PERBANDINGAN KOMPLEKSITAS ALGORITMA DIJKSTRA, BELLMAN-FORD DAN JOHNSON PADA SDN (SOFTWARE-DEFINED NETWORKING)

Rangga Adi Kurnia[1], Drs. Ir. Rumani M., Bc.TT., M.Sc[2], Dr. Marisa W. P. S.T., M.T.[3]

[1,2,3,]Prodi S1 Sistem Komputer, Fakultas Teknik Elektro, Univesitas Telkom
Jln. Telekomunikasi No.1 Terusan Buah Batu Bandung 40257 Indonesia
ranggaak.botak@gmail.com[1], rumani@telkomuniversity.ac.id[2], marisa.paryasto@gmail.com[3]

ABSTRAK

Algoritma pencarian jalur terpendek atau lebih dikenal dengan shortest-path digunakan untuk menentukan rute dalam sebuah graff. Algoritma pencarian jalur terpendek sering kali diimplementasikan pada sebuah jaringan. SDN (Software-Defined Networking) adalah adalah sebuah konsep pendekatan jaringan komputer dimana sistem pengkontrol dari arus data dipisahkan dari perangkat kerasnya. Hal tersebut membuat suatu jaringan mudah diatur dan lebih fleksibel, hal tersebut dikarenakan pada SDN sebuah sistem pengontrol bersifat programmable. Algoritma routing yang akan dibahas dan digunakan pada jaringan SDN adalah algoritma Dijkstra, Bellman-Ford dan Johnson.

Penulis akan melakukan analisis algoritma untuk mengetahui kompleksitas ketiga algoritma tersebut. Dimana bertujuan untuk menentukan algoritma terbaik dalam sebuah topologi dengan menentukan kompleksitas masing-masing algoritma yang diperoleh dari nilai konvergensi dan nilai memori yang dibutuhkan. Tiap algoritma akan diimplementasikan pada controller RYU dan diterapkan pada topologi mesh dan tree yang telah dibuat pada emulator mininet.

Hasil dari pengujian tiap algoritma adalah, semakin besar jaringan maka semakin besar pula nilai konvergensi dan memori yang dibutuhkan. Dan juga jika suatu algoritma bagus pada topologi tertentu, tidak menunjukkan bahwa algoritma tersebut akan bagus juga pada topologi lain.

Kata kunci : SDN, algorita Dijkstra, algorita Bellman-ford, algorita Johnson, kompleksitas algorita

ABSTRACT

Shortest path search algorithm or better known as shortest-path is used to determine the route in a graff. The shortest path search algorithm is often implemented on a network. SDN (Software-Defined Networking) is a computer network approach concept where the control system of the data stream is separated from the hardware. This makes the network easier to manage and more flexible, it is because the SDN a programmable controller system. Routing algorithms to be discussed and used on SDN networks are Dijkstra, Bellman-Ford and Johnson algorithms. The author will perform an algorithmic analysis to determine the complexity of the three algorithms.

The author will perform an algorithmic analysis to determine the complexity of the three algorithms. Where aims to determine the best algorithm in a topology by determining the complexity of each algorithm obtained from the value of convergence and memory value required. Each algorithm will be implemented on the RYU controller and applied to the mesh and tree topology that has been created on the mininet emulator.

The result of testing each algorithm is, the larger the network the greater the value of convergence and memory required. And also if a good algorithm on a certain topology, does not indicate that the algorithm will be good also on other topologies.

Keywords: SDN, Dijkstra algorithm, Bellman-ford algorithm, Johnson algorithm, algorithm complexity

I. PENDAHULUAN

Algoritma pencarian jalur terpendek atau lebih dikenal dengan shortest-path digunakan untuk menentukan rute dalam sebuah graff. Dengan adanya banyak algoritma pencarian jalur terpendek atau shortest-path tentunya memberikan kita banyak pilihan dalam menentukan algorima yang ingin kita gunakan. Disamping itu kita harus menetukan algoritma manakah yang memiliki waktu komputasi paling cepat dan seberapa besar penggunaan ruang memori ketika komputasi berlangsung. Disinilah kompleksitas digunakan. Kompleksitas algoritma terdiri dari dua macam yaitu kompleksitas waktu dan kompleksitas ruang [2]. Kompleksitas waktu pada sebuah algoritma berisi jumlah langkah atau ekpresi bilangan yang dibutuhkan suatu fungsi dari ukuran permasalahan [2]. Sedangkan komplesitas ruang berkaitan dengan sistem memori yang dibutuhkan untuk esksekusi sebuah program [2].

