IT-EXPLORE
Jurnal Penerapan Teknologi Informasi dan Komunikasi
ISSN 2828-7940, e-ISSN 2829-1727, DOI: 10.24246/itexplore.v4i2.2025.pp196-210
Volume 04 Nomor 02 Tahun 2025
Journal homepage: https://ejournal.uksw.edu/itexplore

Simulasi *traffic* dan performa jaringan SDN menggunakan *ONOS Controller* dan standar TIPHON

Yulius Manahil Hendrawan [1], Wiwin Sulistyo [2]
[1,2]Faculty of Information Technology, Satya Wacana Christian University, Salatiga, Central Java, Indonesia

Informasi Artikel

Riwayat Artikel:

Dikirim:
31 Desember 2024
Direvisi:
20 Maret 2025
Diterima:
26 Juni 2025

Kata Kunci:

Software Defined Network,
Onos Controller
Openflow,
VmWare,
Conventional Network

Keywords:

Software Defined Network,
Onos Controller
Openflow,
VmWare,
Conventional Network

This is an open access article under the CC BY-SA license

ABSTRAK

Pertumbuhan pesat perangkat dan lalu lintas data menimbulkan tantangan dalam pengelolaan jaringan konvensional yang sering mengalami kemacetan akibat manajemen lalu lintas yang statis. *Software-Defined Networking* (SDN) hadir sebagai solusi inovatif dengan memisahkan control plane dan data plane, memungkinkan pengelolaan jaringan yang lebih fleksibel dan terpusat. Studi ini mengevaluasi performa SDN berbasis OpenFlow menggunakan ONOS Controller dan membandingkannya dengan jaringan Wi-Fi rumahan konvensional berdasarkan standar TIPHON. Simulasi dilakukan menggunakan *Virtual Machines* (VMWare) dan Mininet yang terhubung ke ONOS Controller melalui protokol OpenFlow, sementara pengujian jaringan konvensional dilakukan melalui *Command Prompt*. Tiga skenario pengujian diterapkan: 1 host ke 1 host, 2 host ke 1 host, dan 2 host ke 2 host secara bersamaan, dengan parameter delay dan packet loss. Hasil menunjukkan bahwa pada skenario 2 host ke 2 host, jaringan konvensional mengalami peningkatan delay dan packet loss hingga 17%, sedangkan SDN tetap stabil dengan delay di bawah 150 ms dan packet loss 0%. Hal ini menegaskan efisiensi SDN dalam mengelola lalu lintas tinggi secara dinamis dan meningkatkan stabilitas jaringan.

ABSTRACT

The rapid increase in connected devices and data traffic poses challenges for traditional networks, which often suffer from congestion due to static traffic management. Software-Defined Networking (SDN) offers an innovative solution by separating the control and data planes, allowing centralized and flexible network control. This study evaluates the performance of OpenFlow-based SDN using the ONOS Controller and compares it with conventional home Wi-Fi networks based on the TIPHON standard. Simulations were conducted using VMWare and Mininet connected to the ONOS Controller via the OpenFlow protocol, while conventional network tests used the Command Prompt. Three data transmission scenarios were tested: 1 host to 1 host, 2 hosts to 1 host, and 2 hosts to 2 hosts, measuring delay and packet loss. Results showed that in the 2 hosts to 2 hosts scenario, the conventional network experienced a delay and packet loss increase of up to 17%, while SDN remained stable with a delay below 150 ms and 0% packet loss. This highlights SDN's efficiency in managing high traffic dynamically and enhancing overall network stability.

Corresponding Author:
Yulius Manahil Hendrawan
Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana
Jl. Diponegoro No.52-60, Sidorejo, Salatiga, Jawa Tengah, Indonesia
Email: yuliusmanahil@gmail.com

1. PENDAHULUAN

Perkembangan teknologi informasi dan komunikasi yang pesat telah mendorong pertumbuhan lalu lintas data yang semakin tinggi dalam jaringan komputer. Seiring dengan meningkatnya jumlah perangkat yang terhubung dan kebutuhan akan layanan berbasis internet, jaringan konvensional menghadapi berbagai tantangan dalam pengelolaan lalu lintas data secara efisien. Salah satu permasalahan utama dalam jaringan konvensional adalah keterbatasan dalam menangani kepadatan lalu lintas, alokasi sumber daya yang tidak optimal, serta rendahnya fleksibilitas dalam menyesuaikan diri terhadap perubahan dinamika jaringan. Konfigurasi jaringan yang bersifat statis dan ketergantungan terhadap perangkat keras juga menyebabkan kesulitan dalam pengelolaan serta pengoptimalan lalu lintas data, terutama pada skala jaringan yang lebih besar.

Selain itu, Jaringan Konvensional sering kali mengalami ketidakefisienan dalam distibusi traffic data, terutama Ketika menghadapi lonjakan beban data yang tinggi. Proses manajemen traffic data yang statis yang menyebabkan pengiriman data mengalami beban yang kurang optimal, Dimana beberapa *traffic* mengalami penumpukan di salah satu jalur sedangkan jalur lain tidak digunakan secara baik. Ketidakefisienan ini berdampak pada performa jaringan, terutama dalam scenario di mana banyak perangkat harus mengakses layanan berbasis internet secara bersamaan. Misalnya dalam linkup kerja dengan banyaknya server atau dalam bermain game dan melakukan *streaming* yang memerlukan *latency* yang rendah. Sebagai Solusi terhadap tantangan tersebut, Software-*Defined Networking* (SDN) diperkenalkan sebagai solusi inovatif dalam arsitektur jaringan modern [1]. SDN memisahkan *control plane* dan *data plane*, memungkinkan manajemen jaringan yang lebih fleksibel, terpusat, dan terprogram. Dengan pendekatan ini, administrator jaringan dapat mengontrol lalu lintas data secara lebih efisien dan responsif terhadap perubahan beban jaringan. Salah satu standar utama dalam implementasi SDN adalah *OpenFlow*, yang memungkinkan komunikasi antara kontroler pusat dan perangkat jaringan [2].

Dalam penelitian berbasis SDN menggunakan *ONOS Controller* dengan protokol OpenFlow, telah terbukti bahwa pendekatan ini efektif dalam mengelola lalu lintas jaringan dengan pengukuran berdasarkan standar TIPHON. Hasil penelitian ini diharapkan dapat memberikan wawasan mengenai keunggulan SDN dalam menghadapi tantangan jaringan modern serta memberikan dasar bagi pengembangan arsitektur jaringan yang lebih efisien dan adaptif. Dengan memahami kelebihan dan keterbatasan dari implementasi SDN dibandingkan dengan jaringan konvensional, penelitian ini diharapkan dapat berkontribusi dalam pengembangan teknologi jaringan yang lebih optimal untuk kebutuhan masa depan. Dengan meningkatnya kebutuhan akan jaringan yang lebih fleksibel dan efisien, penelitian ini diharapkan dapat memberikan wawasan yang berharga bagi akademisi dan praktisi dalam mengembangkan arsitektur jaringan berbasis SDN yang lebih optimal. Hasil penelitian ini juga dapat menjadi dasar bagi pengembangan lebih lanjut, seperti penerapan SDN dalam jaringan skala besar, integrasi dengan teknologi 5G, serta optimalisasi manajemen lalu lintas berbasis kecerdasan buatan (AI).

2. KAJIAN PUSTAKA

2.1. State of The Art

Penelitian mengenai kualitas dan pengembangan jaringan WiFi telah dilakukan dengan berbagai pendekatan untuk mengatasi tantangan yang sering muncul dalam pengelolaan jaringan. Penelitian Erick tentang kualitas sinyal WiFi di Universitas Muslim Indonesia menggunakan metode *Action Research*, yang melibatkan diagnosis, perencanaan tindakan, implementasi, dan evaluasi untuk mengidentifikasi faktor-faktor penyebab lemahnya sinyal [2]. Sementara itu, Aishah et,al menggunakan metode *Received Signal Strength Indicator* (RSSI) untuk mengukur kekuatan sinyal dan estimasi jarak guna menentukan posisi *access point* yang lebih optimal [3]. Selain itu, penelitian Cahyono [4].berfokus pada analisis roaming jaringan WiFi di Universitas Kristen Satya Wacana, mengevaluasi perpindahan pengguna antar *access point* tanpa memerlukan autentikasi ulang.

