PERBANDINGAN SIMULASI REKAYASA *TRAFFIC QUALITY OF SERVICE* (QOS) PADA *SOFTWARE DEFINED NETWORK* DENGAN MENCARI JALUR TERPENDEK MENGGUNAKAN ALGORITMA YEN DAN FLOYD-WARSHALL

Henokh F. Supusepa
PT. Citra Bahasa Global
e-mail: henokhsupusepa@gmail.com

Abstrak

Layanan dan teknologi internet berkembang pesat terhadap kompleksitas, desain, manajemen, dan operasi yang mengarah pada peningkatan jumlah perangkat yang terhubung dan lalu lintas di jaringan telah meningkat sangat cepat dalam beberapa tahun terakhir. Selain itu, muncul perkembangan layanan baru dan penggunaan mobilitas tinggi dan virtualisasi Server, komputasi awan , dan Internet of Things sedang dikembangkan menghasilkan arsitektur jaringan saat ini yang sangat terbatas kemampuannya. Teknologi Software Defined Network adalah konsep baru dalam merancang, mengelola dan mengimplementasikan arsitektur jaringan dengan memisahkan sistem kendali dari sistem penerusan/forward pada peralatan jaringan. Software Defined Network memungkinkan jaringan untuk menyesuaikan lingkungan secara dinamis untuk menyederhanakan pengelolaan dan meningkatkan skalabilitas jaringan melalui implementasi sederhana dari tambahan komponen dan layanan jaringan. Oleh karena itu, persyaratan yang lebih tinggi diajukan untuk meningkatkan kinerja jaringan dan akhirnya pengendalian jaringan menjadi semakin kompleks, tidak konsisten dan tidak fleksibel. Performa jaringan dapat dipengaruhi oleh banyak faktor, seperti konfigurasi jaringan, jumlah permintaan, dan metode manajemen jaringan.

Kata Kunci: Software Defined Network (SDN), *Quality of Service* (QOS), jalur terpendek, Ryu *controller*, Algoritma Yen, Algoritma Floyd-Warshall.

COMPARISON OF TRAFFIC QUALITY OF SERVICE (QOS) ENGINEERING SIMULATION ON DEFINED NETWORK SOFTWARE BY FINDING THE SHORTEST LINE USING YEN AND FLOYD-WARSHALL ALGORITHM

Abstract

Internet services and technology are rapidly expanding in terms of complexity, design, management, and operation leading to an increase in the number of connected devices, and the traffic on the network has increased very rapidly in recent years. Also, emerging new service developments and the use of high mobility and server virtualization, cloud computing, and the Internet of Things are being developed resulting in a network architecture that is currently very limited in capabilities. Software Defined Network technology is a new concept in designing, managing, and implementing network architecture by separating the control system from the forward system on network equipment. Software Defined Network allows the network to dynamically adjust the environment to simplify management and increase network scalability through the simple implementation of additional network components and services. Therefore, higher requirements are put forward to improve network performance and eventually network control becomes increasingly complex, inconsistent, and inflexible. Network performance can be affected by many factors, such as network configuration, number of requests, and network management methods.

Keywords: Software Defined Network (SDN), *Quality of Service* (QOS), shortest Path, Ryu *controller*, Yen Algorithm, Floyd-Warshall Algorithm

1. Pendahuluan

Layanan dan teknologi internet berkembang pesat terhadap kompleksitas, desain, manajemen, dan operasi yang mengarah pada peningkatan jumlah perangkat yang terhubung dan lalu lintas di jaringan telah meningkat sangat cepat dalam beberapa tahun terakhir. Dibandingkan dengan jaringan tradisional yang masih terdapat beberapa kelemahan, salah satunya adalah menangani masalah skalabilitas jaringan [1] dan kurang efisien jika digunakan untuk pengembangan aplikasi pengontrol jaringan, *Software Defined Network* memberikan kemudahan bagi pengguna untuk mengembangkan aplikasi pengontrol jaringan dengan memisahkan fungsi data.

Teknologi Software Defined Network adalah konsep baru dalam merancang, mengelola dan mengimplementasikan arsitektur jaringan dengan memisahkan sistem kendali dari sistem penerusan/forward pada peralatan jaringan. *Software Defined Network* memungkinkan jaringan untuk menyesuaikan lingkungan secara dinamis untuk menyederhanakan pengelolaan dan meningkatkan skalabilitas jaringan melalui implementasi sederhana dari tambahan komponen dan layanan jaringan. *OpenFlow* adalah protokol standar yang digunakan oleh *data plane* dan *control plane* untuk bisa berkomunikasi satu sama lain [2]. *Control Plane* adalah pengontrol yang akan mengontrol jaringan secara terpusat. *Control Plane* terdiri dari *controller* yang akan mengontrol layanan jaringan, seperti rekayasa lalu lintas, manajemen *bandwidth*, dll. Disisi lain, *Data Plane* adalah peralatan jaringan seperti *switch* dan *router*. Oleh karena itu, persyaratan yang lebih tinggi diajukan untuk meningkatkan kinerja jaringan dan akhirnya pengendalian jaringan menjadi semakin kompleks, tidak konsisten dan tidak fleksibel. Salah satu faktor yang mempengaruhi manajemen jaringan adalah perutean jaringan.

Penelitian ini dilakukan dengan tujuan untuk melakukan simulasi perbandingan dan analisis jaringan pemilihan jalur terbaik pada jaringan *Software Defined Network* yang diterapkan algoritma Yen dan algoritma Floyd-Warshall aristektur topologi yang sama, dilihat dari kinerja parameter QoS (*Quality of Service*) dengan menggunakan *protocol OpenFlow* disertai algoritma Yen dan algoritma Floyd-Warshall, diukur dari masing - masing nilai dari parameter QoS yang terdiri dari *Throughput*, *Delay*, dan *Packet Loss* [2]dengan melihat jalur terbaiknya.