SDN (Software-Defined Networking) sebuah konsep pendekatan baru untuk mendesain, membangun dan mengelola jaringan komputer dengan memisahkan control plane dan data plane [12]. Controller SDN yang bersifat programmable memnungkinkan untuk mengimplementasikan suatu routing pada SDN. Routing merupakan penentuan rute terbaik yang akan dilalui informasi yang dikirim dari pengirim menuju penerima. Dalam network, routing biasanya menggunakan suatu algoritma pencarian jalur terpendek.

Menanggapi dari latar belakang diatas mendorong penulis untuk melakukan penelitian perbandingan kompleksitas antara algoritma Dijkstra, Bellman-Ford dan Johnson pada dua topologi berbeda yaitu mesh dan tree didalam jaringan SDN. dan akan dilakukan analisis kinerja dari masing masing algoritma pada topologi mesh dan tree untuk dimenentukan algoritma terbaik untuk masing masing topologi tersebut.

II. LANDASAN TEORI

2.1. Kompleksitas Algoritma

Kompleksitas dari suatu algoritma merupakan ukuran seberapa banyak komputasi yang dibutuhkan algoritma tersebur untuk menyelesaikan masalah. Secara informal, algoritma yang dapat menyelesaikan suatu permasalahan dalam waktu yang singkat memiliki kompleksitas yang rendah, sementara algoritma yang membutuhkan waktu lama untuk menyelesaikan masalahnya mempunyai kompleksitas yang tinggi. Kompleksitas algoritma terdiri dari dua macam yaitu kompleksitas waktu dan kompleksitas ruang [2].

Kompleksitas waktu pada sebuah algoritma berisi jumlah langkah atau ekpresi bilangan yang dibutuhkan suatu fungsi dari ukuran permasalahan [2]. Sedangkan komplesitas ruang berkaitan dengan sistem memori yang dibutuhkan untuk esksekusi sebuah program [2].

2.2. Software-Defined Networking

SDN (Software-Defined Networking) sebuah konsep pendekatan baru untuk mendesain, membangun dan mengelola jaringan komputer dengan memisahkan *control plane* dan *data plane* [7]. Pemisahan ini memungkinkan seseorang untuk menerapkan *control plane* di suatu entitas eksternal yang disebut *controller* [5] dan *data plane* yang termasuk pada forwading packet disebut *switch*[7]. Konsep SDN ini sangat memudahkan operator atau administrator jaringan dalam mengelola jaringannya [1].

2.3. Algoritma Dijkstra

Algoritma Dijkstra, dinamai menurut penemunya, Edsger Dijkstra, adalah algoritma dengan prinsip greedy yang memecahkan masalah lintasan terpendek untuk sebuah graf berarah dengan bobot sisi yang tidak negatif. Berikut merupakan pseudocode algoritma Dijkstra [3]:

```
{ Algoritma Dijkstra }
procedure Dijkstra
        (
        input m: matriks,  a: integer {simpul awal}
        )
{mencari lintasan terpendek dari simpul awal a ke semua simpul lainya
Masukan : matriks ketetanggaan (m) dari graf berbobot G dan simpul awal a
Keluaran: lintasan terpendek dari a ke semua simpul lainnya}
KAMUS
s1,s2,...,sn : interger {larik interger}
d1,d2,...,dn : interger {larik interger}
 i : interger

ALGORITMA
{Langkah 0 (inisialisasi) : }
for i ← 1 to n do
        si ← 0
        di ← mai
endfor

 {Langkah 1: }
        sa ← 1
{karena simpul a adalah simpul asal lintasan terpendek, jadi terpilih dalam lintasan terpendek}
        da ← infinity
{tidak ada lintasan terpendek dari simpul a ke a}

{Langkah 2,3,...,n1 :}
for i ← 2 to n-1 do
{Cari j sedemikian sehingga sj = 0  dan dj = min (d1,d2,...,dn)}
        Sj ← 1
{simpul j sudah terpilih ke dalam lintasan terpendek}
{perbarui di, untuk i = 1,2,3,...,n  dengan : di (baru) = min{di(lama), dj + mji}
endfor
```