Beberapa penelitian juga menyoroti pengembangan sistem untuk meningkatkan efisiensi jaringan. Egar [5] mengembangkan sistem berbasis metode *Certainty Factor* di PT. Telkom Malang guna mendukung penyelesaian gangguan jaringan dan meningkatkan kualitas layanan pelanggan. Susianto dan Anisa melakukan analisis data paket jaringan menggunakan perangkat lunak seperti *Wireshark*, serta *Network Miner* untuk memahami interaksi data dalam jaringan internet [6]. Sementara itu, Permatasari dan

Rushendra menerapkan teknologi *Link Aggregation* dan *Load Balance* dalam jaringan pelanggan MNC Play Jakarta untuk membagi beban lalu lintas internet, yang menghasilkan *throughput* stabil sebesar 10 GB sesuai dengan standar MTU [7].

Selain aspek teknis, penelitian lain juga berfokus pada evaluasi kualitas layanan jaringan. Maria dan Irtawaty menganalisis *Quality of Service* (QoS) jaringan di Politeknik Negeri Balikpapan menggunakan parameter TIPHON, seperti *bandwidth*, *packet loss*, dan *delay*, dengan pendekatan kuantitatif [8]. Penelitian Fahlepi et al menyoroti desain jaringan VLAN untuk meningkatkan efisiensi dan keamanan jaringan di *Montessori School* Yogyakarta [9]. Sementara itu, Sharon dan Supardi [10] mengembangkan jaringan *Wireless Local Area Network* (WLAN) di CV. BIQ dengan menggunakan Mikrotik RB750GL sebagai server untuk meningkatkan fleksibilitas dan efisiensi jaringan, serta menyarankan pemeliharaan rutin agar kinerja tetap optimal. Berdasarkan berbagai penelitian terdahulu, pengelolaan jaringan WiFi terus dikembangkan melalui berbagai metode dan teknologi untuk meningkatkan kualitas layanan. Berbagai pendekatan, seperti optimasi penempatan *access point*, analisis *roaming*, serta penerapan teknologi seperti *Link Aggregation* dan VLAN, telah terbukti efektif dalam meningkatkan kinerja jaringan. Dengan semakin kompleksnya kebutuhan jaringan, disarankan agar penelitian lebih lanjut dilakukan untuk mengembangkan solusi yang lebih efektif serta inovatif dalam meningkatkan stabilitas dan efisiensi jaringan di berbagai lingkungan.

2.2. Landasan Teori

Software Defined Networking (SDN) merupakan sebuah paradigma inovatif dalam arsitektur jaringan modern yang memungkinkan pengelolaan jaringan menjadi lebih dinamis, terpusat, dan mudah diadaptasi sesuai kebutuhan. SDN berfokus pada pemisahan antara *control plane* dan *data plane*, yang memungkinkan administrator jaringan untuk mengontrol lalu lintas data secara lebih fleksibel melalui pemrograman berbasis perangkat lunak. Dengan pendekatan ini, SDN mengatasi keterbatasan jaringan tradisional yang cenderung statis dan bergantung pada perangkat keras. Salah satu teknologi utama yang mendukung SDN adalah protokol *OpenFlow*, yang memungkinkan komunikasi langsung antara *controller* pusat dan perangkat jaringan untuk mengatur lalu lintas data secara efisien [11].

Dalam implementasi SDN, *Mininet* digunakan sebagai platform simulasi jaringan berbasis perangkat lunak. *Mininet* memungkinkan pengguna untuk membuat dan menguji berbagai topologi jaringan secara virtual tanpa memerlukan perangkat keras fisik. Dengan *Mininet*, elemen-elemen jaringan seperti *switch*, *host*, dan *controller* dapat diimplementasikan secara *virtual*, memberikan fleksibilitas tinggi dalam eksperimen dan pengembangan jaringan. Switch yang digunakan dalam Mininet dikelola melalui *Open vSwitch*, yang berjalan sebagai namespace jaringan di sistem operasi Linux. Hal ini memberikan pengalaman yang realistis dalam simulasi jaringan sebelum diterapkan pada infrastruktur jaringan sebenarnya [12].

Salah satu *controller* SDN yang banyak digunakan adalah *ONOS Controller*. ONOS dirancang untuk mengelola jaringan berskala besar dengan pendekatan yang terpusat dan berbasis perangkat lunak. Dengan menyediakan *Application Programming Interface* (API) yang fleksibel, ONOS memungkinkan administrator jaringan untuk memprogram dan menyesuaikan pengaturan jaringan secara dinamis, termasuk dalam pengaturan routing dan manajemen lalu lintas data. Protokol *OpenFlow* menjadi standar utama dalam SDN, berfungsi sebagai penghubung antara *control plane* dan *data plane*. *OpenFlow* memberikan infrastruktur berbasis *flow* serta API standar yang memungkinkan *controller* mengelola *flow table* pada *switch* dan *router* melalui *Secure Channel*. Dengan pendekatan ini, aliran data dalam jaringan SDN dapat dikelola dengan lebih terstruktur, memberikan efisiensi dan skalabilitas yang lebih baik dibandingkan jaringan konvensional [13].

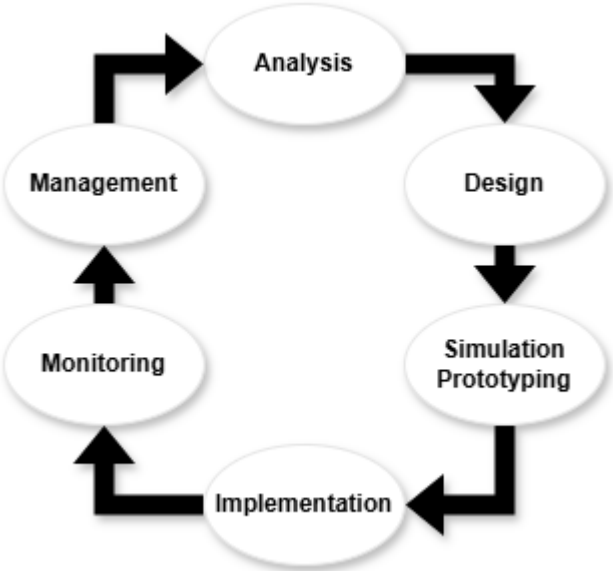
VMWare Workstation merupakan perangkat lunak virtualisasi yang memungkinkan sistem operasi berjalan secara bersamaan dalam lingkungan virtual tanpa mengganggu sistem operasi utama. Dengan arsitektur yang di-*host*, VMWare Workstation memvirtualisasikan I/O sehingga dapat berdampingan dengan sistem operasi *host* yang sudah ada sebelumnya. Arsitektur ini terdiri dari tiga komponen utama: VMApp, VMDriver, dan VMM. VMApp berfungsi sebagai aplikasi yang memungkinkan instalasi sistem operasi lain, sementara VMDriver bertindak sebagai penghubung antara sistem operasi host dan monitor mesin *virtual* (VMM). VMM sendiri menjalankan mesin virtual secara langsung pada perangkat keras, memungkinkan setiap prosesor fisik untuk beroperasi dalam dua dunia, yaitu dunia host dan dunia VMM. Proses peralihan antara kedua dunia ini dikelola oleh VMDriver, meskipun peralihan tersebut lebih berat

dibandingkan dengan peralihan proses normal sehingga dapat menyebabkan *overhead* pada CPU. Oleh karena itu, pengelolaan sumber daya harus dilakukan secara optimal agar performa sistem tetap stabil dan efisien dalam lingkungan virtualisasi [14].

