italiano Wowiling[1], Frengki Simatupang[1], Marojahan M.T Sigiro[1]
Program Studi D3 Teknologi Komputer, Fakultas Vokasi, Institut Teknologi Del, Indonesia
*Email: ekastephanisinambela@gmail.com

Info Artikel

Kata Kunci :
firewall, flow table, open vswitch, pox controller, software-defined networking

Keywords :
firewall, flow table, open vswitch, pox controller, software-defined networking

Tanggal Artikel
Dikirim : 25 September 2025
Direvisi : 24 Desember 2025
Diterima : 30 Desember 2025

Abstrak

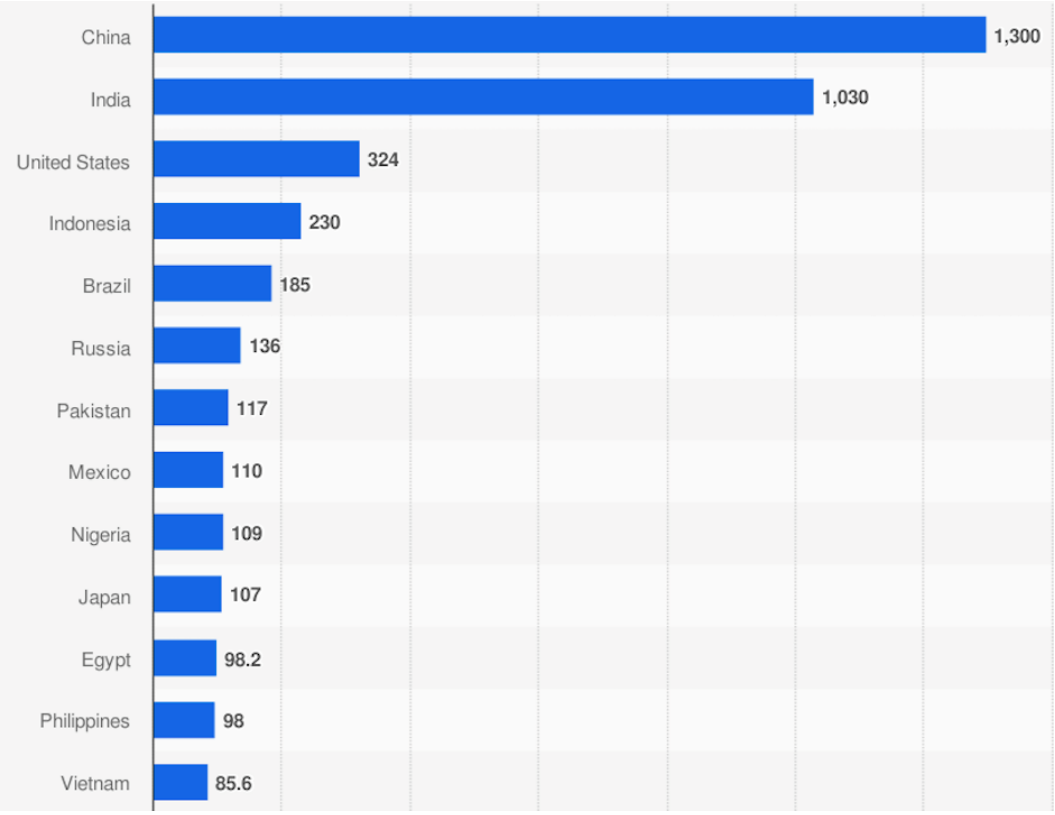
Perkembangan *Software-Defined Networking* (SDN) menghadirkan paradigma baru dalam pengelolaan jaringan melalui pemisahan antara *control plane* dan *data plane*, yang memungkinkan penerapan mekanisme keamanan jaringan secara terpusat dan terprogram. Salah satu mekanisme keamanan penting dalam SDN adalah Firewall berbasis *flow rules*. Penelitian ini mengimplementasikan Firewall berbasis SDN dengan memanfaatkan POX Controller sebagai control plane dan *Open vSwitch* (OvS) sebagai *data plane*. Evaluasi dilakukan melalui skenario pengujian konektivitas yang melibatkan komunikasi antara *controller*, OvS, dan beberapa host dalam jaringan. Hasil pengujian menunjukkan bahwa seluruh skenario berjalan sesuai dengan aturan Firewall yang dirancang. Secara kuantitatif, mekanisme pemfilteran trafik berbasis IP Address mencapai tingkat keberhasilan 100%, ditunjukkan oleh keberhasilan pemblokiran akses dan tetap terjaganya konektivitas pada jalur yang diperbolehkan. Hasil ini membuktikan bahwa integrasi POX Controller dan OvS efektif digunakan sebagai Firewall sederhana berbasis SDN serta berpotensi dikembangkan untuk mekanisme keamanan yang lebih kompleks.

Abstract

The development of Software-Defined Networking (SDN) introduces a new paradigm in network management by separating the control plane from the data plane, enabling centralized and programmable network security mechanisms. One essential security mechanism in SDN is a firewall based on flow rules. This study implements an SDN-based firewall using the POX Controller as the control plane and Open vSwitch (OvS) as the data plane. The system is evaluated through connectivity testing scenarios involving communication between the controller, OvS, and multiple hosts. Experimental results show that all test scenarios operate according to the defined firewall rules. Quantitatively, the IP address–based traffic filtering mechanism achieves a 100% success rate, as indicated by successful blocking of unauthorized access while maintaining permitted network connectivity. These results demonstrate that the integration of POX Controller and OvS is effective as a simple SDN-based firewall and has the potential to be extended to support more advanced network security mechanisms.

1. PENDAHULUAN

Seiring dengan perkembangan zaman, teknologi mengalami kemajuan yang signifikan. Salah satu bentuk perwujudan kemajuan tersebut adalah munculnya teknologi Internet yang digunakan dalam kehidupan Masyarakat secara massif. Internet berfungsi sebagai sarana penyiaran secara global, mekanisme distribusi informasi, serta media interaktif yang memungkinkan kolaborasi antara individu melalui jaringan komputer tanpa Batasan geografis [1], [2]. Keberadaan internet mencerminkan hasil dari investasi berkelanjutan dan komitmen terhadap penelitian serta pengembangan struktur informasi [3], [4]. Berbeda dengan situasi sebelumnya, Masyarakat memperoleh informasi hanya melalui media tradisional seperti surat kabar, namun kini akses terhadap informasi dapat dilakukan dengan lebih cepat dan efisien melalui eksistensi Internet [5], [6].



Gambar 1. Grafik penggunaan internet di dunia [7]

Menurut data yang dipublikasikan oleh Statista pada 2024 pada Gambar 1, China menempati posisi pertama sebagai negara dengan jumlah pengguna Internet terbanyak di dunia, yaitu sekitar 1,3 Miliar pengguna [7]. Angka ini lebih dari tiga kali lipat jumlah pengguna di Amerika Serikat yang berada di peringkat ketiga dengan sekitar 324 juta pengguna. Selain itu, China, India, United States dan Indonesia masing-masing memiliki lebih dari 200 juta pengguna internet, menjadikannya empat negara dengan basis pengguna internet terbesar secara global [7], [8]. Secara keseluruhan, distribusi pengguna internet terbesar terdapat di Kawasan Asia, menunjukkan bahwa pertumbuhan internet di benua tersebut lebih pesat dibandingkan kawasan lain [9]. Jaringan komputer merupakan inovasi yang memunculkan Internet, dengan gambaran sekumpulan perangkat yang saling terhubung dengan tujuan untuk berbagi sumber daya, seperti server, printer, maupun perangkat keras lainnya, serta untuk melakukan pertukaran data digital dan komunikasi elektronik [10]. Konektivitas antar perangkat dalam jaringan dapat diwujudkan melalui berbagai media transmisi, seperti kabel, saluran telepon, gelombang radio, satelit, maupun serat *optic* [11].

Dalam praktik penyediaan layanan, kualitas akses internet melalui infrastruktur jaringan komputer menjadi fundamental yang harus diperhatikan. Oleh karena itu, Pembangunan jaringan yang andal memerlukan perencanaan yang mempertimbangkan *Quality of Service* (QoS). QoS didefinisikan sebagai kemampuan jaringan dalam mengelola lalu lintas data sehingga kebutuhan layanan dari aplikasi dan pengguna tertentu terpenuhi sesuai dengan kebijakan jaringan [12], [13]. Penelitian menunjukkan bahwa mekanisme QoS yang baik berkontribusi terhadap peningkatan performa lalu lintas jaringan, termasuk aspek *delay*, *jitter*, *throughput*, dan *packet loss* [14], [15]. Dengan demikian, kualitas jaringan tidak hanya ditentukan oleh kapasitas *bandwidth*, tetapi juga oleh desain infrastruktur yang menopang pengelolaan lalu lintas data [16].