2.4. Algoritma Bellman-ford

Algoritma Bellman-Ford menghitung jarak terpendek (dari satu sumber) pada sebuah digraf berbobot. Berikut merupakan pseudocode algoritma Bellman-Ford [4]:

```
record titik {
   list sisi2
   real jarak
   titik sebelum
```

```
}
record sisi {
   titik dari
   titik ke
   real bobot
}

function BellmanFord(list semuatitik, list semuasisi, titik dari)
  // Argumennya ialah graf, dengan bentuk daftar titik
  // and sisi. Algoritma ini mengubah titik-titik dalam
  // semuatitik sehingga atribut jarak dan sebelum
  // menyimpan jarak terpendek.

  // Persiapan
  for each titik v in semuatitik:
     if v is dari then v.jarak = 0
     else v.jarak := tak-hingga
     v.sebelum := null

  // Perulangan relaksasi sisi
  for i from 1 to size(semuatitik):
     for each sisi uv in semuasisi:
        u := uv.dari
        v := uv.ke          // uv adalah sisi dari u ke v
        if v.jarak > u.jarak + uv.bobot
           v.jarak := u.jarak + uv.bobot
           v.sebelum := u

  // Cari sirkuit berbobot(jarak) negatif
  for each sisi uv in semuasisi:
     u := uv.dari
     v := uv.ke
     if v.jarak > u.jarak + uv.bobot
        error "Graph mengandung siklus berbobot total negatif"
```

2.5. Algoritma Johnson

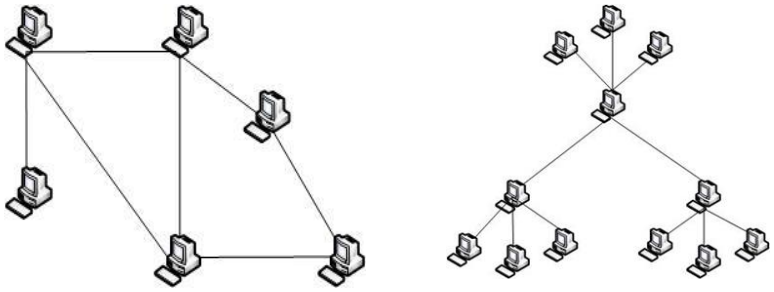
Algoritma Johnson merupakan perpaduan antara algoritma Dijkstra dan algoritma Bellman-Ford. Berikut merupakan pseudocode algoritma Johnson:

```
1.
   create G` where G`.V = G.V + {s},
      G`.E = G.E + ((s, u) for u in G.V), and
      weight(s, u) = 0 for u in G.V
   2.
   if Bellman-Ford(G) == False
      return "The input graph has a negative weight cycle"
   else:
      for vertex v in G`.V:
         h(v) = distance(s, v) computed by Bellman-Ford
      for edge (u, v) in G`.E:
         weight`(u, v) = weight(u, v) + h(u) - h(v)
   3.
      D = new matrix of distances initialized to infinity
      for vertex u in G`.V:
         run Dijkstra(G, weight`, u) to compute distance`(u, v) for all v in G.V
         for each vertex v in G.V:
            D_(u, v) = distance`(u, v) + h(v) - h(u)
      return D
```

2.6. Topologi

Topologi jaringan adalah susunan dari berbagai elemen (link, nodes, dll) didalam sebuah jaringan. Ada dua cara untuk menentukan geometri jaringan yaitu topologi fisik (physical topology) dan topologi logis (logical tolopology). Topologi fisik (physical topology) adalah penempatan berbagai komponen jaringan, termasuk lokasi perangakat dan pemasangan kabel. Sementara tolopologi logis (logical tolopoly) menggambarkan bagaimana data mengalir didalam jaringan, terlepas dari desain fisiknya.

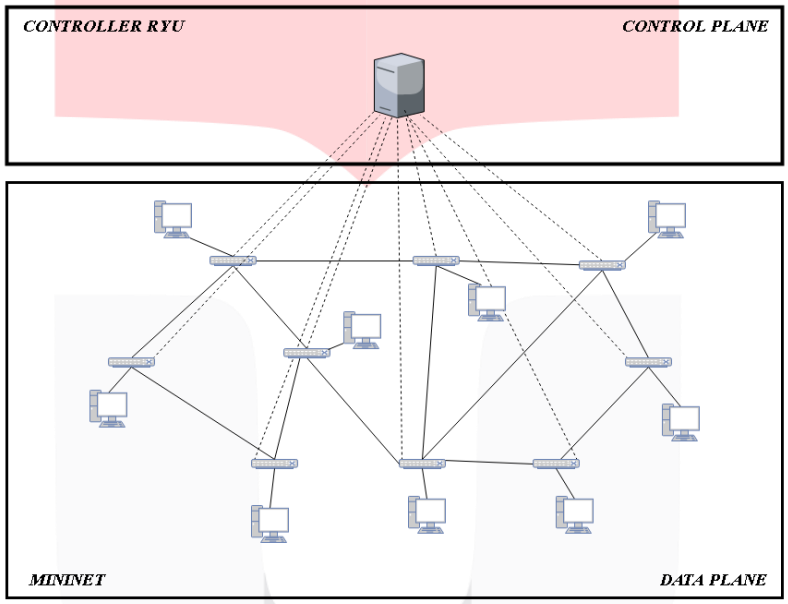
Topologi yang dipakai adalah topologi mesh dan topologi tree.



