POCO – K-Means : Optimasi Penempatan Kontroler pada SDN

Rizki Jamilah Guci[1], Aji Gautama Putrada Satwiko[2], Muhammad Al Makky[3]

[1,2,3]Fakultas Informatika, Universitas Telkom, Bandung
[1,2,3]Kelompok Keahlian Telematika
[1]rizkijamilahguci@students.telkomuniversity.ac.id, [2]ajigp@telkomuniversity.ac.id,
[3]malmakky@telkomuniversity.ac.id

Abstrak

***Software Defined Network* (SDN) adalah sebuah arsitektur jaringan dimana *control plane* yang merupakan entitas eksternal yang disebut sebagai *controller* terstruktur secara terpusat (*Centralized Control*). Dalam skala jaringan besar SDN dapat terdiri dari beberapa controller yang mendistribusikan manajemen jaringan, dimana setiap *controller* terpusat secara logis tetapi terdistribusi secara fisik [1]. *Controller* merupakan pusat jaringan dalam SDN sehingga letak *controller* dalam jaringan sangat penting untuk diperhatikan karena berdampak pada kinerja jaringan yang dihasilkan. Salah satu parameter penting dalam mengukur kinerja jaringan adalah *latency*. *Latency* adalah waktu yang diperlukan untuk mengirim paket dari pengirim ke penerima. Salah satu penyebab latency adalah jarak antara *node* dan *controller* [2]. Sehingga pada penelitian ini, dibutuhkan penggunaan *clustering* untuk meminimalkan jarak antara *node* dan *controller*. Metode *clustering* yang digunakan adalah K-Means dan pengukuran *latency* menggunakan *toolset* POCO yang umum digunakan untuk simulasi optimasi dalam lingkungan MATLAB. Hasil pengujian menunjukkan bahwa metode POCO – K-Means dapat diterapkan untuk penempatan *controller* sesuai dengan skenario dan topologi yang digunakan dalam penelitian ini. Jumlah *cluster* = 4 adalah jumlah *cluster* yang optimal untuk digunakan dengan nilai SSE = 175,5260 dan *Average latency* = 0,13ms.**

Kata kunci : *Software Defined Network*, Penempatan *Controller*, *Latency*, K-Means, POCO

Abstract

Software Defined Network (SDN) is a network architecture where the control plane is an external entity called a controller. On a large scale SDN network consists of several controllers that distribute network management, where each controller has logically centralized but physically distributed [1]**. Controller is the network center in the SDN so that the location of the controller in the network is very important because it has an impact on the performance of the network produced. One important parameter in measuring network performance is latency. Latency is the time needed to send a packet from the sender to the recipient. One of the causes of latency is the distance between node and controller** [2]**. Therefore, it is necessary to use clustering to minimize the distance between node and controller. The clustering method used is K-Means and latency measurements using the POCO toolset that is commonly used for optimization simulations in the MATLAB environment. The test results show that the POCO-K-Means method can be applied to the placement of the controller according to the scenario and topology used in this work. The optimal number of cluster is 4 with SSE = 175.5260 and average latency = 0.13ms.**

Keywords: Software Defined Network, Controller Placement, Latency, K-Means, POCO

1. Pendahuluan

Latar Belakang

Software Defined Network (SDN) merupakan arsitektur jaringan yang memisahkan *control plane* dan *data plane*. Fungsi dari *control plane* dari elemen jaringan seperti *routing, signalling* dan distribusi label, dipindahkan ke satu atau lebih entitas eksternal yang disebut sebagai *controller*. Fungsi dari *data plane* yang tetap berada dalam elemen jaringan seperti *forwarding, queueing* dan *policing*, diinterpretasikan sebagai *switch* [3]. *Controller* yang terletak pada *control plane* adalah otak dari jaringan SDN, dimana semua keputusan tentang jaringan dibuat. *Controller* bekerja untuk aplikasi kontrol, modifikasi keadaan jaringan, dan penjadwalan aliran. *Controller* menentukan arus-arus yang dapat berada di dalam jaringan. Setiap arus di dalam jaringan harus mendapat ijin terlebih dahulu dari *controller* [4]. *Controller* mengacu pada semua fungsi dan proses yang menentukan jalur mana yang digunakan. Fungsi *controller* mencakup konfigurasi sistem, manajemen, dan pertukaran informasi tabel *routing* seperti RIP, OSPF, atau BGP [5]. SDN adalah inovasi yang menyederhanakan manajemen jaringan, sehingga saat ini banyak operator jaringan yang beralih pada SDN karena lebih fleksibel untuk deprogram dibandingkan jaringan konvensional [6].