Paradigma infrastruktur jaringan mengalami pergeseran signifikan dengan munculnya konsep Software Defined Networking (SDN). Berbeda dengan jaringan konvensional yang menggabungkan fungsi *control plane* dan *data plane*, SDN memisahkan kedua fungsi tersebut sehingga administrator dapat mengelola konfigurasi jaringan secara terpusat melalui *controller* [17], [18]. Pemisahan infrastruktur ini memungkinkan pengelolaan ribuan perangkat jaringan secara lebih sederhana, efisien dan responsif terhadap perubahan kondisi lalu lintas pada jaringan. Dibandingkan dengan infrastruktur tradisional, SDN menawarkan fleksibilitas dan kemampuan adaptasi yang lebih baik. Perbedaan mendasar terletak pada struktur *control plane* dan *data plane*, yang menjadikan jaringan berbasis SDN mampu merespon perubahan secara lebih cepat serta memberikan kualitas layanan (QoS) yang lebih baik [19], [20]. Penelitian terkini juga menunjukkan bahwa SDN dapat meningkatkan efisiensi dalam mengurangi *latensi, jitter*, dan *packet loss* melalui mekanisme pengaturan lalu lintas jaringan yang dinamis [21].

Meskipun SDN menawarkan fleksibilitas dan skalabilitas tinggi, infrastruktur ini tetap rentan terhadap ancaman keamanan signifikan yang berpotensi membahayakan informasi pengguna. Kerentanan ini muncul karena perangkat SDN umumnya terhubung ke internet, sehingga meningkatkan kemungkinan akses oleh pihak yang tidak berwenang [22]. Untuk memitigasi risiko tersebut, penerapan mekanisme keamanan seperti Firewall menjadi sangat krusial dalam infrastruktur SDN [23]. Keamanan pada jaringan komputer juga mencakup kebijakan dan mekanisme yang dirancang untuk mencegah akses illegal, penyalahgunaan, modifikasi atau penghentian layanan, sekaligus memastikan ketersediaan sumber daya jaringan [24]. Salah satu ancaman yang sering dihadapi oleh *Denial of Service* (DoS), yaitu serangan yang mengeksploitasi sumber daya sistem hingga layanan menjadi tidak dapat diakses oleh pengguna lain [25]. Strategi pengamanan modern biasanya mengikuti tiga tahapan, yaitu *Prevention* (pencegahan), *Observation* (observasi), dan *Response* (respon) [26]. Beberapa studi terkini membahas desain dan implementasi mekanisme keamanan pada Software-Defined Networking (SDN) melalui pemrograman *flow rules* pada kontroler SDN. Misalnya, penelitian melaporkan implementasi firewall berinspeksi stateful yang diprogram di atas POX Controller dan OpenFlow untuk mendeteksi serta mengatur aliran paket sebagai bagian dari firewall terdistribusi guna meningkatkan skalabilitas pertahanan jaringan, termasuk pemisahan aliran trafik besar dan kecil untuk memitigasi serangan lateral [27]. Selain itu, ulasan komprehensif terhadap kontroler SDN sumber terbuka menguraikan peran kontroler seperti POX serta switch berbasis OpenFlow seperti OvS dalam menyediakan fungsi keamanan dasar melalui *flow-based filtering* dan pengelolaan trafik, serta tantangan operasional lainnya pada implementasi nyata SDN [28]. Studi perbandingan lainnya menunjukkan bahwa POX memiliki keunggulan kesederhanaan dan kemudahan penggunaan dalam konteks implementasi prototipe keamanan, menjadikannya cocok untuk eksperimen firewall SDN pada topologi berskala kecil hingga menengah [29], [30].

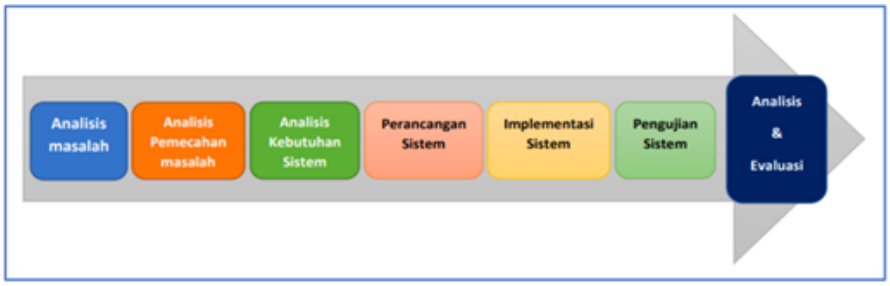
Dalam konteks ini, Firewall memiliki peran sentral. Tanpa mekanisme ini, infrastruktur jaringan sangat rentan terhadap berbagai serangan, termasuk *interception*, *interruption*, *fabrication*, dan *modification* [31], [28]. Namun, penerapan Firewall pada jaringan tradisional berbeda dengan yang ada di SDN. Pada perangkat konvensional, *Firewall* biasanya ditempatkan secara terpusat pada perangkat keras seperti router atau switch, sedangkan dalam SDN, *Firewall* diimplementasikan pada lapisan perangkat lunak yang dikelola secara terpusat melalui *controller* [32], [33], [34]. Keunggulan utama penerapan Firewall di SDN adalah kemampuannya untuk mendistribusikan aturan keamanan secara terpusat, yang tidak hanya meningkatkan skalabilitas jaringan tetapi juga mempercepat respon terhadap ancaman baru. Dengan demikian, integrasi Firewall berbasis SDN memberikan pendekatan yang lebih adaptif dan efisien dibandingkan dengan penerapan tradisional [22], [32], [35].

Penelitian ini bertujuan untuk mengimplementasikan mekanisme Firewall sebagai solusi keamanan pada lingkungan jaringan komputer berbasis SDN. Implementasi dilakukan dengan memanfaatkan POX Controller sebagai *control plane* dan OvS sebagai *data plane*. Pada sistem yang dirancang, POX Controller berfungsi untuk mengatur dan mendistribusikan aturan (*rules*) aliran trafik jaringan, sedangkan OvS berperan dalam meneruskan paket data sesuai dengan kebijakan yang telah ditentukan. Selain itu, penelitian ini menguji efektivitas penerapan aturan keamanan dengan menguji keberhasilan *rules* yang diprogram pada POX Controller dalam mengarahkan paket menuju Firewall melalui OvS. Firewall kemudian melakukan proses inspeksi dan penyaringan paket berdasarkan kebijakan keamanan yang telah ditetapkan. Pengujian dilakukan melalui beberapa skenario yang dirancang sesuai dengan karakteristik lingkungan jaringan berbasis SDN, sehingga dapat diketahui apakah sistem mampu menerapkan kebijakan keamanan secara konsisten dan terpusat.

Kontribusi utama penelitian ini adalah penyajian sebuah model implementasi Firewall terintegrasi dalam arsitektur SDN yang menunjukkan bagaimana pemisahan *control plane* dan *data plane* dapat dimanfaatkan untuk meningkatkan fleksibilitas dan pengelolaan keamanan jaringan. Penelitian ini memberikan gambaran praktis mengenai integrasi POX Controller, OvS, dan *Firewall* dalam satu kesatuan sistem, sehingga dapat menjadi referensi bagi pengembangan dan implementasi mekanisme keamanan jaringan berbasis SDN pada skala laboratorium maupun lingkungan nyata.

2. METODE PENELITIAN

Tahapan penelitian adalah langkah-langkah sistematis yang dilakukan oleh peneliti dalam suatu penelitian untuk mencapai tujuan tertentu. Setiap tahap memiliki fungsi spesifik yang membantu peneliti dalam mengidentifikasi masalah, mencari solusi, serta mengembangkan dan mengevaluasi hasil penelitian. Gambar 2 berikut adalah metode penelitian yang digunakan oleh peneliti.