Gambar 2.1 Topologi mesh dan topologi tree

III. PERANCANGAN SISTEM

3.1. Skema Sistem



Gambar 3.1 Model skema sistem

Sistem ini mensimulasikan jaringan berbasis SDN yang terdiri dari controller dan data plane. Controller yang digunakan adalah controller RYU yang bertugas sebagai pengontrol jalur data, melakukan routing dan menjalankan aturan dari SDN. Data plane menggunakan emulator mininet. Topologi yang terdiri dari switch Openflow dan host dirancang paa mininet.

Pada controller RYU ditanamkan routing dengan algoritma Dijkstra, Bellman-Ford dan Johnson. Sedangkan pada mininet menggunakan topologi mesh dan tree..

3.2. Perancangan Algoritma Pada *Controller* RYU

Digunakan *controller* RYU untuk menjalankan fungsi *control plane*. RYU menyediakan fitur-fitur yang mendukung fungsi *controller*.

Algoritma Dijkstra, Bellman-Ford dan Johnson pada tugas akhir ini diimplementasikan pada RYU *controller* dengan memanfaatkan fitur fitur yang sudah disedikan oleh RYU sendiri. Pada tugas akhir ini dibuat API yang bertugas untuk membangkitkan fitur yang berada dalam ryu controller berupa sebuah class yang dibuat dengan bahasa pemograman python denga komponen sebagai berikut :

1. *Topology Discovery*
   Topology discovery digunakan untuk mengumpulkan informasi *data plane* berupa bentuk topologi, *link, switch, host* dll.

2. *Network View*
   Digunakan networkx library yang berfungsi untuk mengubah topologi yang didapat pada *topology discovery* kebentuk graph berbobot agar memudahkan untuk implementasi algoritma. Networkx library mempunyai fitur untuk menjalankan algoritma Dijkstra, Bellman-Ford dan Johnson yang akan digunakan untuk algoritma pencarian jalur terpendek.

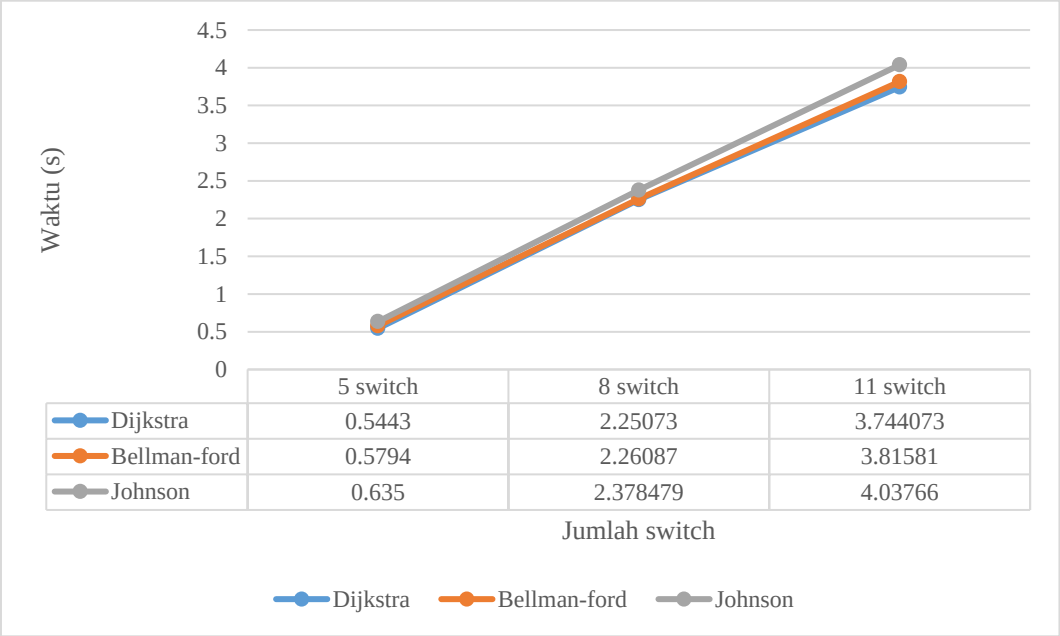
IV. IMPLEMENTASI DAN PENGUJIAN

4.1. Skenario Pengujian

Pengujian kompleksitas dari masing masing algoritma Dijkstra, Bellman-Ford dan Johnson yang diimplementasikan pada jaringan SDN dengan menggunakan emulator mininet dilihat dari segi network performansi. Sedangkan parameter yang diukur untuk melihar performansi dari masing masing algoritma pada penelitian ini adalah network convergence dan resource utilization. Kemudian dilakukan pencarian kompleksitas total dengan nilai delay dan resource utilization, dengan rumus [6]:

$$Kompleksitas\ Total = Memori * (Waktu)^2 \quad (4.1)$$

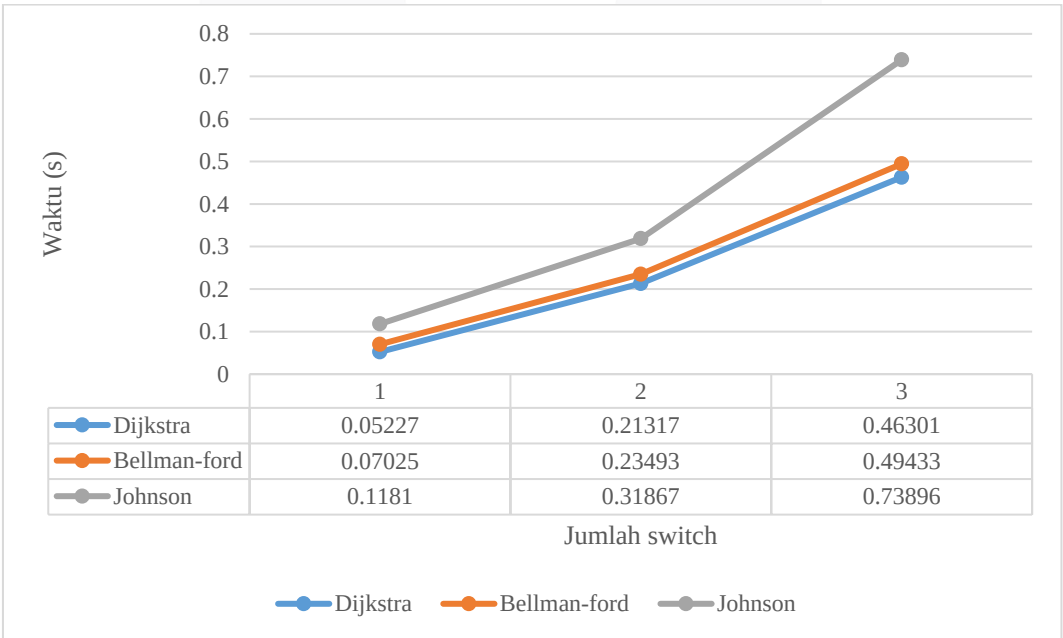
4.2. Pengujian Network Convergence



	5 switch	8 switch	11 switch
Dijkstra	0.5443	2.25073	3.744073
Bellman-ford	0.5794	2.26087	3.81581
Johnson	0.635	2.378479	4.03766

Gambar 4.1 Hasil network convergence pada topologi mesh

Pada Gambar 4.1 di atas algoritma dijsktra terlihat yang paling cepat dalam hal network convergence hal tersebut di karenakan operasi yang dilakukan algoritma Dijkstra lebih banyak dari pada kedua algoritma yang lain, sehingga memakan waktu lebih sedikit. Sedangkan algoritma Johnson terlihat yang paling lama dalam hal network convergence hal tersebut di karenakan operasi yang dilakukan algoritma Johnson lebih banyak dari pada kedua algoritma yang lain, sehingga memakan waktu lebih banyak.



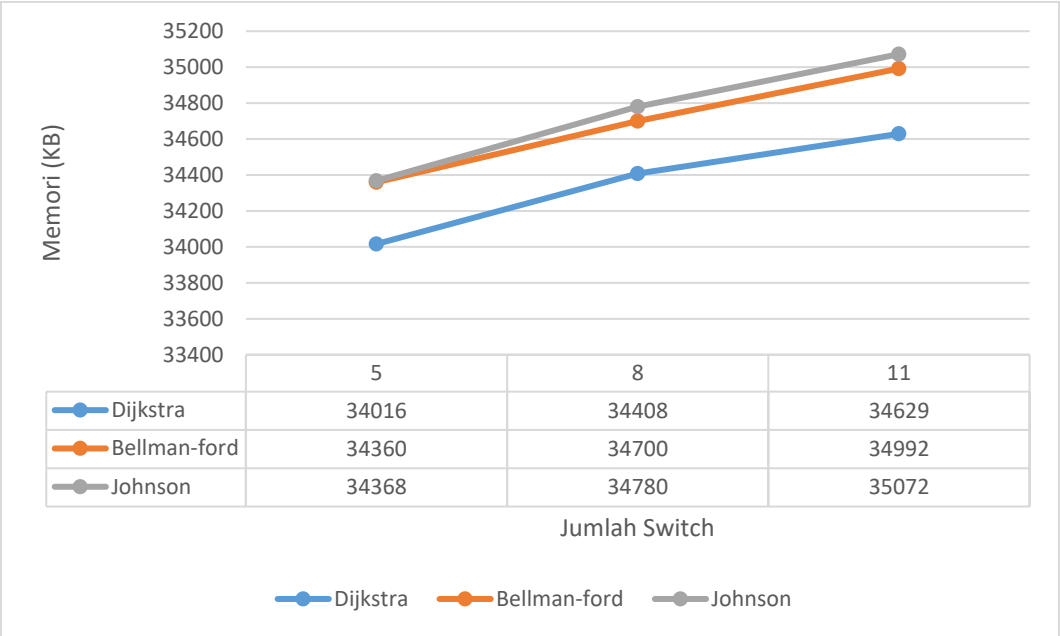
	1	2	3
Dijkstra	0.05227	0.21317	0.46301
Bellman-ford	0.07025	0.23493	0.49433
Johnson	0.1181	0.31867	0.73896