Melalui kombinasi teknologi seperti *Mininet*, *ONOS Controller*, dan protokol *OpenFlow*, *VMWare Workstation*, dan SDN memberikan solusi yang lebih adaptif terhadap dinamika lalu lintas jaringan. Keunggulan SDN dalam pemisahan kontrol dan data memungkinkan optimalisasi pengelolaan jaringan, menjadikannya pilihan yang relevan dalam menghadapi tantangan jaringan modern.

3. METODE PENELITIAN

Penelitian ini menggunakan pendekatan dengan menggunakan metode *Network Development Life Cycle* atau yang dikenal dengan NDLC [15]. Tahapannya dapat dilihat pada Gambar 1 di bawah ini.



Gambar 1. Prosedur Penelitian NDLC

Berikut adalah Tahapan-tahapan alur dalam penelitian diatas (Gambar 3.1.1) yang akan dilakukan :

1) Analysis

   Menentukan perangkat keras dan lunak yang dibutuhkan, yaitu Mininet dan *ONOS Controller*, serta konfigurasi jaringan dasar.

2) Design

   Merancang topologi yang akan digunakan, yaitu 6 switch dan 12 host masing masing switch memiliki 2 host dan memberikan IP pada setiap host.

3) Simulation Prototyping

   Menjalankan Topologi menggunakan Mininet untuk melihat apakah topologi dapat berjalan dengan normal tanpa adanya masalah.

4) Implementation

   Menyambungkan Topologi yang telah dibuat ke *ONOS Controller* dan menjalankan perintah melalui Xterm di Mininet untuk melihat apakah host masing masing dapat tersambung satu sama lain.

5) Monitoring

   Melakukan pengujian *Traffic* dengan melakukan *Ping* untuk melihat konektivitas antar host dan memeriksa *Delay* dan *Packet Loss* sesuai dengan standar TIPHON.

6) Management

Mengevaluasi kinerja jaringan berdasarkan hasil pengujian dan dokumentasi hasilnya sebagai referensi penelitian dan analasis yang lebih lanjut.

4. HASIL DAN PEMBAHASAN

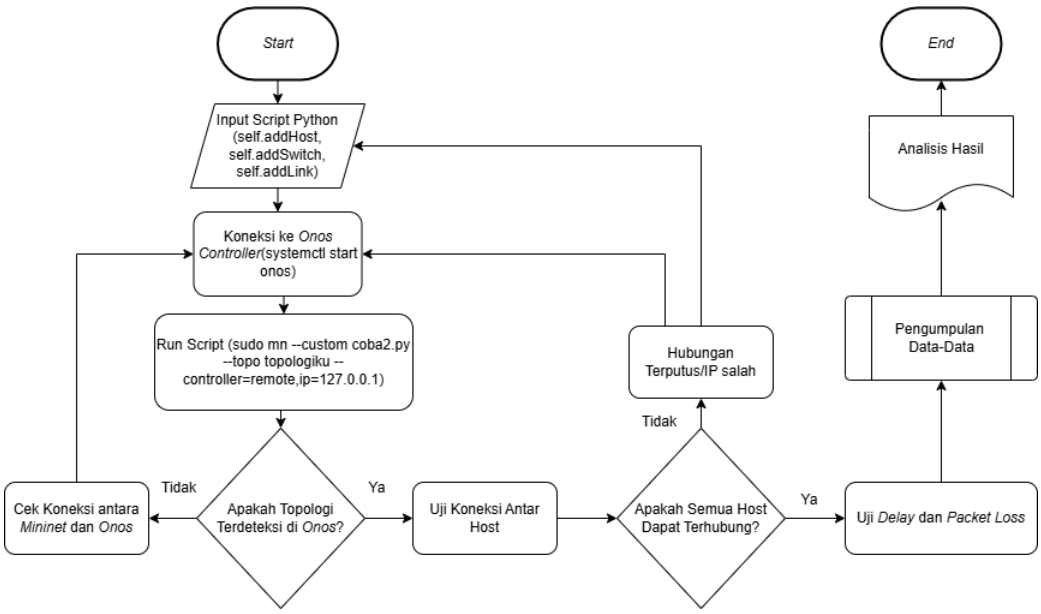
4.1. Design Jaringan

Di tahap ini, peneliti akan menjelaskan *FlowChart* konfigurasi SDN, dan langkah-langkah pada topologi jaringan SDN. Pada langkah penelitan pertama, peneliti mengimplementasikan dengan menuliskan script python yang akan diakses melalui mininet, serta melakukan penambahan IP pada setiap host agar dapat terhubung saat *ONOS Controller* diaktifkan. Tahap kedua peneliti memaparkan struktur topologi yang sebelumnya dibuat menggunakan script Python dan ditampilkan secara *live* melalui web *ONOS Controller*. Pada tahap selanjutnya peneliti melakukan test *Ping* menggunakan mininet dengan perintah *xterm & ping*. Lalu ditahap terakhir ini dilakukan pengujian koneksi jaringan menggunakan standart THIPON, pengambilan data hanya mencakup rata-rata *Delay* dan *Packet Loss*.

Di jaringan komputer, terdapat dua pendekatan utama dalam mengatur dan mengelola jaringan SDN dan Konvensional. Dalam arsitektur konvensional, kedua fungsi ini terdapat pada perangkat jaringan seperti *Router* dan *Switch*. SDN menawarkan kontrol terpusat melalui *Controller*, memungkinkan konfigurasi dan pengelolaan jaringan yang lebih dinamis dan efisien. *Flowchart* dibawah ini akan memaparkan langkah-langkah dalam melakukan konfugirasi pada jaringan SDN.

a. Flowchart Konfigurasi SDN

Pada gambar *FlowChart* seperti yang dapat dilihat pada Gambar 2, konfigurasi SDN dimulai dengan tahap awal (*start*), di mana langkah pertama adalah menginput skrip *Python* untuk mendefinisikan topologi SDN, *host*, switch, dan IP yang digunakan.



Gambar 2. FlowChart Konfigurasi SDN

Skrip ini akan menghasilkan file *Python* untuk dijalankan di Mininet. Selanjutnya, dilakukan koneksi ke *ONOS Controller* dengan perintah "*systemctl start onos*" menggunakan protokol *OpenFlow* untuk komunikasi antara *switch* di *data plane* dan *controller* di *control plane*. Setelah itu, skrip *Python* dijalankan dengan perintah "*sudo mn --custom coba2.py --topo topologiku --controller=remote,ip=127.0.0.1*" menghubungkan topologi Mininet ke *ONOS Controller*. Protokol yang digunakan dalam tahap ini meliputi

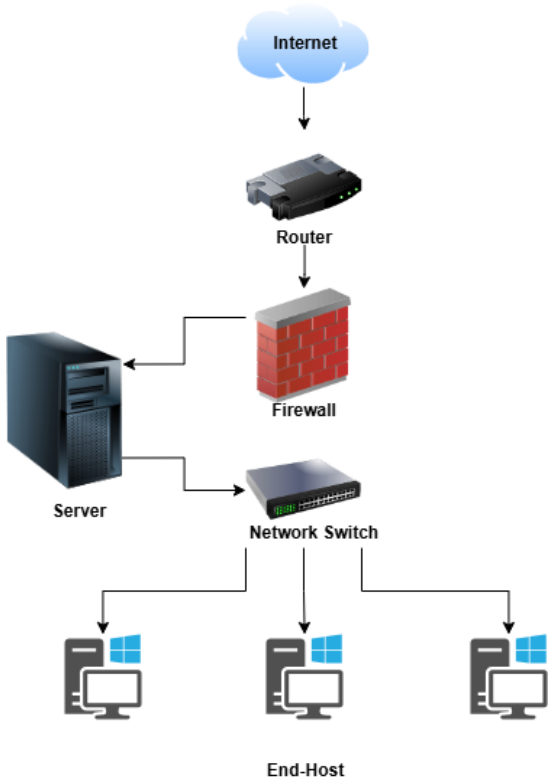
OpenFlow untuk komunikasi utama, OVSDB untuk manajemen switch, serta LLDP dan REST API untuk pertukaran informasi antar perangkat jaringan.