2. Tinjauan Pustaka

A. *Controller* RYU

Ryu adalah framework dari *Software Defined Network* (SDN) yang dapat diimplementasikan sepenuhnya menggunakan bahasa pemrograman *python*. Ryu menyediakan API untuk komponen perangkat lunak, yang memungkinkan pengembang untuk dengan mudah membuat manajemen jaringan baru dan aplikasi kontrol [3]. Ryu mendukung berbagai protokol untuk perangkat jaringan, seperti *OpenFlow*, netconf, OF-config, dll. Dalam protokol *openFlow*, Ryu mendukung penuh versi 1.0, 1.2, 1.3, 1.4, 1.5 dan ekstensi Nicira. Sangat disarankan agar pengontrol ryu digunakan oleh pengembang untuk membangun program di SDN aliran terbuka, karena mendukung berbagai versi aliran terbuka.

B. *OpenFlow*

OpenFlow merupakan antarmuka komunikasi standar yang menjelaskan mekanisme komunikasi antara *control layer* dan *forwarding layer* dalam arsitektur SDN. *OpenFlow* menyediakan akses langsung dalam merubah dan melakukan manajemen forwarding plane pada perangkat jaringan secara fisik dan *virtual*. *Switch* dengan *protocol OpenFlow* di dalam arsitektur SDN, dibagi menjadi tiga bagian. Bagian tersebut adalah *Flow Table*, *Secure Channel* dan protokol *OpenFlow*. *OpenFlow channel* adalah antarmuka yang digunakan untuk menghubungkan komunikasi antara *OpenFlow controller* dan *OpenFlow switch*. Dalam jaringan konvensional, setiap *switch* hanya digunakan untuk meneruskan paket data melalui satu *port*, dan tidak dapat membedakan jenis protokol yang diperlukan. Tapi melalui *OpenFlow protocol* ini, *controller* dapat mengelola konfigurasi dan pengelolaan *switch*, menerima laporan dari *switch*, dan mengirim paket data ke *switch*.

C. *Software Defined Network*

Software Defined Network adalah contoh arsitektur baru di bidang jaringan Komputer yang dinamis, mudah diatur, hemat biaya, dan mudah beradaptasi membuatnya ideal untuk aplikasi dinamis dengan *bandwidth* tinggi saat ini. Arsitektur ini memisahkan kontrol jaringan dan fungsi penerusan ,

sehingga kontrol jaringan dapat diprogram secara langsung, dan infrastruktur dapat diprogram secara langsung. Sedangkan arsitektur yang mendasari jaringan ini yaitu lapisan aplikasi dan layanan jaringan. Tujuan utama SDN adalah untuk mencapai manajemen jaringan yang lebih baik, dengan tingkat hirarki dan kompleksitas yang tinggi, dan untuk memastikan bahwa semua keputusan sistem kontrol dibuat dari titik pusat. SDN memperkenalkan metode untuk meningkatkan level abstraksi dalam konfigurasi jaringan dan menyediakan mekanisme yang dapat secara otomatis merespons perubahan jaringan yang sering dan terus-menerus.

D. Algoritma Yen

Jin Y. Yen [3] menyampaikan pengertian Algoritma Yen pada penelitian nya bahwa "*Yen's algorithm computes single-source K-shortest loopless paths for a graph with non-negative edge cost*". Algoritma Yen merupakan pengembangan algoritma Dijkstra yang di mana algoritma ini menghitung jalur dengan sumber tunggal terpendek tanpa pengulangan grafik dengan tepi cost non-negatif.

```
function YenKSP(Graph, source, destination, K):
    // Determine the shortest path from the source to the destination.
    A[0] = Dijkstra(Graph, source, destination);
    // Initialize the set to store the potential kth shortest path.
    B = [];

    for k from 1 to K:
        // The spur node ranges from the first node to the next to last node in the previous k-shortest path.
        for i from 0 to size(A[k - 1]) - 2:

            // Spur node is retrieved from the previous k-shortest path, k - 1.
            spurNode = A[k-1].node(i);
            // The sequence of nodes from the source to the spur node of the previous k-shortest path.
            rootPath = A[k-1].nodes(0, i);

            for each path p in A:
                if rootPath == p.nodes(0, i):
                    // Remove the links that are part of the previous shortest paths which share the same root path.
                    remove p.edge(i,i + 1) from Graph;

            for each node rootPathNode in rootPath except spurNode:
                remove rootPathNode from Graph;

            // Calculate the spur path from the spur node to the destination.
            // Consider also checking if any spurPath found
            spurPath = Dijkstra(Graph, spurNode, destination);

            // Entire path is made up of the root path and spur path.
            totalPath = rootPath + spurPath;
            // Add the potential k-shortest path to the heap.
            if (totalPath not in B):
                B.append(totalPath);

            // Add back the edges and nodes that were removed from the graph.
            restore edges to Graph;
            restore nodes in rootPath to Graph;

        if B is empty:
            // This handles the case of there being no spur paths, or no spur paths left.
            // This could happen if the spur paths have already been exhausted (added to A),
            // or there are no spur paths at all - such as when both the source and destination vertices
            // lie along a "dead end".
            break;
        // Sort the potential k-shortest paths by cost.
        B.sort();
        // Add the lowest cost path becomes the k-shortest path.
        A[k] = B[0];
        // In fact we should rather use shift since we are removing the first element
        B.pop();

    return A;
```

Gambar 1 *Pseudocode* dari Algoritma Yen

Inputnya adalah *source* (node sumber), *destination*, *Graph*, dan nilai K. Untuk mencari jalur pertama (cost terendah) digunakan algoritma Dijkstra. Kemudian, setelah ditemukan, secara berurutan akan memutuskan link antara node awal (*root*) dan node berikutnya, sehingga mengulangi operasi sebanyak nilai K. Ini dilakukan untuk menemukan jalur selain jalur yang ditemukan terlebih dahulu, hingga jalur K diperoleh. Jika jalur tidak ada lagi, algoritma akan berhenti mencari jalur lain dan mengurutkan jalur yang ditemukan berdasarkan cost terendah.