Gambar 4.2 Hasil network convergence pada topologi tree

Pada Gambar 4.2 di atas algoritma Dijkstra terlihat yang paling cepat dalam hal network convergence hal tersebut di karenakan operasi yang dilakukan algoritma Dijkstra lebih banyak dari pada kedua algoritma yang lain, sehingga memakan waktu lebih sedikit. Sedangkan algoritma Johnson terlihat yang paling lama dalam hal network convergence hal tersebut di karenakan operasi yang dilakukan algoritma

Johnson lebih banyak dari pada kedua algoritma yang lain, sehingga memakan waktu lebih. Sangat signifikan pebedaan antara algoritma Johnson dengan yang lain, memungkinkan perbedaan yang lebih besar jika switch bertambah banyak.

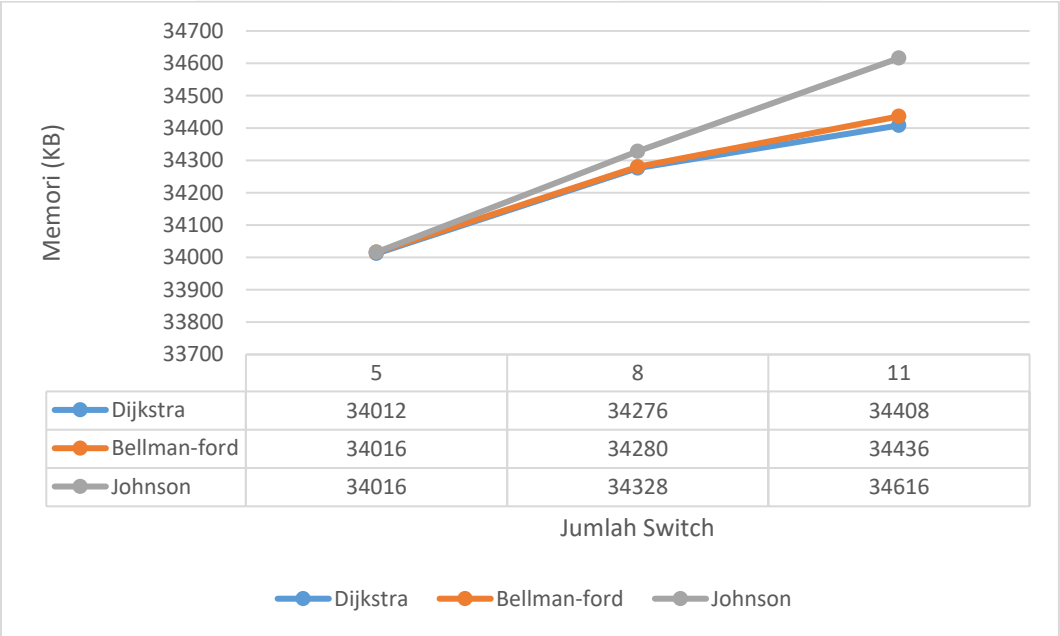
4.3. Pengujian Resource Utilization



	5	8	11
Dijkstra	34016	34408	34629
Bellman-ford	34360	34700	34992
Johnson	34368	34780	35072

Gambar 4.3 Hasil resource utilization pada topologi mesh

Pada Gambar 4.3 di atas hasil pada pengujian resource utilization pada topologi mesh menunjukkan besarnya konsumsi memori yang dipakai selalu meningkat seiring bertambahnya jumlah switch yang digunakan. Pada grafik di atas dapat disimpulkan bahwa algoritma Dijkstra adalah algoritma yang paling sedikit mengkonsumsi memori pada setiap variasi jumlah switch. Sedangkan algoritma Johnson adalah algoritma yang paling boros mengkonsumsi memori ketika controller dijalankan. Hal ini disebabkan karena proses pencarian jalur dengan algoritma dijkstra memerlukan operasi yang lebih sedikit dibandingkan kedua algoritma yang lain, maka konsumsi memori pada algoritma dijsktra lebih sedikit.