Meski demikian, SDN juga memiliki suatu masalah, yaitu lokasi untuk penempatan *controller* pada jaringan. Penempatan *controller* harus berada pada lokasi yang optimal karena dapat mempengaruhi kinerja jaringan [7].

Untuk mengatasi masalah ini, beberapa metrik seperti *latency*, *reliability*, *energy saving*, dan *load balancing* sering dipertimbangkan. Tetapi di antara metrik ini, *latency* merupakan metrik yang paling berperan penting dalam kinerja jaringan karena sangat mempengaruhi kualitas layanan dalam mengembangkan aplikasi *real time* [7]. *Latency* adalah waktu yang diperlukan untuk mengirim paket dari pengirim ke penerima. Semakin tinggi *latency*, maka semakin tinggi resiko kegagalan akses [8]. Umumnya, untuk *local area networks* (LANs) *controller* tunggal cukup untuk mengontrol seluruh jaringan. Namun berbeda dengan LANs, *wide area networks* (WANs) memiliki jangkauan luas dan *latency* yang panjang sehingga penempatan *controller* akan menjadi fokus utama karena lokasi *controller* akan mempengaruhi *latency* yang berakibat pada kinerja jaringan [5][9].

Berawal dari permasalahan tersebut, hal ini perlu sebuah teknik yang dapat menghasilkan solusi dari permasalahan penempatan *controller*. Salah satu metode yang diusulkan untuk masalah penempatan *controller* adalah dengan teknik partisi jaringan. Partisi jaringan merupakan cara yang efektif untuk mempersingkat *latency response* antar *node* [7]. K-Means adalah salah satu metode dengan teknik *clustering*. K-Means termasuk metode berbasis jarak yang membagi data ke dalam sejumlah *cluster*, dan hanya bekerja pada atribut numerik. Oleh karena itu, metode yang digunakan dalam tugas akhir ini adalah K-Means. Kemudian ada satu heuristik, heuristik ini ada dalam suatu program yang dibuat khusus untuk menyelesaikan masalah *Controller Placement Problem* (CPP) yang disebut program *Pareto Optimal Controller Placement* (POCO). K-Means dan POCO digabung untuk memberikan solusi lebih cepat, karena itu dinamakan POCO - K-Means.

Topik dan Batasannya

Dengan latar belakang masalah yang telah dideskripsikan sebelumnya, maka dibuatlah simulasi jaringan pada MATLAB. Adapun batasan masalah pada tugas akhir ini adalah :

1. Hanya berfokus pada penempatan *controller*
2. Parameter yang di ukur adalah *latency*
3. Topologi yang digunakan adalah Internet2 OS3E
4. Setiap *node* di jaringan dapat di pilih sebagai lokasi untuk *controller*
5. Jaringan sepenuhnya dapat diandalkan (tidak ada simpul dan pengontrol yang gagal)

Tujuan

Tujuan dari tugas akhir ini adalah menemukan letak optimal pada jaringan menggunakan metode K-Means dan POCO untuk penempatan *controller*.