Gambar 2. Metodologi penelitian

Pendekatan yang digunakan dalam studi ini terdiri atas tujuh tahapan utama. Tahap pertama adalah analisis masalah yang dilakukan untuk mengidentifikasi isu-isu utama terkait keterbatasan jaringan konvensional dan tantangan keamanan jaringan melalui studi literatur dan observasi. Tahap kedua, analisis pemecahan masalah, berfokus pada evaluasi alternatif solusi arsitektur jaringan, dengan mempertimbangkan efektivitas dan efisiensi, hingga akhirnya dipilih pendekatan berbasis SDN dengan integrasi Firewall. Tahap ketiga adalah analisis kebutuhan sistem, yang mencakup penentuan spesifikasi teknis, baik perangkat lunak maupun konfigurasi jaringan yang diperlukan untuk membangun infrastruktur SDN secara virtual. Selanjutnya, tahap perancangan dilakukan dengan menyusun arsitektur jaringan, meliputi topologi fisik dan logis, serta merancang skenario pengujian yang mencakup komunikasi antar host, pengaturan *flow entry*, dan penerapan *rule*s Firewall. Pada tahap implementasi, rancangan direalisasikan dengan membangun infrastruktur SDN, mengintegrasikannya dengan *controller*, Open vSwitch, dan aplikasi Firewall sesuai dengan topologi yang ditetapkan. Setelah itu, tahap pengujian dilakukan untuk mengevaluasi fungsionalitas sistem, termasuk pengujian komunikasi antar host, pengelolaan *flow table*, serta efektivitas Firewall dalam menyaring paket sesuai dengan *rule* yang ditetapkan pada Firewall. Tahap terakhir adalah analisis hasil dan evaluasi terhadap luaran pengujian dibandingkan dengan tujuan yang ditetapkan. Pada tahap ini juga diidentifikasi area yang masih memerlukan penyempurnaan, serta rekomendasi pengembangan lebih lanjut dalam penerapan SDN dengan mekanisme keamanan Firewall.

2.1 *Analisis Masalah dan Pemecahan Masalah*

2.1.1 Analisis Masalah

Perkembangan teknologi jaringan komputer saat ini ditandai dengan hadirnya SDN sebagai paradigma baru melalui pemisahan fungsi *control plane* dan *data plane*. Berdasarkan arsitekturnya, infrastruktur jaringan dapat dibedakan menjadi dua kategori, yaitu konvensional/tradisional dan modern berbasis SDN. Infrastruktur jaringan konvensional memiliki sejumlah keterbatasan, diantaranya kompleksitas konfigurasi perangkat serta kebutuhan waktu yang relatif lebih lama untuk pengelolaan. Sebaliknya, arsitektur SDN memungkinkan pengaturan jaringan dilakukan secara lebih terpusat dan efisien. Seiring dengan peningkatan penetrasi internet, volume data yang mengalir di dalam jaringan meningkat secara signifikan. Kondisi ini tidak hanya menambah kompleksitas arsitektur jaringan, tetapi juga meningkatkan risiko ancaman keamanan yang berasal dari intruder. Oleh karena itu, mekanisme pengamanan seperti Firewall memegang peran penting dalam menjaga keandalan dan keamanan infrastruktur jaringan. Penggunaan Firewall telah lama menjadi praktik standar pada infrastruktur jaringan konvensional, dan kebutuhan yang sama juga berlaku pada arsitektur SDN. Namun, penerapan Firewall pada SDN menuntut pendekatan yang berbeda karena karakteristik pengelolaan jaringan yang berbasis perangkat lunak dan terpusat. Tantangan utama dalam penelitian ini adalah bagaimana mengimplementasikan Firewall secara virtual pada infrastruktur SDN. Implementasi tersebut memerlukan pemahaman mendalam mengenai arsitektur SDN serta konfigurasi Firewall agar dapat berfungsi optimal dalam menjamin keamanan jaringan.

2.1.2 Analisis Pemecahan Masalah

Berdasarkan permasalahan yang telah dipaparkan sebelumnya, salah satu Langkah untuk mengatasinya adalah dengan memahami komponen utama penyusun infrastruktur jaringan. Pada implementasi SDN berbasis virtual, infrastruktur dibangun dengan memanfaatkan perangkat lunak pendukung seperti sistem operasi dan mesin virtual sebagai media untuk merealisasikan fungsi jaringan, dalam konteks ini, penerapan Firewall menjadi elemen penting untuk menjamin keamanan sistem, baik pada infrastruktur jaringan fisik maupun virtual. Sebelum mengimplementasikan Firewall pada SDN virtual, diperlukan pemahaman mendasar mengenai perbedaan antara infrastruktur fisik (*physical topology*) dan infrastruktur virtual (*logical topology*). Keduanya memiliki karakteristik berbeda, meskipun terdapat kesamaan pada penggunaan *protocol* komunikasi jaringan. Salah satu *protocol* utama dalam arsitektur SDN adalah OpenFlow. Pada topologi fisik, perangkat jaringan yang menggunakan *protocol* ini disebut OpenFlow Switch, sedangkan pada topologi virtual digunakan Open vSwitch. Selain itu, pengembangan SDN virtual juga memerlukan konfigurasi yang tepat pada *control plane* serta pemahaman terhadap fungsi *data plane* yang dijalankan oleh setiap host.

Pada penelitian ini, Firewall diinstal pada mesin utama yang menjadi pusat pengelolaan infrastruktur, yaitu mesin virtual yang menjalankan SDN *controller* dan Open vSwitch. Fungsi utama Firewall adalah melakukan seleksi terhadap paket data sebelum diteruskan ke host tujuan, sesuai dengan kebutuhan sistem. Aturan keamanan yang ditetapkan dalam Firewall (*security policies*) dapat berbasis IP Address, *port number*, *interfaces*, maupun parameter lain yang ditentukan oleh administrator jaringan. Perbedaan mendasar juga terlihat antara penerapan Firewall pada topologi fisik dan virtual. Pada topologi fisik, Firewall umumnya berfungsi untuk melindungi perangkat keras atau client dari serangan *intruder*. Sebaliknya, dalam topologi virtual, Firewall digunakan untuk menjaga integritas sistem operasi dari ancaman seperti *modification attack*, yakni serangan yang bertujuan merusak dan mengubah konfigurasi sistem secara menyeluruh. Dengan demikian, keberadaan Firewall tidak hanya memperkuat keamanan SDN, tetapi juga mendukung fleksibilitas dalam pengelolaan kebijakan jaringan.

2.2 *Analisis Kebutuhan Sistem*

Perangkat lunak yang dibutuhkan untuk membangun lingkungan eksperimen virtual yang berbasis SDN disajikan pada Tabel 1.

Tabel 1. Kebutuhan Perangkat Lunak

No	Perangkat Lunak	Versi	Keterangan
1	VMWare	16.2.4	Mesin virtual
2	Ubuntu	20.04 LTS	Sistem Operasi yang untuk host dan controller
3	POX Controller	0.7.0	Controll plane di infrastruktur jaringan SDN
4	Open vSwitch		Data plane di infrastruktur jaringan SDN
5	Sistem Operasi	Windows 11	Sistem operasi utama (Host)

Selanjutnya, kebutuhan perangkat keras (minimum) yang dibutuhkan untuk membangun lingkungan eksperimen virtual yang berbasis SDN disajikan pada Tabel 2.

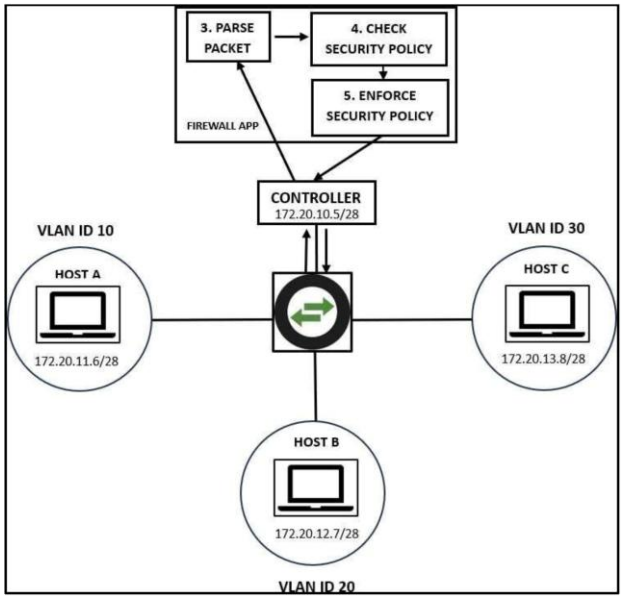
Tabel 2. Kebutuhan Perangkat Keras

No	Perangkat Keras	Versi
1	Jenis processor	11th Gen Intel (R)
2	Processor Core dan speed	Core™ i5-1135G7 @ 2.40GHz 2.42
3	RAM	8GB, system type: 64-bit

2.3 *Perancangan Sistem*

Perancangan infrastruktur jaringan berbasis SDN memiliki perbedaan mendasar dengan jaringan konvensional terutama pada pemisahan fungsi *control plane* dan *data plane*. Dengan adanya pemisahan kedua jenis *controller* memungkinkan fleksibilitas pemrograman jaringan secara langsung sehingga manajemen trafik jaringan menjadi lebih efisien dan adaptif. SDN memanfaatkan protokol OpenFlow sebagai fondasi utama yang memungkinkan *control plane* untuk secara langsung mengatur perilaku perangkat jaringan seperti switch atau router (*data plane*), melalui *rules* atau aturan yang ditentukan dalam *flow table*.

