SMARTICS Journal, Vol.12 No.02 2026. p125-134
ISSN online: 2476-9754, ISSN print: 2623-0429
DOI : https://doi.org/10.21067/smartics.v12i2.14594

Evaluasi Kinerja Software-Defined Networking pada Topologi Fat-Tree untuk Efisiensi Bandwidth

Munaldi[a*], Tole Sutikno[b], Rusydi Umar[c]
[a,b,c]Program Doktor Informatika, Universitas Ahmad Dahlan, Yogyakarta, Indonesia
[*]correspondence email : 2536083056@webmail.uad.ac.id

Abstract*—The increasing demand for digital services requires network architectures that are flexible, programmable, and efficient in utilizing available bandwidth. This study evaluates the performance of Software-Defined Networking (SDN) and a conventional network on an identical Fat-tree k = 4 topology. A quantitative experiment was conducted in Mininet using Open vSwitch, OpenDaylight, OpenFlow 1.3, iPerf3, and ping. Traffic loads were configured at 25%, 50%, and 75% of a 100 Mbps bottleneck-link capacity, with ten repetitions per scenario. Performance was assessed using throughput, round-trip latency, packet loss, and bandwidth efficiency. SDN achieved an average throughput of 81.8 Mbps compared with 60.8 Mbps for the conventional network, reduced average latency from 21.8 ms to 17.1 ms, and reduced packet loss from 2.3% to 0.8%. Bandwidth efficiency increased from 60.8% to 81.8%, equivalent to a relative improvement of 34.5%. A paired-sample t-test across the three load-level means showed a significant throughput difference ($p = 0.018$), whereas latency ($p = 0.088$) and packet loss ($p = 0.105$) were not significant at $\alpha = 0.05$. The findings indicate that centralized flow control and multipath utilization improve performance under increasing traffic loads, although validation on physical testbeds and larger paired samples remains necessary.

***Index Terms*—A Bandwidth Efficiency; Fat-tree; Quality of Service; Software-Defined Networking; Traffic Engineering.**

Abstrak*—Peningkatan kebutuhan layanan digital menuntut arsitektur jaringan yang fleksibel, dapat diprogram, dan efisien dalam memanfaatkan bandwidth. Penelitian ini mengevaluasi kinerja Software-Defined Networking (SDN) dan jaringan konvensional pada topologi Fat-tree k = 4 yang identik. Eksperimen kuantitatif dilakukan menggunakan Mininet, Open vSwitch, OpenDaylight, OpenFlow 1.3, iPerf3, dan ping. Beban trafik ditetapkan sebesar 25%, 50%, dan 75% dari kapasitas link bottleneck 100 Mbps, dengan sepuluh pengulangan pada setiap skenario. Kinerja dievaluasi berdasarkan throughput, latensi round-trip, packet loss, dan efisiensi bandwidth. SDN menghasilkan throughput rata-rata 81,8 Mbps dibandingkan 60,8 Mbps pada jaringan konvensional, menurunkan latensi rata-rata dari 21,8 ms menjadi 17,1 ms, serta menurunkan packet loss dari 2,3% menjadi 0,8%. Efisiensi bandwidth meningkat dari 60,8% menjadi 81,8%, atau meningkat relatif sebesar 34,5%. Uji paired sample t-test terhadap tiga pasangan rata-rata tingkat beban menunjukkan perbedaan throughput yang signifikan ($p = 0,018$), sedangkan latensi ($p = 0,088$) dan packet loss ($p = 0,105$) belum signifikan pada $\alpha = 0,05$. Hasil ini menunjukkan bahwa pengendalian flow terpusat dan pemanfaatan multipath meningkatkan performa pada beban trafik yang semakin tinggi, meskipun validasi pada testbed fisik dan sampel berpasangan yang lebih besar masih diperlukan.

***Kata Kunci*—Efisiensi Bandwidth; Fat-tree; Quality of Service; Rekayasa Trafik; Software-Defined Networking.**

I. PENDAHULUAN

Transformasi digital meningkatkan ketergantungan organisasi pada jaringan untuk mendukung sistem informasi, komputasi awan, *Internet of Things*, konferensi video, dan layanan real-time. Variasi jumlah pengguna dan karakteristik aplikasi membuat pola trafik semakin dinamis. Oleh karena itu, kualitas layanan tidak hanya ditentukan oleh kapasitas link, tetapi juga oleh kemampuan jaringan dalam mengatur jalur, antrean, dan prioritas trafik secara adaptif.

Pada jaringan konvensional, fungsi control plane dan data plane melekat pada perangkat yang sama. Kebijakan forwarding, routing, dan pencegahan loop dikonfigurasi secara terdistribusi. Pendekatan tersebut tetap relevan untuk jaringan sederhana, tetapi pada jaringan yang berkembang dapat meningkatkan beban administrasi, memunculkan inkonsistensi konfigurasi, serta memperlambat respons terhadap perubahan kebutuhan trafik [1]–[3].

Software-Defined Networking (SDN) memisahkan control plane dari data plane dan menempatkan keputusan jaringan pada controller yang mempunyai pandangan logis terhadap topologi. Melalui protokol southbound seperti OpenFlow, controller dapat memasang flow rule, mengubah jalur, dan menerapkan traffic engineering secara terpusat. Arsitektur ini mendukung otomatisasi, monitoring berbasis flow, dan pemanfaatan multipath yang lebih terkoordinasi [1], [2].

