

Kerangka Kerja Konsep *Multi Tenant* pada *Software Defined Network* yang Mendukung Isolasi Kuat Antar *Tenant* dengan Metode *Network Slicing*

Yudi Awal Nugraha*, Mochamad Teguh Kurniawan

Universitas Telkom, Indonesia

Email: yudiawalnugrahaa@student.telkomuniversity.ac.id*,

teguhkurniawan@telkomuniversity.ac.id

Keywords:

Software Defined Network (SDN); Multi-Tenant; Tenant Isolation; OpenFlow; Network Slicing, Network Security; Quality of Service.

Abstract

The increasing adoption of multi-tenant architectures in Software Defined Networking (SDN) has intensified challenges related to tenant isolation, network security, and quality of service (QoS). Existing studies generally implement VLAN-based isolation or network slicing independently, resulting in limited protection against cross-tenant attacks and inconsistent resource allocation. This study aims to develop and validate a comprehensive multi-tenant SDN framework that integrates VLAN-based isolation, network slicing, dynamic OpenFlow policy enforcement, and adaptive QoS management into a unified architecture. An experimental proof-of-concept approach was employed using the Mininet emulator, the Python-based Ryu controller, Open vSwitch, and the OpenFlow protocol. The proposed framework was evaluated through connectivity testing, QoS measurements, and Distributed Denial-of-Service (DDoS) SYN Flood attack simulations under multiple attack scenarios. The experimental results demonstrate that the framework achieved complete inter-tenant isolation while improving average throughput by 27.4% (from 490,284 Kbps to 624,619 Kbps), reducing packet loss by 84.5% (from 1.621% to 0.252%), and decreasing average delay from 0.440 ms to 0.388 ms. During DDoS attacks involving up to 30 attacker nodes, unaffected tenants maintained excellent QoS performance according to TIPHON standards. In conclusion, the proposed integrated framework provides a robust and scalable solution for secure multi-tenant SDN environments by simultaneously enhancing traffic isolation, network performance, and resilience against cyberattacks, making it suitable for future cloud computing and data center deployments.

Kata Kunci:

Software Defined Network (SDN); Multi-Tenant; Isolasi Tenant; OpenFlow; Network Slicing; Keamanan Jaringan; Quality of Service

Abstrak

Penerapan arsitektur *multi-tenant* pada *Software Defined Network* (SDN) semakin berkembang, namun masih menghadapi tantangan dalam menjamin isolasi antar-tenant, keamanan jaringan, serta kualitas layanan (*Quality of Service/QoS*). Penelitian sebelumnya umumnya menerapkan isolasi berbasis VLAN atau *network slicing* secara terpisah sehingga belum mampu memberikan perlindungan menyeluruh terhadap serangan lintas tenant maupun alokasi sumber daya yang optimal. Penelitian ini bertujuan merancang dan memvalidasi kerangka kerja SDN *multi-tenant* yang mengintegrasikan mekanisme isolasi berbasis VLAN, *network*

slicing, dynamic policy enforcement berbasis *OpenFlow*, serta QoS adaptif dalam satu arsitektur terpadu. Penelitian menggunakan metode eksperimen dengan pendekatan *proof of concept* melalui implementasi pada *emulator Mininet, Ryu Controller* berbasis *Python, Open vSwitch*, dan protokol *OpenFlow*. Evaluasi dilakukan melalui pengujian konektivitas, pengukuran QoS, serta simulasi serangan DDoS SYN Flood pada berbagai skenario. Hasil penelitian menunjukkan bahwa framework mampu mencapai isolasi penuh antar-tenant, meningkatkan throughput rata-rata sebesar 27,4% (490.284 Kbps menjadi 624.619 Kbps), menurunkan packet loss sebesar 84,5% (1,621% menjadi 0,252%), serta menurunkan delay rata-rata dari 0,440 ms menjadi 0,388 ms. Pada skenario serangan DDoS hingga 30 node penyerang, tenant yang tidak menjadi target tetap mempertahankan kualitas layanan kategori sangat baik berdasarkan standar TIPHON. Dengan demikian, *framework* yang diusulkan terbukti efektif dalam meningkatkan keamanan, performa jaringan, dan ketahanan terhadap serangan siber sehingga berpotensi diterapkan pada lingkungan *cloud computing* dan data center modern.

PENDAHULUAN

Perkembangan internet telah menjadi salah satu inovasi teknologi paling transformatif dalam sejarah peradaban manusia. Internet berkembang dari konsep teoretis sederhana menjadi infrastruktur global yang kompleks, mengubah cara manusia berkomunikasi, bekerja, belajar, dan berinteraksi secara fundamental. Pemahaman terhadap perkembangan internet menjadi penting karena merupakan salah satu faktor antisipatif dalam menentukan arah perkembangan teknologi di masa depan serta dampaknya terhadap berbagai aspek global (Botez, 2023; Irum, 2019; Islam, 2021; Nisa, 2024; Tolin & Piccaluga, 2024). Dalam perkembangannya, infrastruktur jaringan internet seringkali terhambat oleh masalah keamanan, sehingga pengembangan mekanisme keamanan jaringan merupakan komponen penting yang harus dipertimbangkan secara serius, sebab tidak ada jaringan yang sepenuhnya aman dari serangan (Barakabitze, 2019; Barki, 2016; Bezahaf, 2020; Ifeanyichukwu, 2023; Kreutz et al., 2015; Panda et al., 2021).