Selain itu, dari aspek keamanan mendukung implementasi Firewall dan mekanisme *control* akses berbasis aturan *flow* untuk menentukan apakah paket diteruskan, diblokir, dimodifikasi, atau diarahkan ke controller. Dengan demikian pada penelitian ini dilakukan simulasi terhadap penerapan Firewall pada SDN yang dirancang secara virtual.



Gambar 3. Topologi fisik dan topologi logic jaringan berbasis SDN

Gambar 3 memperlihatkan topologi fisik penerapan Firewall pada infrastruktur jaringan berbasis SDN secara virtual yang terdiri dari tiga host (Host A, B dan C), sebuah SDN *Controller* (*control plane*) yang menggunakan *POX Controller* yang akan mengontrol perilaku OpenFlow *switches* dengan cara menerima informasi dari *switch* dan mengirimkan instruksi (flow *rules*) kembali sesuai kebijakan jaringan yang ditetapkan. Selanjutnya, Open vSwitch (OVS) yang kompatibel dengan protokol OpenFlow yang memungkinkan pengelolaan lalu lintas jaringan dengan mengimplementasikan aturan *flow table* entry yang dikirimkan oleh SDN *Controller* melalui protokol OpenFlow pada lingkungan virtualisasi dengan fleksibilitas yang tinggi. Untuk mendukung konektivitas antar komponen jaringan yang terhubung secara virtual, maka diperlukan pengalamatan IP Address dan alokasi VLAN ID seperti disajikan pada Tabel 3.

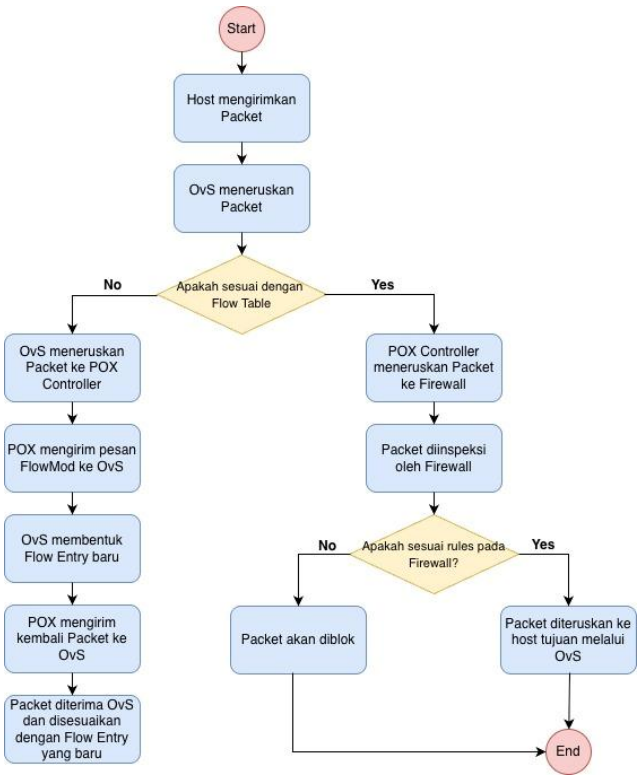
Tabel 3. Pengalamatan IP dan VLAN ID setiap komponen

Host	IP Address	VLAN
POX Controller dan *Open vSwitch*	172.20.10.5/28	-
Host A	172.20.11.6/28	VLAN ID 10
Host B	172.20.12.7/28	VLAN ID 20
Host C	172.20.13.8/28	VLAN ID 30

Rules Firewall dirancang dengan mengikuti pola yang diatur dalam bentuk *tuple* berisi parameter tertentu. *Rule* akan dijadikan acuan untuk menyaring suatu paket dengan menerapkan aksi seperti paket diteruskan (*forwarding*), dimodifikasi (*modification*) atau dibuang (*drop*). Bentuk pola *rules* yang ditetapkan pada Firewall sebagai berikut: **{<input pattern>, <action>}**. *Rules* pada Firewall dapat dirumuskan dengan tingkat kompleksitas yang beragam seperti "*allow all*", "*deny all*", bahkan dapat menyaring paket berdasarkan port TCP yang spesifik.

2.3.1 Flowchart Sistem

Mekanisme pengiriman paket dalam arsitektur SDN dengan menggunakan OVS dan POX Controller dan penerapan Firewall sebagai sistem keamanan, perlu menjalankan serangkaian alur yang disusun secara sistematis.



Gambar 4. Flowchart mekanisme pengiriman paket pada jaringan berbasis SDN

Gambar 4 menyajikan flowchart yang menggambarkan alur kerja pengiriman paket pada arsitektur SDN dengan integrasi Firewall sebagai lapisan keamanan tambahan. Proses komunikasi dimulai Ketika sebuah host mengirimkan paket yang pertama kali diterima oleh OVS. Selanjutnya, OVS memeriksa paket terhadap entri yang ada di *flow table*. Apabila paket tidak sesuai dengan entri pada *flow table*, OVS akan meneruskan paket tersebut ke POX Controller. *Controller* kemudian mengirimkan instruksi FlowMod untuk menambahkan entri baru pada *flow table* OVS. Setelah entri baru dibuat, paket dikembalikan ke OVS agar dapat diproses sesuai aturan terbaru yang telah ditentukan. Sebaliknya, apabila paket sudah sesuai dengan entri yang ada, maka paket langsung diarahkan menuju *controller* untuk tahap pemrosesan selanjutnya. Pada tahap berikutnya, paket akan dikelola oleh Firewall yang berfungsi untuk melakukan inspeksi mendalam terhadap paket dengan membandingkannya terhadap aturan kebijakan keamanan yang telah ditetapkan. Apabila paket memenuhi kriteria sesuai dengan *rules* pada Firewall, maka paket diteruskan ke host tujuan melalui OVS. Namun, apabila paket dinyatakan tidak sesuai dengan *rules*, maka paket tersebut akan diblokir untuk mencegah potensi ancaman terhadap jaringan. Secara keseluruhan, alur kerja ini menegaskan peran sinergis antara OVS, POX Controller dan Firewall dalam pengelolaan serta pengamanan jaringan SDN. OVS bertindak sebagai pengelola lalu lintas data di *data plane*, POX Controller mengatur pengendalian jalur komunikasi pada control plane, sedangkan Firewall memperkuat aspek keamanan jaringan dengan memberikan mekanisme seleksi paket berbasis *rules*. Integrasi ini tidak hanya meningkatkan fleksibilitas kemudahan pengendalian jaringan, tetapi juga memperkuat ketahanan infrastruktur terhadap potensi ancaman keamanan.

2.3.2 Skenario Pengujian

Skenario pengujian yang diterapkan untuk menguji penerapan *Firewall* pada infrastruktur jaringan berbasis SDN, sebagai berikut:

a. Pengujian keterhubungan antar komponen pada jaringan berbasis SDN

Tim merancang pengujian untuk memastikan infrastruktur berjalan sesuai dengan fungsinya dengan cara menguji konektivitas antar komponen yang saling terhubung berdasarkan topologi fisik dan logik yang telah dibuat. Selain itu, pengujian ini juga dijalankan untuk memastikan bahwa setiap komponen menjalankan fungsinya secara optimal. Pengujian konektivitas antar komponen dilakukan sesuai dengan Tabel 4.

Tabel 4. Pengujian konektivitas antar komponen pada jaringan berbasis SDN

No Pengujian	Komponen	Cara Pengujian
Pengujian 1	Host A, Host B dan Host C	Konektivitas antar host melalui Open vSwitch dan *controller*
Pengujian 2	Open vSwitch	Menambah dan menghapus *tabel flow entry*
Pengujian 3	SDN *Controller*	Melakukan konfigurasi terhadap *control plane*

b. Pengujian Firewall pada infrastruktur berbasis SDN

Selanjutnya dilakukan pengujian Firewall sebagai mekanisme keamanan jaringan yang dijalankan pada arsitektur SDN yang telah diuji. Firewall akan diimplementasikan pada SDN *Controller* dengan mendefinisikan *rules* yang berfungsi untuk mengatur lalu lintas jaringan. Pengujian difokuskan pada pemblokiran paket yang dikirim oleh salah satu host, ketika paket tersebut tidak memenuhi kriteria, sehingga dapat dibuktikan bahwa Firewall dapat menjalankan perannya dalam mengontrol lalu lintas jaringan untuk menjaga integritas dan keamanan jaringan. *Rules* yang dirancang untuk ditetapkan pada table *flow entry* adalah blocking terhadap IP Address yang spesifik dari setiap host secara bergantian. Untuk memastikan bahwa Firewall sukses memberikan luaran sesuai *rules* yang berlaku diuji coba dengan menjalankan perintah PING.

c. Pengujian modifikasi rules Firewall

Pengujian berikutnya adalah dengan melakukan modifikasi *rules* Firewall secara dinamis, dengan mengubah pemblokiran paket antar host melalui penyaringan berdasarkan IP Address. Pengujian ini dilakukan untuk menguji bahwa POX Controller adaptif ketika terjadi perubahan *rules*. Terdapat 2 *rules* yang akan ditetapkan yaitu pemblokiran paket yang berasal dari IP Address 172.20.11.6/28 yang merupakan Host A, dan pemblokiran paket yang berasal dari IP Address 172.20.12.7/28 yang merupakan Host B. Pengujian akan dilakukan dengan cara mengirimkan paket dari kedua Host ke Host lainnya dengan menggunakan perintah PING.