Penelitian mutakhir menunjukkan bahwa kemampuan SDN dalam membaca kondisi jaringan dan menyesuaikan jalur dapat memperbaiki *load balancing* dan *Quality of Service*. Pei et al. [5] menunjukkan bahwa rekayasa trafik berbasis *deep reinforcement learning* dapat mengurangi kemacetan dan memperbaiki distribusi beban. Gunavathie dan Umamaheswari [6] melaporkan bahwa routing berbasis prediksi trafik meningkatkan throughput sekaligus menurunkan delay dan jitter. Ashraf et al. [7] menegaskan bahwa evolusi trafik perlu dikelola dengan mempertimbangkan utilisasi link dan overhead pemasangan rule.

Topologi Fat-tree relevan untuk pengujian karena menyediakan beberapa jalur berbiaya sama pada lapisan core, aggregation, dan edge. Pada konfigurasi k = 4, jaringan terdiri atas empat core switch, delapan aggregation switch, delapan edge switch, dan enam belas host. Struktur ini memungkinkan evaluasi yang jelas terhadap kemampuan arsitektur jaringan dalam mendistribusikan flow dan memanfaatkan jalur redundan [4], [12].

Meskipun demikian, sentralisasi logis pada SDN tidak selalu menjamin performa yang lebih baik. Interaksi Packet-In/Flow-Mod, jarak controller–switch, kapasitas controller, dan mekanisme failover dapat menambah latensi atau membentuk bottleneck baru. Studi mengenai data center dan controller SDN menunjukkan bahwa control-plane delay dan controller workload tetap menjadi isu penting ketika jumlah flow serta skala topologi meningkat [4], [10], [13].

State of the art memperlihatkan kemajuan pada algoritma traffic engineering, prediksi trafik, dan load balancing. Namun, banyak studi berfokus pada algoritma atau perbandingan antar-controller, sedangkan bukti eksperimen langsung antara jaringan konvensional dan SDN pada topologi, kapasitas link, pasangan host, durasi, dan tingkat beban yang sama masih terbatas. Selain itu, efisiensi bandwidth sering dilaporkan tanpa denominator yang eksplisit sehingga hasil sulit diverifikasi.

Berdasarkan kesenjangan tersebut, novelty penelitian ini terletak pada evaluasi terkontrol jaringan konvensional dan SDN pada Fat-tree k = 4 menggunakan tiga tingkat beban trafik yang identik, definisi efisiensi bandwidth berbasis throughput aktual terhadap kapasitas link bottleneck, serta validasi kuantitatif melalui paired sample t-test pada rata-rata setiap tingkat beban.

Penelitian ini bertujuan untuk: (1) membandingkan throughput jaringan konvensional dan SDN pada beban 25%, 50%, dan 75%; (2) mengevaluasi latency, packet loss, dan efisiensi bandwidth setelah penerapan kontrol terpusat; serta (3) mengidentifikasi kecenderungan performa ketika beban meningkat dan menilai signifikansi perbedaannya.

Kontribusi penelitian ini adalah sebagai berikut: (1) mengevaluasi kinerja Software-Defined Networking pada topologi Fat-tree menggunakan throughput, latency, packet loss, dan efisiensi bandwidth; (2) membandingkan performa jaringan konvensional dan SDN pada beberapa tingkat beban trafik dalam lingkungan pengujian yang terkendali; (3) memvalidasi hasil eksperimen menggunakan paired sample t-test sehingga perbedaan performa dapat dibuktikan secara kuantitatif; dan (4) memberikan rekomendasi penerapan SDN untuk meningkatkan efisiensi bandwidth pada jaringan institusi.

Artikel ini disusun sebagai berikut. Bagian 2 menjelaskan desain eksperimen, lingkungan pengujian, topologi, skenario trafik, parameter pengukuran, prosedur pengambilan data, dan analisis statistik. Bagian 3 menyajikan hasil serta pembahasan komparatif, Bagian 4 menguraikan keterbatasan penelitian, dan Bagian 5 menyimpulkan temuan beserta arah penelitian selanjutnya.