Organisasi Tulisan

1. Pendahuluan
   Bagian ini berisi penjelasan latar belakang, topik dan batasannya, tujuan dan penjelasan singkat dari susunan tugas akhir ini.
2. Studi Terkait
   Bagian ini menjelaskan teori dan literatur mendukung yang digunakan.
3. Sistem yang Dibangun
   Bagian ini memaparkan sistem yang dibuat dan diimplementasikan.
4. Evaluasi
   Bagian ini berisi analisa dan hasil pengujian yang telah dilakukan.
5. Kesimpulan
   Bagian ini menjelaskan kesimpulan dari hasil pengujian.

2. Studi Terkait

2.1 Studi Literatur

Masalah penempatan *controller* adalah salah satu masalah yang paling banyak mendapat perhatian dari kalangan peneliti. Penelitian [7] mengusulkan teknik partisi jaringan untuk menyederhanakan masalah penempatan *controller*. Penelitian ini menggunakan algoritma K-Means yang dimodifikasi untuk mempersingkat *latency* antara pengontrol dan saklar di sub-jaringan. Hasil simulasi menunjukkan bahwa *latency* maksimum dengan algoritma K-Means yang telah dimodifikasi jauh lebih pendek dari rata-rata *latency* yang dicapai K-Means standar.

Penelitian lain [4] mengusulkan algoritma Greedy yang mekanismenya diilustrasikan menggunakan topologi Internet2 OS3E. Penelitian ini mengambil *latency* sebagai pertimbangan penting untuk menyelesaikan masalah penempatan *controller*. Dan untuk pengukuran *latency* menggunakan toolbox bernama Kalman Filter yang diimplementasikan dalam MATLAB dengan antarmuka MEX. *Latency* yang diukur adalah *average latency* dan *worstcase latency*.

2.2 MATLAB

MATLAB [10] merupakan software yang digunakan untuk pemrograman, analisis, serta komputasi teknis dan matematis berbasis matriks. Dalam MATLAB tersedia *toolbox* yang dapat digunakan untuk aplikasi-aplikasi khusus, seperti pengolahan sinyal, sistem kontrol, logika *fuzzy*, jaringan saraf tiruan, optimasi, pengolahan citra digital, bioinformatika, simulasi dan berbagai teknologi lainnya.

Dalam MATLAB tersedia *tool* yang dapat digunakan untuk membuat *graphic user interface* (GUI) atau antarmuka pengguna grafis. GUI menjadi perantara dalam interaksi antara *user* dengan komputer melalui ikon grafis. GUI memudahkan *user* dalam mengoperasikan program komputer yang telah dibuat.

2.3 K-Means

Metode K-Means [11][12][13] adalah algoritma klasterisasi yang mempartisi objek ke dalam beberapa *cluster*. Secara umum, proses K-Means dapat dilihat pada gambar 1. Tujuan dari proses ini adalah untuk meminimalisasikan variasi dalam suatu *cluster*. K-Means banyak digunakan karena kemudahan implementasinya dan waktu proses yang relatif cepat. Kompleksitas waktu K-Means adalah O(nkt), dimana n adalah jumlah keseluruhan objek data, k adalah jumlah *cluster*, dan t adalah jumlah iterasi. K-Means merupakan metode partisi yang mengelompokkan objek data n ke dalam *cluster* k dengan jarak minimum antara pusat *cluster* dan objek data.

Pada masing-masing tahapan K-Means, berikut penjelasan dari gambar 1.

1. Tentukan k sebagai jumlah *cluster* yang akan dibentuk
2. Tentukan k *centroid* (titik pusat *cluster*) secara random
3. Hitung jarak setiap objek ke masing-masing *centroid* dari masing-masing *cluster*. Untuk menghitung jarak antara objek dengan *centroid* adalah dengan menggunakan *Euclidean Distance* seperti pada persamaan (1)

$$D(i,j) = \sqrt{(x_{1i} - x_{1j})^2 + (x_{2i} - x_{2j})^2 + \ldots + (x_{ki} - x_{kj})^2} \quad (1)$$

Dimana :

D (i,j) = Jarak objek ke-i ke pusat *cluster* j
x_{ki} = Objek ke-i pada atribut ke-k
x_{kj} = Titik pusat ke-j pada atribut ke-k

