Jurnal Teknik Informatika (JUTIF)
Vol. 5, No. 4, August 2024, pp. 33-42

DOI: https://doi.org/10.52436/1.jutif.2024.5.4.2000
p-ISSN: 2723-3863
e-ISSN: 2723-3871

ROUTING OPTIMIZATION ON SOFTWARE DEFINED NETWORK ARCHITECTURE USING BREADTH FIRST SEARCH ALGORITHM

David Armanda[*1], Fransiska Sisilia Mukti[2], Danang Arbian Sulistyo[3]

[1,2,3]Informatics Engineering, Faculty of Technology and Design, Asia Institut of Technology and Business Malang, Indonesia
Email: [1]davidarmanda010@gmail.com, [2]ms.frans@asia.ac.id, [3]danangarbian@asia.ac.id

(Article received: April 22, 2024; Revision: May 27, 2024; published: July 29, 2024)

Abstract

Software Defined Network (SDN) is a network modelling that separates the control plane and data plane. SDN is a new form of paradigm used for large-scale networks, one of which is for routing. Most types of routing used today use single-path routing. Single-path only uses one path as data transmission. This will result in reduced performance on the network which is often referred to as network congestion. In this test, the routing algorithm used is Breadth First Search (BFS) by modifying the path so that congestion on the network can be minimised. The BFS algorithm is implemented using Mininet emulator, Ryu Controller, and fat-tree topology. In the test, 20 scenarios were used with a bandwidth of 50 - 1000 Mbps within 15 seconds. Tests were conducted to measure the performance of the BFS algorithm, namely the path and QOS (Quality Of Service) parameters which include testing delay, packet loss, jitter, and throughput. The data obtained in testing using the conventional BFS algorithm is compared with the modified BFS algorithm data in the same test method. In path testing, the modified BFS algorithm is superior and in parameter testing, it is produced with a degraded percentage value in delay (65%), packet loss (99%), jitter (84%), and throughput has increased by (41%). So the modified BFS algorithm is superior due to the utilisation of path modification for routing optimisation which is more effective in handling network congestion.

Keywords: *BFS modification, Conventional BFS, QoS, routing, SDN.*

OPTIMASI ROUTING PADA ARSITEKTUR SOFTWARE DEFINED NETWORK MENGGUNAKAN ALGORITMA BREADTH FIRST SEARCH

Abstrak

Software Defined Network (SDN) merupakan pemodelan jaringan yang memisahkan antara *control plane* dan *data plane*. SDN adalah bentuk paradigma baru yang digunakan untuk jaringan dalam skala besar, salah satunya untuk *routing*. Sebagian besar jenis *routing* yang digunakan saat ini menggunakan *routing single-path*. *Single-path* hanya menggunakan satu jalur sebagai pengiriman data. Hal ini akan mengakibatkan performa pada jaringan semakin berkurang yang sering disebut dengan kemacetan jaringan. Dalam pengujian ini, algoritma *routing* yang digunakan adalah *Breadth First Search* (BFS) dengan memodifikasi pada bagian jalur sehingga kemacetan pada jaringan dapat diminimalisir. Algoritma BFS diimplementasikan menggunakan emulator *Mininet*, *Ryu Controller*, dan topologi *Fat-Tree*. Dalam pengujian, digunakan 20 skenario dengan *bandwidth* 50 - 1000 Mbps dalam waktu 15 detik. Pengujian dilakukan untuk mengukur kinerja dari algoritma BFS yaitu jalur dan parameter QOS (*Quality Of Service*) yang meliputi pengujian *delay*, *packet loss*, *jitter*, dan *throughput*. Data yang didapatkan pada pengujian menggunakan algoritma BFS konvensional dibandingkan dengan data algoritma BFS modifikasi dalam metode pengujian yang sama. Pada pengujian jalur, algoritma BFS modifikasi lebih unggul dan pada pengujian parameter, dihasilkan dengan nilai prosentase yang mengalami degradasi pada *delay* (65%), *packet loss* (99%), *jitter* (84%), dan *throughput* mengalami peningkatan sebesar (41%). Maka algoritma BFS modifikasi lebih unggul karena pemanfaatan modifikasi jalur untuk optimasi *routing* yang lebih efektif dalam penanganan kemacetan jaringan.

Kata kunci: *BFS konvensional, BFS modifikasi, QoS, routing, SDN.*

1. PENDAHULUAN

Peningkatan jumlah aplikasi jaringan meningkatkan kebutuhan terhadap kemampuan

pengiriman data yang efisien [1]. Internet berkembang sangat pesat khususnya pada jaringan komputer, komponenyang digunakan untuk membangun jaringan komputer seperti *switch*, *router*, dan perangkat lain perlu dikonfigurasi dengan benar untuk menghindari masalah jaringan [2]. Paradigma baru telah dikembangkan yaitu SDN.

Arsitektur internet baru yang telah diusulkan untuk memperbaiki arsitektur jaringan dengan konsep *control plane* dan *data plane* yaitu, *Software Defined Network* (SDN) [3]. Konsep utama pada SDN adalah *Control Plane* akan terpisah dari *Data Plane*. *Control Plane* memiliki fungsi ketika mengatur paket-paket yang dikirimkan dalam suatu jaringan, sedangkan *Data Plane* merupakan perangkat yang memiliki fungsi dalam melakukan kegiatan pengiriman maupun penerusan paket berdasarkan informasi yang diberikan oleh *Control Plane* [4]. SDN memiliki arsitektur yang memungkinkan jaringan komputer untuk kebutuhan aplikasi maupun kebutuhan pengguna, menyederhanakan manajemen dan memungkinkan skalabilitas jaringan [5]. Jika dibandingkan dengan menggunakan perutean statis, perutean dinamis dapat memperbarui *routing table* sesuai dengan status jaringan secara *real time* [6]. SDN dapat mengontrol konektivitas suatu jaringan serta aliran *traffic* dari jaringan yang melewatinya [7].

Ada istilah rute atau jalur yang digunakan untuk penentuan jalur terbaik dalam proses *routing*. Rute titik awal ke titik akhir, hasil yang baik adalah jalur terpendek. Masalah penentuan jalur terpendek sangat menarik untuk dibahas karena sangat berkaitan dengan optimasi *routing* [8]. Proses pengiriman paket data melalui beberapa jalur dalam jaringan komputer yang berbeda disebut dengan *routing*. *Routing* pada jaringan mempengaruhi kinerja jaringan dalam hal manajemen [9]. *Routing* akan mempengaruhi kecepatan akses internet dengan pemilihan jalur yang dipengaruhi oleh beberapa hal, antara lain perangkat yang digunakan, topologi yang dipakai, ketepatan administrator jaringan dalam melakukan konfigurasi jaringan, sampai kepada kapasitas *bandwidth* yang tersedia untuk pangaksesan internet [10].