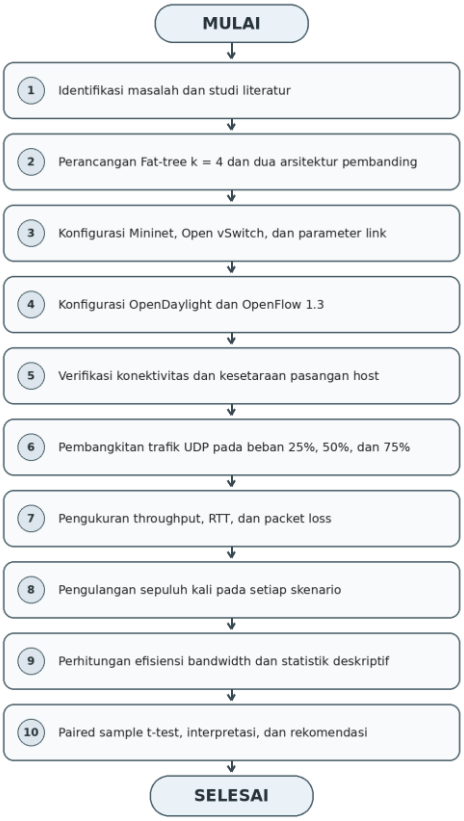
II. METODE PENELITIAN

A. Desain Penelitian dan Tahapan Eksperimen

Penelitian ini menggunakan desain eksperimen kuantitatif berpasangan. Dua arsitektur dibangun pada topologi dan parameter link yang sama. Skenario konvensional menjalankan Open vSwitch dalam mode standalone dengan forwarding normal dan Spanning Tree Protocol untuk mencegah loop, sehingga sebagian jalur redundan tidak digunakan secara bersamaan. Skenario SDN menjalankan Open vSwitch dalam mode secure dan terhubung ke OpenDaylight; flow rule dipasang melalui OpenFlow 1.3 untuk mendistribusikan flow pada jalur yang tersedia. Pasangan host, offered load, durasi, dan urutan pengujian

dibuat identik agar selisih hasil terutama merefleksikan perbedaan mekanisme pengendalian jaringan.

Tahapan penelitian meliputi identifikasi masalah, studi literatur, perancangan Fat-tree k = 4, konfigurasi lingkungan virtual, verifikasi konektivitas controller–switch, pembangkitan trafik, pengukuran QoS, pengulangan eksperimen, perhitungan efisiensi bandwidth, analisis statistik, dan interpretasi hasil. Alur penelitian ditunjukkan pada Gambar 1.



Gambar 1. Alur penelitian dan tahapan eksperimen

B. Lingkungan Pengujian

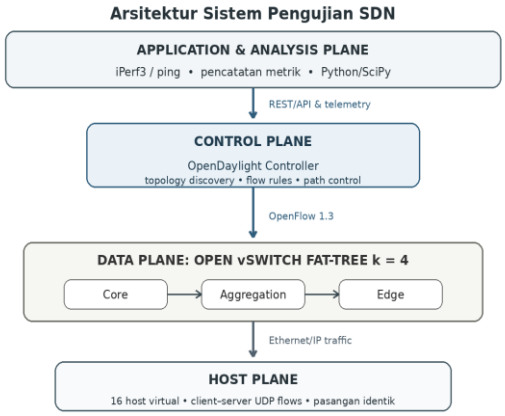
Pengujian dilakukan pada lingkungan virtual agar konfigurasi dapat dikendalikan dan direplikasi. Mininet membangun host, Open vSwitch, dan link virtual; OpenDaylight bertindak sebagai controller; OpenFlow 1.3 digunakan pada southbound interface; iPerf3 membangkitkan trafik UDP dan mencatat throughput serta packet loss; sedangkan ping digunakan untuk memperoleh round-trip time. Analisis deskriptif dan paired sample t-test dilakukan menggunakan Python dan SciPy.

Tabel 1. Spesifikasi lingkungan pengujian

Komponen	Spesifikasi
Prosesor	Intel Core i7-11700, 3,6 GHz, 8 core
Memori	16 GB DDR4
Sistem operasi	Ubuntu 20.04 LTS
Emulator jaringan	Mininet 2.3.0
Controller SDN	OpenDaylight Phosphorus SR1
Southbound protocol	OpenFlow 1.3
Virtual switch	Open vSwitch
Traffic generator	iPerf3 (UDP)
Latency probe	ping/ICMP
Analisis statistik	Python 3.8+ dan SciPy

C. Arsitektur Sistem

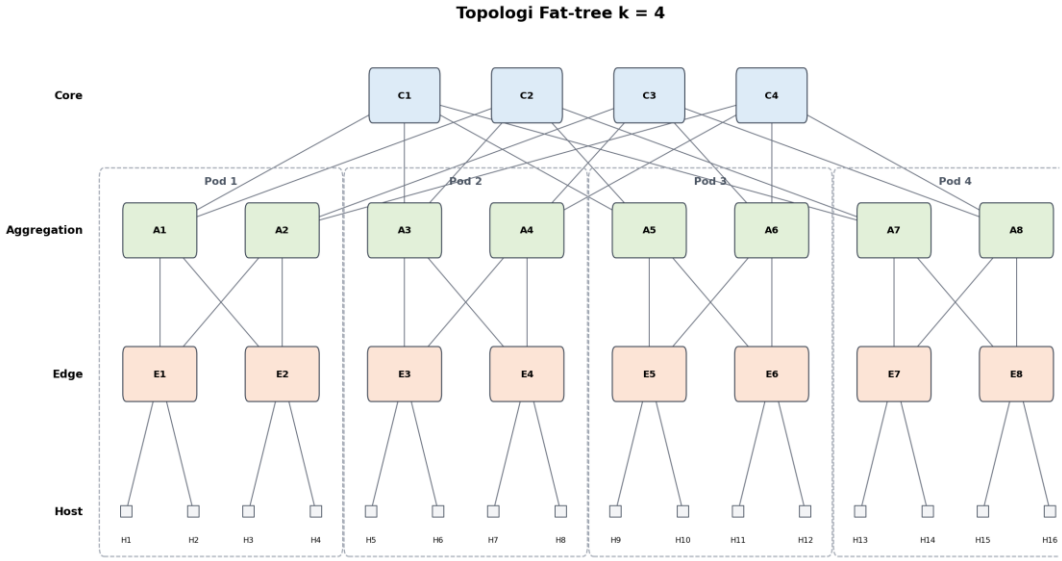
Arsitektur pengujian memisahkan application/analysis plane, control plane, data plane, dan host plane. OpenDaylight menerima informasi topologi dan statistik flow, kemudian berkomunikasi dengan Open vSwitch melalui OpenFlow 1.3. iPerf3 dan ping berjalan pada host untuk menghasilkan trafik serta metrik. Hubungan antarkomponen ditunjukkan pada Gambar 2.



