Analisis Perbandingan Kinerja Algoritme *Dijkstra, Bellman-Ford,* dan *Floyd-Warshall* Untuk Penentuan Jalur Terpendek Pada Arsitektur Jaringan *Software Defined Network*

Aprillia Arum Pratiwi[1], Widhi Yahya[2], Mahendra Data[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]aprilliaarumpratiwi@gmail.com, [2]widhi.yahya@ub.ac.id, [3]mahendra.data@ub.ac.id

Abstrak

Software defined network (SDN) merupakan konsep jaringan yang terpusat dan fleksibel dibandingkan dengan jaringan tradisional yang ada sekarang. SDN mulai dikembangkan beberapa tahun terakhir ini dan sudah banyak diimplementasikan salah satunya adalah routing jaringan. Routing merupakan proses pencarian jalur komunikasi yang digunakan untuk mengirimkan paket dari pengirim ke penerima. Algoritme routing yang diimplementasikan bertugas untuk menentukan jalur terpendek yaitu algoritme *Dijkstra*, *Bellman-Ford*, dan *Floyd-Warshall*. Ketiga algoritme tersebut akan diimplementasikan menggunakan emulator *Mininet* dan *controller Ryu*. Untuk penentuan jalur dibutuhkan penentuan bobot *link* atau *cost. Cost* pada penelitian ini berdasarkan dengan perhitungan bagi antara *reference bandwidth* sebesar 1000 Mbps dan *link bandwidth* yang menggunakan tiga jenis besaran kapasitas yaitu 10 Mbps, 100 Mbps, dan 1000 Mbps. Pengujian dilakukan dengan parameter yaitu validasi, *convergence time, throughput, recovery time* dan *packet loss*. Berdasarkan hasil validasi, sistem berjalan sesuai dengan perhitungan perhitungan manual masing-masing algoritme. Pada pengujian *convergence time*, *Dijkstra* lebih unggul dengan rata-rata 0,0087 detik dibandingkan *Bellman-ford* 0,0094 detik dan *floyd-warshall* 0,02025 detik. Pada pengujian *throughput* ketiga algoritme tidak terlalu memiliki perbedaan yang jauh. Berdasarkan pengujian *recovery time* algoritme *floyd-warshall* memiliki *recovery time* lebih cepat dari algoritme lain. Berdasarkan pengujian *packet loss Dijkstra* masih unggul dalam menangani *packet loss* saat pengiriman.

Kata kunci: *Software Defined Network, Ryu, Mininet, Algoritme Dikstra, Algorime Bellman-Ford, Algoritme Floyd-Warshall*

Abstract

Software defined network (SDN) is a centralized and flexible network concept compared to traditional networks that exist today. SDN has been developed in the last few years and has been widely implemented, one of which is routing networks. Routing is the process of finding the communication path used to send packets from the sender to the recipient. The implemented routing algorithm is tasked to determine the shortest paths, namely Dijkstra, Bellman-Ford, and Floyd-Warshall algorithms. The three algorithms will be implemented using the Mininet and Ryu controller emulators. For determining the path required the determination of the link or cost weight. Cost in this study is based on the calculation of the reference bandwidth of 1000 Mbps and the bandwidth link that uses three types of capacity quantities, namely 10 Mbps, 100 Mbps, and 1000 Mbps. Testing is done with parameters, namely validation, convergence time, throughput, recovery time and packet loss. Based on the results of validation, the system runs according to the calculation of the manual calculation of each algorithm. In convergence time testing, Dijkstra was superior with an average of 0.0087 seconds compared to Bellman-ford 0.0094 seconds and Floyd-Warshall 0.02025 seconds. In testing the three throughput algorithms do not have far distinction. Based on recovery time testing, the Floyd-Warshall algorithm has faster recovery time than other algorithms. Based on the testing of packet loss Dijkstra is still superior in handling packet loss when sending.