Routing sering terjadi kemacetan hal ini terjadi ketika jumlah paket yang ditransmisikan melalui jaringan telah mendekati kapasitas penanganan paket yang dapat dilakukan dan penentuan nilai metrik akan mempengaruhi hasil kumputasi jalur. Padahal semakin tinggi tingkat kepadatan lalu lintas, komunikasi data pada jalur yang memiliki lalu lintas data yang padat akan berakibat tingginya resiko pengurangan terhadap paket data dikirim dikarenakan penurunan kapasitas *bandwidth*. Salah satu upaya yang dapat dilakukan untuk mengatasi permasalahan kemacetan jaringan adalah dengan mendesain jalur pengiriman data yang tepat (lebih dikenal dengan istilah *routing*) [11].

Selain itu dampak pada *client* yang mengalami kegagalan atau kemacetan pengiriman jaringan yang berdampak pada request TCP (*Transmission Control Protocol*) dan HTTP (*Hypertext Transfer – Transfer Protocol*) tidak stabil [12]. Terkait TCP dan HTTP menyambung dengan *Quality of Service* (QoS) digunakan untuk tolak ukur dalam melakukan analisa pengelolaan *bandwidth* dibagi secara merata guna kualitas koneksi jaringan lebih stabil [13]. QoS memberikan kualitas yang baik dan stabil apabila perutean bersifat dinamis, sehingga akan mencapai efisiensi dan fleksibilitas yang tinggi [14].

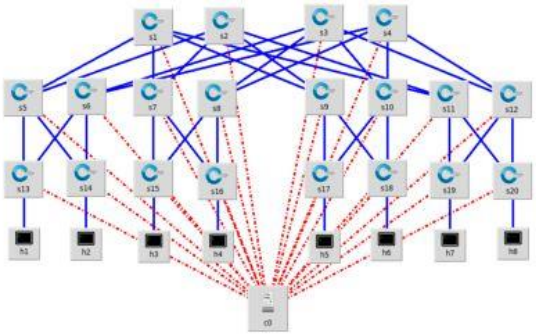
Pada pengujian sebelumnya yang dilakukan oleh performa BFS pada pencarian jalur mengalami kemacetan yang menyebabkan *routing* tidak berkerja secara maksimal [9]. Berbagai pengujian telah dilakukan dengan mengadopsi algoritma BFS sebagai optimasi *routing* sebagai acuan [15]. mengusulkan sebuah algoritma dengan menggunakan perbandingan dalam pengiriman *packet forwarding* pada arsitekrut SDN. Usulan metode tersebut disimulasikan dengan menggunakan *virtual machine*. Hal yang sama diusukan dalam Aprilia Kartika Sriastunsi dkk pada pengujian tersebut mengusulkan *routing* dengan algoritma BFS dimana pada pengujian trafik QoS pada *packet loss* sebesar 4,98% prosentase yang jauh dari stabil [9].

Untuk memperbaiki pengujian *routing* yang belum efektif pada algoritma BFS, maka pengujian ini mengusulkan skema yang baru dari pengujian sebelumnya, yaitu dengan modifikasi algoritma BFS pada pencarian jalur. Tujuan utama dari modifikasi jalur adalah dalam proses pengiriman paket data jaringan yang lebih efesien dan fleksibilitas. Karena pemanfaatan jalur pada *routing* diharapkan mampu meningkatkan skalabilitas pada proses *routing* jaringan dan memperkuat daya pengiriman paket data layanan jaringan (QoS). Alat pengujian menggunakan *tools Iperf* merupakan *tools* yang digunakan untuk mengukur kecepatan *bandwitdth* dan kualitas layanan jaringan QoS [16].

2. METODE PENELITIAN

2.1. Desain Topologi *Fat-Tree*

Pada tahap desain topologi jaringan menggunakan topologi *fat-tree* dengan membuat *design* topologi menggunakan mininet. Mininet adalah fitur dari linux sebagai *emulator* jaringan SDN yang dapat mensimulasikan kinerja antara *end-host, switch, router, controller*, dan link dalam sebuah kernel linux [6].



Gambar 1. Topologi *Fat-Tree*

Mengutip dari gambar 1, desain topologi yang menggunakan 8 *host*, 20 *switch*, dan 1 *controller*, dimana topolgi ini disebut dengan topologi *fat-tree* merupakan topologi yang sangat kompleks. Pada host terdapar terdapat address list digunakan untuk membuat daftar alamat IP Address yang dapat digunakan dalam untuk konfigurasi [17].

Topologi *fat-tree* diperkanalkan pertama kali oleh ilmuan Bernama *al-fares et al* untuk membangun sebuah jaringan dalam pusat data. Kemudian seiring berjalannya waktu ada banyak pengujian yang menggunakan topologi ini dalam pengujian jaringan pusat data [4]. Banyak keuntungan yang dapat diperoleh oleh topologi *fat-tree* dalam pengujian ini. Topologi *fat-tree* bersifat *fault tolerance* yang berarti dapat mentoleransi pengiriman saat terjadi suatu kegagalan dalam mengirimkan data.banyaknya jalur yang tersedia memungkinkan untuk menggunakan jalur lain dalam mencari jalur alternatif.

2.2. Algoritma BFS

Hampir sama seperti DFS, *Breadth First Search* (BFS) juga merupakan algoritme yang bersifat *unweighted graph* untuk melakukan pencarian jalur pada graf atau *tree*. Namun pencarian jalur dilakukan secara melebar atau per level (tingkatan) *tree*. Ekspansi dilakukan ke setiap *node* tetangga pada level yang sama. Algoritme ini menggunakan struktur data *queue* (antrian) yang memiliki kompleksitas sebesar O(V+E) dan kegunaan *queue* hampir sama seperti *stack* pada DFS. Namun ekspansi dilakukan secara melebar di setiap level sebelum melanjutkan ke level node berikutnya [9].