E. Algoritma Floyd-Warshall

Algoritma Floyd-Warshall adalah algoritma untuk menghitung jalur terpendek [6]. Algoritma tersebut dapat menemukan semua jarak ke setiap *node* (*all pairs shortest path*), yang artinya dapat digunakan untuk menghitung bobot minimum semua jalur yang menghubungkan sepasang titik, Dan selesaikan semua operasi sekaligus untuk semua titik berpasangan. Algoritma Floyd-Warshall memiliki input graf berbobot terarah (V, E), yang merupakan titik (node / vertex V) dan sisi (edge E). Di sisi E boleh memiliki nilai bobotnya negatif, tetapi siklus dengan bobot negatif atau *loop* di jalur tidak diperbolehkan. Dan dibawah ini merupakan Pseudocode dari algoritma Floyd-Warshall.

```
Initialize: n //n = jumlah node pada topologi
Initialize: dist |n|x|n| = inf //array untuk menyimpan jarak minimum
Initialize: nxt |n|x|n| = null //array untuk menyimpan index node

def FloydWarshall()
        for each vertex v do
          dist[v][v] <- 0
          next[v][v] -< v

        for each edge(u,v) do
          dist[u][v] <- w(u,v)
          next[u][v] <- v

        for k from 1 to |n| do
          for i from 1 to |n| do
            for j from 1 to |n| do
                if dist[i][j]>dist[i][k] + dist[k][j] then
                  dist[i][j] <- dist[i][k] + dist[k][j]
                    next[i][j] <- next [i][k]
```

Gambar 2 *Pseudocode* dari Algoritma Floyd-Warshall

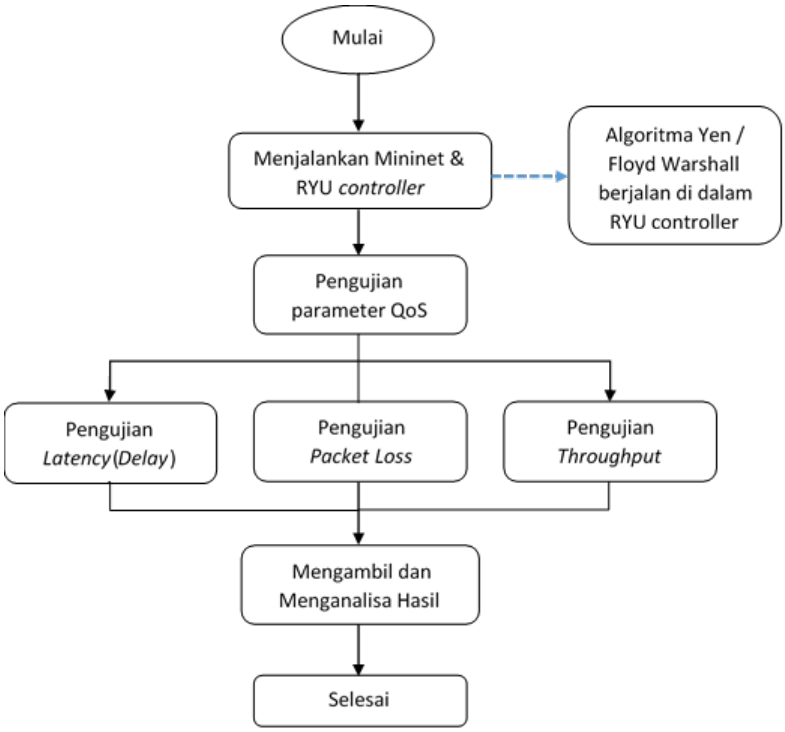
3. Metode Penelitian

Untuk membangun suatu sistem perlu dilakukan beberapa langkah agar situasi pembangunan dapat ditentukan dan dapat diberikan pengawasan jika terjadi penyimpangan. Demikian pula dalam pengembangan jaringan komputer khususnya dalam pembangunan berskala besar hendaknya dilakukan secara bertahap agar jaringan dapat mencapai tujuannya dan berfungsi sebagai media percepatan arus informasi dan pertukaran informasi. Sesuai dengan pembahasan penelitian, maka metode pengembangan sistem yang saya pilih menggunakan SPDLC *(Security Policy Development Life Cycle).* SPDLC adalah metode mendefinisikan strategi untuk memperbarui pengorganisiran dari sistem jaringan. Wahsheh & Foss [4] berpendapat, pengembangan sistem SPDLC yang diambil melakukan penelitian dalam 5 tahap, yaitu tahap *Analysis, Design, Implementation, Enforcement dan Enhancement*.