Keywords: *Software Defined Network, Ryu, Mininet, Dikstra Algoritthm, Bellman-Ford Algoritthm, Floyd-Warshall Algoritthm*

1. PENDAHULUAN

Dewasa ini teknologi komunikasi berkembang semakin pesat. Perkembangan pada teknologi komunikasi dirancang untuk memenuhi kebutuhan akan komunikasi yang cepat dan efisien. Salah satunya jaringan komunikasi kini mulai diganti dengan jaringan yang bersifat *programable*. Jaringan *software defined network* merupakan salah satu evolusi teknologi jaringan yang sesuai dengan tuntutan yang sedang berkembang saat ini. *Software defined network* atau SDN memiliki konsep yang berbeda dengan jaringan tradisional. Pada jaringan trasional *control plane* dan *data plane* terdapat pada satu perangkat. Sedangkan SDN memisahkan *control plane* sebagai *controller* dan *data plane* yang dimaksudkan untuk memudahkan proses pengaturan, pengelolaan dan *monitoring* jaringan (S.Sharma,et al.,2014). Tujuan utama dari jaringan SDN adalah untuk mempermudah penerapan aplikasi jaringan lain pada *controller* seperti *load balancing, network virutalization, Intrusion detection, routing* dan lainnya.

Controller merupakan bagian dari SDN yang bertanggung jawab untuk mendefinisikan jaringan, mengatur masalah *availability*, laju traffic data, *routing* dan *forwarding* dan lain-lain. Suatu protokol *routing* memiliki tabel yang berisi informasi jalur yang akan dilalui oleh suatu paket data, dari informasi tersebut ditentukan suatu jalur yang akan digunakan untuk melewatkan paket berdasarkan alamat tujuannya. Dalam mengoptimalkan kinerja protokol *routing* diperlukan Algoritme *routing protocol* untuk menentukan rute terpendek pada berbagai macam topologi. Untuk membangun suatu *routing protocol* yang baik, maka diperlukan Algoritme *routing protocol* yang dapat menentukan jalur terpendek pada berbagai topologi tanpa melakukan konfigurasi ulang (Ulfa Kurniawati, 2016). Algoritme *Djikstra, Bellman-Ford* dan *Floyd-Warshal* memiliki cara pencarian yang berbeda.

Dari penelitian yang sudah ada, penulis akan melakukan analisis perbandingan dari Algoritme *Dijkstra* dan *Bellman-Ford* pada *Software Defined Network* (SDN). Untuk penelitian lebih lanjut, penulis berminat untuk menambahkan Algoritme *Floyd-Warshall* sebagai pilihan Algoritme *routing* yang akan di bandingkan performanya dengan Algoritme *Dijkstra* dan *Bellman-Ford*. Penerapan dari ketiga Algoritme tersebut dengan melihat parameter uji meliputi *convergence time, throughput, recovery time* dan *resource usage*. Ketiga algoritme akan dijalankan menggunakan *controller* RYU dan menggunakan emulator *Mininet*. Ryu merupakan perangkat lunak berbasis komponen yang didefinisikan pada topologi jaringan. Ryu menyediakan komponen perangkat lunak dengan API terdefinisi dengan baik sehingga memudahkan pengembang untuk membuat aplikasi manejemen dan kontrol jaringan baru. Ryu mendukung berbagai protokol untuk mengelola perangkat jaringan seperti *OpenFlow, Netconf, OF-config* dan lain-lain. Pada penelitian ini peneliti menggunakan protokol *Openflow* untukkomunikasi yang terjadi dilakukan oleh *controller* dengan perangkat-perangkat jaringan lainnya. Sementara Mininet adalah suatu software emulatir yang memungkinkan untuk melakukan *prototyping* pada jaringan yang luas dengan hanya menggunakan satu mesin. Mininet merupakan salah satu *software* yang paling sering diigunakan pada SDN. Mininet menggunakan pendekatan *virtualisasi* untuk membuat suatu sistem. Seperti host, *switch* dan *controller virtual*. Mininet biasanya digunakan sebagai simulasi, *verifikasi*, *testing tool* dan *resource*. Salah satu keunggulanyang dimiliki oleh mininet asalah mampu membuat topologi sesuai yang diinginkan, mampu membuat topologi yang cukup kompleks, lebih besar dan topologi internet. Sehingga mendapatkan hasil pengujian yang dapat dibandingkan untuk menentukan Algoritme manakah yang memiliki hasil paling optimal jika diterapkan pada jaringan *software defined network* (SDN).