Selanjutnya, topologi diuji apakah terdeteksi di ONOS. Jika tidak, konektivitas antara *Mininet* dan *ONOS Controller* diperiksa ulang. Jika terdeteksi, dilakukan uji konektivitas antar host menggunakan perintah "*pingall*" untuk memastikan semua host dapat saling terhubung. Protokol ICMP digunakan untuk uji ini. Apabila semua host terhubung, pengujian dilanjutkan ke tahap pengukuran *delay* dan *packet loss*. Pada tahap ini, digunakan perintah "*iperf*" dengan salah satu *host* diatur sebagai *server* (iperf -s) dan *host* lain mengirimkan data ke *server* dengan perintah "*iperf -c -t 30 &*". Protokol TCP digunakan untuk mengukur *throughput* dan *delay*, sedangkan UDP digunakan untuk mengukur *packet loss* dan *jitter*.

Data dari pengujian "*iperf*" dan "*pingall*" kemudian dikumpulkan dan diolah secara statistik. Standar TIPHON digunakan untuk menghitung rata-rata *delay* dan *packet loss*. Hasil pengujian dianalisis untuk dibuat dalam bentuk tabel, sekaligus mengidentifikasi faktor-faktor yang memengaruhi performa pengiriman data. Proses konfigurasi dan pengujian jaringan SDN selesai di tahap akhir (*end*).

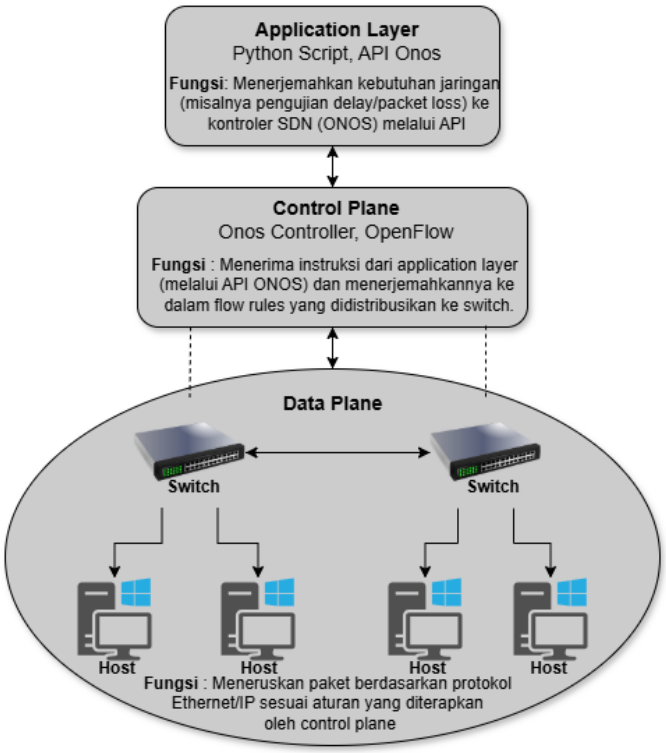
b. Arsitektur Jaringan Konvensional dan SDN

Pada arsitektur konvensional (Gambar 3), pengaturan lalu lintas data dilakukan oleh perangkat keras jaringan seperti *router*, *firewall*, *switch*, dan *server*. *Firewall* berfungsi sebagai penjaga keamanan dengan menyaring data berdasarkan aturan tertentu, sementara *server* menyediakan layanan penting seperti penyimpanan data, aplikasi, dan otentikasi pengguna.



Gambar 3. Arsitektur Jaringan Konvensional

Berbeda dengan arsitektur konvensional, SDN (Gambar 4) menawarkan pendekatan yang lebih fleksibel. SDN memisahkan antara *control plane* (pengendali jaringan) dan *data plane* (jalur data), memungkinkan pengelolaan jaringan dilakukan secara terpusat melalui perangkat lunak. *Control plane* pada SDN, yang menggunakan *ONOS Controller* dan protokol *OpenFlow*, bertugas mengatur lalu lintas jaringan. Caranya, *control plane* menerima instruksi dari lapisan aplikasi, menerjemahkannya menjadi aturan aliran (*flow rules*), dan mengirimkan aturan tersebut ke *switch*.

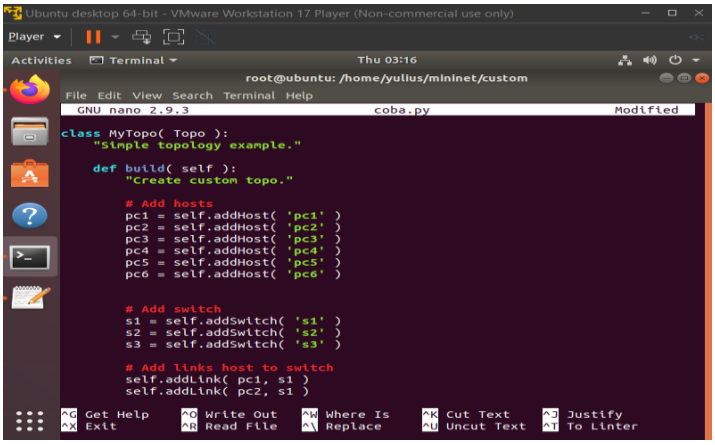


Gambar 4. Arsitektur Jaringan SDN

Perbedaan mendasar antara kedua arsitektur ini terletak pada bagaimana kontrol jaringan dilakukan. Pada jaringan konvensional, kontrol tersebar di setiap perangkat, sehingga perubahan konfigurasi harus dilakukan secara manual pada masing-masing perangkat. Sementara itu, SDN menggunakan kontrol terpusat melalui *controller*, yang membuat pengelolaan jaringan menjadi lebih dinamis, otomatis, dan mudah dikonfigurasi tanpa perlu mengubah perangkat keras secara langsung. Keunggulan-keunggulan inilah yang menjadikan SDN sebagai solusi untuk meningkatkan efisiensi dan fleksibilitas jaringan modern.

4.2. Konfigurasi Jaringan SDN

Pada tahap pertama pengujian, peneliti membuat topologi menggunakan skrip Python yang diimplementasikan pada *Mininet* dan dikontrol oleh *ONOS Controller*. Skrip ini mendefinisikan koneksi antara *host* dan *switch* menggunakan perintah seperti *self.addHost, self.addSwitch*, dan *self.addLink*.



Gambar. 5. Python Script

Gambar 5 menunjukkan skrip *Python* awal yang digunakan untuk topologi sederhana. Sedangkan Gambar 6 menunjukkan bahwa peneliti melakukan penambahan *host*, switch dan juga IP pada setiap host. Penambahan IP pada setiap *host* agar setiap *host* dapat terhubung dan juga stabil saat nantinya *ONOS Controller* diaktifkan. Perlu diperhatikan *Mininet* dan *ONOS Controller* dapat saling terhubung agar *traffic* serta informasi topologi dapat terhubung langsung ke *ONOS Controller*. Dilakukannya pengujian SDN ini agar pemahaman bagaimana menilai *traffic* dapat bekerja dengan baik dan juga efisien pada SDN yang telah diterapkan

```
  GNU nano 2.9.3                    coba2.py

class MyTopo(Topo):
    "Topology dengan hanya switch."

    def build(self):
        # Menambahkan host
        pc1 = self.addHost('pc1', ip='192.168.1.1/24')
        pc2 = self.addHost('pc2', ip='192.168.1.2/24')
        pc3 = self.addHost('pc3', ip='192.168.1.3/24')
        pc4 = self.addHost('pc4', ip='192.168.1.4/24')
        pc5 = self.addHost('pc5', ip='192.168.1.5/24')
        pc6 = self.addHost('pc6', ip='192.168.1.6/24')
        pc7 = self.addHost('pc7', ip='192.168.1.7/24')
        pc8 = self.addHost('pc8', ip='192.168.1.8/24')
        pc9 = self.addHost('pc9', ip='192.168.1.9/24')
        pc10 = self.addHost('pc10', ip='192.168.1.10/24')
        pc11 = self.addHost('pc11', ip='192.168.1.11/24')
        pc12 = self.addHost('pc12', ip='192.168.1.12/24')

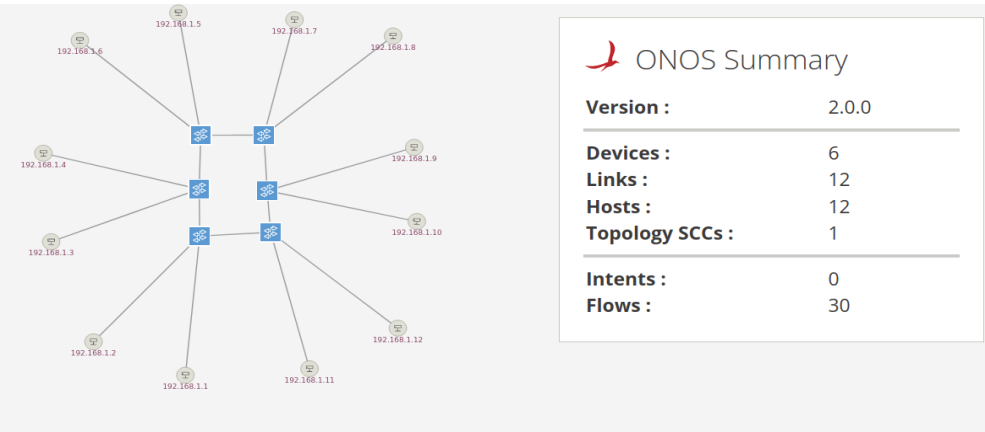
        # Menambahkan switch
        s1 = self.addSwitch('s1')
        s2 = self.addSwitch('s2')
        s3 = self.addSwitch('s3')
        s4 = self.addSwitch('s4')

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Linter
```