Software Defined Network (SDN) telah berkembang menjadi paradigma atau konsep baru dalam mendesain, mengelola, dan mengimplementasikan jaringan, terutama untuk mendukung kebutuhan dan inovasi dalam jaringan komputer yang semakin kompleks (Elliott, 2021). SDN memberikan harapan nyata untuk mengatasi keterbatasan infrastruktur jaringan konvensional dengan menawarkan kemampuan pemrograman jaringan secara dinamis dan pengelolaan terpusat. Konsep utama SDN adalah sentralisasi kendali jaringan, di mana semua pengaturan berada pada control plane yang terpisah dari data plane, sehingga pengelolaan jaringan dapat dilakukan melalui pengontrol berbasis perangkat lunak (Mousa et al., 2016). SDN memiliki empat fitur keunggulan dibandingkan jaringan tradisional, yaitu pengendalian aliran data secara dinamis, pengelolaan dan pemantauan status jaringan secara menyeluruh, fungsi pengendalian yang dapat diprogram, dan struktur paket data yang lebih sederhana (Khan, 2020). SDN menyederhanakan pengelolaan yang kompleks dari berbagai aliran data, memfasilitasi programmability, dan meningkatkan virtualisasi dengan memisahkan serta

mengabstraksi fungsi-fungsi control plane dari elemen-elemen jaringan ke dalam entitas terpusat yang disebut SDN *controller*.

Dalam perkembangannya, SDN juga menghadapi berbagai ancaman keamanan yang semakin kompleks, mulai dari isolasi tenant yang tidak memadai, serangan terhadap *controller* terpusat, hingga kebocoran data antar tenant. Meskipun SDN menawarkan fleksibilitas luar biasa dalam pengelolaan jaringan multi-tenant, implementasinya masih menghadapi tantangan serius terkait isolasi dan keamanan antar-tenant (Jain & Gupta, 2024; Liu & Xu, 2019; Putra et al., 2023; Zou, 2018). Dalam skenario multi-tenant, setiap tenant berbagi infrastruktur fisik yang sama, namun memerlukan jaminan penuh bahwa lalu lintas data mereka sepenuhnya terisolasi dari lalu lintas data tenant lain. Tanpa mekanisme isolasi yang kuat, berbagai risiko keamanan dapat muncul, seperti kebocoran data, akses tidak sah antar-tenant, dan serangan lateral movement yang membahayakan privasi serta integritas data seluruh pengguna jaringan (Caroline et al., 2022; Giorgetti et al., 2021; Jammal et al., 2014; Sabopelo & Isong, 2024; Shin, 2014; Xavier, 2016).

Permasalahan isolasi tenant yang tidak efektif dapat menimbulkan beberapa dampak negatif yang signifikan. Pertama, serangan tenant-hopping, di mana penyerang dari satu tenant berhasil mengakses sumber daya tenant lain. Kedua, gangguan kinerja (*performance interference*), di mana aktivitas berlebih satu tenant dapat memengaruhi *Quality of Service* (QoS) tenant lain secara keseluruhan. Giorgetti et al. (2021) membuktikan bahwa tanpa mekanisme isolasi performa yang eksplisit, seluruh slice pada infrastruktur yang sama akan mengalami degradasi kualitas secara bersamaan ketika total traffic mendekati kapasitas link, sebuah kondisi yang tidak dapat ditoleransi dalam lingkungan multi-tenant. Ketiga, kompleksitas teknis dalam melakukan network slicing yang aman, mencakup konfigurasi VLAN yang benar, penggunaan protokol tunneling yang tepat, dan pengelolaan aturan aliran granular yang sangat kompleks (Karatas et al., 2017; Kim & Feamster, 2013).

Penelitian terkait di lingkup nasional telah dilakukan oleh Putra et al. (2023) yang mengimplementasikan network slicing menggunakan *FlowVisor* dan *POX Controller* pada topologi *Abilene* dengan metode *TCP port slicing*. Hasil penelitian tersebut mengonfirmasi keberhasilan isolasi flowspace antar slice berdasarkan TCP port dengan nilai QoS dalam kategori baik. Namun penelitian tersebut memiliki keterbatasan pada tiga aspek kritis: (1) mekanisme isolasi berlapis yang menggabungkan *VLAN-based isolation* dan *network slicing* secara terintegrasi; (2) ketahanan isolasi terhadap serangan *cross-tenant* nyata seperti DDoS SYN Flood dalam skenario *multi-tenant* dengan tiga tenant sekaligus; dan (3) pengujian enforcement kebijakan QoS adaptif per-tenant berbasis *Service Level Agreement* menggunakan *controller Ryu* dengan protokol *OpenFlow*.