2. METODOLOGI PENELITIAN

Bagian ini menjelaskan langkah-langkah yang akan dilakukan dalam penyusunan tugas akhir ini. Beberapa tahapan sebagai metodologi penelitian yaitu perumusan masalah, studi literatur, perancangan, implementasi dan pengujian, analisis dan hasil, dan kesimpulan. Berikut diagram alir tahapan metodologi penelitian yang digunakan:

Gambar 2.1 Diagram alir metode penelitian

Berdasarkan gambar 2.1 dapat di uraikan langkah-langkah metodologi penelitian yang akan dilakukan, yaitu:

1. Perumusan masalah dalam penelitian ini dirumuskan berdasarkan masalah yang sedang terjadi. Dari perumusan masalah yang disusun maka, peneliti dapat mengambil langkah untuk memberikan slusi dari permasalahan yang teah dirumuskan.
2. Studi literatur penelitian sebelumnya yang terkait, *Software Defined Network*, *OpenFlow*, *controller* (RYU), *mininet*, *Routing*, Algoritme *Routing*, Parameter kinerja jaringan, dan topologi
3. Perancangan atau desain sistem, topologi, dan lingkungan untuk menerapkan algoritme routing pada SDN
4. Implementasi sistem pada *OpenFlow* SDN
5. Pengujian sistem routing pada jaringan SDN
6. Analisis hasil pengujian berdasarkan parameter uji yang telah ditentukan
7. Penarikan kesimpulan bersadarkan hasil analisis pengujian yang dilakukan terhadap sistem.

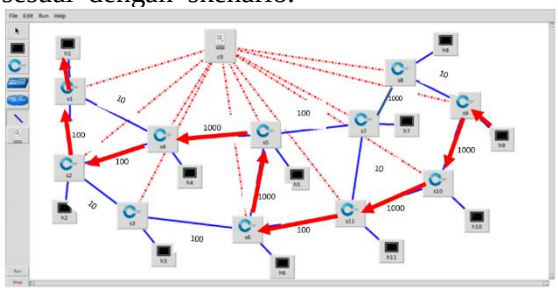
3. PERANCANGAN DAN ANALISIS

Setelah simulasi dilakukan berdasarkan skenario yang telah dibuat. Maka akan menghasilkan suatu *report* yang berisi data-data mengenai hasil dari simulasi yang telah dilakukan. Data tersebut akan diambil dan disajikan ke dalam grafik yang nantinya akan dilakukan analisis terhadap kinerja protokol *routing*.

Hasil simulasi yang digunakan untuk menganalisis kinerja dari algoritme *Dijkstra*, *Bellman-Ford* dan *Floyd-warshall* adalah dengan melakukan beberapa pengujian. Terdapat lima pengujian yaitu validasi, *convergence time*, *throughput*, *recovery time*, dan *packet loss*.

1) Pengujian validasi

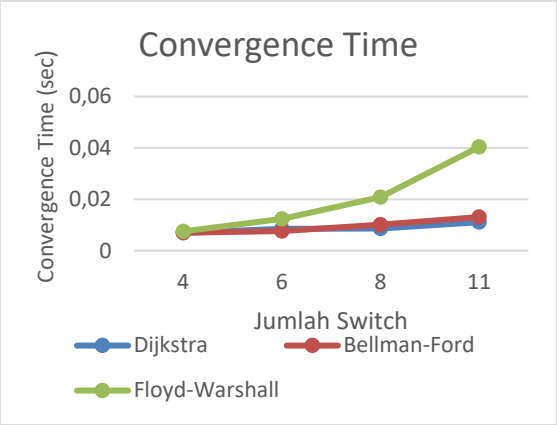
Pengujian validasi dilakukan dengan perhitungan secara matematis sesuai dengan mekanisme perhitungan masing-masing algoritme. Dalam perhitungan manual akan ditentukan node sumber host 9 dan node tujuan adalah host 1. Kemudian sistem juga akan melakukan pengiriman dari host 9 ke host 1. Maka akan diperoleh hasil dari keduanya, jika memiliki hasil yang sama maka sistem yang dibuat dapat dinyatakan sesuai dengan skenario.