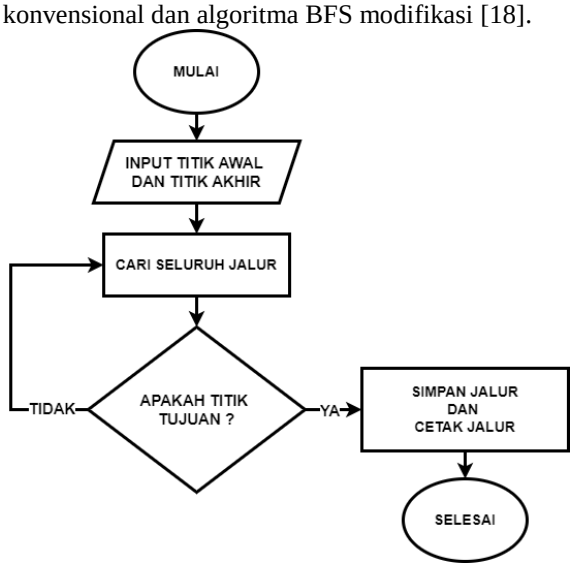
```
1  procedure BFS(Graph, source):
2      create a queue Q
3      enqueue source onto Q
4      mark source
5      while Q is not empty:
6          dequeue an item from Q into v
7          for each edge e incident on v in Graph:
8              let w be the other end of e
9              if w is not marked:
10                 mark w
11                 enqueue w onto Q
```

Gambar 2. *Pseudocode* Algoritma BFS

Pada gambar 2, *pseudocode* algoritme BFS BFS bekerja dengan cara melakukan traversal terhadap semua simpul yang bersebelahan dengan simpul awal secara melebar dan dilakukan secara tertahap kedalaman demi kedalaman (*depth by depth*). Algoritma BFS diimplementasikan menggunakan prinsip *First In First Out* (FIFO) [18].

2.3. Pencarian Jalur Algoritma BFS Konvensional

Proses pencarian jalur pengujian ini menggunakan algoritma *Breadth First Seacrh* (BFS) yang akan dibandingkan dengan algoritma BFS konvensional dan algoritma BFS modifikasi [18].



Gambar 3. Alur Pencarian Jalur Algoritma BFS Konvensional

Alur pencarian jalur algoritma BFS konvensional pada gambar 3, dijelaskan pada point berikut :

1) Input titik awal dan titik akhir pada algoritma BFS konvensional, yaitu (host1 tujuan ke host 8) untuk melakukan pengujian jalur
2) Dari titik awal, dimulai proses melakukan pencarian seluruh jalur untuk mengidentifikasi node mana yang akan dilewati dan dilanjutkan melakukan pencarian ke level tetangga dengan menginisialisasi ke setiap nodenya.
3) Kondisi menyatakan apakah titik tujuan, maka kondisi iya akan melanjutkan dengan merekomendasikan pencarian jalur dari algoritma BFS konvensional dan mencetak jalur yang dihasilkan. Bila kondisi tidak maka melakukan perulangan pada pencarian ke level tetangga (node sejajarnya) sampai berhasil dan jalur yang direkomendasikan sudah berhasil di tampilkan.

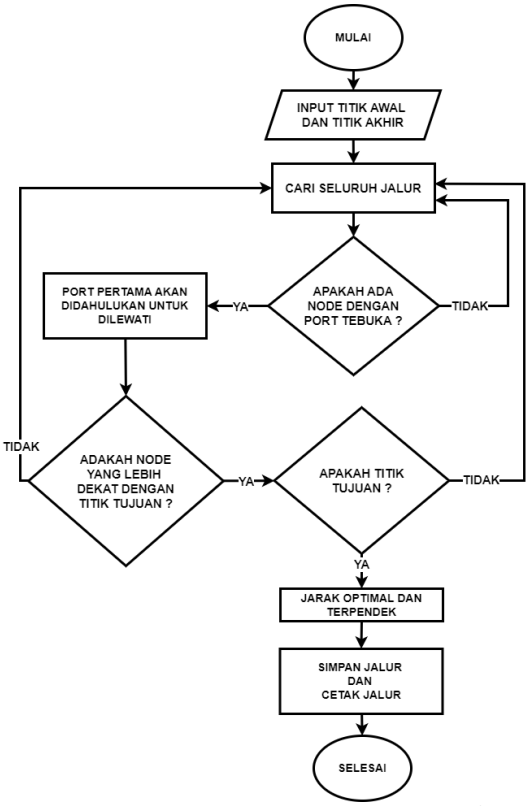
2.4. Pencarian Jalur Algoritma BFS Modifikasi

Algoritma BFS digunakan pada optimasi *routing* dalam pengujian [9] yang menerapkan algoritma BFS konvensional bahwa pengujian tersebut memerlukan waktu yang cukup lama dalam pemilihan jalur yang mengakibatkan proses *routing* tidak stabil dan mengalami penurunan QoS. Maka dari itu skema pengujian ini mengusulkan dengan

memodifikasi algoritma BFS pada jalur untuk menangani kemacetan jaringan.

Pada alur pencarian algoritma BFS yang dimodifikasi mengekspensi node terpendek dan mengutip pada gambar 4, dapat dilihat alur pencarian jalur algoritma BFS Modifikasi dijelaskan pada point berikut :

1) Input titik awal dan titik akhir pada algoritma BFS modifikasi.
2) Setelah terinput maka akan melakukan pencarian seluruh jalur untuk mengidentifikasi *node* mana yang akan dilewati.
3) Ada kondisi menyatakan, apakah ada node dengan port terbuka, jika iya maka akan diproses untuk melewati *port* pertama. Bila tidak, maka akan melakukan perulangan ke cari seluruh jalur sampai berhasil menemukan *port* terbuka. *port* adalah jalur yang digunakan untuk menghubungkan komputer/server dalam sebuah jaringan. *Port* terbuka berfungsi untuk identifikasi *node* mana yang belum terhubung, karena *port* merupakan bagian penting untuk proses *routing.*
4) Terdapat kondisi mencari node yang paling dekat dengan titik tujuan. Jika kondisi iya, maka akan melanjutkan ke kondisi titik tujuan. Bila menyatakan tidak, maka akan melakukan perulangan ke cari seluruh jalur sampai proses *routing* berhasil. Kondisi perbandingan node adalah *point* utama dalam algoritma BFS modifikasi untuk proses optimasi *routing* dikarenakan jalur yang akan dilewati sudah direkomendasikan dengan perbandingan *node* dan *port* terbuka untuk mencapai ke tujuan atau titik akhir.
5) Pada kondisi menyatakan apakah titik tujuan, jika iya maka akan merekomendasikan jarak optimal dan terpendek untuk proses *routing*. Bila tidak, maka akan melakukan perulangan lagi ke cari seluruh jalur sampai berhasil.
6) Proses simpan jalur dan cetak jalur adalah rekomendasi dari algoritma BFS modifikasi.



Gambar 4. Alur Pencarian Jalur Algoritma BFS Modifkasi

Algoritma BFS modifikasi dalam pencarian jalur yang memiliki kesamaan akan ringkas dengan port yang bisa terhubung, kecepatan dan tingkat akurasi sangat efisien. Memiliki kelebihan yang dalam penelurusan *node* algoritma dilakukan secara bertahap, mengeksplorasi kedalaman demi kedalaman. Hal ini membuat algoritma BFS yang dimodifikasi bekerja lebih optimal dibandingkan algoritma BFS konvensional apabila node tujuan berada tepat dalam jangkauan kedalam yang sudah ditentukan. Algoritma BFS menggunakan konsep FIFO (*First In First Out*) [18].