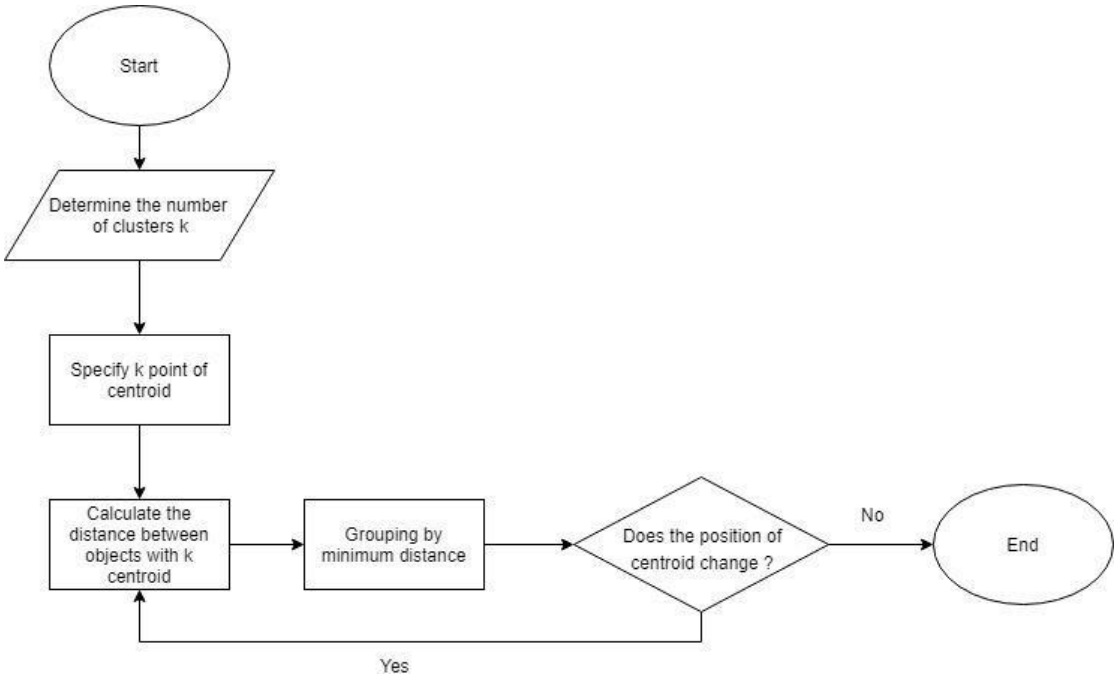
4. Alokasikan masing-masing objek ke dalam *cluster* dengan jarak terdekat
5. Lakukan iterasi, kemudian tentukan posisi *centroid* baru dengan menghitung rata-rata semua objek dalam *cluster* menggunakan persamaan (2)

$$C = \frac{\sum_{i=1}^{n} X_i}{n} \quad (2)$$

Dimana :

i = 1, 2, 3, … , n
C = *Centroid* pada *cluster*
Xi = objek ke-i
n = Banyaknya objek / jumlah objek yang menjadi anggota *cluster*

6. Ulangi langkah 3 jika posisi *centroid* baru tidak sama



Gambar 1 Flowchart K-Means

2.4 Metode Elbow

Metode Elbow [12] merupakan suatu metode yang digunakan untuk menghasilkan informasi dalam menentukan jumlah *cluster* terbaik dengan cara melihat persentase hasil perbandingan antara jumlah *cluster* yang akan membentuk siku pada suatu titik. Metode Elbow ini memberikan ide atau gagasan dengan cara memilih nilai *cluster* dan kemudian menambah nilai *cluster* tersebut untuk dijadikan model data dalam penentuan jumlah *cluster* terbaik. Selain itu persentase perhitungan yang dihasilkan menjadi pembanding antara jumlah *cluster* yang ditambah. Hasil persentase yang berbeda dari setiap nilai jumlah *cluster* dapat ditunjukkan dengan menggunakan grafik sebagai sumber informasinya. Jika nilai jumlah *cluster* pertama dengan nilai jumlah *cluster* kedua memberikan sudut dalam grafik atau nilainya mengalami penurunan paling besar maka jumlah *cluster* tersebut yang terbaik.