Gambar 3.1 Alur paket ICMP dikirimkan dari host 9 ke host 1

2) Pengujian *convergence time*

Pengujian *convergence time* dilakukan dengan tujuan mengetahui berapa waktu yang diperlukan oleh *controller* untuk membentuk tabel routing yang akan dijadikan sebagai jalur untuk mengirim paket dan menghasilkan *cost* yang kecil. Pengujian dilakukan dengan melakukan pengiriman paket ICMP (*internet control message protocol*). Dengan *command ping* dari host sumber ke host tujuan. Kemudian *timer* pada program *controller* akan berjalan. Setelah mendapatkan *reply* dari host tujuan, *controller* akan menampilkan jalur terpendek dan *convergence time*.



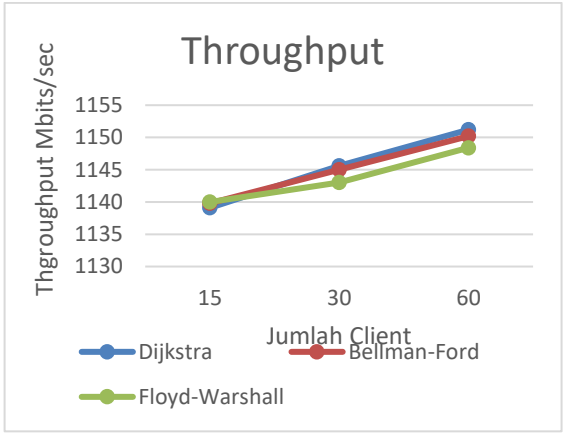
Gambar 3.2 Grafik pengujian *convergence time*

Pada gambar merupakan grafik perbandingan pengujian *convergence time* yang dilakukan sebanyak 10 kali pengujian. Pengujian dilakukan dengan jumlah *switch* yang berbeda yaitu sedikit, sedang dan banyak atau dalam jumlah angka yaitu 4, 6, 8, 11 *switch*. Dari hasil yang didapatkan untuk setiap variasi jumlah *switch* pada topologi akan mengalami perbedaan *convergence time*. Perbedaan hasil ini dapat dilihat dari grafik bahwa *convergence time* bertambah seiring dengan bertambahnya jumlah *switch*. Dari hasil yang diperoleh dapat diketahui bahwa algoritme *Dijkstra* lebih unggul atau memiliki *convergence time* lebih cepat dari algoritme *Belman-Ford* dan *Floyd-Warshall*.

3) Pengujian *Throughput*

Pengujian *throughput* dilakukan untuk mengetahui kecepatan pengiriman paket pada suatu jaringan. Pengujian ini dilakukan sebanyak 5 kali percobaan. Pengiriman paket TCP dilakukan dengan menggunakan *tools iperf* dari host 9 sebagai client ke host 1 sebagai *server*. Pengujian dilakukan dengan variasi jumlah koneksi client yaitu 15, 30 dan 60. Hal ini bertujuan untuk mengetahui pengaruh dari perbedaan jumlah koneksi *client* terhadap *throughput*.

Dapat dilihat dari grafik berikut nilai throughput mengalami kenaikan seiring dengan bertambahnya jumlah koneksi *client*.

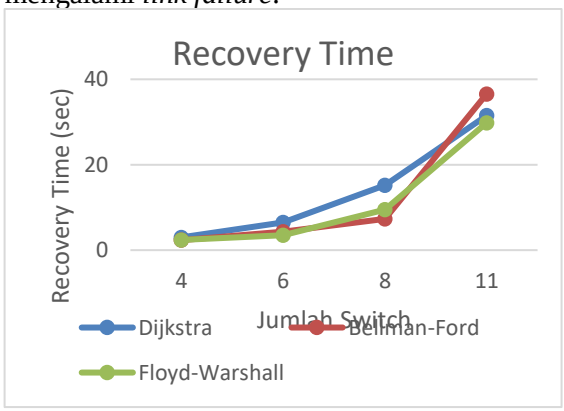


Gambar 3.3 Grafik pengujian *Throughput*

Hal ini dikarenakan dengan bertambahnya jumlah koneksi maka akan bertambah pula jumlah pengiriman yang dilakukan untuk setiap koneksi. Dari hasil di atas dapat dilihat algoritme *Dijkstra* memiliki throughput lebih baik dibandingkan algoritme *Bellman-Ford* dan *Floyd-Warshall*.