2.5. Metrik Penilaian Jalur

Jalur yang dihasilkan disini adalah jalur tanpa bobot. Oleh karena itu diperlukan bobot untuk memberikan *cost* dari setiap traversal. Pada pengujian pembobotan *link* dari jalur dihitung menggunakan Cisco OSPF yang ditunjukkan oleh persamaan (1) dibawah ini [18]:

$$ew = \frac{BR}{BL} \quad (1)$$

Dimana *ew* adalah *edge weigth* (*link cost*), B_R adalah *reference bandwidth* yang besarnya 100 Mbps sesuai pada protokol OSPF dari Cisco dan B_L adalah *link bandwidth.*

2.6. *Quality Of Service* (QoS)

Quality Of Service adalah kemampuan suatu jaringan untuk menyediakan layanan yang baik

dengan menyediakan bandwidth, mengatasi jiiter dan delay. Paramter QOS adalah *latency*, *jitter*, *packet loss*, *throughput*, MOS, *echo cancellation* dan PDD. QOS sangat ditentunkan oleh kualitas jaringan yang digunakan [19]. Beberapa paramater yang akan diujikan pada pengujian adalah *delay*, *jitter*, *packet loss* dan *throughput*. QoS berkaitan erat dengan data multimedia, layanan multimedia, dan *real-time* multimedia.

2.6.1. *Delay* (*Latency*)

Delay (*Latency*) didefinisikan sebagai total waktu tunda suatu paket yang diakibatkan oleh proses transmisi dari satu titik lain yang menjadi tujuannya. Sebuah pengujian mendefinisikan bahwa delay merupakan waktu yang dibutuhkan data untuk menempuh jarak dari asal ke tujuan [20]. *Delay* di dalam jaringan dapat digolongkan sebagai berikut *delay processing*, *delay packetization*, *delay serialization*, *delay jitter buffer* dan *delay network* [19]. Mengutip dari tabel 1, merupakan standarisasi nilai *delay* sebagai acuan untuk pengujian parameter QoS [21].

Tabel 1. Standarisasi Nilai Delay Versi TIPHON (1999)

Kategori Latensi	Besar Delay (ms)	Indeks
Sangat Bagus	< 150 ms	4
Bagus	150 ms s/d 300 ms	3
Sedang	300 ms s/d 450 ms	2
Jelek	> 450 ms	1

Berikut adalah standart pada persamaan (2) nilai *delay* versi TIPHON (1999) yang digunakan sebagai acuan dalam pengujian [21]:

$$Delay = \frac{Packet\ Length}{Link\ Bandwidth} x\ 100 \qquad (2)$$

2.6.2. *Jitter*

Jitter atau variasi kedatangan paket, hal ini diakibatkan oleh variasi – variasi dalam Panjang antrian, dalam waktu pengolahan data, dan juga dalam waktu penghimpunan ulang paket – paket di akhir perjalanan *jitter*. *Jitter* lazimnya disebut variasi *delay* berhubungan erat dengan *latency*, yang menunjukkan banyaknya variasi *delay* pada tranmisi data di jaringan. Selain itu kecepatan terima dan kirim paket dari setiap *node* juga dapat menyebabkan *jitter* [20]. *Delay* antrian pada *router* dan *switch* dapat menyebabkan *jitter* [19]. Mengutip dari tabel 2, merupakan standarisasi nilai *jitter* sebagai acuan untuk pengujian parameter QoS [21].

Tabel 2. Standarisasi Nilai Jitter Versi TIPHON (1999)

Kategori Jitter	Jitter (ms)	Indeks
Sangat Bagus	0 ms	4
Bagus	0 ms s/d 75 ms	3
Sedang	75 ms s/d 125 ms	2
Jelek	125 ms s/d 225 ms	1

Berikut adalah standarisasi pada persamaan (3) nilai *jitter* versi TIPHON (1999) yang digunakan sebagai acuan dalam pengujian [21]:

$$Jitter = \frac{Total\ Variasi\ Delay}{Total\ Packet\ yang\ Diterima} x\ 100 \qquad (3)$$

2.6.3. *Packet Loss*

Packet Loss adalah merupakan suatu parameter yang menggambarkan suatu kondisi yang menunjukkan jumlah total paket yang hilang. Salah satu penyebab *packet loss* adalah antrian yang melebihi kapasitas *buffer* pada setiap *node*. Dalam penerapan kehidupan sehari-hari, packet loss akan mempengaruhi kualitas layanan yang menggunakan protokol UDP (*User Datagram Protocol*) dan ICMP (Internet *Control Messege Protocol*) [20]. Beberapa penyebab terjadinya *packet loss* yaitu :

1) *Congestion*, disebabkan terjadinya antrian yang berlebihan dalam jaringan.
2) *Node* yang bekerja melebihi kapasitas buffer.
3) *Memory* yang terbatas pada node.
4) *Policing* atau control terhadap jaringan untuk memastikan bahwa jumlah trafik yang mengalir sesuai dengan besarnya *bandwidth*. Jika besarnya trafik yang mengalir di dalam jaringan melebihi dari kapasitas *bandwidth* yang ada maka policing control akan membuang kelebihan trafik yang ada [19].

Mengutip dari tabel 3, merupakan standarisasi nilai *packet loss* sebagai acuan untuk pengujian paremater QoS [21].

Tabel 3. Standarisasi Nilai Packet Loss Versi TIPHON (1999)

Kategori Degredasi	Packet Loss	Indeks
Sangat Bagus	0 %	4
Bagus	3 %	3
Sedang	15 %	2
Jelek	24 %	1

Berikut adalah standarisasi pada persamaan (4) nilai *packet loss* versi TIPHON (1999) yang digunakan sebagai acuan dalam pengujian [21]:

$$Packet\ Loss = \frac{(Paket\ data\ dikirim - Paket\ data\ diterima)}{Paket\ data\ yang\ dikirim} x\ 100\% \qquad (4)$$

2.6.4. *Throughput*

Throughput yaitu kecepatan (*rate*) transfer data efektif, yang diukur dalam *bit per second* (bps). *Throughput* merupakan jumlah total kedatangan paket yang sukses yang diamati pada destination selama interval waktu tertentu dibagi oleh durasi interval waktu tersebut. Ada definisi dari pengujian lain yang menyatakan *throughput* sebagai jumlah total kedatangan paket yang diamati pada tujuan paket selama waktu tertentu dibagi oleh durasi interval waktu tersebut. Semakin besar throughput, maka semakin baik kinerja suatu jaringan komputer [20]. Dalam standar TIPHON *throughput* dihitung dalam persen, untuk mendapatkan nilai *throughput* dalam persen hasil perhitungan *throughput* kemudian dibagi dengan besarnya nilai *bandwidth* dan dikalikan 100 % untuk mengetahui besarnya persentase nilai

throughput yang sebenarnya [19]. Mengutip dari tabel 4, merupakan standarisasi nilai *throughput* sebagai acuan untuk pengujian paremater QoS [21].