Berdasarkan identifikasi kesenjangan penelitian tersebut, penelitian ini bertujuan merancang, mengimplementasikan, dan memvalidasi kerangka kerja multi-tenant SDN yang mengintegrasikan mekanisme *VLAN-based isolation*, *network slicing*, dan *adaptive QoS* dalam satu framework komprehensif. Tiga kontribusi utama yang dihasilkan adalah: (1) integrasi hibrid *VLAN-based isolation* dan *network slicing* yang dikontrol terpusat melalui *Ryu controller*, memungkinkan isolasi berlapis dari Layer 2 hingga application layer; (2) mekanisme flow rules dinamis dengan *policy enforcement* berbasis *OpenFlow* yang secara otomatis mendeteksi dan memblokir traffic anomali antar tenant; serta (3) sistem QoS adaptif per-tenant dengan bandwidth allocation real-time berbasis SLA. Validasi dilakukan melalui

serangkaian eksperimen terstruktur mencakup pengukuran kualitas jaringan dan simulasi serangan DDoS dengan berbagai skala untuk membuktikan efektivitas isolasi yang diusulkan.

METODE PENELITIAN

Jenis dan Pendekatan Penelitian

Penelitian ini menggunakan metode penelitian eksperimental dengan pendekatan *proof of concept*. Metode eksperimental dipilih karena penelitian bertujuan untuk merancang, mengimplementasikan, dan menguji secara empiris mekanisme isolasi tenant dalam *framework* SDN multi-tenant. Pendekatan *proof of concept* bertujuan membuktikan bahwa *framework* yang diusulkan mampu mencapai fungsi atau hasil yang diinginkan, sekaligus sebagai argumen yang memperkuat kelayakan pengembangan prototipe untuk dilanjutkan di lingkungan produksi yang lebih luas. Penelitian dilaksanakan dalam lingkungan simulasi terkontrol menggunakan emulator Mininet yang merepresentasikan topologi jaringan SDN multi-tenant secara realistis.

Alur penelitian mencakup enam tahapan sistematis: (1) Studi Literatur – pengumpulan dan analisis referensi terkait SDN, arsitektur multi-tenant, mekanisme isolasi, network slicing, dan protokol OpenFlow dari jurnal ilmiah, paper konferensi, dan dokumentasi teknis; (2) Analisis Kebutuhan – identifikasi kebutuhan fungsional dan non-fungsional sistem; (3) Perancangan Sistem – merancang arsitektur *framework*, topologi jaringan, mekanisme VLAN tagging, dan algoritma deteksi anomali; (4) Implementasi – mengimplementasikan rancangan menggunakan Mininet, Ryu controller, dan Open vSwitch; (5) Pengujian dan Evaluasi – pengujian fungsional, keamanan, performa, dan skalabilitas; (6) Dokumentasi – penyusunan laporan hasil penelitian.

HASIL DAN PEMBAHASAN

Hasil Pengujian Konektivitas Jaringan SDN

a. Konektivitas Tanpa *Network Slicing*

Pengujian konektivitas tanpa *network slicing* dilakukan menggunakan perintah pingall pada emulator Mininet. Topologi yang dibangun terdiri dari 4 switch (s0, s1, s2, s3) dan 6 host (h1a, h1b, h1c, h2a, h2b, h2c) yang seluruhnya berada dalam satu subnet flat network 10.0.0.0/24 tanpa isolasi antar host. Pada kondisi ini, seluruh host dapat saling berkomunikasi secara bebas, dan hasil pengujian menunjukkan 0% packet loss dengan 30/30 paket berhasil diterima.

Kondisi tanpa *network slicing* ini memperlihatkan bahwa seluruh tenant berbagi sumber daya jaringan yang sama tanpa pembatasan bandwidth, prioritas traffic, maupun isolasi aliran data. Akibatnya, potensi interferensi antar tenant sangat tinggi. Jaringan flat tanpa slicing ini rentan terhadap berbagai ancaman: trafik dari satu tenant yang tinggi dapat merebut bandwidth tenant lain, paket yang seharusnya bersifat privat dapat dilihat oleh tenant lain melalui teknik sniffing sederhana, dan serangan yang ditujukan pada satu tenant secara langsung berdampak pada kinerja seluruh pengguna jaringan.

b. Konektivitas Dengan *Network Slicing*

Setelah *framework network slicing* diterapkan, jaringan berhasil dibagi menjadi tiga slice yang sepenuhnya terisolasi. Hasil pengujian pingall menunjukkan hanya 6 dari 30 paket berhasil diterima (20%), dengan pola komunikasi yang sangat terstruktur: keberhasilan penuh

pada seluruh komunikasi intra-tenant (6/6 paket berhasil di antara host-host dalam satu tenant) dan kegagalan total pada seluruh komunikasi lintas tenant (0/24 paket berhasil). Kondisi ini secara akademis bukan merupakan indikasi kegagalan sistem. Sebaliknya, drop rate 80% tersebut justru merupakan konfirmasi keberhasilan penegakan kebijakan isolasi antar-tenant oleh controller Ryu melalui flow rules OpenFlow.