Gambar. 6. Penambahan IP, Host, Switch

4.3. Struktur Topologi SDN

Di tahap ini, peneliti memaparkan struktur topologi yang telah dibuat menggunakan script python pada tahap sebelumnya, termasuk jumlah perangkat yang terhubung satu sama lain. Melalui web ONOS dapat terlihat dengan jelas bentuk topologi yang telah dibentuk secara langsung, dengan adanya topologi ini peneliti dapat memahami skenario agar hasil yang didapatkan akurat.



Gambar. 7. Topologi SDN melalui ONOS

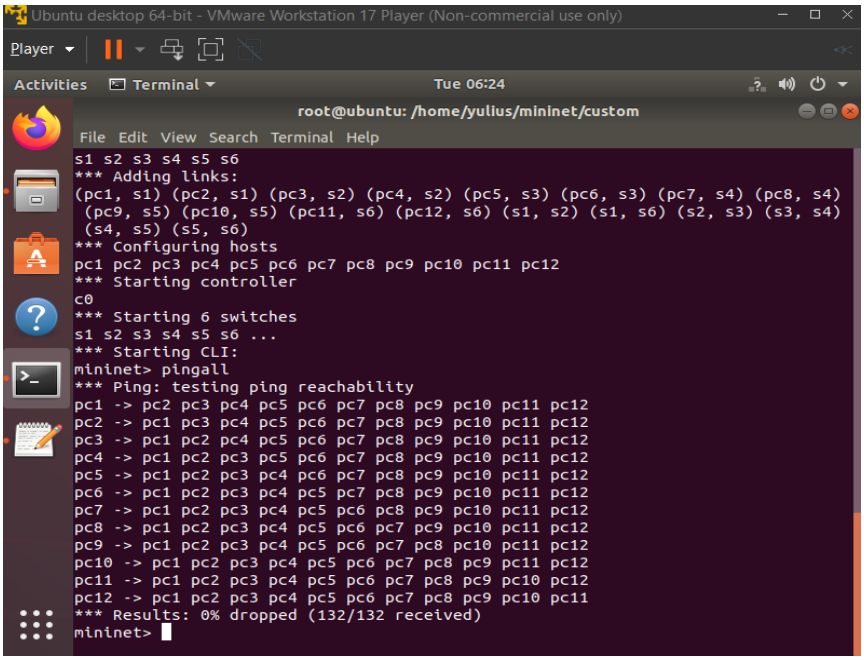
Topologi ini divisualisasikan melalui web ONOS, terdiri dari 6 switch dan 12 host, sebagaimana ditunjukkan pada Gambar 7. Meskipun topologi SDN yang telah divisualisasikan melalui ONOS menunjukan koneksi yang stabil antara perangkat, jaringan ini sangat bergantung pada *controller* sebagai pusat kendali utama. Dalam penelitian ini, implementasi *redundant controller* tidak diterapkan, sehingga apapibla *ONOS Controller* mengalami kegagalan, seluruh jaringan dapat mengalami gangguan dan

menyababkan perangkat kehilangan konektivitas. Hal ini menunjukan bahwa *controller* berperan penting sebagai *Single Point of Failure* (SPoF) dalam jaringan SDN, berbeda dengan jaringan konvensional yang Dimana distribusi kendali lebih tersebar di masing-masing perangakat.

Meskipun penelitian ini tidak menguji serangan DDoS terhadap *controller*, penting untuk mempertimbangkan resiko keamanan dalam implementasi SDN. Serangan terhadap *controller* dapat menggaggu keseluruhan jaringan. Disarankan untuk penelitian kedepannya dapat mengeksplorasi lebih lanjut mengenai Implementasi *redundant controller* untuk memastikan jaringan tetap stabil meskipun terjadi kegagalan pada salah satu *controller*.

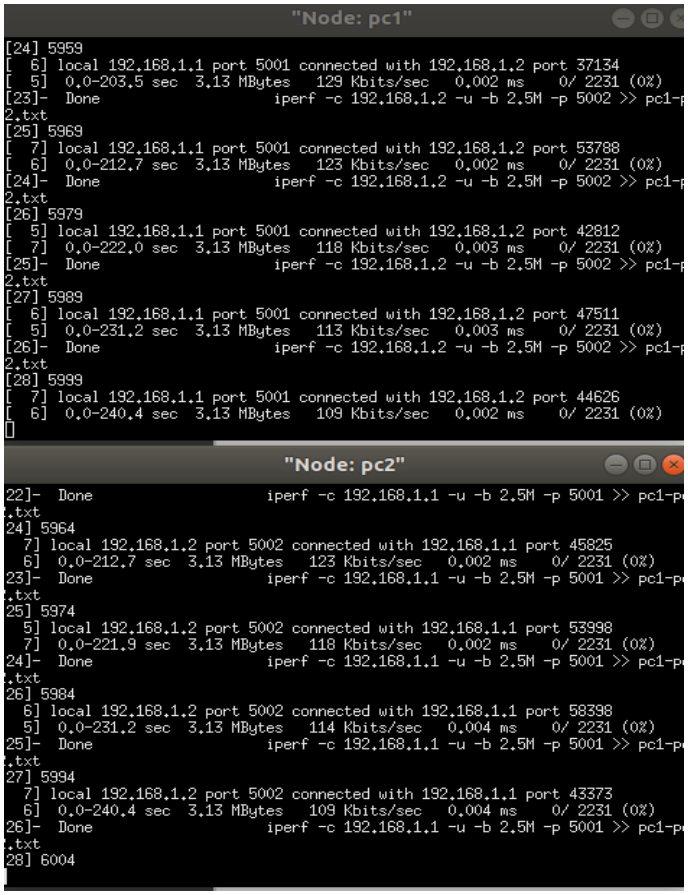
4.4. Pengujian Traffic Data Dengan Mininet

Setelah topologi diintegrasikan ke ONOS menggunakan perintah *"sudo mn --custom coba2.py --topo topologiku --remote=controller,ip=127.0.0.1"*, dilakukan uji koneksi antar host menggunakan perintah "*pingall*" (Gambar 8). Uji ini bertujuan memastikan semua *host* terhubung, ditandai dengan tidak adanya simbol "X" dalam hasil uji. Jika ada *host* yang gagal terhubung, konfigurasi diperbaiki.