Gambar 2. Arsitektur sistem dan pemisahan control plane–data plane

D. Topologi Fat-tree dan Konfigurasi Link

Topologi yang digunakan adalah Fat-tree k = 4 yang secara kanonik terdiri atas empat pod. Setiap pod memiliki dua aggregation switch dan dua edge switch; setiap edge switch terhubung ke dua host. Dengan demikian, topologi mencakup 4 core switch, 8 aggregation switch, 8 edge switch, 16 host, dan 48 link virtual. Struktur multipath ini memungkinkan perbandingan antara jalur redundan yang dibatasi STP pada skenario konvensional dan jalur yang dapat dimanfaatkan melalui flow rule pada skenario SDN.



Gambar 3. Topologi Fat-tree k = 4 pada lingkungan pengujian

Tabel 2. Konfigurasi topologi dan link

Komponen/Parameter	Jumlah/Nilai	Keterangan
Nilai k	4	Port logis per switch
Pod	4	2 aggregation dan 2 edge per pod
Core switch	4	Backbone
Aggregation switch	8	Dua per pod
Edge switch	8	Dua per pod
Host	16	Dua per edge switch
Jumlah link	48	16 host–edge, 16 edge–aggregation, 16 aggregation–core
Bandwidth link	100 Mbps	Kapasitas nominal setiap link
Delay link	5 ms	Delay emulasi per link
Queue size	100 paket	Batas antrean link
Skenario SDN	OpenDaylight/OpenFlow 1.3	OVS mode secure
Skenario konvensional	OVS standalone + STP	Tanpa external controller

E. Skenario dan Konfigurasi Trafik

Beban trafik dibagi menjadi rendah (25 Mbps), sedang (50 Mbps), dan tinggi (75 Mbps), yang ekuivalen dengan 25%, 50%, dan 75% kapasitas link 100 Mbps. Setiap skenario dijalankan selama 60 detik dan diulang sepuluh kali pada masing-masing arsitektur, sehingga diperoleh 30 observasi per arsitektur. UDP dipilih agar offered load dapat ditetapkan secara eksplisit dan tidak dipengaruhi mekanisme congestion control.

Tabel 3. Skenario pengujian berdasarkan tingkat beban

Skenario	Offered load	Durasi	Pengulangan
S1	25 Mbps (25%)	60 detik	10 kali
S2	50 Mbps (50%)	60 detik	10 kali
S3	75 Mbps (75%)	60 detik	10 kali

Pasangan host sumber–tujuan dibuat sama pada kedua arsitektur dan dijalankan secara berurutan. Interval pelaporan iPerf3 ditetapkan 1 detik, ukuran datagram 1.470 byte, dan probe RTT dilakukan menggunakan ping selama trafik berlangsung.

Tabel 4. Konfigurasi pembangkitan trafik dan pencatatan data

Parameter	Nilai konfigurasi
Traffic generator	iPerf3
Protokol	UDP
Offered load	25, 50, dan 75 Mbps
Durasi	60 detik per pengulangan
Pengulangan	10 kali per skenario dan arsitektur
Ukuran datagram	1.470 byte
Reporting interval	1 detik
Mode	Client–server
Latency probe	ping/ICMP
Output utama	Throughput, RTT, dan packet loss

F. Parameter Pengukuran

Definisi parameter mengacu pada prinsip benchmarking jaringan dalam RFC 2544 dan parameter performa transfer paket IP pada ITU-T Y.1540 [14], [15]. Throughput menyatakan jumlah bit yang berhasil diterima per satuan waktu. Latency dilaporkan sebagai rata-rata round-trip time (RTT). Packet loss adalah proporsi paket yang tidak diterima dibandingkan paket yang dikirim. Efisiensi bandwidth menyatakan persentase throughput aktual terhadap kapasitas link bottleneck nominal 100 Mbps.

$$\text{Throughput (Mbps)} = (\text{Data diterima (byte)} \times 8) / (\text{Durasi (s)} \times 10^6) \quad (1)$$

$$\text{Latency/RTT rata-rata (ms)} = \Sigma RTT_i / N \quad (2)$$

$$\text{Packet Loss (\%)} = ((\text{Paket dikirim} - \text{Paket diterima}) / \text{Paket dikirim}) \times 100\% \quad (3)$$

$$\text{Efisiensi Bandwidth (\%)} = (\text{Throughput aktual} / 100 \text{ Mbps}) \times 100\% \quad (4)$$

$$\text{Peningkatan Relatif (\%)} = ((\text{Nilai SDN} - \text{Nilai Konvensional}) / \text{Nilai Konvensional}) \times 100\% \quad (5)$$

Untuk latency dan packet loss, perbaikan dihitung dengan pembilang Nilai Konvensional − Nilai SDN. Sebagai contoh, efisiensi bandwidth SDN berdasarkan throughput rata-rata 81,8 Mbps adalah (81,8/100) × 100% = 81,8%. Dibandingkan jaringan konvensional 60,8%, peningkatan relatifnya adalah ((81,8 − 60,8)/60,8) × 100% = 34,5%, sedangkan selisih absolutnya 21,0 poin persentase.