A. Perancangan Aplikasi

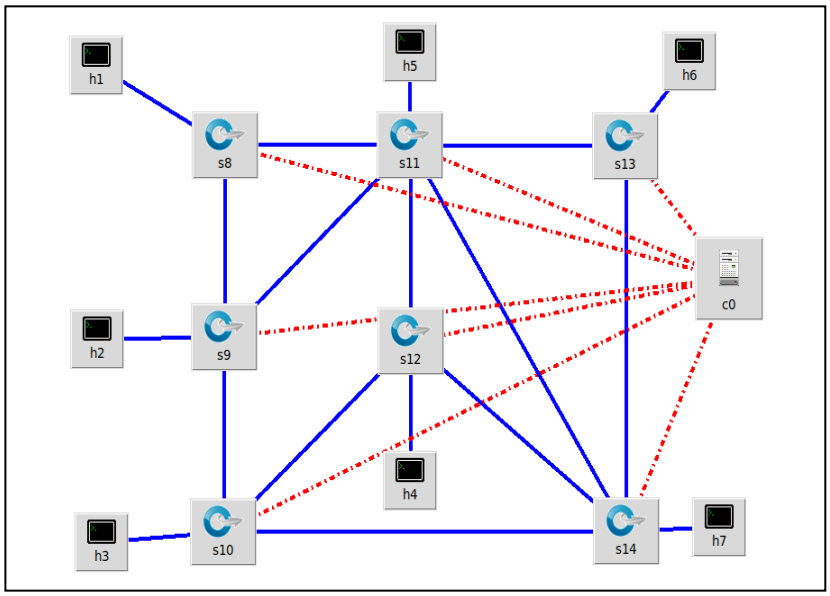
Skenario yang penulis lakukan terdiri dari 2 bagian. Pertama, pengujian dilakukan dengan menggunakan algoritma Yen. Yang kedua, pengujian dilakukan dengan menggunakan algoritma Floyd-Warshall. Pengujian ini dilakukan dengan melakukan pengiriman *protocol* ICMP dengan total waktu sebagai sampelnya yaitu sebanyak 50 kali. Untuk nilai parameter ujinya yaitu *Latency*, *Packet Loss*, dan *Throughput*. Pada tahap pengujian, pengukuran Latency , Packet Loss, dan Throughput dilakukan dengan *host* tujuan sebagai *server* dan *host* pengirim sebagai *client*. Lalu penulis menggunakan Iperf untuk mengukur performansi dari sisi TCP. Dan Wireshark digunakan untuk meng-*capture traffic* pada jaringan dalam membantu menghitung nilai parameter QoS tertentu. Ini dilakukan untuk bisa melihat

kualitas lalu lintas data yang dikirim ke penerima dan untuk mengetahui pengaruh kepadatan lalu lintas terhadap pemilihan jalur.