Verifikasi konektivitas menggunakan perintah net pada CLI Mininet mengonfirmasi bahwa setiap kelompok host hanya dapat berkomunikasi dalam slice jaringannya sendiri sesuai subnet yang telah ditentukan. Switch s1 mengelola host Tenant A, switch s2 mengelola host Tenant B, dan switch s3 mengelola host Tenant C, yang kesemuanya terhubung ke switch inti s0 dan dikendalikan oleh controller c0. Hasil ini secara empiris membuktikan bahwa SDN dengan pendekatan network slicing merupakan solusi yang efektif untuk manajemen jaringan multi-tenant yang aman dan terisolasi, tanpa memerlukan perubahan pada infrastruktur fisik jaringan.

Pengukuran *Quality of Service* (QoS)

a. QoS Jaringan SDN Tanpa *Network Slicing*

Pengukuran QoS dilakukan menggunakan iPerf3 untuk throughput, ping untuk delay, dan kombinasi keduanya untuk packet loss dan jitter. Pengukuran dilakukan selama interval waktu yang konsisten untuk setiap link guna memastikan representativitas data. Tabel 1 menyajikan hasil pengukuran QoS per link pada kondisi tanpa network slicing.

Tabel 1. Hasil Pengukuran QoS SDN Tanpa Network Slicing

No	Link	Throughput (Kbps)	Packet Loss (%)	Delay (ms)	Jitter (ms)
1	s1-eth2 / h1a	464.755	1,768	0,392	6,684
2	s2-eth2 / h2a	183.228	1,664	0,976	0,354
3	s1-eth3 / h1b	442.211	1,595	0,404	2,160
4	s2-eth3 / h2b	554.649	1,628	0,322	3,046
5	s3-eth2 / h1c	614.868	1,446	0,289	3,816
6	s3-eth3 / h2c	681.998	1,630	0,261	4,235
Rata-rata		490.284	1,621	0,440	3,324
Indeks TIPHON		4	4	4	3
Kategori		Sangat Baik	Sangat Baik	Sangat Baik	Baik

Analisis terhadap Tabel 1 menunjukkan bahwa jaringan SDN tanpa network slicing sudah beroperasi dengan kualitas yang baik. Throughput bervariasi antara 183.228 Kbps (h2a, terendah) hingga 681.998 Kbps (h2c, tertinggi), dengan rata-rata 490.284 Kbps yang mendapat indeks 4 (sangat baik). Variasi throughput yang cukup besar (koefisien variasi ~38%) mengindikasikan tidak adanya mekanisme alokasi bandwidth yang adil antar tenant – kondisi noisy neighbor yang berpotensi merugikan tenant tertentu. Parameter packet loss sangat baik di kisaran 1,446% hingga 1,768%, masih di bawah ambang batas 2% standar TIPHON untuk kategori sangat baik. Parameter delay sangat rendah dengan nilai maksimum 0,976 ms dan

rata-rata 0,440 ms. Parameter jitter menjadi satu-satunya yang tidak mencapai indeks 4 (sangat baik), dengan nilai rata-rata 3,324 ms, meskipun masih jauh di bawah ambang 75 ms standar TIPHON.

b. QoS Jaringan SDN dengan *Network Slicing*

Tabel 2. Hasil Pengukuran QoS SDN dengan *Network Slicing*

No	Link	Throughput (Kbps)	Packet Loss (%)	Delay (ms)	Jitter (ms)
1	s1-eth2 / h1a	572.320	0,000	0,458	2,077
2	s2-eth2 / h2a	644.210	0,000	0,403	3,980
3	s1-eth3 / h1b	558.128	0,003	0,468	3,900
4	s2-eth3 / h2b	522.596	0,000	0,497	3,113
5	s3-eth2 / h1c	839.497	1,496	0,210	3,548
6	s3-eth3 / h2c	610.965	0,014	0,293	4,836
Rata-rata		624.619	0,252	0,388	3,575
Indeks TIPHON		4	4	4	3
Kategori		Sangat Baik	Sangat Baik	Sangat Baik	Baik