4) Pengujian *Recovery Time*

Pengujian *recovery time* bertujuan untuk menghitung berapa lama waktu yang dibutuhkan oleh suatu sistem jaringan untuk menemukan jalur baru setelah mengalami *failure*. Pengujian dilakukan dengan mengirimkan paket ICMP dengan *command ping* seperti pengujian *convergence time* sebelumya, setelah mendapat jalur lalu dilakukan pemutusan sebuah link pada topologi. Setelah dilakukan pemutusan link controller akan membuat jalur baru yang tidak mengalami *link failure*.



Gambar 3.4 Grafik pengujian *Recovery Time*

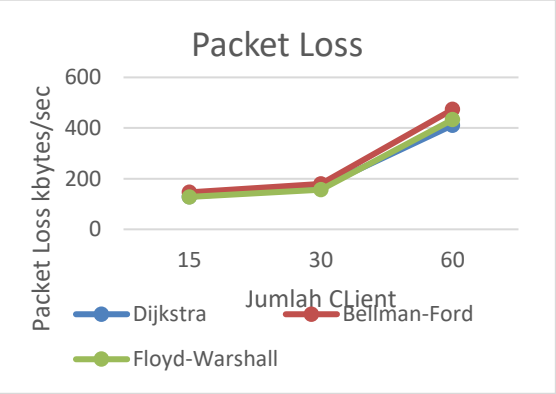
Gambar di atas merupakan grafik pengujian recovery time yang telah dilakukan sebanyak 10 kali percobaan dengan variasi jumlah *switch* yang berbeda yaitu sedikit, sedang, dan banyak atau dalam jumlah angka adalah 4, 6,

8, 11 *switch.*

Dari hasil pengujian yang telah diketahui semakin banyak *switch* pada suatu topologi maka akan membaut *controller* membutuhkan waktu lebih lama untuk membentuk jalur baru setelah dilakukannya *link failure*. Berbeda dengan hasil pengujian *convergence time* yang menunjukkan algoritme *Dijkstra* lebih unggul dari algoritme lainnya, namun pada penujian *recovery time* algoritme *Floyd-Warshall* lebih unggul dari kedua algoritme *Dijkstra*. Dikarenakan setelah pemutusan *link*, algoritme *Floyd-Warshall* masih memiliki informasi jalur-jalur lain. Informasi tersebut diperoleh dari pencarian jalur sebelumya.

5) Pengujian *Packet Loss*

Pengujian *packet loss* dilakukan dengan tunuan untuk mengetahui kegagalan dalam pengiriman suatu paket dari *client* ke *server*. Pengujian dilakukan sebanyak 5 kali perobaan dengan variasi jumlah client yaitu 15, 30, dan 60 untuk mengetahui pengaruh perbedaan jumlah koneksi *client* terhadap *packet loss*. Pengujian dilakukan dengan *tools iperf* untuk mengirim paket UDP dengan *rate transfer* sebesar 1 Mbits/sec dari host 9 (*client*) ke host 1 (*server*).



Gambar 3.5 Grafik pengujian *packet loss*

Berdasarkan hasil pengujian *packet loss* menunjukkan bahwa penambahan jumlah client mempengaruhi nilai rata-rata *packet loss*. Hal ini dikarenakan semakin banyak jumlah *client* yang mengirim paket maka sumberdaya akan dibagikan pada jaringan tersebut semakin sedikit sehingga paket yang dikirimkan akan banyak yang hilang. Dari grafik pengujian nilai *packet loss* pada ketiga algoritme *Bellman-Ford* mengalami kenaikan nilai paket loss paling tinggi dibanding algoritme *Dijkstra* dan *Floyd-Warshall*.