Gambar 3 Siklus Perancangan Pengujian Sistem

B. Topologi



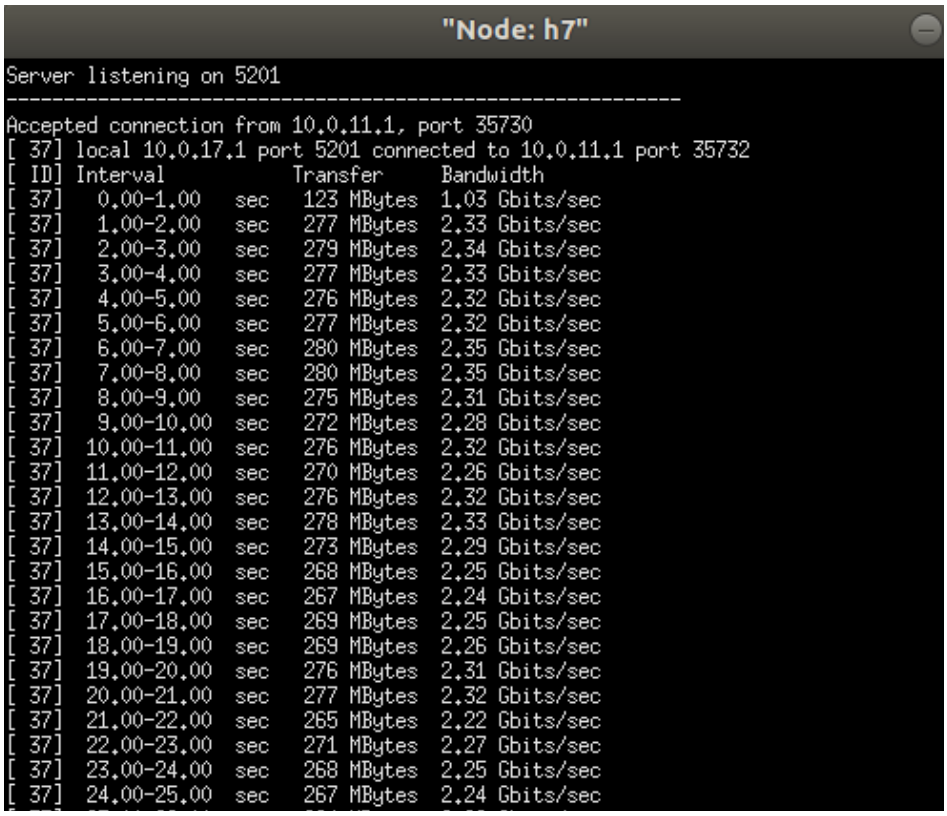
Gambar 4 Simulasi Topologi Jaringan

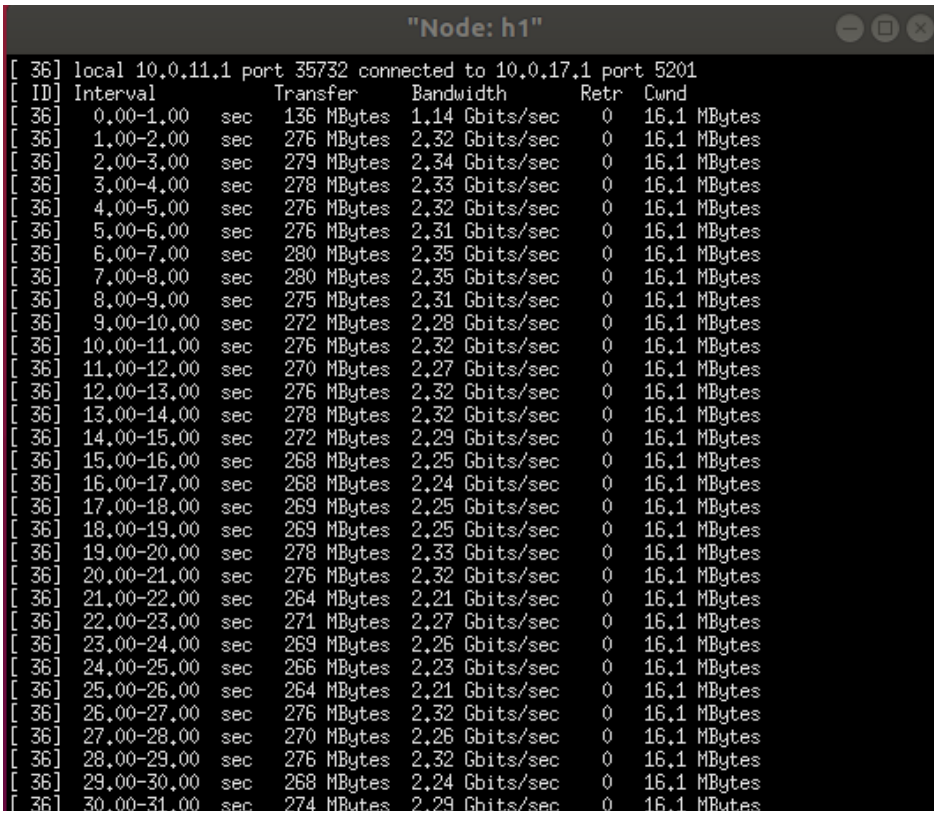
Pada Gambar 4, Simulasi Topologi Jaringan di atas merupakan topologi yang ada dalam penelitian [5] dengan judul: *Fail Path Analysis on OpenFlow Network Using Floyd-Warshall Algorithm*, yang diambil dari Gambar 3 *Mesh Topology*. Simulasi topologi di atas merupakan topologi yang akan diterapkan dengan menggunakan Algoritma Yen dan Algoritma Floyd-Warshall. *Link* (Kabel) dari *Switch* S8 ke S9 memiliki *delay* 3 detik dan *bandwidth* 100MBps. *Link* (Kabel) dari *Switch* S9 ke S10 memiliki *delay* 9 detik dan *bandwidth* 100MBps. *Link* (Kabel) dari *Switch* S8 ke S11 memiliki *delay* 10 detik dan *bandwidth* 100MBps. *Link* (Kabel) dari *Switch* S9 ke S11 memiliki *delay* 7 detik dan *bandwidth* 100MBps.

Link (Kabel) dari *Switch* S10 ke S12 memiliki *delay* 3 detik dan *bandwidth* 100MBps. *Link* (Kabel) dari *Switch* S10 ke S14 memiliki *delay* 4 detik dan *bandwidth* 100MBps. *Link* (Kabel) dari *Switch* S11 ke S13 memiliki *delay* 2 detik dan *bandwidth* 100MBps. *Link* (Kabel) dari *Switch* S11 ke S14 memiliki *delay* 2 detik dan *bandwidth* 100MBps. *Link* (Kabel) dari *Switch* S11 ke S12 memiliki *delay* 5 detik dan *bandwidth* 100MBps. *Link* (Kabel) dari *Switch* S12 ke S14 memiliki *delay* 2 detik dan *bandwidth* 100MBps. Dan *link* (Kabel) dari *Switch* S13 ke S14 memiliki *delay* 8 detik dan *bandwidth* 100MBps.

4. Hasil & Analisis

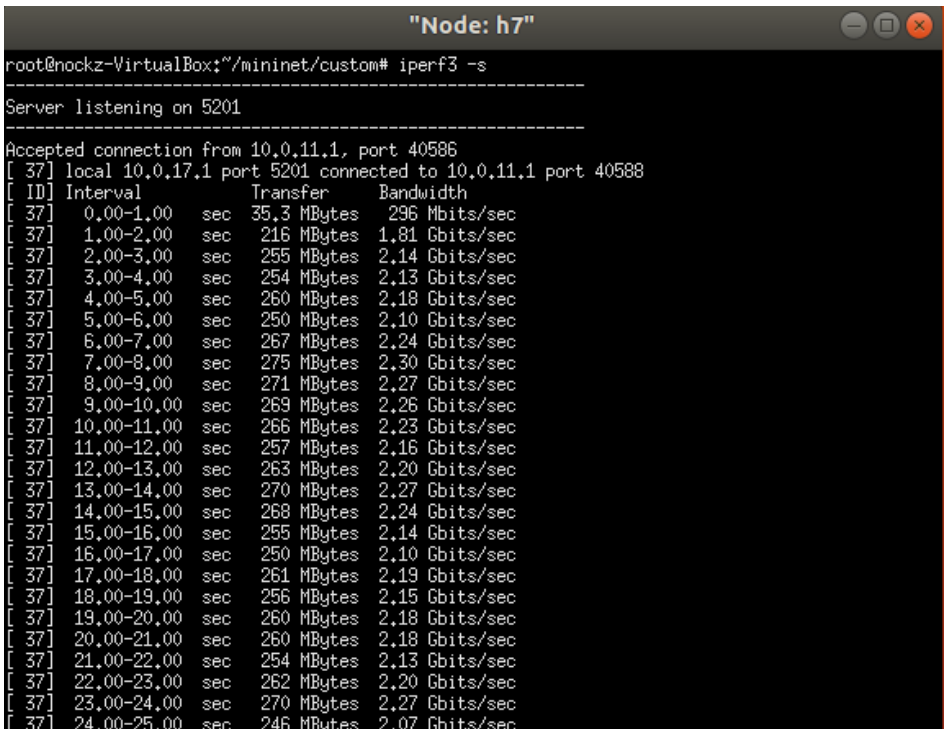
Pada bagian ini berisi tentang laporan seluruh hasil yang diperoleh dari penelitian. Pada bagian ini berisi analisa, hasil serta pembahasan dari topik penelitian, yang bisa di buat terlebih dahulu metodologi penelitian. Pengujian dilakukan terhadap 3 parameter *Quality of Service*, yaitu *Throughput, Delay* dan *Packet Loss* dengan memanfaatkan *command* iperf dan aplikasi pemantau jaringan yaitu *Wireshark*, dimana h7 yang bertindak sebagai *server* menggunakan *command*: iperf3 –s dan untuk h1 yang bertindak sebagai *client* menggunakan *command*: iperf3 –c 10.0.17.1 –t 50. Dan akan ditambahkan dengan *command* –P n pada *client*, dimana n merupakan angka untuk jumlah *client parallel* yang dibuat. Di bawah ini merupakan tampilan pengujian parameter pada saat menggunakan *command* iperf dan hasil pengujian ke-3 parameter tersebut.