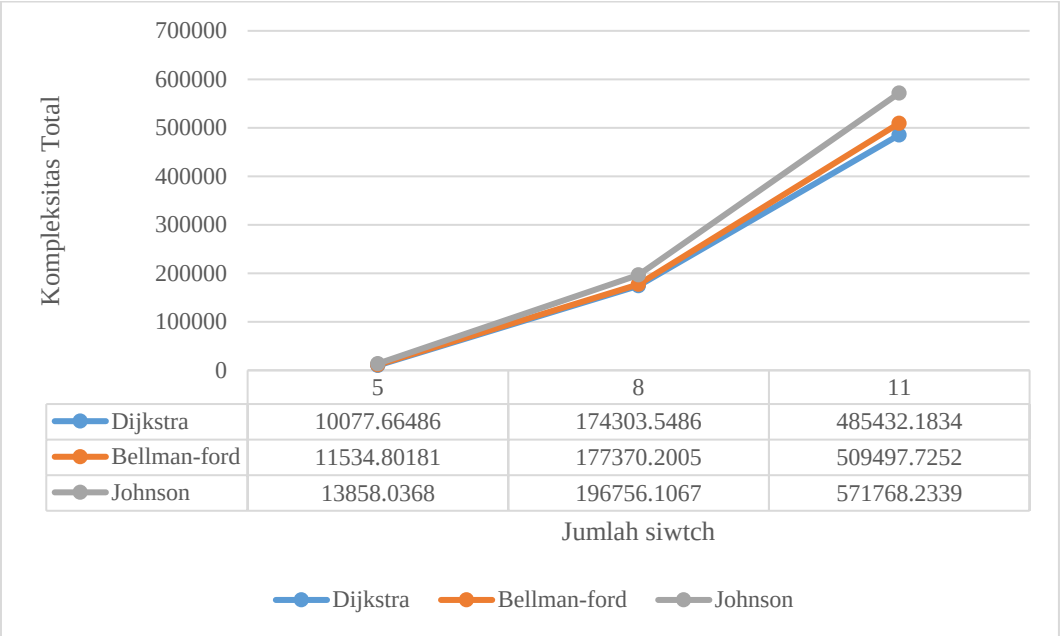


	5	8	11
Dijkstra	34012	34276	34408
Bellman-ford	34016	34280	34436
Johnson	34016	34328	34616

Gambar 4.4 Hasil resource utilization pada topologi tree

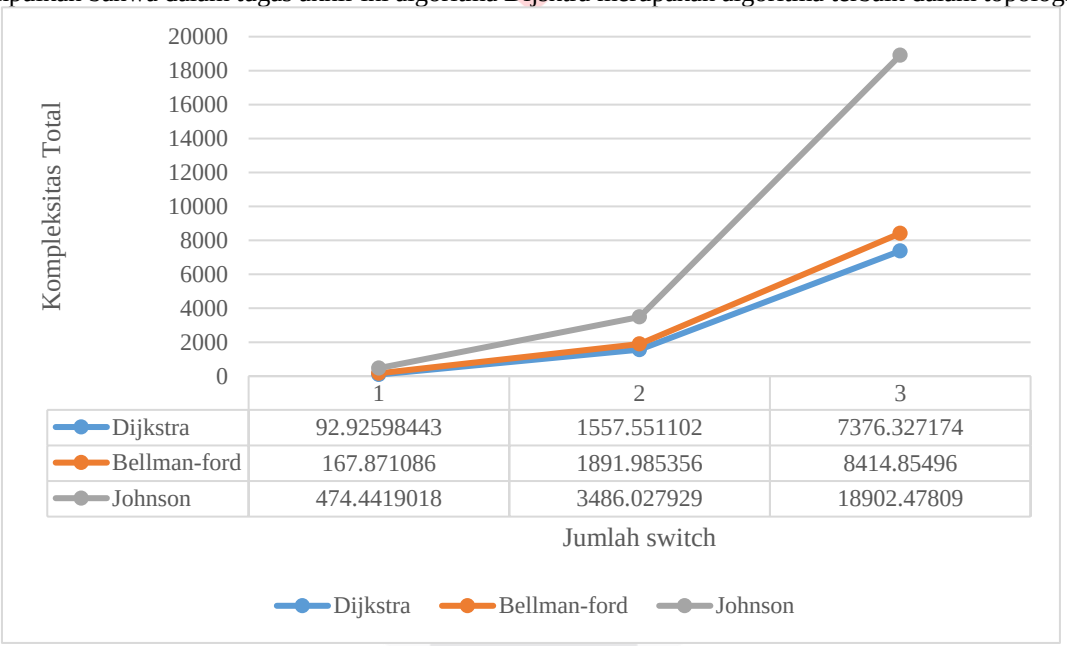
Pada Gambar 4.4 di atas hasil pada topologi tree menunjukkan bahwa algoritma Dijkstra sebagai algoritma yang paling kecil mengkonsumsi memori, sedangkan algoritma Johnson algoritma yang paling boros mengkonsumsi memori. Hal ini disebabkan karena proses pencarian jalur dengan algoritma dijkstra memerlukan operasi yang lebih sedikit dibandingkan kedua algoritma yang lain, maka konsumsi memori pada algoritma dijsktra lebih sedikit.