Hasil pengukuran dengan network slicing menunjukkan peningkatan performa yang signifikan. Throughput rata-rata meningkat menjadi 624.619 Kbps dengan indeks 4. Yang paling mencolok adalah perbaikan drastis pada packet loss: 3 dari 6 link mencatatkan packet loss 0% (nol sama sekali), sementara nilai tertinggi hanya 1,496%, jauh lebih baik dibandingkan kondisi tanpa slicing. Rata-rata packet loss turun menjadi 0,252%, sebuah perbaikan sebesar 84,5%. Penurunan packet loss yang signifikan ini dapat dijelaskan oleh mekanisme QoS queue pada OVS yang memastikan setiap tenant mendapatkan bandwidth yang terjamin: paket tidak perlu bersaing dalam antrian yang sama dengan paket tenant lain, sehingga kemungkinan drop akibat buffer overflow jauh lebih kecil. Delay sedikit membaik menjadi 0,388 ms, sementara jitter mengalami sedikit kenaikan menjadi 3,575 ms, kemungkinan karena overhead pemrosesan queue tambahan, namun tetap dalam kategori baik.

c. Analisis Perbandingan QoS Komprehensif

Tabel 3. Perbandingan Nilai Rata-rata Parameter QoS Sebelum dan Sesudah Network Slicing

Parameter QoS	Tanpa NS	Dengan NS	Perubahan Absolut	Perubahan (%)	Indeks Tanpa NS	Indeks Dengan NS
Throughput (Kbps)	490.284	624.619	+134.335	+27,4% ↑	4	4
Packet Loss (%)	1,621	0,252	-1,369	-84,5% ↓	4	4
Delay (ms)	0,440	0,388	-0,052	-11,8% ↓	4	4
Jitter (ms)	3,324	3,575	+0,251	+7,6% ↑	3	3
Rata-rata Indeks	—	—	—	—	3,75	3,75

Berdasarkan Tabel 3, dapat disimpulkan bahwa penerapan *network slicing* secara konsisten meningkatkan kualitas jaringan pada parameter *throughput*, *packet loss*, dan *delay*, dengan hanya sedikit peningkatan pada jitter yang tetap berada dalam batas kategori baik. Pola ini sejalan dengan temuan Putra et al. (2023) bahwa penerapan *network slicing* mengharuskan setiap paket diperiksa melalui *flowspace* sebelum diteruskan, menambah sedikit *overhead* pemrosesan yang tercermin pada kenaikan jitter. Secara keseluruhan, efek positif *network slicing* jauh lebih besar dari efek negatifnya, dengan indeks rata-rata QoS tetap sama di 3,75 namun dengan kualitas yang lebih konsisten dan terprediksi antar tenant.

Pengujian Ketahanan Terhadap Serangan DDoS SYN Flood

a. Metodologi Pengujian Serangan

Simulasi serangan DDoS SYN Flood dilakukan menggunakan tool hping3 yang merupakan alat pengujian jaringan berbasis command line yang mendukung pengiriman paket TCP/UDP/ICMP secara terprogram. Perintah serangan yang digunakan adalah: hping3 -S -p 80 -i u100000 --flood [target IP] &. Parameter -S mengaktifkan flag SYN pada setiap paket TCP, -p 80 menentukan port HTTP sebagai target, -i u100000 mengatur interval 100.000 mikrodetik, dan --flood mengaktifkan mode pengiriman paket masif tanpa menunggu respons. Tanda & memungkinkan proses berjalan di background. Menurut Islam et al. (2021), aktivasi parameter flood pada hping3 memaksa mesin penyerang melepaskan paket SYN dalam jumlah yang tidak terbatas, sangat krusial untuk menguji ketahanan mekanisme pertahanan.

Pengujian dilakukan dalam dua kondisi berbeda (tanpa dan dengan *network slicing*) untuk memungkinkan perbandingan langsung. Jumlah penyerang divariasikan dari 5, 10, 15, 20, 25, hingga 30 node aktif secara serentak untuk menganalisis pola degradasi kinerja dan titik kritis kapasitas. Analisis trafik dilakukan secara real-time menggunakan Wireshark yang merekam packet rate pada interface yang relevan.

Analisis Efektivitas Isolasi dan Skalabilitas Framework

a. Analisis Efektivitas Mekanisme Isolasi Berlapis

Keberhasilan *framework* ini dalam mengisolasi *traffic* antar *tenant* dapat dianalisis dari dua perspektif: isolasi logis berbasis VLAN dan isolasi *resource* berbasis *network slicing*. Pada level isolasi logis, mekanisme VLAN tagging memastikan bahwa paket yang masuk ke switch tanpa VLAN tag langsung diberikan tag sesuai port ingress, dan paket dengan VLAN tag yang tidak cocok dengan konfigurasi *port egress* akan di-drop. Proses ini terjadi di level *hardware forwarding* dalam OVS, sehingga *overhead*-nya sangat minimal. Pada level isolasi *resource*, mekanisme *queue-based bandwidth enforcement* pada OVS memastikan bahwa *traffic* dari satu tenant tidak dapat mengkonsumsi bandwidth yang telah dialokasikan untuk tenant lain, bahkan ketika trafik burst terjadi secara bersamaan.