Untuk mendapatkan perbandingannya adalah dengan menghitung SSE (*Sum of Square Error*) dari masing-masing nilai jumlah *cluster*. Karena semakin besar jumlah *cluster* K maka nilai SSE akan semakin kecil. Rumus SSE pada K-Means bisa dilihat pada persamaan (3)

$$SS\blacklozenge = \sum_{\blacklozenge\blacklozenge-1}^{K} \sum_{\blacklozenge\blacklozenge\blacklozenge\blacklozenge} \| X\blacklozenge - \blacklozenge\blacklozenge \|_2 \quad (3)$$

Algoritma Metode Elbow dalam menentukan jumlah K terbaik pada K-Means

1. Mulai
2. Inisialisasi awal nilai K
3. Naikkan nilai K
4. Hitung SSE dari setiap nilai K
5. Melihat hasil SSE dari nilai K yang turun secara drastis
6. Tetapkan nilai K yang berbentuk siku
7. Selesai

2.5 *Pareto Optimal Controller Placement* (POCO)

POCO [14][15] adalah sebuah *toolset* pada MATLAB untuk menentukan penempatan *controller* berdasarkan parameter *latency*. Digunakan untuk menemukan solusi terbaik dari semua penempatan *controller*. POCO dirancang untuk mengukur kinerja jaringan pada berbagai fungsi objektif. Kerangka ini dipakai sebagai program GUI *open source* yang ada dalam MATLAB. Kerangka kerja POCO dirancang untuk mengevaluasi penempatan *controller* yang optimal dalam berbagai skenario jaringan.

POCO mengeksplorasi bagian dari pencarian ruang dengan metode *Pareto Simulated Annealing* (PSA). *Simulated Annealing* (SA) adalah algoritma pencarian metaheuristik yang dirancang untuk menemukan solusi perkiraan untuk masalah yang mungkin tidak dapat dikomputasi secara komputasional. Alasan untuk menggunakan heuristik semacam ini adalah untuk masalah penempatan

controller yang jaringannya terlalu besar, dan perilaku jaringan yang berubah terlalu cepat. POCO mengeksplorasi pencarian ruang dengan probabilitas. POCO mengevaluasi metrik kinerja dari setiap kombinasi yang mungkin dari memilih simpul untuk menjadi pengontrol. Penggunaan algoritma seperti PSA ini memungkinkan untuk mencapai solusi dalam jangka waktu yang lebih singkat.

2.6 *Performance Metric*

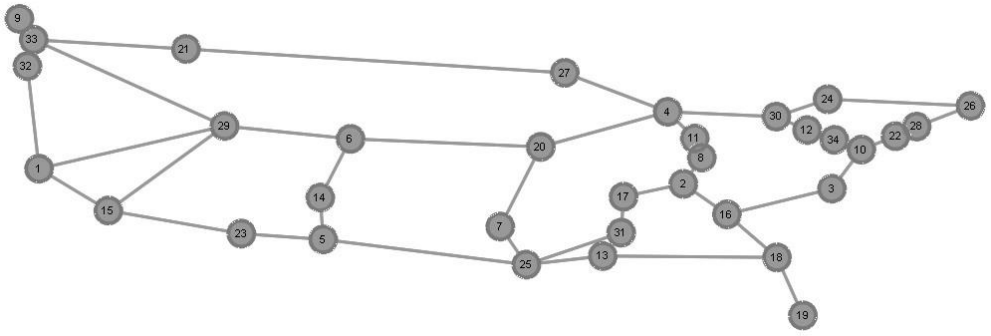
Latency [2] adalah waktu yang dibutuhkan untuk mengirimkan informasi dari pengirim ke penerima. *Latency* diukur dalam satuan *miliseconds* (ms). *Latency* dalam jaringan disebabkan oleh tiga elemen berbeda.