Gambar 8. Run Script Python di Mininet

Gambar 9 menunjukkan aksi "*iperf -s -u -p 5001(port)*" perintah tersebut untuk memberikan akses agar dapat mengirim beban data berat dan mengatifkan penggunaan UDP alih-alih TCP. Dan "*for i in {1..30}; do iperf -c <ip tujuan> -u -p <port yang dituju> >> nama_file.txt & ping <ip tujuan> -c 10 >> nama_file.txt; done*" untuk mengirimkan data sebanyak 30x ke *Host* yang diinginkan dari file .txt tersebut dapat kita dapatkan data transfer file berserta *Delay* dan *Packet Loss*. pada perintah xterm pc1 dan pc2 menunjukan bahwa *host* tersebut dapat terhubung walaupun saat pengiriman terjadi penurunan *Bandwidth* tetapi masih dalam keadaan stabil.



Gambar 9. Pengiriman Beban Data Berat Pc1 ke Pc2 dan sebaliknya

4.5. Pengujian *Delay* dan *Packet Loss*

Setelah dilakukannya pengumpulan data, tahapan berikutnya dilakukan uji koneksi jaringan menggunakan standar TIPHON atau *Telecommunications and Internet Protocol Harmonization Over Network*. Pengukuran menggunakan TIPHON dapat membantu mengukur kualitas jaringan yang mencakup berbagai aspek yaitu *Latency, Delay, Jitter, Troughput, dan Packet Loss*. Dari data yang telah diperoleh, data tersebut akan dianalisis apakah kualitas jaringan tersebut memenuhi standar TIPHON. Pengujian ini hanya mencakup pada uji *Delay* serta *Packet Loss*. Pada uji *Delay*, TIPHON memiliki standar ukur dari kategori "sangat baik" kurang dari 150ms hingga kategori "buruk" diatas 450ms, dan untuk *Packet Loss* 0 hingga 2% memiliki kategori "sangat baik" dan lebih dari 25% memiliki kategori "buruk".

Tabel 1. *Delay Standart* TIPHON

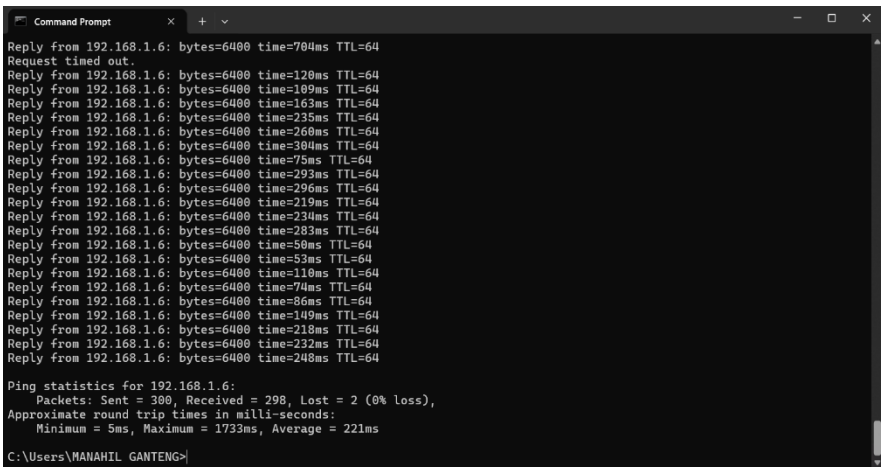
No	Kategori	Besar *Delay*	Indeks
1	Sangat Baik	<150 ms	4
2	Baik	150 – 300 ms	3
3	Cukup	300 – 450 ms	2
4	Buruk	>450 ms	1

Tabel 2 *Packet Loss* Standart TIPHON

No	Kategori	Besar *Packet Loss*	Indeks
1	Sangat Baik	0% - 2%	4
2	Baik	3% - 14%	3
3	Cukup	15% – 24%	2
4	Buruk	<25%	1

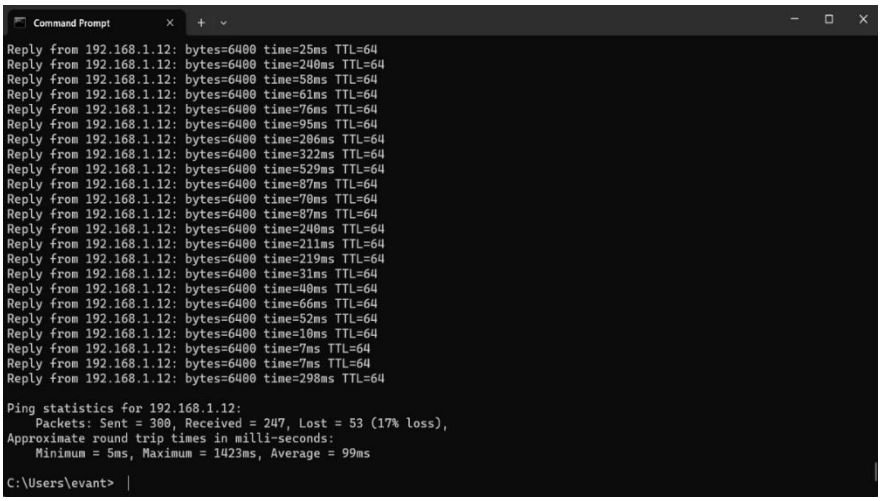
Pada setiap *Host* yang telah memiliki IP akan sebelumnya telah dilakukan uji *Ping* dengan mengirim paket dari 1 *host* ke 11 *host* lainnya, selanjutnya penguji melakukan pengujian pengiriman beban data yang lebih berat menggunakan perintah "*for i in {1..30}; do iperf -c <ip tujuan> -u -p <port yang dituju> >> nama_file.txt & ping <ip tujuan> -c 10 >> nama_file.txt; done*" pengujian ini dilakukan dengan cara 1-1, 2-1, 2-2 yaitu 1 *Host* mengirim ke 1 *Host* lainnya, lalu 2 *Host* mengirim data secara bersamaan ke 1 *Host*, yang terakhir mengirim 2 *Host* mengirim ke 2 *Host* bersamaan.

Dibagian pengujian terakhir penguji menggunakan 2 *Xterm Host* yang sama untuk melaukan pengiriman ke 1 *Host* begitu pula dengan *Host* lainnya. Hasil ini akan dibandingkan dengan penggunaan jaringan Konvensional di dunia nyata dengan total 4 *device* dan pengujian dilakukan menggunakan *Command Prompt* bawaan windows dengan perintah "*ping <ip_tujuan> -n 300 -l 6400*" disini penguji menggunakan 300 dan beban 6400 *byte* menyamakan dengan pengujian di *Linux*, hanya saja di *Linux* dapat menggunakan pengulangan 30 kali dengan mengirim persepuluh kali, sedangkan di *Command Prompt* dilakukan secara langsung ke 300 kali. Berikut contoh data dari *Command Prompt* yang dilakukan pengiriman secara bersamaan.