Efektivitas pendekatan berlapis ini terbukti dari hasil pengujian: tidak ada satu pun paket yang berhasil melintas dari satu tenant ke tenant lain selama seluruh durasi pengujian, baik dalam kondisi normal maupun saat serangan DDoS berlangsung. Isolation rate yang dicapai adalah 100% untuk komunikasi antar subnet tenant yang berbeda, sebuah angka yang membuktikan integritas mekanisme isolasi yang diimplementasikan. Hasil ini sejalan dengan temuan Irum et al. (2019) yang menunjukkan bahwa kombinasi VLAN isolation dengan flow-

based enforcement menghasilkan tingkat isolasi yang lebih tinggi dibandingkan masing-masing teknik secara individual.

b. Analisis *Flow Setup Time* dan *Overhead Controller*

Salah satu aspek kritis dalam implementasi SDN reaktif adalah *flow setup time*, yaitu waktu yang dibutuhkan controller untuk merespons paket pertama yang tidak memiliki *flow rule* yang cocok (*packet-in event*) dan menginstall flow rule baru ke switch yang relevan. Pengukuran menggunakan *Wireshark* menunjukkan bahwa *flow setup time* dalam *framework* ini berkisar antara 2-8 ms untuk instalasi flow rule tunggal, sebuah nilai yang sangat baik untuk lingkungan emulasi berbasis Mininet.

Overhead controller Ryu dalam menangani *traffic multi-tenant* relatif rendah berkat optimasi yang diterapkan, di antaranya: penggunaan *caching flow entries* per tenant untuk menghindari pemrosesan berulang, *proactive flow installation* untuk pola traffic yang sudah diprediksi (seperti *gateway ARP*), dan *idle timeout* yang dikonfigurasi secara optimal berdasarkan karakteristik traffic masing-masing tenant. Pendekatan optimasi ini konsisten dengan rekomendasi Panda et al. (2021) mengenai konfigurasi *idle timer* yang optimal untuk meminimalkan delay rata-rata paket. *Overhead* keseluruhan *controller Ryu* dalam mengelola tiga tenant secara bersamaan tidak menyebabkan peningkatan *delay* yang terukur (perbedaan *delay* antara kondisi tanpa dan dengan *slicing* hanya 0,052 ms), mengkonfirmasi bahwa *framework* ini skalabel untuk jumlah tenant yang lebih besar.

Tabel 4. Ringkasan Metrik Performa Framework Multi-Tenant SDN yang Diusulkan

Aspek	Putra et al. (2023)	Zou et al. (2018)	Kurniawan et al. (2021)	Penelitian Ini
<i>Controller</i>	POX + FlowVisor	Ryu	POX + FlowVisor	Ryu
Mekanisme Isolasi	TCP port slicing	Permission 3-level	FlowVisor slicing	VLAN + Network Slicing
Jumlah <i>Tenant</i> Diuji	2 slice	N/A (framework)	2 slice	3 tenant
Pengujian Serangan	Tidak ada	Tidak ada	Tidak ada	DDoS SYN Flood (6 skenario)
Pengukuran QoS	Ya (4 parameter)	Tidak ada	Ya (parsial)	Ya (4 parameter lengkap)
<i>Overhead Latency</i>	Tidak dikuantifikasi	< 0,1 ms	Tidak dikuantifikasi	< 0,1 ms
Pendekatan	Evaluasi QoS	Framework permission	Isolasi flowospace	Framework terintegrasi

Berdasarkan Tabel 4, framework yang diusulkan dalam penelitian ini memiliki cakupan yang lebih komprehensif dibandingkan penelitian terdahulu. Kebaruan utama terletak pada kombinasi unik antara: (1) penggunaan controller Ryu modern yang lebih powerful dibandingkan POX+FlowVisor; (2) integrasi mekanisme isolasi berlapis (VLAN + network slicing) yang memberikan perlindungan lebih granular; (3) pengujian dengan tiga tenant aktif sekaligus yang lebih merepresentasikan kondisi lingkungan multi-tenant nyata; dan (4) validasi

ketahanan terhadap serangan DDoS nyata dengan enam skenario berbeda yang belum dilakukan oleh penelitian-penelitian sebelumnya.

Limitasi dan Arah Pengembangan

Meskipun hasil penelitian ini menunjukkan keefektifan yang tinggi, terdapat beberapa limitasi yang perlu diakui secara akademis. Pertama, seluruh pengujian dilakukan dalam lingkungan emulasi Mininet yang berjalan pada satu mesin fisik. Karakteristik latency, throughput, dan perilaku jaringan dalam emulasi Mininet mungkin tidak sepenuhnya identik dengan jaringan fisik nyata, terutama untuk aspek jitter yang sangat sensitif terhadap kondisi hardware. Pengujian pada infrastruktur fisik dengan *switch OpenFlow* nyata diperlukan untuk validasi yang lebih komprehensif.