Tabel 4. Standarisasi Nilai Throughput Versi TIPHON (1999)

Kategori	Throughput	Indeks
Sangat Bagus	100 %	4
Bagus	75 %	3
Sedang	50 %	2
Jelek	< 25 %	1

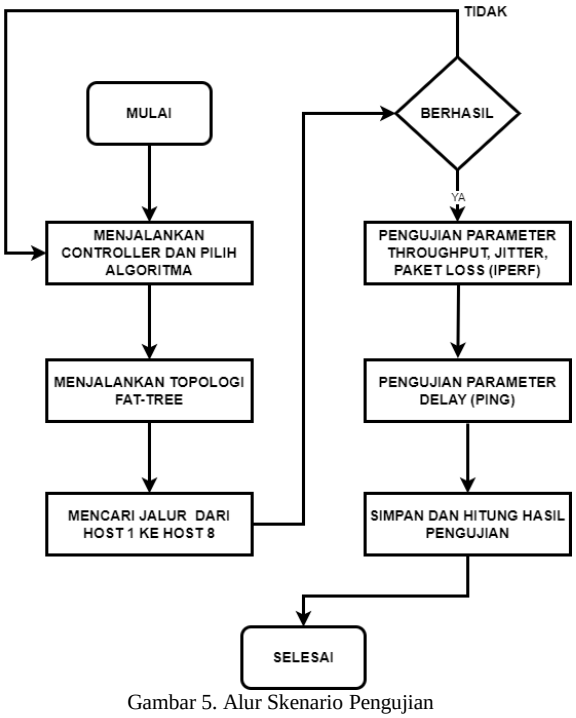
Berikut adalah standarisasi pada persamaan (5) nilai *Throughput* versi TIPHON (1999) yang digunakan sebagai acuan dalam pengujian [21]:

$$Throughput = \frac{Packet\ Data\ Diterima}{Lama\ Pengamatan} x\ 100 \qquad (5)$$

2.7. Skenario Pengujian

Pada skenario pengujian jalur yaitu algoritma BFS konvensional dibandingkan dengan algoritma BFS modifikasi untuk pengambilan data. Selanjutnya dilakukan pengujian parameter QoS dengan tujuan untuk mendapatkan data akurasi dalam pengujian ini.

Mengutip pada gambar 5, merupakan implementasi dari pengujian yang meliputi menjalankan *controller* dan pilih algoritma dilanjutkan dengan menjalankan topologi pada mininet, kemudian cari jalur dari host 1 ke host 8. Jika berhasil maka dilanjutkan dengan pengujian *delay* dengan cara ping ip address, seteleh mendapatkan nilai rata-rata, dilanjutkan dengan pengujian parameter *throughput, jitter* dan *packet loss* secara bersamaan karena parameter ini sudah menjadi kesatuan dalam pengujian.



Gambar 5. Alur Skenario Pengujian

Pada langkah terakhir simpan dan hitung nilai rata-rata dari semua pengujian parameter guna mendapatkan hasil dalam komparasi algoritma BFS konvensional dan algoritma BFS modifikasi. Nilai rata-rata diambil dalam analisa hasil pengujian semua parameter. Untuk menyatakan perbandingan dari dua algoritma BFS konvensional dan algoritma BFS modifikasi pada Diskusi, maka mengutip dari persamaan (6) sebagai acuan dari TIPHON [21] yaitu sebagai berikut:

$$Nilai\ Prosentase = \frac{(Paket\ data\ dikirim - Paket\ data\ diterima)}{Paket\ data\ yang\ dikirim} x\ 100\% \qquad (6)$$

3. HASIL DAN PEMBAHASAN

Untuk mengevaluasi performansi dari protokol *routing* yang diusulkan algoritma BFS modifikasi, sebuah simulasi dilakukan menggunakan simulator linux ubuntu 20.04 dengan mengembangkan pada pencarian jalur alternatif terpendek untuk membantu dalam proses *routing* berbasis SDN lebih efektif. Dalam simulasinya, pengujian ini mengadopsi parameter-parameter yang pada umumnya digunakan dalam pengujian lainnya untuk mengevaluasi performansi dari modifikasi algoritma BFS yang diusulkan dan berikut tabel pengujian QoS.

Tabel 5. Skenario Parameter Pengujian

Parameter	Nilai (Skenario)
Delay	50 – 1000 Mbps (20)
Packet Loss	50 – 1000 Mbps (20)
Jitter	50 – 1000 Mbps (20)
Throughput	50 – 1000 Mbps (20)

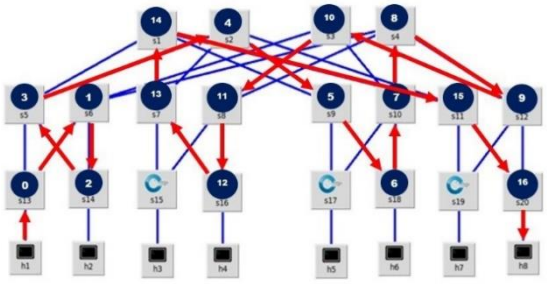
Pada tabel 5, menunjukkan dalam parameter utama yang dijadikan sebagai acuan dalam melakukan komparasi antara algoritma BFS konvensional dan algoritma BFS modifikasi adalah parameter QoS digunakan sebagai evaluasi kualitas kinerja jaringan dengan jangka waktu. Selanjutnya pada pengujian jalur mempengaruhi terhadap parameter yang diuji karena dampak pada pemilihan jalur akan terlihat pada kualitas jaringan QoS.

3.1. Hasil Pencarian Jalur

Prosedur pertama dalam proses pengujian jalur dengan cara ping host 1 ke host 8. Pengujian ini guna mencari jalur yang dilewati dalam algoritma BFS konvensional dan algoritma BFS modifikasi.

3.1.1. Hasil Pencarian Jalur Menggunakan Algoritma BFS Konvensional

Hasil pengujian dengan topologi *fat-tree* dapat dilihat pada gambar 6, dimana tanda panah merah menunjukkan jalur yang sudah dilewati oleh algortima BFS konvensional.