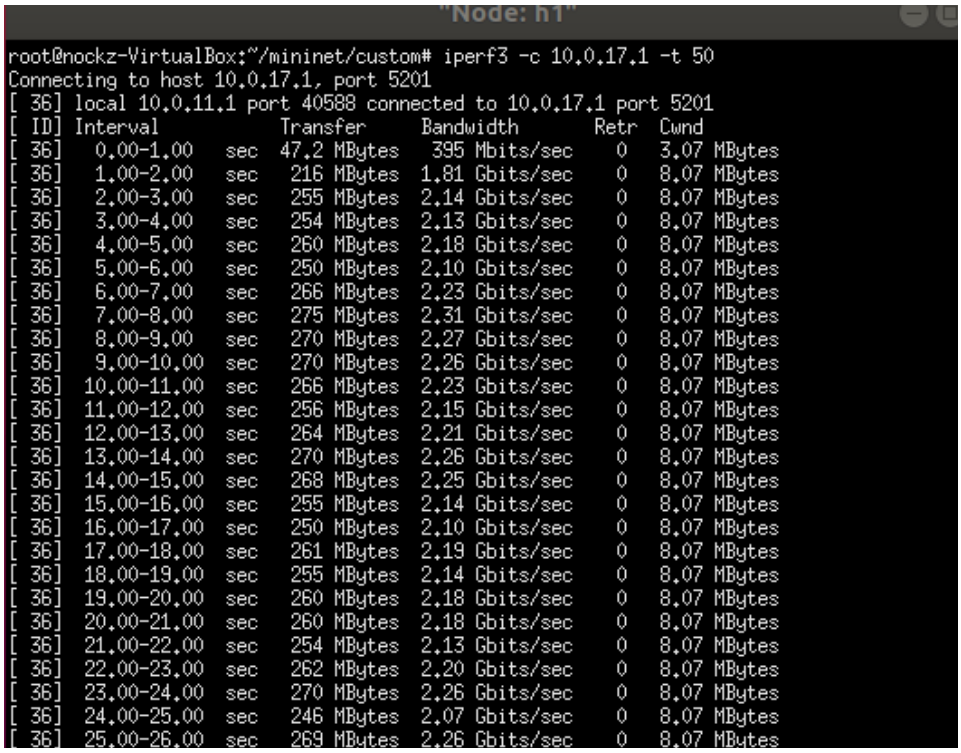




Gambar 5 Pengujian *throughput* dengan menggunakan *iperf* pada Algoritma Yen

Berikut ini merupakan gambar pengujian throughput dengan menggunakan algoritma Floyd-Warshall.

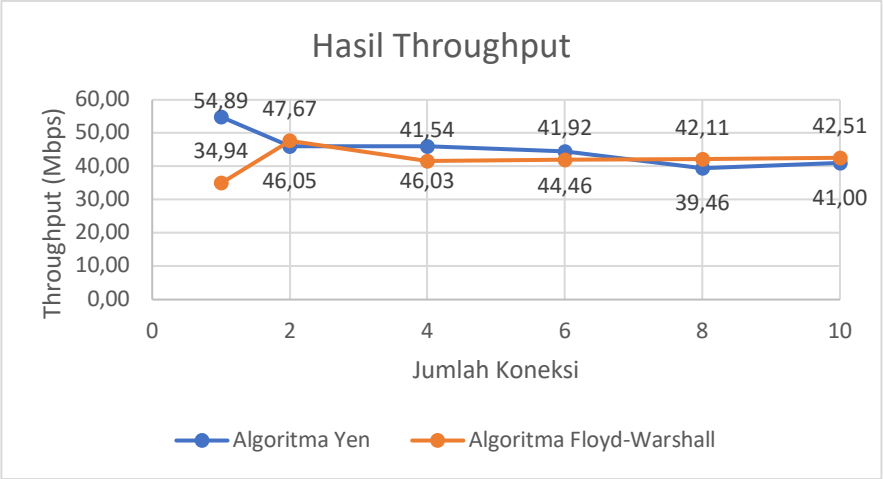




Gambar 6 Pengujian *throughput* dengan menggunakan *iperf* pada Algoritma Floyd-Warshall

1) Hasil Uji *Throughput*

Pengujian throughput dilakukan untuk melihat berapa banyak data yang dapat dikirim oleh algoritma dalam waktu tertentu. Tes ini dilakukan menggunakan *command* iperf. Pengujian dilakukan dengan membuat koneksi TCP antara host 1 yang bertindak sebagai client dan host 7 yang bertindak sebagai server. Lalu komunikasi yang terjadi antara host 1 dan host 7 dicatat dengan menjalankan aplikasi Wireshark dan melihat hasil analisa pada koneksi protocol TCP.