G. Prosedur Pengambilan Data

1) Membangun Fat-tree k = 4 dengan parameter link pada Tabel 2.
2) Menjalankan skenario konvensional, mengaktifkan forwarding normal dan STP, lalu memverifikasi konektivitas seluruh host.
3) Menjalankan OpenDaylight, menghubungkan Open vSwitch melalui OpenFlow 1.3, dan memverifikasi registrasi switch.
4) Menetapkan pasangan client–server yang sama pada kedua arsitektur.
5) Membangkitkan trafik UDP sesuai S1, S2, dan S3 serta melakukan probe RTT secara paralel.
6) Mencatat throughput, RTT, paket dikirim, dan paket diterima; kemudian mengulangi setiap skenario sepuluh kali.
7) Menghitung statistik deskriptif, efisiensi bandwidth, perbaikan relatif, dan paired sample t-test.

H. Analisis Statistik

Paired sample t-test dipilih karena nilai jaringan konvensional dan SDN diperoleh pada kondisi beban yang sama. Hipotesis nol (H_0) menyatakan rata-rata selisih performa kedua arsitektur sama dengan nol, sedangkan hipotesis alternatif (H_1) menyatakan rata-rata selisih tidak sama dengan nol. Uji dilakukan dua arah pada tingkat signifikansi $\alpha = 0,05$ menggunakan Python/SciPy. Tabel inferensial menggunakan tiga pasangan rata-rata tingkat beban ($n = 3$; $df = 2$), sehingga hasil p-value diperlakukan sebagai bukti eksploratif dan perlu dikonfirmasi menggunakan seluruh pasangan data mentah pada studi lanjutan.

III. HASIL DAN PEMBAHASAN

A. Data Pengujian Berulang pada Skenario S2

Tabel 5 menyajikan sepuluh pengulangan jaringan SDN pada beban sedang 50 Mbps. Data ini digunakan untuk menunjukkan variasi pengukuran sebelum hasil setiap tingkat beban diringkas dan dibandingkan dengan jaringan konvensional.

Tabel 5. Hasil sepuluh pengulangan SDN pada beban sedang (50%)

Iterasi	Throughput (Mbps)	Latency (ms)	Packet loss (%)
1	82,4	15,2	0,8
2	84,1	14,6	0,6
3	83,0	15,0	0,7
4	85,2	14,3	0,5
5	82,8	15,1	0,8
6	83,7	14,8	0,7
7	84,5	14,5	0,6
8	81,9	15,5	0,9
9	83,3	14,9	0,7
10	84,0	14,7	0,6

Statistik deskriptif pada sepuluh pengulangan menunjukkan bahwa throughput rata-rata sebesar 83,49 Mbps (SD = 1,00), latency 14,86 ms (SD = 0,36), dan packet loss 0,69% (SD = 0,12). Penyebaran yang rendah menunjukkan bahwa skenario S2 relatif stabil, tetapi tidak menggantikan kebutuhan untuk melaporkan raw paired observations pada seluruh tingkat beban.

Tabel 6. Statistik deskriptif sepuluh pengulangan SDN pada beban sedang

Parameter	Mean	Minimum	Maksimum	SD
Throughput	83,49 Mbps	81,9 Mbps	85,2 Mbps	1,00
Latency	14,86 ms	14,3 ms	15,5 ms	0,36
Packet loss	0,69%	0,5%	0,9%	0,12

B. Hasil Pengukuran Throughput

Throughput mencerminkan kemampuan jaringan mengantarkan data aplikasi. Nilai yang menurun ketika offered load meningkat menunjukkan kemacetan, antrean, atau pemilihan jalur yang tidak optimal. Hasil perbandingan disajikan pada Tabel 7.

Tabel 7. Perbandingan throughput jaringan konvensional dan SDN

Kondisi beban	Konvensional (Mbps)	SDN (Mbps)	Peningkatan (%)
Rendah (25%)	78,4	94,2	20,2
Sedang (50%)	61,7	83,5	35,3
Tinggi (75%)	42,3	67,8	60,3
Rata-rata	60,8	81,8	34,5

SDN menghasilkan throughput lebih tinggi pada seluruh tingkat beban. Peningkatan relatif bertambah dari 20,2% pada beban rendah menjadi 60,3% pada beban tinggi. Pola tersebut menunjukkan bahwa manfaat pengendalian flow semakin terlihat ketika antrean dan kontensi link meningkat. Pada skenario konvensional, STP mempertahankan satu pohon forwarding sehingga sebagian kapasitas redundan tidak aktif; pada skenario SDN, flow dapat ditempatkan pada jalur alternatif.