3. HASIL DAN PEMBAHASAN

3.1 Implementasi *POX Controller*

POX Controller sebagai SDN *controller* berfungsi untuk mengontrol perilaku OvS dengan cara menerima informasi dari *switch* dan mengirimkan instruksi (*flow rules*) kembali sesuai dengan kebijakan jaringan yang ditetapkan. POX Controller akan diinstal pada sebuah terminal dengan sistem operasi Ubuntu berbasis Linux dan di kode menggunakan Bahasa pemrograman Python. Untuk memastikan *POX Controller* telah berjalan dengan sukses dapat menggunakan perintah ““./pox.py –version”, dan memastikan struktur file pada direktori POX Controller telah terbentuk. POX Controller menggunakan modul forwarding.l2_learning,yang berfungsi sebagai learning switch pada Layer 2. Log akan menampilkan bahwa POX versi 0.7.0 berhasil dijalankan dan terkoneksi dengan OvS melalui protokol OpenFlow, ditandai dengan identifikasi datapath ID. Pesan bahwa paket masuk ke *controller* menandakan OvS belum memiliki aturan yang sesuai, sehingga paket diteruskan ke POX untuk diproses. POX kemudian mempelajari alamat MAC host dan mengirimkan *rules* bar uke OvS. Dengan demikian, hasil ini membuktikan peran POX Controller sebagai *control plane* yang mengelola aturan lalu lintas jaringan, sementara OvS berfungsi sebagai *data plane* yang meneruskan paket berdasarkan instruksi controller.

3.2 Implementasi *Bridge*

Bridge adalah perangkat atau komponen yang berfungsi menghubungkan dua atau lebih segmen jaringan yang menggunakan protokol yang sama. *Bridge* bekerja pada lapisan data link (Layer 2) dalam model referensi OSI (*Open Systems Interconnection*). Tujuan utama dari *Bridge* adalah mengirimkan paket data di antara segmen jaringan yang terhubung secara efisien. Untuk menjalankan arsitektur SDN dibutuhkan sebuah *bridge* untuk menghubungkan semua mesin virtual yang akan dihubungkan seperti antara host dan server. IP Address untuk SDN Controller, OVS dan Host dikonfigurasi sesuai dengan Tabel 4 pengalamatan IP pada bagian perancangan sistem dengan metode statis pada setiap mesin virtual yang digunakan. IP Address akan dikonfigurasi pada sistem operasi Ubuntu yang dijalankan pada setiap mesin virtual secara manual dengan menetapkan IP address, subnet mask dan gateway. Konfigurasi ini bertujuan untuk memastikan host terhubung dengan benar ke jaringan sesuai topologi yang digunakan, sekaligus mendukung komunikasi data yang stabil dalam lingkungan uji coba SDN.

3.3 Implementasi *Open vSwitch*

OvS diinstal pada mesin virtual yang sama dengan POX Controller. Layanan OvS berhasil aktif dijalankan dapat dipantau melalui daemon sistem yang akan menampilkan status "active (excited)" atau status eksekusi sukses. Hal ini membuktikan bahwa OVS telah siap untuk digunakan dalam mengelola lalu lintas jaringan, serta dapat berfungsi sebagai *data plane* yang berinteraksi dengan SDN Controller. Melalui perintah ovs-vsctl list ports interface dapat menampilkan daftar port yang terhubung pada OvS. Pada penelitian ini *bridge br0* pada OVS memiliki tiga antarmuka fisik yaitu ens37, ens38, dan ens39, serta tiga port VLAN yaitu vlan10, vlan20 dan vlan30. Konfigurasi ini menegaskan bahwa OvS mampu mengintegrasikan antarmuka fisik dan virtual untuk mendukung segmentasi jaringan berbasis VLAN, sehingga memberikan fleksibilitas dalam pengelolaan lalu lintas jaringan pada lingkungan SDN.

3.4 Implementasi *Firewall* pada Arsitektur SDN

POX Controller dan OvS dijalankan secara virtual dan dihubungkan dengan menggunakan metode *bridge*, dengan mengeksekusi perintah **"ovs-vsctl set-controller <nama_bridge> tcp:<alamat_IP_POX>:<port_POX>"**. Pada implementasinya, POX Controller diberi Alamat 172.20.10.5 dengan port 6633. Untuk memastikan konektivitas keduanya terbentuk, maka pada sisi POX Controller dijalankan modul ***openflow.discovery*** dan ***forwarding.l2_learning***, yang ditandai dengan status "**is_*connected***" pada *controller*. Selanjutnya, Firewall diimplementasikan pada *POX Controller* sebagai wadah untuk merumuskan aturan keamanan jaringan (*rules*) yang spesifik untuk mengontrol lalu lintas jaringan misalnya dengan memberikan kriteria penyaringan paket yang melalui jaringan berdasarkan IP Address, port, protokol tertentu atau kriteria lainnya. *Rules* yang diuji pada penelitian ini adalah penyaringan terhadap IP Address tertentu dengan menguji bloking IP Address antar-host secara bergantian, dan *bloking* protokol ICMP. *Rule* yang akan digunakan adalah DROP dengan detail sebagai berikut.

((DROP) msg.actions.append(of.ofp_action_output(port=of.OFPP_NONE))event.connection.send(msg))

Perintah PING akan dijalankan untuk memastikan *rules* Firewall dapat dijalankan dengan sukses sesuai dari konfigurasi yang ditetapkan untuk masing-masing skenario bloking antar-host.

3.5 Implementasi Host

Terdapat 3 host yang akan dipantau pada lalu lintas jaringan yang diberikan IP Address dan didaftarkan pada VLAN pada OVS, sesuai dengan Tabel 5. Setiap Host akan dijalankan pada masing-masing mesin virtual dengan sistem operasi Ubuntu Desktop 20 LTS, lalu konfigurasi pengalamatan ditetapkan secara statik.

Tabel 5. Pengalaman IP Address dan VLAN ID Host

Host	IP Address	VLAN ID
Host A	172.20.11.6/28	10
Host B	172.20.12.7/28	20
Host C	172.20.13.8/28	30

3.6 Pengujian dan Pembahasan Hasil

Pengujian dilakukan sesuai dengan skenario pengujian dengan pembahasan hasil yang diuraikan pada sub berikut.

3.6.1 Pengujian dan pembahasan hasil keterhubungan antar komponen

Pengujian antar komponen dilakukan untuk memastikan konektivitas antar komponen sukses melalui perintah PING yang ditujukan kepada masing-masing IP Address yang ditetapkan pada setiap komponen. Hasil pengujian disajikan pada Tabel 6 yang menunjukkan bahwa seluruh komponen telah berhasil terhubung yang dibuktikan dengan status REPLY dari setiap uji konektivitas antar komponen yang dituju. Keterhubungan antar komponen ini menjadi fondasi yang penting untuk menjalankan pengujian skenario selanjutnya.