1. *Propagation time* : waktu yang diperlukan untuk satu paket data melakukan perjalanan dari satu tempat ke tempat lain, yang secara langsung berkaitan dengan jarak fisik antara kedua tempat
2. *Routing/Switching time* : saat transmisi data terjadi *delay* karena *buffering*
3. *Congestion* : ketika jaringan padat, paket-paket antri untuk dikirim sehingga menyebabkan transmisi ulang, dan akibatya paket mengalami keterlambatan sampai ke penerima

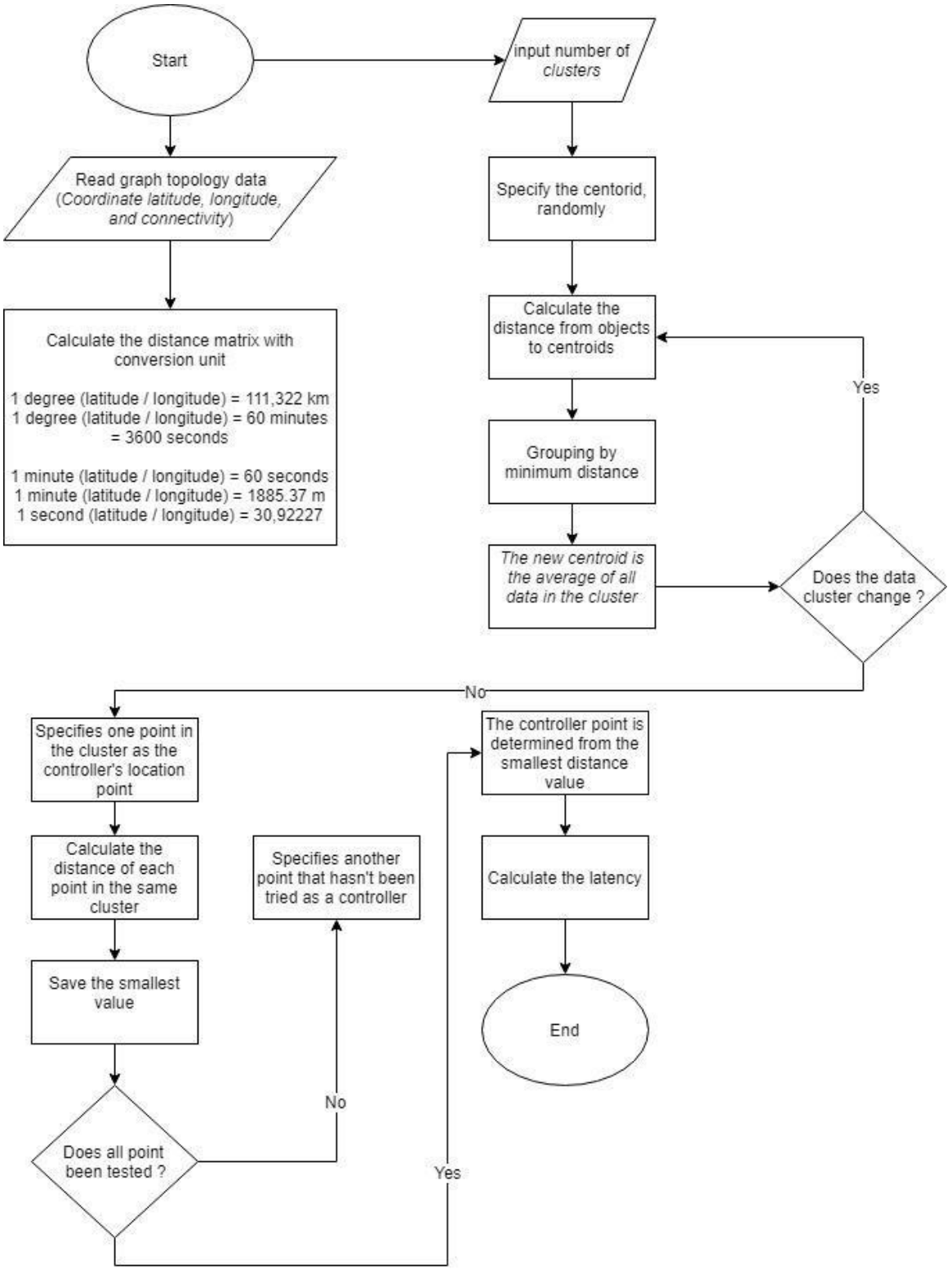
3. Sistem yang Dibangun

3.1 Gambaran Umum Sistem

Penelitian ini menggunakan topologi internet2 OS3E seperti yang terlihat pada gambar 2. Proses pembangunan sistem dimulai dengan pengumpulan data yang terkait dengan penelitian. Kemudian implementasi dilakukan hingga pengujian untuk mengetahui keberhasilan saat diuji. Proses lengkap alur kerja sistem dapat dilihat pada gambar 3.



Gambar 2 Topologi jaringan yang digunakan



Gambar 3 Flowchart Alur Kerja Sistem