Temuan ini konsisten dengan penelitian Pei et al. [5] dan Gunavathie dan Umamaheswari [6] yang menunjukkan bahwa routing adaptif berdasarkan kondisi trafik dapat meningkatkan pemanfaatan jaringan. Besarnya peningkatan tidak dapat dibandingkan secara langsung karena topologi, algoritma, dan offered load berbeda; yang konsisten adalah arah pengaruh pengaturan jalur terhadap throughput.

C. Hasil Pengukuran Latency

Latency dilaporkan sebagai RTT rata-rata. Kenaikan RTT berkaitan dengan waktu antre, pemrosesan, dan panjang jalur. Hasil perbandingan disajikan pada Tabel 8.

Tabel 8. Perbandingan latency/RTT jaringan konvensional dan SDN

Kondisi beban	Konvensional (ms)	SDN (ms)	Penurunan (%)
Rendah (25%)	12,4	9,8	21,0
Sedang (50%)	18,7	14,9	20,3
Tinggi (75%)	34,2	26,7	21,9
Rata-rata	21,8	17,1	21,3

SDN menurunkan RTT rata-rata dari 21,8 ms menjadi 17,1 ms atau sekitar 21,3%. Pada beban tinggi, selisih 7,5 ms menunjukkan bahwa distribusi flow membantu mengurangi antrean pada link yang padat. Arah temuan ini sejalan dengan routing berbasis prediksi trafik yang dilaporkan mampu menurunkan delay [6].

Interpretasi latency harus mempertimbangkan overhead control plane. Paket pertama dari flow baru dapat memicu Packet-In dan menunggu pemasangan rule. Jia et al. [4], Sheikh et al. [13], dan Abuabdallah [10] menunjukkan bahwa controller workload, first-packet delay, topologi, dan lingkungan virtual memengaruhi performa. Oleh karena itu, penurunan RTT dalam penelitian ini berlaku pada ukuran topologi dan *single-controller* yang diuji.

D. Hasil Pengukuran Packet Loss

Packet loss menunjukkan persentase paket yang tidak sampai ke tujuan. Kehilangan paket dapat meningkat ketika offered load mendekati kapasitas link dan antrean penuh. Hasil pengukuran ditunjukkan pada Tabel 9.

Tabel 9. Perbandingan packet loss jaringan konvensional dan SDN

Kondisi beban	Konvensional (%)	SDN (%)	Penurunan (%)
Rendah (25%)	1,1	0,4	63,6
Sedang (50%)	2,0	0,7	65,0
Tinggi (75%)	3,8	1,3	65,8
Rata-rata	2,3	0,8	65,2

Packet loss rata-rata menurun dari 2,3% menjadi 0,8%, atau berkurang 65,2%. Penurunan paling besar terjadi pada beban tinggi, dari 3,8% menjadi 1,3%. Secara mekanistik, distribusi flow ke jalur yang lebih longgar mengurangi kemungkinan queue overflow. Temuan tersebut sejalan dengan kajian load balancing berbasis keadaan jaringan yang menekan kemacetan [5], [8], [11].

Bagi jaringan institusi, packet loss yang lebih rendah berpotensi meningkatkan kestabilan konferensi video, sistem akademik, dan layanan sensor. Namun, kebijakan QoS tetap perlu mengatur prioritas trafik secara eksplisit; distribusi jalur saja tidak menggantikan klasifikasi layanan dan queue scheduling.

E. Efisiensi Bandwidth

Efisiensi bandwidth dihitung dari throughput aktual dibagi kapasitas link bottleneck nominal 100 Mbps. Denominator ini konsisten dengan konfigurasi link dan memungkinkan hasil diverifikasi secara langsung. Berdasarkan throughput rata-rata, jaringan konvensional mencapai efisiensi 60,8%, sedangkan SDN mencapai 81,8%.

Tabel 10. Perbandingan efisiensi bandwidth

Arsitektur	Efisiensi rata-rata	Perhitungan/interpretasi
Konvensional	60,8%	60,8 Mbps / 100 Mbps
SDN	81,8%	81,8 Mbps / 100 Mbps
Peningkatan relatif	34,5%	Selisih absolut 21,0 poin persentase

Peningkatan relatif 34,5% menunjukkan bahwa SDN mengantarkan proporsi kapasitas nominal yang lebih besar. Hasil ini konsisten dengan tujuan traffic engineering untuk menurunkan maximum link utilization dan memindahkan flow dari link kritis menuju jalur yang kurang padat [5], [7]. Implikasinya, organisasi perlu mengevaluasi ketidakseimbangan utilisasi sebelum menambah kapasitas fisik. Namun, penghematan biaya tidak diukur secara langsung sehingga tidak boleh diklaim sebagai hasil empiris penelitian ini.

F. Uji Paired Sample t-Test

Uji berpasangan membandingkan tiga pasangan rata-rata S1, S2, dan S3 antara jaringan konvensional dan SDN. Dengan n = 3 pasangan dan df = 2, hasil harus ditafsirkan secara hati-hati karena statistical power rendah.