Tabel 6. Hasil pengujian konektivitas antar komponen

Uji konektivitas	Bukti	Status
POX Controller → OVS	server01@sdn01:~$ sudo ovs-vsctl show 07ebef0a-9c89-45c8-ac53-1329bbcedafa Bridge br0 Controller "tcp:172.20.10.5:6633" is_connected: true Port br0 Interface br0 type: internal	Berhasil
POX Controller → Host A	server01@sdn01:~/pox$ ping 172.20.11.6 PING 172.20.11.6 (172.20.11.6) 56(84) bytes of data. 64 bytes from 172.20.11.6: icmp_seq=1 ttl=64 time=8.14 ms	Berhasil
POX Controller → Host B	server01@sdn01:~/pox$ ping 172.20.12.7 PING 172.20.12.7 (172.20.12.7) 56(84) bytes of data. 64 bytes from 172.20.12.7: icmp seq=1 ttl=64 time=14.1 ms	Berhasil
POX Controller → Host C	server01@sdn01:~/pox$ ping 172.20.13.8 PING 172.20.13.8 (172.20.13.8) 56(84) bytes of data. 64 bytes from 172.20.13.8: icmp seq=1 ttl=64 time=12.7 ms	Berhasil
Host A → Host B	hosta@hosta:~/Desktop$ ping 172.20.12.7 PING 172.20.12.7 (172.20.12.7) 56(84) bytes of data. 64 bytes from 172.20.12.7: icmp_seq=1 ttl=63 time=5.60 ms	Berhasil
Host B → Host C	hostb@hostb:~/Desktop$ ping 172.20.13.8 PING 172.20.13.8 (172.20.13.8) 56(84) bytes of data. 64 bytes from 172.20.13.8: icmp_seq=1 ttl=63 time=350 ms	Berhasil
Host A → Host C	hosta@hosta:~/Desktop$ ping 172.20.13.8 PING 172.20.13.8 (172.20.13.8) 56(84) bytes of data. 64 bytes from 172.20.13.8: icmp_seq=1 ttl=63 time=4.38 ms	Berhasil

Tabel 6 menunjukkan hasil uji konektivitas pada lingkungan SDN yang telah dirancang, meliputi koneksi antara POX Controller, OvS, serta komuniasi antar host (Host A, Host B, dan Host C). Pengujian dilakukan menggunakan perintah ovs-vsctl show untuk memverifikasi koneksi *control plane* serta perintah PING untuk menguji konektivitas pada *data plane*. Berdasarkan hasil pengujian, koneksi antara POX Controller dan OvS berhasil dilakukan, yang ditunjukkan oleh status is_connected: true, yang menandakan bahwa *controller* mampu berkomunikasi dan mengelola OvS secara terpusat, sesuai dengan prinsip dasar arsitektur SDN. Selanjutnya, pengujian konektivitas dari POX Controller ke seluruh host menunjukkan bahwa seluruh paket ICMP berhasil diterima, yang mengindikasikan bahwa jalur komunikasi antar *control plane* dan *data plane* berfungsi dengan baik.

Selain itu, pengujian konektivitas antar host (Host A ke Host B, Host B ke Host C, dan Host A ke Host C) juga menunjukkan hasil dengan status berhasil. Hal ini mengindikasikan bahwa aturan *flow* yang diprogram melalui POX Controller telah diterapkan secara tepat pada OvS, sehingga paket data dapat diteruskan sesuai dengan rules yang telah ditentukan. Secara keseluruhan, dari 7 skenario pengujian konektivitas yang dilakukan, seluruhnya menunjukkan status berhasil, sehingga tangkat keberhasilan pengujian mencapai 100%. Hasil ini membuktikan bahwa arsitektur jaringan SDN yang dirancang mampu menyeediakan konektivitas *end-to-end* yang andal.

3.6.2 Pengujian dan pembahasan hasil kinerja OVS

Pengujian OVS bertujuan memastikan bahwa switch virtual berfungsi pada SDN dalam hal pengalihan paket maupun integrasi dengan *controller*. Pada OVS perlu diuji konektivitas VLAN, serta *flow table* yang diterima dari *controller*.

Tabel 7. *Flow table* pada OvS

No	Flow Table Entry
1	OPFT_FEATURES_REPLY (xid=0x2) : dpid:0000000c29c7b915
2	n_tables:254, n_buffers:0
3	capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
4	actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst mod_nw_src mod_nw_dst mod_nw tos mod_tp_src mod_tp_dst

Hasil pengujian yang telah dilakukan menunjukkan bahwa *flow table* berhasil diterima oleh OVS seperti yang disajikan pada Tabel 7, yang kemudian *flow entry* pada *flow table* akan diterapkan pada setiap VLAN *interface* antar-host, seperti yang disajikan pada Tabel 8.

Tabel 8. VLAN ID dan MAC Address Host pada OvS

No	Host	Interface	VLAN ID	MAC Address
1	Host A	Ens37	10	82:cb:84:3b:01:58
2	Host B	Ens38	20	1e:ed:06:3b:2b:e6
3	Host C	Ens39	30	9a:1e:8b:b5:9cc:61

Selain itu, pada OVS juga dapat memperlihatkan status masing-masing konfigurasi jaringan host dengan menggunakan perintah "*sudo ovs- ofctl show br0*", seperti yang disajikan pada Tabel 9. Jika seluruh konfigurasi host dapat ditampilkan, host telah berhasil terkoneksi secara sukses dengan OVS.

Tabel 9. Konfigurasi Host pada OvS

Komponen	Host A / ens37	Host B / ens38	Host C / ens39
Config	0	0	0
State	0	0	0
Current	1GB-FD COPPER AUTO_NEG	1GB-FD COPPER AUTO_NEG	1GB-FD COPPER AUTO_NEG
Supported	10MB-HD 100MB-HD	10MB-HD 100MB-HD	10MB-HD 100MB-HD
Speed	1000Mbps now, 1000Mbps max	1000Mbps now, 1000Mbps max	1000Mbps now, 1000Mbps max

Pada OvS juga terdapat *flow entry* yang akan menjadi *rules* untuk meneruskan paket menuju host tujuan. Pengujian yang perlu dilakukan adalah penambahan dan penghapusan *flow entry* pada OvS. Hasil pengujian penambahan *flow entry* sukses dilakukan, seperti pada Gambar 5.

```
server01@sdn01:~/pox$ sudo ovs-ofctl dump-flows br0
server01@sdn01:~/pox$ sudo ovs-ofctl add-flow br0 in_port=1,actions=output:2
server01@sdn01:~/pox$ sudo ovs-ofctl add-flow br0 in_port=1,actions=output:2
server01@sdn01:~/pox$ sudo ovs-ofctl dump-flows br0
 cookie=0x0, duration=2.313s, table=0, n_packets=0, n_bytes=0, in_port=ens37
ions=output:ens38
```

Gambar 5. Penambahan dan penghapusan *flow entry* pada OvS

Pada Gambar 5 terdapat komponen yang ditambahkan yaitu **"in_port=1, actions=output:2"**, hasil penambahan *flow entry* dapat dilihat dengan perintah **"sudo ovs-ofctl dump-flows br0"**, sedangkan untuk melakukan penghapusan entry dengan perintah **"sudo ovs-ofctl dump-flows br0"**.

3.6.3 Pengujian dan pembahasan hasil kinerja POX Controller

POX Controller diuji dengan menjalankan beberapa modul dengan fungsi yang spesifik. (1). Modul l2_learning akan mengimplementasikan pembelajaran *MAC Address* pada layer *Data Link*. Fungsi utama dari modul ini adalah meneruskan paket pada jaringan berbasis Ethernet. POX akan menyimpan *MAC Address* paket yang diterima pada Tabel 8 *MAC Address*, dan meneruskan paket ke tujuan sesuai dengan informasi tersebut. (2). Modul l3_learning akan mengimplementasikan pembelajaran *IP Address* pada *layer Network*. *IP Address* pada paket yang diterima akan disimpan pada *routing table* dan paket akan dirutekan berdasarkan informasi dari *routing table* tersebut.

```
Server01@sdn01:~/pox$ ./pox.py openflow.discovery forwarding.l2_learning
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
WARNING:version:Support for Python 3 is experimental.
INFO:core:POX 0.7.0 (gar) is up.
INFO:openflow.of_01:[00-0c-29-c7-b9-15 1] connected
Server01@sdn01:~/pox$ ./pox.py openflow.discovery forwarding.l3_learning
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
WARNING:version:Support for Python 3 is experimental.
INFO:core:POX 0.7.0 (gar) is up.
INFO:openflow.of_01:[00-0c-29-c7-b9-15 1] connected
INFO:root:Antar Host berhasil melakukan PING dan Paket masuk ke controller
INFO:root:Antar Host berhasil melakukan PING dan Paket masuk ke controller
INFO:root:Antar Host berhasil melakukan PING dan Paket masuk ke controller
```

Gambar 6. Aktivitas pengiriman paket antar host yang diproses oleh Controller

Gambar 6 merupakan hasil implementasi modul l2_learning dan l3_learning pada POX Controller. Kedua modul tersebut berfungsi untuk memantau aktivitas pengiriman paket antar host. Uji coba dilakukan dengan menggunakan perintah PING dan hasil yang didapatkan adalah paket diproses pada POX Controller.