Kedua, modul *Anomaly Detector* dalam *framework* saat ini menggunakan pendekatan threshold-based yang sederhana berdasarkan statistik *packet rate*. Pendekatan ini efektif untuk mendeteksi serangan volumetrik seperti DDoS, namun mungkin kurang efektif untuk serangan yang lebih halus dan berpola rendah seperti *slow-rate attacks* atau *sophisticated port scanning*. Integrasi algoritma *machine learning* yang dilatih per-tenant, seperti yang direkomendasikan oleh Jain dan Gupta (2024), dapat secara signifikan meningkatkan kemampuan deteksi anomali pada serangan yang lebih canggih.

Ketiga, *framework* saat ini menguji tiga tenant dengan total enam host. *Skalabilitas framework* pada jumlah tenant yang jauh lebih besar (misalnya 50+ tenant dengan ratusan host) perlu dievaluasi, terutama terkait pertumbuhan ukuran *flow table* pada *switch OpenFlow* dan *load* pada *controller Ryu*. Hasil pengujian awal skalabilitas menunjukkan pola scaling yang linier sesuai penambahan tenant, namun evaluasi kuantitatif yang lebih rigorous pada skala besar tetap diperlukan untuk memvalidasi klaim skalabilitas secara menyeluruh.

KESIMPULAN

Penelitian ini berhasil membangun dan memvalidasi kerangka kerja *multi-tenant* pada *Software Defined Network* (SDN) yang mengintegrasikan mekanisme *VLAN-based isolation*, *network slicing*, dan *Quality of Service* (QoS) adaptif melalui kontroler Ryu dengan protokol *OpenFlow* dalam lingkungan emulasi Mininet. Berdasarkan serangkaian eksperimen yang dilakukan, beberapa kesimpulan utama dapat ditarik: *Framework* yang diusulkan berhasil menciptakan isolasi antar-tenant yang kuat. Hasil pengujian pingall menunjukkan hanya 20% paket berhasil diterima (seluruhnya komunikasi intra-tenant), sementara 80% sisanya di-drop sebagai bukti keberhasilan penegakan kebijakan isolasi melalui *flow rules* kontroler Ryu. Tidak ada paket tunggal pun yang berhasil melintas antar subnet tenant yang berbeda.

Penerapan *network slicing* memberikan dampak positif yang signifikan terhadap kualitas jaringan: *throughput* meningkat 27,4% (dari 490.284 Kbps menjadi 624.619 Kbps), *packet loss* menurun drastis 84,5% (dari 1,621% menjadi 0,252%), dan *delay* membaik dari 0,440 ms menjadi 0,388 ms. Hanya parameter jitter yang mengalami sedikit kenaikan (7,6%) akibat *overhead* pemrosesan *flowspace*, namun tetap berada dalam kategori baik berdasarkan standar TIPHON. Kedua kondisi memperoleh indeks rata-rata QoS 3,75, namun jaringan dengan *network slicing* menunjukkan kualitas yang lebih konsisten dan terprediksi.

Pengujian serangan DDoS SYN Flood membuktikan superioritas *framework* yang diusulkan secara dramatis. Tanpa *network slicing*, server target dapat dilumpuhkan dalam 8-50

detik tergantung jumlah penyerang (5–30 node), dengan dampak yang menyebar ke seluruh tenant. Dengan network slicing, Tenant A yang menjadi target serangan langsung tetap beroperasi normal dengan 0% packet loss dan delay rata-rata 0,027 ms. Lebih kritis, Tenant B dan Tenant C yang berada di slice berbeda sama sekali tidak merasakan dampak serangan dan mempertahankan indeks QoS sempurna (4 – Best) sepanjang durasi pengujian.

Pendekatan terintegrasi yang menggabungkan *VLAN-based isolation*, *network slicing*, *dynamic flow rules*, dan *adaptive QoS* dalam satu *framework* terbukti lebih efektif dibandingkan pendekatan single-mechanism. *Framework* ini memberikan *blueprint* implementasi SDN *multi-tenant* yang dapat menjamin isolasi jaringan yang kuat, kualitas layanan yang terprediksi, dan ketahanan terhadap serangan, yang berpotensi diterapkan pada lingkungan jaringan *data center* dan *cloud computing* nyata.

Untuk penelitian selanjutnya, beberapa arah pengembangan yang direkomendasikan meliputi: (1) pengujian *framework* pada infrastruktur fisik nyata dengan skala tenant yang lebih besar; (2) integrasi modul *machine learning* untuk *anomaly detection* yang lebih adaptif dan akurat; (3) pengembangan mekanisme *auto-scaling slice* berdasarkan perubahan kebutuhan SLA secara dinamis; serta (4) evaluasi performa *framework* dalam skenario jaringan dengan topologi yang lebih kompleks dan jumlah *flow rules* yang lebih besar.