Tabel 11. Hasil paired sample t-test berdasarkan rata-rata tingkat beban

Parameter	Mean konvensional	Mean SDN	|t|	p-value	Keputusan
Throughput	60,80 Mbps	81,83 Mbps	7,44	0,018	Signifikan
Latency	21,77 ms	17,13 ms	3,14	0,088	Tidak signifikan
Packet loss	2,30%	0,80%	2,83	0,105	Tidak signifikan

Throughput menghasilkan $p = 0{,}018$ sehingga H_0 ditolak pada $\alpha = 0{,}05$. Perbedaan throughput didukung secara statistik pada data agregat tingkat beban. Latency menghasilkan $p = 0{,}088$ dan packet loss $p = 0{,}105$; keduanya menunjukkan arah perbaikan secara deskriptif, tetapi bukti inferensial belum cukup kuat. Pengujian lanjutan perlu menggunakan seluruh data berpasangan per iterasi dan melaporkan interval kepercayaan.

G. Pembahasan Komparatif

Tabel 12. Ringkasan hasil perbandingan performa

Parameter	Konvensional	SDN	Interpretasi
Throughput	60,8 Mbps	81,8 Mbps	Naik 34,5%
Latency	21,8 ms	17,1 ms	Turun 21,3%
Packet loss	2,3%	0,8%	Turun 65,2%
Efisiensi bandwidth	60,8%	81,8%	Naik relatif 34,5%

Secara keseluruhan, SDN menunjukkan throughput dan efisiensi bandwidth yang lebih tinggi serta latency dan packet loss yang lebih rendah. Keunggulan terbesar muncul pada beban tinggi, mendukung argumentasi bahwa nilai utama SDN terletak pada kemampuan menyesuaikan kebijakan flow ketika distribusi trafik tidak seimbang. Hasil ini memperkuat studi traffic engineering dan load balancing mutakhir [5]–[8], [11].

Temuan efisiensi juga sejalan dengan Ashraf et al. [7] yang menempatkan utilisasi link dan keterbatasan flow table sebagai sasaran optimasi bersama. Pemanfaatan lebih banyak jalur dapat memperbaiki efisiensi, tetapi pemasangan atau perubahan rule yang terlalu sering menambah overhead. Kebijakan flow harus menyeimbangkan responsivitas, kestabilan, dan beban controller.

Hasil penelitian tidak berarti bahwa SDN otomatis unggul pada seluruh kondisi. Apabila controller tidak stabil, kanal controller–switch terganggu, atau rule flow tidak dirancang dengan baik, performa dapat menurun. Kajian controller menunjukkan variasi kinerja antarsistem dan pentingnya topologi, workload, dan konfigurasi [10], [13], [18]. Karena itu, implementasi institusi sebaiknya dimulai melalui proof of concept pada backbone, laboratorium, atau server farm, diikuti pengujian beban, keamanan, dan failover.

Indikator implementasi juga perlu diperluas melampaui throughput, latency, dan packet loss. Organisasi perlu memantau controller CPU, flow setup time, rule-table occupancy, failover time, keamanan kanal OpenFlow, dan kesiapan sumber daya manusia. Dengan demikian, penelitian ini memberikan bukti awal mengenai manfaat SDN pada Fat-tree $k = 4$ sekaligus menetapkan batas generalisasi hasil.

H. Keterbatasan Penelitian

Pertama, eksperimen dilakukan pada emulator Mininet sehingga belum memasukkan variasi perangkat keras, kualitas kabel, switching latency fisik, dan gangguan operasional pada jaringan produksi. Hasil menunjukkan perilaku relatif dalam lingkungan terkendali, bukan jaminan nilai absolut pada infrastruktur nyata. Kedua, penelitian hanya menggunakan satu OpenDaylight controller. Beban controller, redundansi, konsistensi state, penempatan controller, serta skenario controller failure dan failover belum dievaluasi. Ketiga, pengujian menggunakan trafik UDP sintetis dengan tiga tingkat offered load dan durasi 60 detik. Pola burst, campuran TCP–UDP, trafik aplikasi nyata, variasi ukuran paket, dan trafik east–west yang lebih kompleks belum dicakup. Uji inferensial juga menggunakan tiga pasangan rata-rata skenario sehingga statistical power terbatas. Keempat, penelitian belum mengevaluasi keamanan controller, autentikasi kanal OpenFlow, serangan DDoS, flow-table exhaustion, konsumsi energi, dan biaya migrasi. Penelitian selanjutnya perlu menggunakan testbed fisik, raw paired observations yang lebih besar, multi-controller dan failover, trafik campuran, serta evaluasi keamanan dan biaya operasional.