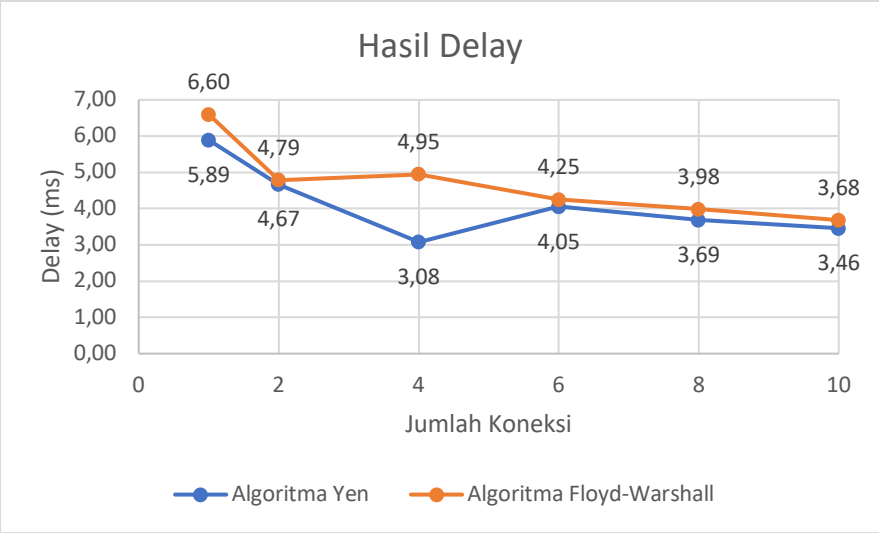


Gambar 7 Hasil Pengujian *Throughput*

Hasil dari nilai *throughput client* yang dijalankan pada Algoritma Yen dan Algoritma Floyd-Warshall ditampilkan pada Gambar 7. Nilai yang ditampilkan pada grafik merupakan nilai throughput yang mampu diberikan oleh masing-masing algoritma, yang dimana seiring bertambahnya jumlah koneksi client, *throughput* mengalami penurunan. Dari hasil yang terlihat di gambar, Algoritma Floyd memiliki nilai throughput lebih rendah dari pada Algoritma Yen.

2) Hasil Uji *Delay*

Dalam menguji Delay, hasil uji tersebut diambil dari nilai rata – rata delay yang terjadi pada saat dijalankan command iperf, dan melakukan test koneksi antara host 1 yang bertindak sebagai client dan host 7 yang bertindak sebagai server. Hasil uji dapat dilihat melalui catatan analisa Wireshark pada saat command iperf dijalankan dengan menghitung total waktu dibagi dengan total paket yang diterima.

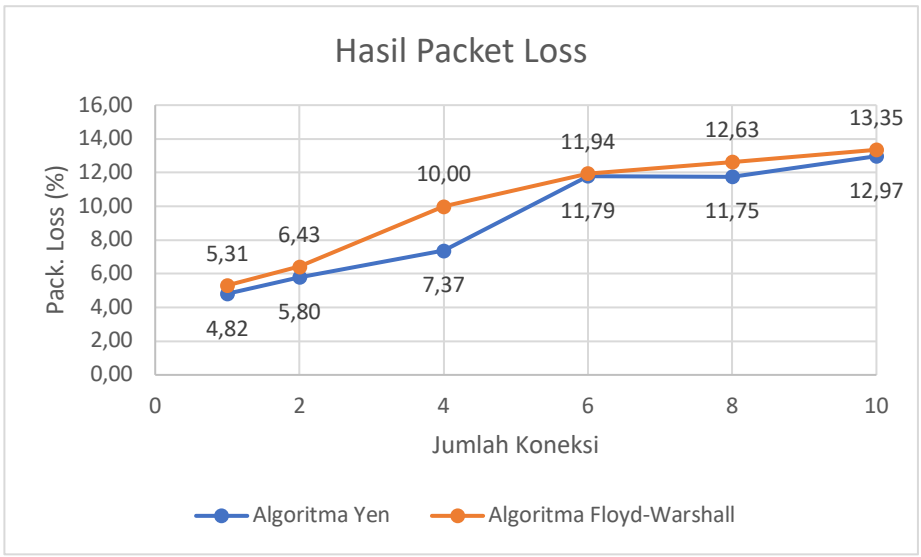


Gambar 8 Hasil Uji *Delay*

Dari grafik yang ditunjukan pada Gambar 8, dapat dilihat bahwa Delay yang dimiliki pada saat topologi menggunakan Algoritma Yen lebih kecil dibandingkan Algoritma Floyd-Warshall.

3) Hasil Uji *Packet Loss*

Dalam menguji Packet Loss, dilihat jumlah paket yang tidak dapat dikirim dari client ke server. Tes kehilangan paket dilakukan menggunakan command iperf. Pengujian diselesaikan dengan membuat sambungan secara bersamaan antara host 1 yang bertindak sebagai client dan host 2 yang bertindak sebagai server, lalu melihat hasil analis yang dicatat oleh Wireshark dengan memfilter protocol tcp.analysis.lost_segment.