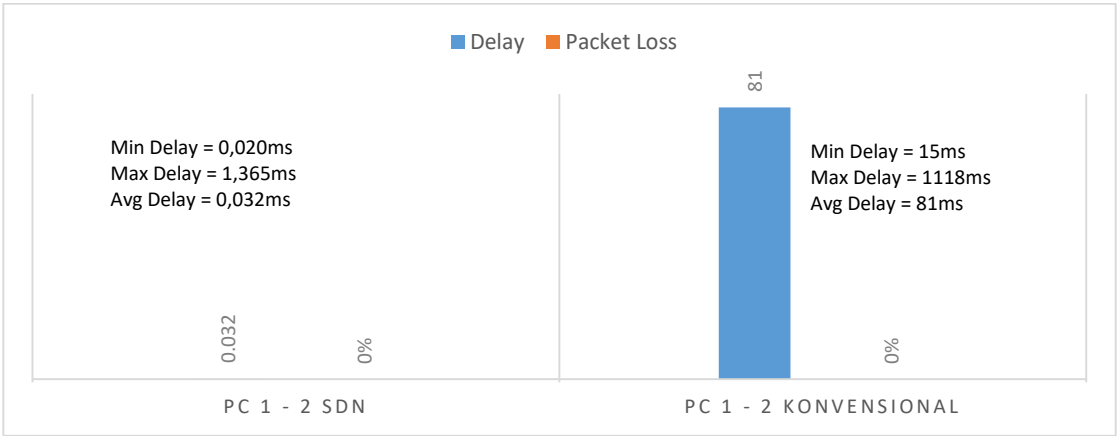
Gambar 10. pengiriman paket dari pc 1- 3 Konvensional

Gambar 10 menunjukkan uji koneksi ke IP 192.168.1.6 dari 300 paket dengan beban 6400 *bytes* yang telah dikirim hanya 298 paket yang diterima 2 diantaranya hilang. dari data tersebut menunjukan koneksi masih cukup baik karena sebagian besar paket data terkirim dengan sukses, dan waktu tempuh yang dicapai masih cukup baik

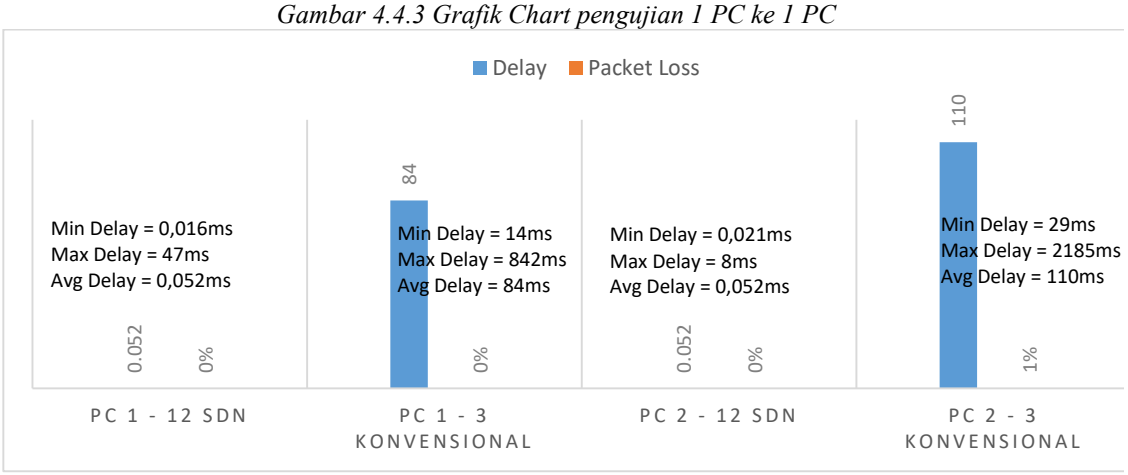


Gambar 11. Pengiriman paket dari pc 2-4 Konvensional

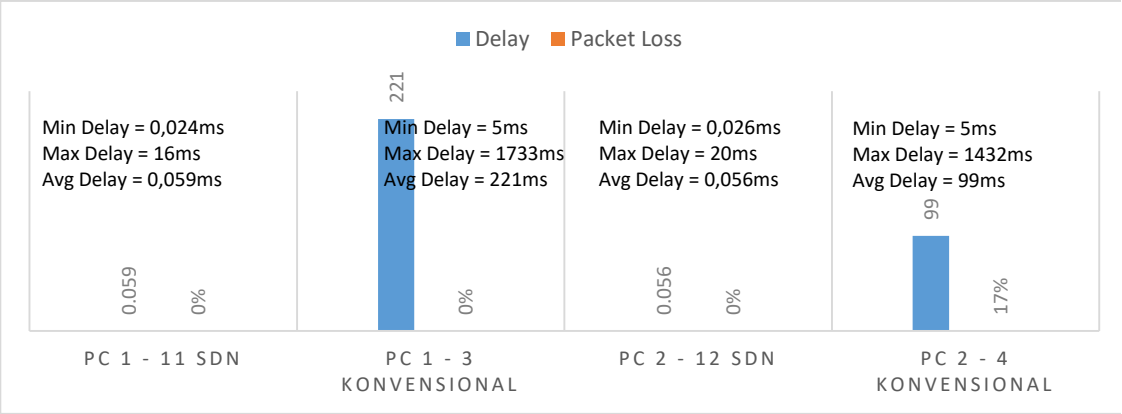
Pada koneksi yang ditunjukan pada Gambar 11 koneksi ke IP 192.168.1.12, terlihat mengalami masalah yang signifikan, Tingkat kehilangan paket mencapai 17% dengan 53 dari 300 paket tidak terkirim ke tujuan. Kemungkinan penyebabnya antara lain masalah pada kepadatan jaringan serta gangguan interferensi.



Gambar 4.4.3 Grafik Chart pengujian 1 PC ke 1 PC



Gambar 12. Grafik Chart pengujian 2 PC ke 1 PC



Gambar 13. Grafik Chart pengujian 2 PC ke 2 PC

Setelah melakukan pengujian pada setiap *host* dalam jaringan, diperoleh data yang menunjukan *Delay* dan *Packet Loss* yang baik untuk SDN dan hasil yang cukup untuk Konvensinal. Pada pengujian pengiriman data (Gambar 12) 1 PC ke 1 PC pada jaringan SDN maupun Konvensional dengan menggunakan standar TIPHON menunjukan hasil yang Sangat Baik dengan masing masing *Delay* berada di bawah 150 ms dan *Packet Loss* menjukan 0%, pada pengujian selanjutnya (Gambar 4.4.4) yaitu 2 PC ke 1 PC kedua jaringan masih menunjukan hasil yang masih terhitung Sangat Baik walaupun terjadi lonjakan pada salah satu PC 2-3 pada jaringan Konvensional yang menunjukan bahwa *Delay* masih berada di bawah 150 ms dan *Packet Loss* hanya 1%.

Selanjutnya pada Gambar13 merupakan hasil pengujian 2 PC ke 2 PC SDN masih berada di kondisi yang sangat baik sedangkan pada Konvensional di PC 1 – 3 terjadi *Delay* yang cukup tinggi tetapi menurut standar TIPHON masih termasuk kondisi yang Baik dan stabil karena *Packet Loss* masih menunjukan 0% sedangkan pada PC 2 – 4 *Delay* yang ditunjukan Sangat Baik yaitu 99ms tetapi pada *Packet Loss* berada di 17% yang mengidentifikasikan bahwa dari 300 data yang terkirim ada 51 data yang tidak terkirim ke PC yang dituju. Pada pengujian ke 3 ini kemungkinan disebabkan karena pengiriman paket data secara bersamaan pada PC 1-3 dan PC 2-4 walaupun PC 1-3 melewati *Traffic* yang panjang tetapi tetap konsisten sedangkan pada PC2-4 walaupun *Traffic* yang digunakan cepat karena terjadinya interferensi dengan PC 1-3 yang cenderung lebih stabil dan mungkin *Router/Switch* tidak dapat memproses data yang dikirim secara bersamaan sehingga terbuang dan terjadinya *Packet Loss*.

5. SIMPULAN DAN SARAN

Berdasarkan hasil pengujian yang telah dilakukan pada jaringan SDN yang telah dibuat diperoleh berbagai temuan sebagai berikut.