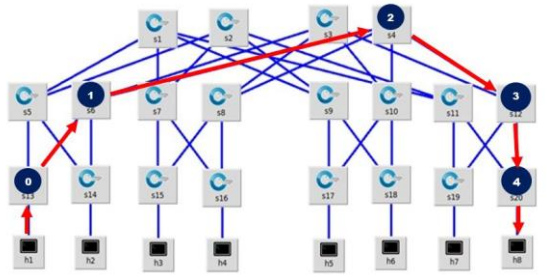
Gambar 6. Pencarian Jalur Algoritma BFS Konvensional

Hasil ditemukan jalur – jalur dari algoritma BFS konvensional dari host 1 ke host 8 dengan melewati cukup banyak *switch*, dimana kondisi yang memerlukan waktu sangat lama dalam mencari jalur karena dibutuhkan *routing* dengan melewati *switch*, maka menggunakan algortima BFS terjadi keterlambatan, kemacetan, dan membutuhkan waktu yang lama untuk mencapai akhir sehingga jalur memiliki kesamaan.

3.1.2. Hasil Pencarian Jalur Menggunakan Algoritma BFS Modifikasi

Hasil pengujian topologi *fat-tree* dapat dilihat pada gambar 7, menunjukkan algoritma bfs modifikasi memiliki jalur lebih pendek dan singkat.

Pencarian Jalur menggunakan algoritma BFS yang dimodifikasi menghasilkan jalur yang independent dan tidak adanya persamaan jalur yang sudah dilalui, sehingga waktu yang ditempuh untuk sampai ke tujuan akhir sangat singkat dengan memanfaatkan jalur yang sudah ditentukan. Terdapat perbedaan dalam pemilihan jalur bukti sebagai berikut.



Gambar 7. Pencarian Jalur Algoritma BFS Modifikasi

Tabel 6. Hasil Pencarian Jalur

Algoritma BFS Konvensional	Algoritma BFS Modifikasi
(13, 6, 14, 5, 2, 9, 18, 10, 4, 12, 3, 8, 16, 7, 1, 11, 20)	(13, 6, 4, 12, 20)

Pada tabel 6 menunjukkan dalam pencarian jalur pada algoritma BFS konvensional berjumlah 17 *rute/switch* yang dilewati untuk sampai ke host 8 dikarenakan belum ada modifikasi terhadap jalur dibandingkan dengan algoritma BFS modif dengan total 5 dikarenakan pemanfaatan modifikasi pada jalur sehingga dampak sangat signifikan dan efektif dalam menyelesaikan permasalahan kemacetan pada jalur. Algoritma BFS modifikasi jauh lebih unggul dalam menyelesaikan optimasi *routing* pada pengujian jalur dengan pemangkasan jalur dari 17 menjadi 5.

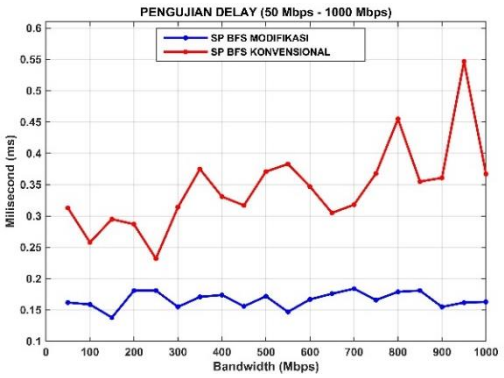
3.2. Pengujian Parameter QoS

Salah satu cara untuk mengukur kinerja layanan internet adalah dengan menggunakan parameter QoS yang terdiri dari *Delay*, *Throughput*, *Jitter* dan *Packet Loss*. Pengujian dilakukan pada SP (*Single-Path*) algoritma BFS konvensional dan SP (*Single-Path*) algoritma BFS Modifikasi. Skenario pengujian dilakukan dari bandwidth 50 – 1000 Mbps dengan kelipatan 50 Mbps dalam 20 kali pengujian dalam 15 detik.

3.2.1. Hasil Pengujian *Delay*

Pada gambar grafik 8, dapat dilihat hasil dari pengujian *delay* dihasilkan dengan nilai rata-rata yaitu, 0,34495 ms artinya algoritma BFS konvensional bekerja tidak stabil dengan memerlukan waktu yang lama. Sedangkan pada algoritma BFS modifikasi jauh lebih unggul pada pengujian delay dengan nilai rata – rata yaitu 0,16645 ms artinya stabil dalam pengiriman paket dengan waktu yang singkat.

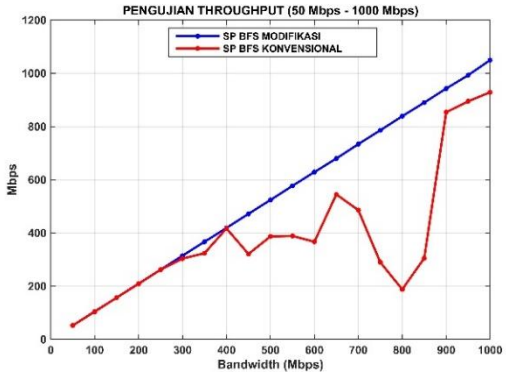
Maka dapat disimpulkan bahwa pengujian *delay* dari algortima BFS modifikasi lebih unggul dengan memiliki nilai lebih kecil yang berdampak positif pada proses pengiriman paket *bandwidth* stabil dengan waktu yang singkat dibandingkan dengan algoritma BFS konvensional pada setiap bandwidth mengalami naik turun dengan waktu yang lama, sehingga algoritma BFS modifikasi cukup stabil dan berdampak positif pada kualitas jaringan (QoS).



Gambar 8. Hasil Pengujian *Delay*

3.2.2. Hasil Pengujian *Throughput*

Pada pengujian *throughput* digambar 9, dapat dilihat hasil pengujian terdapat perbedaan yang signifikan pada algoritma BFS modifikasi yang lebih unggul.

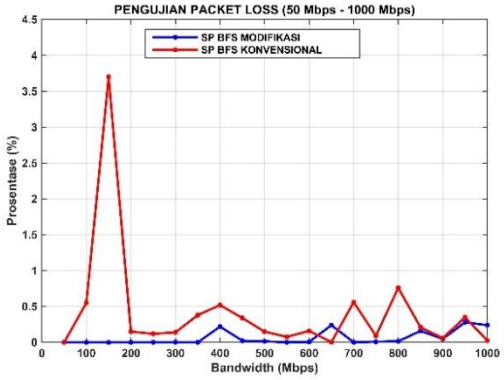


Gambar 9. Hasil Pengujian *throughput*

Hasil pada gambar 9 menunjukkan pengujian bagus dalam mempertahankan kestabilan *bandwidth* dengan nilai rata-rata yaitu 550,22 Mbps, sedangkan pada algoritma BFS konvensional pada nilai *throughput* mengalami degradasi (penurunan), tidak stabil dengan nilai rata-rata 389,27 Mbps. Konsistens pengiriman paket bandwidth terjadi pada algoritma BFS modifikasi dari 50 – 1000 Mbps bahkan tidak mengalami penurunan sedikitpun dibandingkan dengan algoritma BFS konvensional yang terjadi pada *bandwidth* tertentu yang mengalami penurunan yang jauh mengurangi dari paket *bandwidth* yang dikirim. Penyebabnya adalah kestabilan terhadap paket yang dikirim karena dengan jalur yang singkat dengan waktu yang cepat, karena pada algoritma BFS modifikasi pemanfaatan jalur yang singkat yang berdampak positif untuk kualitas jaringan (QoS).