4.4. Pengujian Kompleksitas Total



Gambar 4.5 Hasil kompleksitas total pada topologi mesh

Pada Gambar 4.1 di atas dapat disimpulkan bahwa kompleksitas total yang dihitung dari topologi mesh menunjukkan algoritma Johnson memiliki kompleksitas total yang paling besar, diikuti Bellman-ford lalu Dijkstra dengan kompleksitas paling kecil. Hasil tersebut menyimpulkan bahwa dalam tugas akhir ini algoritma Dijsktra merupakan algoritma terbaik dalam topologi mesh.



Gambar 4.5 Hasil kompleksitas total pada topologi tree

Pada Gambar 4.1 di atas dapat disimpulkan bahwa kompleksitas total yang dihitung dari topologi mesh menunjukkan algoritma Johnson memiliki kompleksitas total yang paling besar, diikuti Bellman-ford lalu Dijkstra dengan kompleksitas paling kecil. Hasil tersebut menyimpulkan bahwa dalam tugas akhir ini algoritma Dijkstra merupakan algoritma terbaik dalam topologi tree.

V. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan dari penelitian dan pengujian yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Pengujian waktu *network convergence* pada topologi mesh dan tree dengan variasi *switch* 5, 8, dan 11 menyimpulkan bahwa algoritma Dijkstra memiliki hasil *network convergence* yang paling kecil dari ketiga algoritma tersebut. Hal tersebut dikarenakan operasi algoritma Dijkstra lebih sedikit ketimbang dua algoritma lain.
2. Pengujian *Resource Utilization* pada topologi mesh dan tree menyimpulkan bahwa algoritma Dijkstra adalah algoritma yang paling baik dalam hal konsumsi memori, karena algoritma Dijsktra yang paling sedikit mengkonsumsi memori.
3. Pengujian kompleksitas total pada topologi mesh dapat disimpulkan bahwa algoritma Dijkstra adalah algoritma yang paling bagus untuk diimplementasikan pada topologi mesh dan tree.

5.2. Saran

Beberapa saran untuk mengembangkan penelitian ini selanjutnya adalah sebagai berikut:

1. Membandingkan algoritma shortest path lain seperti, BFS, Hill climbing, Kruskal, dll.
2. Menggunakan variasi dan jumlah *switch* yang lebih banyak.

Daftar Pustaka

[1] Abdillah, Desiaonto., Ummah, Izzatul. (2016). *Perancangan Simulasi Jaringan Virtual Berbasis Software-Define Networking*.

[2] Azizah, Ulfah Nur. (2013). *Perbandingan Detektor Tepi Prewit dan Detektor Tepi Laplacian Berdasarkan Kompleksitas Waktu dan Citra Hasil*. Universitas Pendidikan Indonesia.

[3] I Nyoman Prama Pradnyana. (2010). *Pencarian Rute Terpendek Tempat Penting Melalui Mobile GMaps dengan Menggunakan Algoritma Dijkstra*. Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika. Institut Teknologi Bandung. Bandung.

[4] Kamayudi, Apri. Studi dan Implementasi Algoritma Dijkstra, Bellman-Ford dan Flyod-Warshall dalam Menagani Masalah Lintasan Terpendek dalam Graf. Program Studi Teknik Informatika. Institut Teknologi Bandung. Bandung.

[5] P.Fonseca, R. Bennesby, E. Mota and A, Passito (2012). *A replication component for resilient openflow-based networking.*

[6] Paryasto, M. W. (2012). Arsitektur Unit Pengali Composite Field Kombinasi MH-KOA Untuk Eliptic Curve Cryptography.

[7] Risdianto, Aris Cahyadi., Arif, Muhammad., Eueung, Mulyana. (2015). *Buku Komunitas SDN-RG*. Bandung : GitBook.