1. Implementasi Topologi : Topologi jaringan SDN berhasil diimplemantasikan menggunakan *Python Script* pada *Mininet* dan dikontrol oleh *Onos Controller*. Topologi *Custom Mesh* yang digunakan penguji terbukti efektif dalam menghubungkan seluruh perangkat.
2. Pengujian Konektivitas : pengujian "*iperf*" menunjukan bahwa seluruh *host* dalam jaringan dapat saling berkomunikasi dengan baik. Tidak ditemukannya kendala yang signifikan dalam konektivitas antar perangkat
3. Pengukuran Kinerja : pada pengujian pertama dan kedua, *Delay* serta *Packet Loss* yang didapat masih berada di indeks 4 yaitu Sangat Baik. Tetapi pada pengujian ke tiga SDN menunjukan hasil yang Sangat Baik sedangkan Konvensional menunjukan adanya peningkatan pada *Delay* dan *Packet Loss* pada PC 1-3 memiliki *Delay* tinggi tetapi stabil dengan Packet Loss 0%. Sedangkan PC 2-4 memiliki *Delay* yang rendah tetapi *Packet Loss* mencapai 17%. Hal ini menunjukan adanya interferensi konvensional dan router/switch dapat mempengaruhi performa disaat pengiriman.
4. Perbandingan SDN dan Konvensional : pada pengamatan yang dilakukan Jaringan SDN terbukti lebih unggul dalam hal *Delay* dan *Packet Loss*, terutama pada kondisi beban yang tinggi dan

pengiriman secara bersamaan. Hal ini menunjukan bahwa SDN lebih efisien dibandingkan dengan jaringan Konvensional. Sedangkan Jaringan Konvensional memiliki keterbatasan, cenderung mengalami peningkatan *Delay* dan *Packet Loss* yang signifikan dan penurunan performa ketika beban jaringan yang dikirimkan besar, terutama disaat banyaknya perangkat ang mengirimkan data secara bersamaan

Selanjutnya berdasarkan hasil pengujian yang telah dilakukan, penelitian ini membuktikan keunggulan signifikan jaringan SDN dibandingkan dengan jaringan Konvensinal. Implementasi topologi jaringan yang dikontrol oleh ONOS controller telah menghasilkan peningkatan kinerja dalam hal *Delay* dan *Packet Loss*. Dari analisis data menunjukan bahwa SDN mampu mendistribusikan *traffic* secara lebih efisien, sehingga meminimalkan adanya interferensi antar paket dan menjaga stabilitas jaringan, bahkan dalam beban jaringan yang tinggi. Hal ini mengindikasikan bahwa arsitektur SDN memiliki fleksibelitas dan Skalabilitas yang lebih baik dalam mengakomodasi aplikasi *modern* yang semakin kompleks. Inovasi utama dalam penelitian ini adalah penerapan standar TIPHON dalam pengujian performa SDN yang memungkinkan pendekatan yang lebih terstruktur dalam menilai kinerja jaringan serta mengindentifikasi area yang memerlukan optimasi lebih lanjut.

Hasil pengujian menunjukkan bahwa SDN merupakan teknologi yang menjanjikan untuk menggantikan arsitektur jaringan Konvensional yang kaku. Berdasarkan penelitian kali ini, disarankan untuk para peneliti selanjutnya meneliti mengenai, menguji skalabilitas kinerja jaringan SDN dengan skala yang lebih besar untuk mengidentifikasi batasan dan potensi SDN tersebut, mengevalusasi algoritma *routing* yang berbeda untuk membuat kinerja SDN semakin optimal dalam skenario yang spesifik, mengintegrasi SDN dengan teknologi emerging jaringan 5G, maupun AI. Dari hasil penelitian yang didapatkan pasti memiliki implikasi yang luas untuk industri telekomunikasi dan informasi. Mengadopsi SDN kedepannya dapat meningkatakan efisien jaringan, mengurangi biaya serta dapat dilakukannya penelitan untuk pengembangan layanan jaringan yang lebih inovatif.

DAFTAR PUSTAKA

[1] E. R. Huddiniah, E. M. Safitri, S. A. Priyambada, M. Nasrullah, and N. D. Angresti, “Optimasi Rute Untuk Software Defined Networking-Wide Area Network (SDN-WAN) Dengan Openflow Protocol,” *Inform. Mulawarman J. Ilm. Ilmu Komput.*, vol. 13, no. 1, p. 7, 2018, doi: 10.30872/jim.v13i1.1006.

[2] E. I. Alwi, “Analisis Kualitas Sinyal Wifi Pada Universitas Muslim Indonesia,” *Informatics Journal.* vol. 4, no. 1, 2019.

[3] S. S. Aishah Garnis, Suroso, “Pengkajian Kualitas Sinyal dan Posisi WiFi Access Point dengan Metode RSSI di Gedung KPA Politeknik Negeri Sriwijaya”, *Media Neliti,* 2017.

[4] A. Cahyono, “Analisis Roaming Jaringan Wireless (Studi Kasus : Gedung Perkuliahan di UKSW Diponegoro”, *Repository UKSW*, pp. 5–15, 2018.

[5] E. A. Prasetya, “Identifikasi Gangguan Jaringan Dengan Metode Certainty Factor Di PT.Telkom Malang Guna Peningkatan Pelayanan Customer,” vol. 09, pp. 1–5, 2021.

[6] D. Susianto *et al.,* "Implementasi dan Analisis Jaringan Menggunakan Wireshark, Cain and Abels, Network Minner (Studi Kasus : AMIK Dian Cipta Cendikia)”, Media Neliti, vol. XVI, pp. 120–125, 2018.

[7] Permatasari, Eka. *Analisa Pembagian Penggunaan Bandwidht Untuk Jaringan Internet Pelanggan Mnc Play Area Jakarta Dengan Menggunakan Teknologi Link Aggregation Dan Load Balance*. Diss. Universitas Mercu Buana Jakarta, 2022..

[8] Ulfah, M., Irtawaty, A. S., Soekarno, J., Km, H., & Indonesia, B. Pengukuran dan Evaluasi Kinerja Parameter Quality of Service (QoS) Jaringan Internet SMA Negeri 9 Balikpapan. *Prosiding Seminar Nasional Inovasi Teknologi Terapan, vol*, *19*, 234-241.

[9] A. F. Panjaitan, R. Supardi, and A. Al Akbar, "Penerapan Framework Man in the Middle Menggunakan Linux Pada Lembaga Penyiaran Publik Rri Bengkulu," *Jtis*, vol. 4, no. 3, pp. 35–41, 2021.

[10] R. S. Desmon Sharon, Sapri, "Membangun Jaringan Wireless Local Area Network (Wlan) Pada Cv.Biq Bengkulu," *J. Media Infotama*, vol. 10, no. 1, pp. 35–41, 2014, [Online]. Available: https://jurnal.unived.ac.id/index.php/jmi/article/view/230/207

[11] R. Fauzan, J. Gusti, A. Ginting, and J. T. Telekomunikasi, "Analisis Performansi Jaringan Software Define Network (SDN) Menggunakan Aruba Van Controller," *J. Pengemb. IT*, vol. 5, no. 3, pp. 64–69, 2020.

[12] E. Safrianti, "Optimasi Manajemen Jaringan Data UNRI Menggunakan Teknologi Jaringan Terdefenisi Perangkat Lunak," *JTEV (Jurnal Tek. Elektro dan Vokasional)*, vol. 5, no. 1.1, p. 5, 2019, doi: 10.24036/jtev.v5i1.1.106133.

[13] M. H. Subowo, "Analisis Kinerja Jaringan Software-Defined Networking (SDN) Berbasis Open Network Operating System (ONOS) dengan Menggunakan Protokol OpenFlow 1.5: Studi Kasus pada Algoritma Flow Scheduling Shortest Path First (SPF) dan Weighted Round Robin (WRR)," *J. Teknol. Inform. dan Komput.*, vol. 9, no. 1, pp. 266–273, 2023, doi: 10.37012/jtik.v9i1.1519.

[14] R. A. Nuryadin, T. A. Ramadhani, J. Karaman, and M. Reza, "Analisis Perbandingan Performa Virtualisasi Server Menggunakan Vmware Esxi, Oracle Virtual Box, Vmware Workstation 16 Dan Proxmox," *METHOMIKA J. Manaj. Inform. dan Komputerisasi Akunt.*, vol. 7, no. 2, pp. 175–180, 2023, doi: 10.46880/jmika.vol7no2.pp175-180.

[15] S. A. Haris, H. Suhartono, and H. Herlawati, "Menjaga Kestabilan Jaringan Load Balancing Nth Dengan Teknik Failover Pada PT. Jakarta Samudera Sentosa Jakarta," *PIKSEL Penelit. Ilmu Komput. Sist. Embed. Log.*, vol. 6, no. 1, pp. 49–60, 2018, doi: 10.33558/piksel.v6i1.1399.