3.2.3. Hasil Pengujian *Packet Loss*

Pada hasil pengujian *packet loss* digambar grafik 10, dapat dilihat hasil pengujian parameter *packet loss*. Pengujian *packet loss* dapat ditimbulkan karena pengiriman paket gagal dalam proses *routing*.



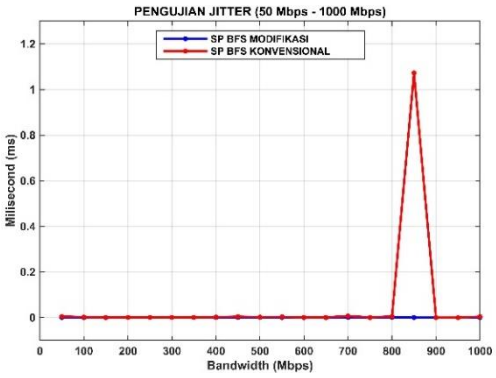
Gambar 10. Hasil Pengujian *Packet Loss*

Mengutip dari gambar 10, nilai rata-rata adalah 0,417325% atau 0,42 % untuk algoritma BFS Konvensional artinya pengujian tidak stabil dikarenakan jalur yang dilewati oleh paket *bandwidth* dengan jumlah banyak dan memerlukan waktu yang lama, sedangkan algoritma BFS yang dimodifikasi memiliki nilai rata-rata adalah 0,063025% atau 0,06% artinya proses pengujian sangat stabil dikarenakan jalur yang dilewati oleh paket sangat singkat dalam waktu yang cepat.

Penyebab dampak negatif pada algoritma BFS Konvensional adalah jalur yang dilewati sangat banyak sehingga berpotensi pengiriman paket *bandwidth* tidak stabil dan terjadi pengurangan prosentase *bandwidth* beserta kualitas jaringan yang dihasilkan.

3.2.4. Hasil Pengujian *Jitter*

Pada hasil pengujian jitter digambar 11, dapat dilihat hasil pengujian dengan perbedaan signifikan terutama pada *bandwidth* 850 Mbps yang meningkat artinya tidak stabil dengan menggunakan algoritma BFS konvensional, sedangkan pada algoritma BFS modifikasi memiliki nilai akurasi yang lebih rendah diantara 0,1 ms artinya nilai pengujian jitter tidak berdampak negatif pada performa jaringan.



Gambar 11. Hasil Pengujian *Jitter*

Dari hasil pengujian gambar 11 pengujian *jitter* adalah algoritma BFS konvensional bekerja tidak stabil dengan waktu yang lama dalam pengiriman paket *bandwidth*, sedangkan pada algoritma BFS modifikasi lebih unggul dalam pengujian *jitter* yang memiliki nilai lebih rendah artinya pengiriman paket sangat stabil dengan waktu yang singkat sehingga berdampak pada kualitas jaringan (QoS).

4. DISKUSI

Dari hasil pengujian parameter *Quality Of service* (QoS) diatas yaitu *delay*, *throughput*, *jitter*, dan *packet loss* dihasilkan dengan nilai rata – rata menggunakan algoritma BFS konvensional dan algoritma BFS modifikasi. Maka pada pengujian ini mengambil data keseluruhan parameter pengujian parameter QoS sebagai bahan perbandingan pada tabel sebagai berikut:

Tabel 7. Hasil Pengujian Parameter QoS

Parameter	BFS Konvensional	BFS Modifikasi
Delay	0,34495 ms	0,16645 ms
Throughput	389,27 Mbps	550,22 Mbps
Jitter	0,05575 ms	0,00045 ms
Packet Loss	0,417325 %	0,063025 %

Hasil pengujian parameter QoS pada tabel 7, menunjukkan bahwa pengujian parameter QoS

mengalami degradasi dengan nilai prosentase pada *delay* sebesar 65%, *packet loss* 99%, *jitter* 84%, dan *throughput* mengalami peningkatan sebesar 41%, mengutip dari standart QoS versi tiphon [21], bahwa degradasi pada *delay*, *jitter* dan *packet loss* mengalami penurunan artinya optimasi *routing* berjalan efektif dan stabil menggunakan algoritma BFS modifikasi dibandingkan dengan algoritma BFS konvensional yang mengalami ketidakstabilan saat pengiriman paket *bandwidth* penyebab utama adalah jalur yang dilewati cukup panjang dan memerlukan waktu yang lama yang mengakibatkan terjadi kemacetan jaringan. Hasil pengujian bisa dibandingkan dengan pengujian sebelumnya dari [15] menghasilkan nilai rata-rata terutama pada *packet loss* sebesar 81,3 % sangat jauh sekali dangan isi pada tabel 7, setelah dilakukan modifikasi jalur pada algoritma BFS.

5. KESIMPULAN

Penerapan algoritma BFS modifikasi mampu melakukan fleksibilitas dengan baik, ditunjukkan pada bukti pada hasil pengujian jalur dan pengujian parameter QoS. Algoritma BFS modifikasi memberikan performansi jaringan yang lebih baik dalam optimasi *routing*. Dibuktikan dengan terjadi degradasi dengan nilai prosentase pada *delay* sebesar 65%, *packet loss* 99%, *jitter* 84%, dan *throughput* mengalami peningkatan sebesar 41%, artinya algoritma BFS modifikasi memiliki optimasi routing berjalan dengan baik dan sanggup membuat perutean menjadi lebih pendek dalam jangka waktu singkat. Grafik parameter pengujian juga menunjukkan perubahan signifikan setelah menerapkan pemanfaatan jalur. Dengan adanya skema ini, jaringan dapat ditingkatkan lebih efisien, membantu proses *routing* lebih efektif untuk penanganan kemacetan jaringan dan memberikan kualitas jaringan QoS lebih baik.