IV. KESIMPULAN

Penelitian ini menunjukkan bahwa SDN pada topologi Fat-tree k = 4 menghasilkan performa lebih baik daripada jaringan konvensional dalam lingkungan emulasi yang dikendalikan. Throughput rata-rata meningkat dari 60,8 Mbps menjadi 81,8 Mbps, latency menurun dari 21,8 ms menjadi 17,1 ms, dan packet loss menurun dari 2,3% menjadi 0,8%. Dengan kapasitas link bottleneck 100 Mbps, efisiensi bandwidth meningkat dari 60,8% menjadi 81,8%, setara peningkatan relatif 34,5%. Paired sample t-test pada tiga rata-rata tingkat beban menunjukkan perbedaan throughput yang signifikan (p = 0,018), sedangkan latency dan packet loss belum signifikan. SDN dapat direkomendasikan untuk diuji secara bertahap pada segmen jaringan institusi yang memiliki trafik dinamis, dengan perhatian pada kapasitas controller, keamanan, failover, dan kompetensi pengelola. Penelitian berikutnya perlu memvalidasi hasil pada perangkat fisik, memperbesar jumlah pasangan data, serta mengevaluasi multi-controller, mixed traffic, dan failure scenarios.

V. DAFTAR PUSTAKA

[1] Priyadarsini, M., & Bera, P. (2023). Software defined networking architecture, traffic management, security, and placement: A survey. Computer Networks, 192, 108047. https://doi.org/10.1016/j.comnet.2021.108047

[2] Hodaei, A., & Babaie, S. (2023). A survey on traffic management in software-defined networks: Challenges, effective approaches, and potential measures. Wireless Personal Communications, 118, 1507–1534. https://doi.org/10.1007/s11277-021-08100-3

[3] Mohammadi, R., Akleylek, S., Ghaffari, A., & Shirmarz, A. (2022). Taxonomy of traffic engineering mechanisms in software-defined networks: A survey. Telecommunication Systems, 81, 475–502. https://doi.org/10.1007/s11235-022-00947-6

[4] Jia, Z., Liu, Q., & Sun, Y. (2023). sRetor: A semi-centralized regular topology routing scheme for data center networking. Journal of Cloud Computing, 12, 150. https://doi.org/10.1186/s13677-023-00521-8

[5] Pei, X., Sun, P., Hu, Y., Li, D., Chen, B., & Tian, L. (2024). Enabling efficient routing for traffic engineering in SDN with deep reinforcement learning. Computer Networks, 241, 110220. https://doi.org/10.1016/j.comnet.2024.110220

[6] Gunavathie, M. A., & Umamaheswari, S. (2024). Traffic-aware optimal routing in software defined networks by predicting traffic using neural network. Expert Systems with Applications, 239, 122415. https://doi.org/10.1016/j.eswa.2023.122415

[7] Ashraf, U., Ahmed, A., Avallone, S., & Imputato, P. (2024). Traffic evolution in Software Defined Networks. Computer Networks, 255, 110852. https://doi.org/10.1016/j.comnet.2024.110852

[8] Liao, X., Bao, B., Cui, W., & Liu, D. (2024). Load balancing and topology dynamic adjustment strategy for power information system network: A deep reinforcement learning-based approach. Frontiers in Energy Research, 11, 1342854. https://doi.org/10.3389/fenrg.2023.1342854

[9] Dai, W., Foerster, K.-T., Fuchssteiner, D., & Schmid, S. (2023). Load-optimization in reconfigurable data-center networks: Algorithms and complexity of flow routing. ACM Transactions on Modeling and Performance Evaluation of Computing Systems, 8(3), 1–30. https://doi.org/10.1145/3597200

[10] Abuabdallah, A. G. (2026). Performance-based evaluation of SDN controllers: OpenDayLight and RYU. Journal of Electrical Systems and Information Technology, 13, 35. https://doi.org/10.1186/s43067-026-00343-z

[11] Farahi, R. (2025). A comprehensive overview of load balancing methods in software-defined networks. Discover Internet of Things, 5, 6. https://doi.org/10.1007/s43926-025-00098-5

[12] Prabhu, S., Manimozhi, V., Davoodi, A., & García Guirao, J. L. (2024). Fault-tolerant basis of generalized fat trees and perfect binary tree derived architectures. The Journal of Supercomputing, 80, 15783–15798. https://doi.org/10.1007/s11227-024-06053-5

[13] Sheikh, M. N. A., Hwang, I.-S., Raza, M. S., & Ab-Rahman, M. S. (2024). A qualitative and comparative performance assessment of logically centralized SDN controllers via Mininet emulator. Computers, 13(4), 85. https://doi.org/10.3390/computers13040085

[14] Bradner, S., & McQuaid, J. (1999). Benchmarking methodology for network interconnect devices (RFC 2544). RFC Editor. https://doi.org/10.17487/RFC2544

[15] International Telecommunication Union. (2023). Internet protocol data communication service—IP packet transfer and availability performance parameters: Amendment 2 (ITU-T Recommendation Y.1540).

[16] Constantine, B., Forget, G., Geib, R., & Schrage, R. (2015). Traffic management benchmarking (RFC 7640). RFC Editor. https://doi.org/10.17487/RFC7640

[17] Wang, G., Zhao, Y., Huang, J., & Wang, W. (2017). The controller placement problem in software defined networking: A survey. IEEE Network, 31(5), 21–27.

https://doi.org/10.1109/MNET.2017.1600182
[18]Gupta, N., Maashi, M. S., Tanwar, S., Badotra, S., Aljebreen, M., & Bharany, S. (2022). A comparative study of software defined networking controllers using Mininet. Electronics, 11(17), 2715. https://doi.org/10.3390/electronics11172715