3.2 Proses Penentuan Letak Kontroler

Seperti yang telah ditulis pada flowchart, K-Means dijalankan terlebih dahulu untuk mendapatkan *cluster* dan anggota *cluster*. Setelah *cluster* didapatkan, dilanjutkan dengan menghitung jarak setiap titik di dalam cluster yang sama untuk mendapatkan letak *controller*. Jarak terkecil akan diambil sebagai titik letak *controller.*

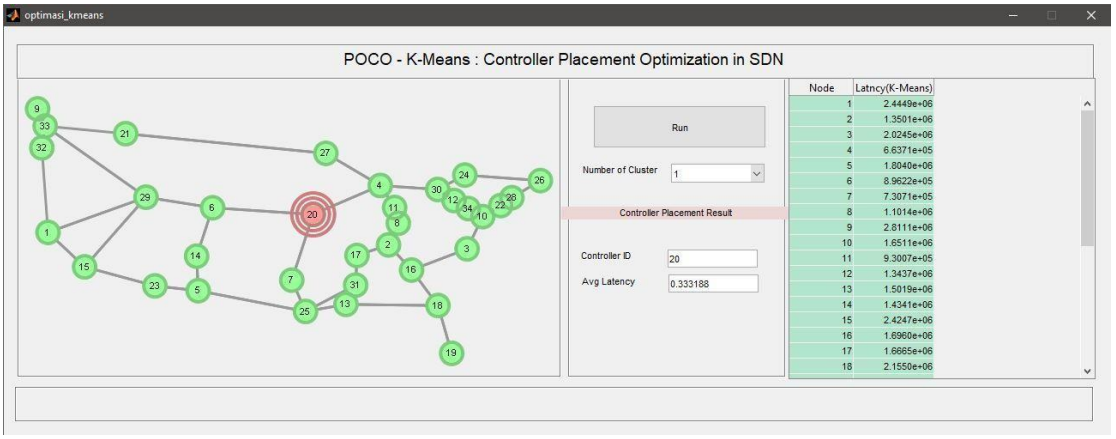
3.3 Pengukuran Latency

Dalam kasus SDN, *latency* antara *node* dan c*ontroller* sebanding dengan jarak antara *node* ke *controller*. Pengukuran *latency* menggunakan toolset POCO yang sudah umum digunakan untuk simulasi optimasi dalam lingkungan MATLAB. POCO merupakan proses pencarian ruang, yaitu melakukan percobaan semua kombinasi letak titik *controller*, selanjutnya menghitung *latency* semua titik dalam topologi. Pengukuran *latency* ini dilakukan untuk menentukan apakah letak *controller* yang sudah ditemukan layak untuk dijadikan sebagai letak *controller*.