4. KESIMPULAN

Berdasarkan hasil pengujian dan analisis yang telah dilakukan, dapat ditarik kesimpulan dari penelitian ini, yaitu sebagai berikut:

Penulis telah berhasil melakukan penelitian tentang analisis kinerja algoritme *Dijkstra, Bellman-Ford,* dan *Floyd-Warshall* pada arsitektur jaringan *Software Defined Network* untuk penentuan jalur terpendek dengan hasil cost paling kecil. Berdasarkan hasil dari pengujian yang telah dilakukan, maka analisa dari

a. Berdasarkan hasil pengujian validasi setiap algoritme menentukan jalur yang sama dari perhitungan manual maupun pada sistem, jalur yang dihasilkan yaitu 9-10-11-6-5-4-2-1.
b. Berdasarkan hasil pengujian *convergence time*, *convergence time* meningkat seiring bertambahnya jumlah *switch.* Dengan hasil algoritme Dijkstra lebih unggul karena memiliki waktu lebih cepat dibandingkan algoritme *Bellman-Ford* dan *Floyd-Warshall. Floyd-warshall* memiliki *convergence time* paling buruk yang membutuhkan waktu lama untuk membentuk jalur pengiriman paket.
c. Berdasarkan hasil pengujian *throughput* untuk mengukur kualitas jaringan pada algoritme *Dijkstra* memiliki jumlah throughput lebih baik dari algoritme *Bellman-Ford* dan *Floyd-warshall.*
d. Berdasarkan hasil pengujian *recovery time* algoritme *Floyd-Warshall* memiliki waktu paling baik atau paling cepat dibandingkan dengan algoritme *Dijkstra* dan *Bellman-Ford*.
e. Berdasarkan hasil pengujian *packet loss* algoritme *Dijkstra* memiliki jumlah *packet loss* paling rendah dibandingkan algoritme *Bellman-Ford* dan *Floyd-Warshall*

5. SARAN

Beberapa saran berdasarkan hasil penelitian ini untuk pengembangan penelitian selanjutnya, yaitu sebagai berikut:

1. Perlu dilakukan penambahan jumlah *switch* untuk mengetahui kinerja algoritme pada topologi yang lebih kompleks
2. Perlu dilakukan implementasi pada *real network*.

6. DAFTAR PUSTAKA

Nadeau & Gray, "SDN- *Software Defined Network*", 2013

Mulyana Eueung, "Buku Komunitas SDN-RG",*Published with Gitbook*,2015

OpenNetworking Foundation, 2012. *Software Defined Networking: The New Norm for Networks.*

Riza Abu.; dkk., 2016 "Analisis Performansi Perutingan Link State Menggunakan Algoritma *Djikstra* pada *Platform Software Defined Network* (SDN)", Fakultas Teknik Elektro, Universitas Telkom Bandung

Munir, Rinaldi. 2005. "Strategi Algoritmik".Bandung. Institut Teknologi Bandung.

Kartadie, Rikie.; Satya, Barka., Februari 2015, "Uji Performa *Controller Floodlight* dan *OpenDayLight* sebagai Komponen Utama Arsitektur *Software Defined Network*", STMIK AMIKOM Yogyakarta.

G. Patel, A.S. Athreya, and S. Erukulla., "*OpenFlow based Dynamic Load Balanced Switching*, " COEN 233, Project Report, 2013

Kurniawati Ulfa, "Analisis Perbandingan Algoritma *Djikstra* dan *Bellman-Ford* dengan menggunakan *Software Defined Network*", Fakultas Ilmu Komputer,Universitas Brawijaya,2016

Ibrahim,2016 "Analisis Perbandingan Algoritma *Floyd-Warshall* dan *Djikstra* untuk Menentukan Jalur Terpendek pada jaringan *OpenFLow*", Fakultas Ilmu Komputer, Universitas Brawijaya

Gopinda Al Araaf, Mei 2016 **"**Implementasi Algoritma *Bellman-Ford* dan *Floyd Warshall* untuk Mencari Rute Terpendek (Stud Kasus: Rute Jakarta-Jogja)", Sistem Informasi STMIK AMIKOM Yogyakarta,

Lazuardi Erico, 2016 "Metode Pemilihan Jalur Routing Adaptif Berdasarkan Kemacetan Jaringan Dengan Algoritma *Dijkstra* pada *OpenFlow*", Fakultas Ilmu Komputer, Universitas Brawijaya