Gambar 9 Hasil Uji *Packet Loss*

Nilai packet loss pada Gambar 9 yang dimiliki oleh Algoritma Yen lebih besar nilai persentasenya dibandingkan dengan Algoritma Floyd-Warshall. Perhitungan packet loss dilakukan untuk melihat seberapa besar nilai persentase dari pengiriman paket yang gagal untuk masing –

masing algoritma pada topologi jaringan yang diuji. Dan ternyata seiring bertambahnya jumlah koneksi client, maka packet loss akan semakin sering terjadi. Rata-rata persentase packet loss yang dimiliki Algoritma Yen adalah 8.60% sedangkan untuk Algoritma Floyd-Warshall adalah 6.38%.

Dari hasil uji ketiga parameter di atas, dapat disimpulkan ke dalam bentuk tabel menjadi seperti gambar di bawah ini.

Jumlah Koneksi Paralel		1	2	4	6	8	10
Throughput	*Algoritma Yen*	54.89	46.05	46.03	44.46	39.46	41.00
	Algoritma Floyd-Warshall	34.94	47.67	41.54	41.92	42.11	42.51
Delay	*Algoritma Yen*	5.89	4.67	3.08	4.05	3.69	3.46
	Algoritma Floyd-Warshall	6.60	4.79	4.95	4.25	3.98	3.68
Pack. Loss	*Algoritma Yen*	4.82	5.80	7.37	11.79	11.75	12.97
	Algoritma Floyd-Warshall	5.31	6.43	10.00	11.94	12.63	13.35

Gambar 10 Tabel Hasil Uji Parameter

5. Kesimpulan

Berdasarkan daripada hasil pengujian dan analisis dari penelitian yang dilakukan, dapat ditarik kesimpulan sebagai berikut. Berdasarkan daripada hasil pengujian dan analisis dari penelitian yang dilakukan, dapat ditarik kesimpulan sebagai berikut. (1) Penerapan Algoritma Yen dan Algoritma Floyd-Warshall pada arsitektur jaringan SDN (*Software Defined Network*) berjalan dengan sangat baik. Dari hasil parameter yang diberikan kedua algoritma tersebut, Algoritma Yen memiliki performa lebih baik dalam menangani *Quality of Service* dengan parameter *delay* dan *packet loss* sedangkan Algoritma Floyd-Warshall memberikan performa rata – rata *throughput* yang lebih baik. (2) Sistem dapat mencari beberapa jalur sesuai kebutuhan, dan memilih jalur terbaik sebagai media transmisi data. Mengukur kinerja jaringan berdasarkan hasil tes *throughput*, ditemukan bahwa seiring bertambahnya jumlah koneksi *client*, hasil nilai *throughput* dari Algoritma Floyd-Warshall memiliki nilai hasil rata – rata yang lebih rendah yaitu 41.78Mbps dibandingkan dengan Algoritma Yen dengan nilai 45.32Mbps. Dalam pengujian parameter *Delay*, seiring bertambahnya jumlah koneksi *client*, Algoritma Yen mampu memberikan nilai *delay* yang lebih kecil dengan nilai rata-rata 4.14ms, daripada yang diberikan oleh Algoritma Floyd-Warshall dengan *delay* 4.71ms. Untuk hasil pengujian *packet loss*, Algoritma Yen cukup mampu meminimalisir terjadinya Packet Loss dalam jaringan dengan rata-rata persentase 9.08% dibandingkan dengan Algoritma Floyd-Warshall yang rata-rata nilai *packet loss*-nya sebesar 9.94%. (3)Performansi yang diberikan oleh Algoritma Yen pada penelitian ini memberikan hasil yang cukup baik, namun dalam pengujian Throughput, Algoritma Floyd-Warshall memiliki performasi rata – rata yang lebih baik dibandingkan Algoritma Floyd-Warshall.

6. Daftar Pustaka

[1] A. R. Sudiyatmoko, S. N. Hertiana and R. M. Negara, "Analisis Performansi Perutingan Link State Menggunakan Algoritma Djikstra Pada Platform Software Defined Network (SDN)," *INFOTEL,* vol. 8, no. 1, pp. 40-46, 2016.

[2] N. Hunaifi and R. F. Akbar, "ANALISIS KINERJA JARINGAN BERBASISKAN SOFTWARE DEFINITION NETWORK DENGAN PROTOKOL OPENFLOW DI RRI BANDUNG," *Infotronik,* vol. 4, no. 1, 2019.

[3] J. I. Sihotang, "STUDI KELAYAKAN KUALITAS LAYANAN PENYAMPAIAN KONTEN MULTIMEDIA MELALUI JARINGAN BERKAPASITAS TERBATAS," *TeIKa,* vol. 7, no. 1, 2017.

[4] Ryu SDN Framework Community, "Build SDN Agilely," 2017. [Online]. Available: https://ryu-sdn.org/.

[5] J. Y. Yen, "AN ALGORITHM FOR FINDING SHORTEST ROUTES FROM ALL SOURCE NODES TO A GIVEN DESTINATION IN GENERAL NETWORKS," *Quarterly of Applied Mathematics,* vol. 27, no. 4, pp. 526-530, 1970.

[6] Ramadiani, D. Bukhori, Azainil and N. Dengen, "Floyd-warshall algorithm to determine the shortest path based on android," in *IOP Conference Series: Earth and Environmental Science*, Samarinda, 2017.

[7] L. A. Wahsheh and J. Alves-Foss, "Security Policy Development: Towards a Life-Cycle and Logic-Based Verification Model," *American Journal of Applied Sciences,* vol. 5, no. 9, pp. 1117-1126, 2008.

[8] E. F. Rohman, Ritzkal and Y. Afrianto, "Fail Path Analysis on Openflow Network Using Floyd-Warshall Algorithm," *Manajemen, Teknologi Informatika dan Komunikasi (Mantik),* vol. 4, no. 3, pp. 1546-1550, 2020.