DAFTAR PUSTAKA

[1] M. M. Ridhoilah, A. Basuki, and K. Amron, "Penggunaan Jalur dalam Multipath Routing secara Proporsional Berdasarkan Kebutuhan Bandwidth Pengguna pada Software Defined Networking," 2020. [Online]. Available: http://j-ptiik.ub.ac.id

[2] D. A. Mutiara, K. N. Isnaini, and D. Suhartono, "Network Programmability for Network Issue Using Paramiko Library," *J. Tek. Inform.*, vol. 4, no. 4, pp. 751–758, 2023, doi: 10.52436/1.jutif.2023.4.4.691.

[3] A. Shafira, S. N. Hertiana, and L. V. Yovita, "Analisis Performansi Integrasi Sistem Named Data Networking Dan Software Defined Network Untuk Komunikasi Data," vol. 1, no. 1, pp. 28–35, 2023, doi: https://doi.org/10.25124/jnst.v1i1.6727.

[4] D. Krisyanto, R. A. Siregar, and K. Amron, "Pencarian Jalur Alternatif berbasis Algoritme Yen's pada Jaringan Software Defined Network," vol. 6, no. 3, pp. 1165–1174, 2022, [Online]. Available: http://j-ptiik.ub.ac.id

[5] I. P. A. E. Pratama and K. C. Bakkara, "Pengujian QoS Pada Implementasi SDN Berbasis Mininet dan OpenDaylight Menggunakan Topologi Tree," *J. Sisfokom (Sistem Inf. dan Komputer)*, vol. 10, no. 2, pp. 170–175, 2021, doi: 10.32736/sisfokom.v10i2.1141.

[6] M. Khatami, "Studi ; Performansi ; Protokol Routing ; IS- IS ; Pada Arsitektur Jaringan ; Software ; Defined ; Network ;(SDN)," vol. 10, no. 2, 2023.

[7] F. Nisa and S. Ramadona, "Sistem Pencegahan Serangan Distributed Denial Of Service Pada Jaringan SDN," *J. Sistim Inf. dan Teknol.*, vol. 5, no. 3, pp. 1–8, 2023, doi: 10.60083/jsisfotek.v5i3.269.

[8] I. Arthalia Wulandari and P. Sukmasetyan, "Implementasi Algoritma Dijkstra untuk Menentukan Rute Terpendek Menuju Pelayanan Kesehatan," *J. Ilm. Sist. Inf.*, vol. 1, no. 1, pp. 30–37, 2022, doi: 10.24127/jisi.v1i1.1953.

[9] A. K. Sriastunti, R. Primananda, and W. Yahya, "Implementasi Routing pada OpenFlow Software-Defined Network dengan Algoritme Depth-First Search dan Breadth-First Search," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 3, no. 8, pp. 8112–8120, 2019.

[10] Y. S. Putra, M. T. Indriastuti, and F. S. Mukti, "OPTIMALISASI NILAI THROUGHPUT JARINGAN LABORATORIUM MENGGUNAKAN METODE HIERARCHICAL TOKEN BUCKET (STUDI KASUS: STMIK ASIA MALANG)," *Netw. Eng. Res. Oper.*, vol. 5, no. 2, p. 83, Oct. 2020, doi: 10.21107/nero.v5i2.161.

[11] R. Zuhdianto and F. S. Mukti, "a Clustering Optimization for Energy Efficiency in Wireless Sensor Network Using K-Means Algorithm," *J. Tek. Inform.*, vol. 4, no. 1, pp. 225–234, 2023, doi: 10.52436/1.jutif.2023.4.1.523.

[12] S. Shidqi, D. A. Sulistyo, and F. A. Ahda, "Pembuatan Infrastruktur Database Menggunakan Metode Replikasi Untuk Pelanggan Jagoan Hosting," *J. Ilm. Teknol. Inf. Asia*, vol. 16, no. 1, p. 65, 2022, doi: 10.32815/jitika.v16i1.702.

[13] Siddik.mohd, "ANALISIS QUALITY OF SERVICE JARINGAN LOCAL AREA

NETWORK MENGGUNAKAN MIKROTIK ROUTERBOARD750 Mohd . Siddik Program Studi Sistem Informasi , STMIK Royal Kisaran email : mohdsiddik27@gmail.com PENDAHULUAN Saat ini penggunaan Jaringan komputer bukan lagi meru," *J. Teknol. dan Sist. Inf.*, vol. V, no. 2, pp. 113–118, 2019.

[14] I. N. Khoerotunisa, S. N. Hertiana, and R. M. Negara, "Analysis of User Mobility Performance on Software Defined Wireless Network Using Dijkstra Algorithm," *J. Tek. Inform.*, vol. 2, no. 2, pp. 127–133, 2021, doi: 10.20884/1.jutif.2021.2.2.84.

[15] U. B. Hanafi, M. A. D. Arie, M. R. Fikri, and T. Irfan, "Analisis perbandingan algoritma Dijkstra dan Breadth First Search pada packet forwarding arsitektur SDN," *JITEL (Jurnal Ilm. Telekomun. Elektron. dan List. Tenaga)*, vol. 2, no. 1, pp. 57–66, Mar. 2022, doi: 10.35313/jitel.v2.i1.2022.57-66.

[16] V. Gueant, "iPerf - The TCP, UDP and SCTP network bandwidth measurement tool." 2023.

[17] A. Pramudita, "Implementation of Load Balancing With Per Connection Classifier and Failover and Utilization of Telegram Bot (Case Study : Pt Tujuh Media Angkasa) Implementasi Load Balancing Dengan Per Connection Classifier Dan Failover Serta Pemanfaatan Bot Telegram (," vol. 5, no. 1, pp. 273–282, 2024.

[18] I. Suryo Wicaksono and P. Hari Trisnawan, "Implementasi Multipath Routing menggunakan Algoritme Iterative Deepening Depth First Search pada OpenFlow Software-Defined Networking," 2021. [Online]. Available: http://j-ptiik.ub.ac.id

[19] A. Turmudi and F. Abdul Majid, "Analisis QoS (Quality Of Service) Dengan Metode Traffi Shaping Pada jaringan Interent (Studi Kasus : PT Toyonaga Indonesia)," *Sigma*, vol. 9, no. 4, pp. 2407–3903, 2019, [Online]. Available: https://jurnal.pelitabangsa.ac.id/index.php/sigma/article/view/445

[20] A. Z. Firmansah and A. Hadi, "KOMPARASI LOAD BALANCING METODE PCC DAN NTH PADA MIKROTIK IMPLENTASI DI AL IRSYAD TENGARAN 7 BATU," *JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 8, no. 1, pp. 21–26, Feb. 2023, doi: 10.29100/jipi.v8i1.3261.

[21] ETSI, "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); General aspects of Quality of Service (QoS)," *Etsi Tr 101 329 V2.1.1*, vol. 1, pp. 1–37, 2020.