3.6.4 Pengujian dan pembahasan hasil bloking paket menggunakan Firewall

Setelah seluruh komponen dipastikan dapat menjalankan fungsinya, selanjutnya adalah pengujian fungsi Firewall dengan tujuan untuk memblok paket. *Rules* Firewall akan diujikan untuk *IP Address* yang ditetapkan secara spesifik pada sebuah modul pox.py yang berbasis Python. Uji coba yang dilakukan adalah bloking komunikasi antar host secara bergantian dengan menggunakan perintah PING. Hasil yang diharapkan dari pengujian ini adalah paket yang diterima dari host yang diblok akan dibuang (drop) sesuai dengan *rule* pada Firewall. Paket yang dikirimkan melalui perintah PING adalah protokol ICMP Paket-paket tersebut adalah paket ICMP (*Internet Control Message Protocol*). Pengujian pertama adalah pemblokiran paket yang dikirimkan oleh Host A menuju Host B dan C, serta *Controller*. Hasil pengujian terhadap kasus ini disajikan pada Tabel 10.

Tabel 10. Hasil pengujian filtering paket di Firewall dengan pemblokiran Host A berdasarkan IP Address

Pengujian	Bukti	Status
Aktivasi *rules Firewall* pada Controller	server01@sdn01:~/pox$ ./pox.py log.level --DEBUG firewall POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al. DEBUG:core:POX 0.7.0 (gar) going up... DEBUG:core:Running on CPython (3.8.10/May 26 2023 14:05:08) DEBUG:core:Platform is Linux-5.4.0-152-generic-x86_64-with-glibc2.29 WARNING:version:Support for Python 3 is experimental. INFO:core:POX 0.7.0 (gar) is up. DEBUG:openflow.of_01:Listening on 0.0.0.0:6633 INFO:openflow.of_01:[00-0c-29-c7-b9-15 1] connected	Berhasil
Host A → Host B & Host C	hosta@hosta:~/Desktop$ ping 172.20.12.7 PING 172.20.12.7 (172.20.12.7) 56(84) bytes of data. From 172.20.11.6 icmp_seq=9 Destination Host Unreachable hosta@hosta:~/Desktop$ ping 172.20.13.8 PING 172.20.13.8 (172.20.13.8) 56(84) bytes of data. From 172.20.11.6 icmp_seq=1 Destination Host Unreachable	Berhasil
Host A → Controller	hosta@hosta:~/Desktop$ ping 172.20.10.5 PING 172.20.10.5 (172.20.10.5) 56(84) bytes of data. From 172.20.11.6 icmp_seq=1 Destination Host Unreachable	Berhasil
Host B → Host A	hostb@hostb:~/Desktop$ ping 172.20.11.6 PING 172.20.11.6 (172.20.11.6) 56(84) bytes of data. From 172.20.12.7 icmp_seq=9 Destination Host Unreachable	Berhasil
Host C → Host A	hostc@hostc:~/Desktop$ ping 172.20.11.6 PING 172.20.11.6 (172.20.11.6) 56(84) bytes of data. From 172.20.13.8 icmp_seq=9 Destination Host Unreachable	Berhasil

Pada Tabel 10 dapat disimpulkan bahwa paket yang berasal dari Host A berhasil diblok (*drop*), sehingga Host A tidak dapat melakukan komunikasi ke Host B dan Host C serta *Controller*. Begitu juga dengan arah sebaliknya, Host B dan C, serta *Controller* tidak dapat melakukan komunikasi dengan Host A. Dengan demikian hasil pengujian menunjukkan bahwa aktivasi

modul Firewall pada POX Controller berhasil dilakukan, yang ditandai dengan keluaran log debug dan status koneksi OpenFlow yang aktif antara controller dan OvS. Kondisi ini mengonfirmasi bahwa *control plane* SDN mampu menjalankan fungsi keamanan secara terpusat, serta mendistribusikan *flow rules* ke *data plane* sesuai dengan *rules* yang telah ditentukan.

Pengujian konektivitas antar host dan antara host dengan *controller* memperlihatkan bahwa seluruh lrafik yang tidak diizinkan berhasil diblokir, ditunjukkan oleh pesan *Destination Host Unreachable* pada setiap skenario uji. Dari total lima skenario pengujian, seluruhnya memberikan hasil sesuai dengan *rules* Firewall yang dirancang, sehingga tingkat keberhasilan pengeujian mencapai 100%. Temuan ini menegaskan efektivitas integrasi Firewall berbasis SDN dalam mengendalikan lalu lintas jaringan secara dinamis dan konsisten melalui mekanisme pemrograman *flow rules* pada OvS.

Tabel 11. *Rules* pada *Firewall* untuk pemblokiran paket berdasarkan IP Address

```
def handle_packet(event):
  packet = event.parsed
  # Melakukan filtering berdasarkan IP Address
  if packet.find('ipv4'):
    ip_packet = packet.find('ipv4')
    src_ip = ip_packet.srcip
    dst_ip = ip_packet.dstip
    # Menetapkan IP Address Host B
    Host b_ip = '172.20.12.7'
```

Selanjutnya, *rules* Firewall dapat diubah dengan kasus baru misalnya pemblokiran terhadap Host B, dengan melakukan perubahan sesuai *rules* pada Tabel 11. Sesuai dengan *rules* Firewall, maka IP Address Host B akan diblokir, dengan hasil pengujian yang disajikan pada Tabel 12.

Tabel 12. Hasil pengujian filtering paket di *Firewall* dengan pemblokiran Host B berdasarkan IP Address

Pengujian	Bukti	Status
Aktivasi *rules Firewall* pada Controller	server01@sdn01:~/pox$ ./pox.py log.level --DEBUG firewall POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al. DEBUG:core:POX 0.7.0 (gar) going up... DEBUG:core:Running on CPython (3.8.10/May 26 2023 14:05:08) DEBUG:core:Platform is Linux-5.4.0-152-generic-x86_64-with-glibc2.29 WARNING:version:Support for Python 3 is experimental. INFO:core:POX 0.7.0 (gar) is up. DEBUG:openflow.of_01:Listening on 0.0.0.0:6633 INFO:openflow.of_01:[00-0c-29-c7-b9-15 1] connected	Berhasil
Host B → Host A & Host C	hostb@hostb:~/Desktop$ ping 172.20.11.6 PING 172.20.11.6 (172.20.11.6) 56(84) bytes of data. From 172.20.12.7 icmp_seq=1 Destination Host Unreachable hostb@hostb:~/Desktop$ ping 172.20.13.8 PING 172.20.13.8 (172.20.13.8) 56(84) bytes of data. From 172.20.12.7 icmp_seq=1 Destination Host Unreachable	Berhasil
Host B → Controller	hostb@hostb:~/Desktop$ ping 172.20.10.5 PING 172.20.10.5 (172.20.10.5) 56(84) bytes of data. From 172.20.12.7 icmp_seq=1 Destination Host Unreachable	Berhasil
Host A → Host B	hosta@hosta:~/Desktop$ ping 172.20.12.7 PING 172.20.12.7 (172.20.12.7) 56(84) bytes of data. From 172.20.11.6 icmp_seq=1 Destination Host Unreachable	Berhasil
Host C → Host B	hostc@hostc:~/Desktop$ ping 172.20.12.7 PING 172.20.12.7 (172.20.12.7) 56(84) bytes of data. From 172.20.13.8 icmp_seq=1 Destination Host Unreachable	Berhasil

Sesuai hasil pengujian pada Tabel 12 dapat disimpulkan bahwa perubahan *rules* secara fleksibel dapat dilakukan melalui *Controller*. Seluruh paket yang berasal dari Host B berhasil diblok (*drop*) dari Host A dan C, serta *Controller*, sehingga dapat disimpulkan bahwa percobaan berhasil memenuhi skenario pengujian dengan persentase keberhasilan 100%.

4. KESIMPULAN

Berdasarkan hasil implementasi dan pengujian, integrasi POX Controller dengan OvS berhasil dijalankan untuk mendukung fungsi Firewall berbasis SDN. Hasil uji menunjukkan bahwa mekanisme pemblokiran berdasarkan IP Address telah berjalan sesuai dengan aturan (*rule*) yang ditetapkan, sehingga sistem mampu memberikan kontrol lalu lintas jaringan secara dinamis dan terpusat. Temuan ini menegaskan bahwa pendekatan SDN dengan memanfaatkan POX Controller dan OvS dapat menjadi solusi efektif dalam penerapan Firewall yang dapat diprogram secara fleksibel. Hal ini ditunjukkan melalui hasil pengujian filtering paket pada Firewall yang mencapai 100% keberhasilan dalam setiap skenario uji yang telah dirancang dan dijalankan. Untuk penelitian selanjutnya, pengembangan sistem dapat diarahkan pada peningkatan fungsi Firewall agar tidak hanya terbatas pada pemblokiran IP Address, tetapi juga mencakup *port-based filtering*, maupun deteksi serangan berbasis pola (*signature-based* maupun *anomaly-based detection*). Selain itu, uji coba pada lingkungan jaringan yang lebih kompleks dan berskala besar juga direkomendasikan, sehingga solusi Firewall berbasis SDN ini dapat divalidasi lebih lanjut untuk kebutuhan implementasi nyata di infrastruktur jaringan.