REFERENSI

- Barakabitze, A. A. (2019). 5G Network Slicing using SDN and NFV: A Survey. In *arXiv preprint*. <https://arxiv.org/abs/1912.02802>
- Barki, L. (2016). Detection of Distributed Denial of Service Attacks in Software Defined Networks. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. <https://doi.org/10.1109/ICACCI.2016.7732445>
- Bezahaf, M. (2020). Internet Evolution: Critical Issues. *IEEE Internet Computing*, 24(4), 5–14. <https://doi.org/10.1109/MIC.2020.3001519>
- Botez, R. (2023). Efficient Network Slicing with SDN and Heuristic Algorithm for Low Latency Services in 5G/B5G Networks. *Sensors*, 23(13), 6053. <https://doi.org/10.3390/s23136053>
- Caroline, R., Rahmat, B., & Dewanta, F. (2022). Implementation and Analysis of Network Slicing Based on Software Defined Network. *E-Proceeding of Engineering*, 8(6), 3016.
- Elliott, S. (2021). Proof of Concept Research. *Philosophy of Science*, 88(2), 258–280. <https://doi.org/10.1086/711503>
- Giorgetti, A., Scano, D., & Valcarengi, L. (2021). Guaranteeing Slice Performance Isolation With SDN. *IEEE Communications Letters*, 25(11), 3537–3541. <https://doi.org/10.1109/LCOMM.2021.3113175>
- Ifeanyichukwu, O. (2023). Application of Three Probability Distributions. *African Journal of Mathematics and Statistics Studies*, 6(4), 77–80.
- Irum, S. (2019). Multi-tenant Isolation in Software Defined Networks. In *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST)* (Vol. 263). Springer. https://doi.org/10.1007/978-3-030-05195-2_36
- Islam, M. M. (2021). Cyber-Security: Dos Attack Outcomes are Dangerous. *European Journal of Electrical Engineering and Computer Science (EJECE)*, 5(3), 54–59. <https://doi.org/10.24018/ejece.2021.5.3.336>
- Jain, K., & Gupta, A. (2024). Machine Learning-Powered Tenant Isolation in Multi-Tenant Architectures. *Nanotechnology Perceptions*, 20(7), 22–31.

- Jammal, M., Singh, T., Shami, A., Asal, R., & Li, Y. (2014). Software defined networking: State of the art and research challenges. *Computer Networks*, 72, 74–98.
- Karatas, G., Can, F., Dogan, G., Konca, C., & Akbulut, A. (2017). Multi-Tenant Architectures in the Cloud: A Systematic Mapping Study. In *2017 International Conference on Computer Science and Engineering (IDAP)*. <https://doi.org/10.1109/IDAP.2017.8090268>
- Khan, L. U. (2020). Network Slicing: Recent Advances, Taxonomy, Requirements, and Open Research Challenges. *IEEE Access*, 8, 36009–36028.
- Kim, H., & Feamster, N. (2013). Improving Network Management with Software Defined Networking. *IEEE Communications Magazine*, 51(2), 114–119.
- Kreutz, D., Ramos, F. M. V, Verissimo, P., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, 103(1).
- Liu, J., & Xu, Q. (2019). Machine Learning in Software Defined Network. In *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITNEC)*. <https://doi.org/10.1109/ITNEC.2019.8729331>
- Mousa, M., Bahaa-Eldin, A. M., & Sobh, M. (2016). Software Defined Networking Concepts and Challenges. In *2016 11th International Conference on Computer Engineering & Systems (ICCES)*. <https://doi.org/10.1109/ICCES.2016.7821979>
- Nisa, I. S. N. (2024). Analisis Quality of Service (QoS) Menggunakan Standar Parameter TIPHON. *Teknomatika*, 17(1), 1–9.
- Panda, G., Chaudhry, S., & Yuksel, M. (2021). Optimizing Flow Idle Timer in Reactive SDN. In *2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)* (pp. 1–6). <https://doi.org/10.1109/NFV-SDN53031.2021.9665124>
- Putra, F. F., Dewanta, F., & Hertiana, S. N. (2023). Analisis QoS Pengaplikasian Network Slicing pada Topologi Abilene Jaringan SDN Menggunakan FlowVisor dan POX Controller. *Jurnal Infotel*, 15(2), 113–121.
- Sabopelo, R., & Isong, B. (2024). *An Integrated Framework for Controllers Placement and Security in SDN Ecosystem*. 6(1).
- Shin, Y. (2014). The Study on Configuration of Multi-Tenant Networks in SDN Controller. In *2014 International Conference on Information and Communication Technology Convergence (ICACT)*. <https://doi.org/10.1109/ICACT.2014.6779153>
- Tolin, G., & Piccaluga, A. (2024). Managing proof-of-concept (PoC) programs in public research organizations. *European Journal of Innovation Management*, 27(9), 516–541.
- Xavier, M. G. (2016). *Understanding Performance Interference in Multi-Tenant Cloud Databases and Web Applications*.
- Zou, D. (2018). A Fine-grained Multi-tenant Permission Management Framework for SDN and NFV. *IEEE Access*, 6. <https://doi.org/10.1109/ACCESS.2018.2828132>