DAFTAR PUSTAKA

[1] B. M. Leiner *et al.*, "A brief history of the internet," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 5, pp. 22–31, Oct. 2009, doi: 10.1145/1629607.1629613.
[2] M. Castells, *The Rise of the Network Society*, 2nd ed. Oxford, UK: Wiley-Blackwell, 2010.
[3] T. Berners-Lee, M. Fischetti, and M. L. Foreword, *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. New York, NY, USA: Harper San Francisco, 2000.
[4] K. C. Laudon and J. P. Laudon, *Management Information Systems: Managing the Digital Firm*, 16th ed. Harlow, UK: Pearson Education, 2020.
[5] P. T. Jaeger, J. Lin, and J. M. Grimes, "Cloud Computing and Information Policy: Computing in a Policy Cloud?," *Journal of Information Technology & Politics*, vol. 5, no. 3, pp. 269–283, Oct. 2008, doi: 10.1080/19331680802425479.
[6] M. Hilbert and P. López, "The World's Technological Capacity to Store, Communicate, and Compute Information," *Science (1979)*, vol. 332, no. 6025, pp. 60–65, Apr. 2011, doi: 10.1126/science.1200970.
[7] Statista, "Countries with the largest digital populations in the world as of October 2025 (in millions)," https://www.statista.com/statistics/262966/number-of-internet-users-in-selected-countries/.
[8] Internet World Stats, "World internet usage and population statistics 2023," https://www.internetworldstats.com/stats.htm.
[9] M. Graham and S. Dutton, *Society and the Internet: How Networks of Information and Communication are Changing Our Lives*. Oxford, UK: Oxford University Press, 2019.
[10] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*, 5th ed. Upper Saddle River, NJ, USA: Prentice Hall, 2011.
[11] W. Stallings, *Data and Computer Communications*, 10th ed. Harlow, UK: Pearson, 2014.
[12] W. M. Zuberek and D. Strzeciwilk, "Modeling Quality of Service Techniques for Packet-Switched Networks," *Dependability Engineering, IntechOpen*, 2017.
[13] J. A. Pérez, V. H. Zárate, and C. Cabrera, "A Network and Data Link Layer QoS Model to Improve Traffic Performance," 2006, pp. 224–233. doi: 10.1007/11807964_23.
[14] S. Wood and S. Chatterjee, "Network Quality of Service for the Enterprise: A Broad Overview," *Information Systems Frontiers*, vol. 4, no. 1, pp. 63–84, Apr. 2002, doi: 10.1023/A:1015390607862.
[15] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, Apr. 2014, doi: 10.1145/2602204.2602219.
[16] N. Ghani, A. Shami, C. Assi, and M. Y. A. Raja, "Quality of service in Ethernet passive optical networks," in *2004 IEEE/Sarnoff Symposium on Advances in Wired and Wireless Communications*, IEEE, pp. 161–165. doi: 10.1109/SARNOF.2004.1302866.
[17] M. N. A. Sheikh, I.-S. Hwang, M. S. Raza, and M. S. Ab-Rahman, "A Qualitative and Comparative Performance Assessment of Logically Centralized SDN Controllers via Mininet Emulator," *Computers*, vol. 13, no. 4, p. 85, Mar. 2024, doi: 10.3390/computers13040085.
[18] S. A. Ibrahim Hussein, F. W. Zaki, and M. M. Ashour, "Performance evaluation of software-defined wide area network based on queueing theory," *IET Networks*, vol. 11, no. 3–4, pp. 128–145, May 2022, doi: 10.1049/ntw2.12039.

[19] K. T. Mehmood, S. Atiq, I. A. Sajjad, M. M. Hussain, and M. M. A. Basit, "Examining the Quality Metrics of a Communication Network with Distributed Software-Defined Networking Architecture," *Computer Modeling in Engineering & Sciences*, vol. 141, no. 2, pp. 1673–1708, 2024, doi: 10.32604/cmes.2024.053903.

[20] D. K. Ryait and M. Sharma, "Performance Evaluation of SDN Controllers," 2023, pp. 1009–1021. doi: 10.1007/978-981-99-5166-6_68.

[21] I. Ali, S. Hong, and T. Cheung, "Quality of Service and Congestion Control in Software-Defined Networking Using Policy-Based Routing," *Applied Sciences*, vol. 14, no. 19, p. 9066, Oct. 2024, doi: 10.3390/app14199066.

[22] I. Alsmadi and D. Xu, "Security of Software Defined Networks: A survey," *Comput Secur*, vol. 53, pp. 79–108, Sep. 2015, doi: 10.1016/j.cose.2015.05.006.

[23] H. Ahmadvand, C. Lal, H. Hemmati, M. Sookhak, and M. Conti, "Privacy-Preserving and Security in SDN-Based IoT: A Survey," *IEEE Access*, vol. 11, pp. 44772–44786, 2023, doi: 10.1109/ACCESS.2023.3267764.

[24] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "Sdn Security: A Survey," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, IEEE, Nov. 2013, pp. 1–7. doi: 10.1109/SDN4FNS.2013.6702553.

[25] K. Kalkan, G. Gur, and F. Alagoz, "Defense Mechanisms against DDoS Attacks in SDN Environment," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 175–179, 2017, doi: 10.1109/MCOM.2017.1600970.

[26] K. Kurniabudi *et al.*, "Network anomaly detection research: a survey," *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, vol. 7, no. 1, Mar. 2019, doi: 10.52549/ijeei.v7i1.773.

[27] S. P., B. P. Kavin, S. R. Srividhya, R. V., K. C., and W.-C. Lai, "Performance Evaluation of Stateful Firewall-Enabled SDN with Flow-Based Scheduling for Distributed Controllers," *Electronics (Basel)*, vol. 11, no. 19, p. 3000, Sep. 2022, doi: 10.3390/electronics11193000.

[28] A. Mardaus, E. Biernacka, R. Wójcik, and J. Domżał, "Open Source Software-Defined Networking Controllers—Operational and Security Issues," *Electronics (Basel)*, vol. 13, no. 12, p. 2329, Jun. 2024, doi: 10.3390/electronics13122329.

[29] C. Jayawardena, J. Chen, A. Bhalla, and L. Bui, "Comparative Analysis of POX and RYU SDN Controllers in Scalable Networks," *International journal of Computer Networks & Communications*, vol. 17, no. 2, pp. 35–51, Mar. 2025, doi: 10.5121/ijcnc.2025.17203.

[30] Y. Meng, C. Ke, and Z. Huang, "A model transformation based security policy automatic management framework for software-defined networking," *Comput Secur*, vol. 142, p. 103850, Jul. 2024, doi: 10.1016/j.cose.2024.103850.

[31] M. Rahouti, K. Xiong, Y. Xin, S. K. Jagatheesaperumal, M. Ayyash, and M. Shaheed, "SDN Security Review: Threat Taxonomy, Implications, and Open Challenges," *IEEE Access*, vol. 10, pp. 45820–45854, 2022, doi: 10.1109/ACCESS.2022.3168972.

[32] J. G. V. Pena and W. E. Yu, "Development of a distributed firewall using software defined networking technology," in *2014 4th IEEE International Conference on Information Science and Technology*, IEEE, Apr. 2014, pp. 449–452. doi: 10.1109/ICIST.2014.6920514.

[33] A. Kanwal, M. Nizamuddin, W. Iqbal, W. Aman, Y. Abbas, and S. Mussiraliyeva, "Exploring Security Dynamics in SDN Controller Architectures: Threat Landscape and Implications," *IEEE Access*, vol. 12, pp. 56517–56553, 2024, doi: 10.1109/ACCESS.2024.3390968.

[34] Ahmad Turmudi Zy, Isarianto, A. M. Rifa'i, A. Nugroho, and A. Ghofir, "Enhancing Network Security: Evaluating SDN-Enabled Firewall Solutions and Clustering Analysis Using K-Means through Data-Driven Insights," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 9, no. 1, pp. 69–76, Jan. 2025, doi: 10.29207/resti.v9i1.6056.

[35] I. P. Hariyadi, I. M. Y. Dharma, R. Azhar, and S. Suriyati, "Implementasi Software-Defined Network Terintegrasi Firewall pada Proxmox untuk Pengontrolan Konfigurasi Jaringan dan Pengamanan Layanan Container," *JTIM: Jurnal Teknologi Informasi dan Multimedia*, vol. 7, no. 1, pp. 107–122, Jan. 2025, doi: 10.35746/jtim.v7i1.644.