4. Evaluasi

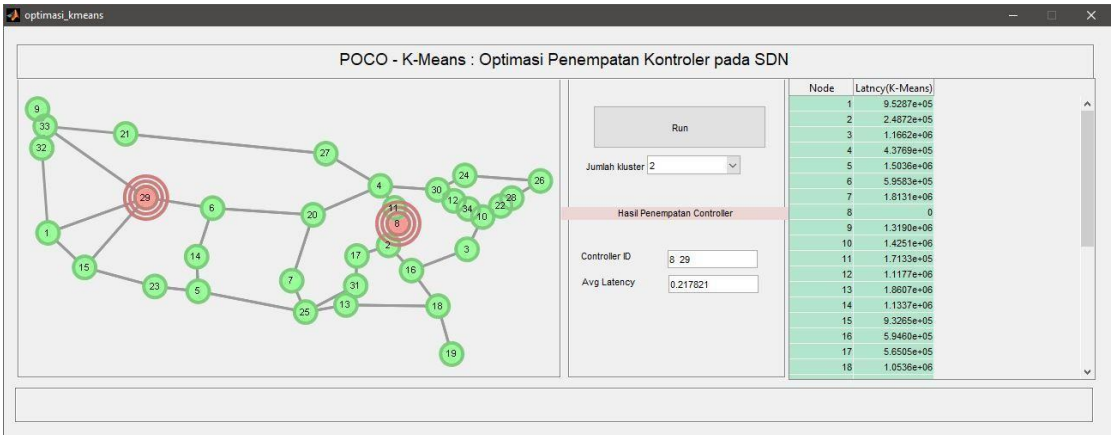
Pada bab-bab sebelumnya, telah dijelaskan mengenai teori-teori yang berkaitan, metodologi sistem, dan perancangan sistem. Pada bab ini, akan dijelaskan mengenai hasil yang diperoleh dalam pengujian beserta analisa pengujian. Pengujian ini untuk mengetahui apakah algoritma yang telah dipilih dapat menentukan letak controller yang optimal.

4.1 Hasil Pengujian



Gambar 4 Letak *Controller* dan *Average Latency* untuk Jumlah *Cluster* = 1

Pada gambar 4 dengan jumlah *cluster* = 1 didapatkan letak *controller* yang optimal untuk jaringan, yaitu pada node 20. Dengan *average latency* adalah 0.33ms, dan anggota *cluster* adalah semua *node* yang berada dalam jaringan.

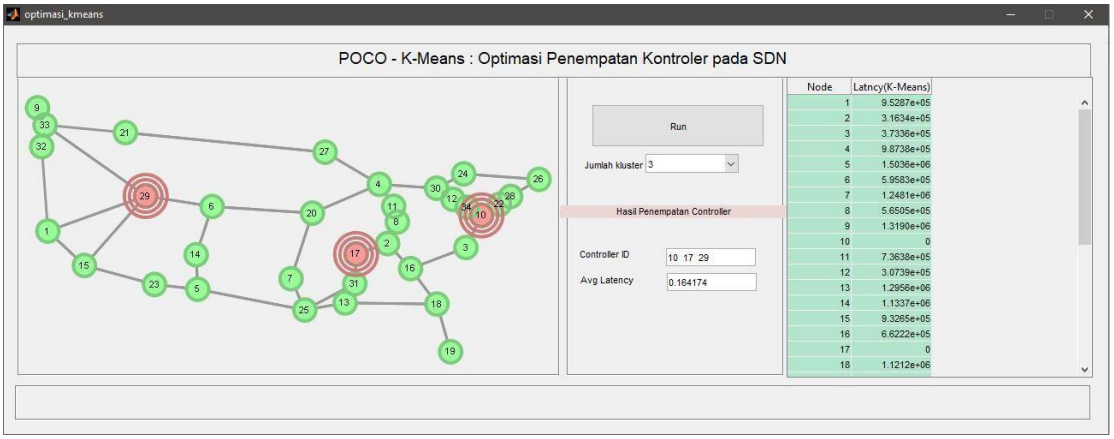


Gambar 5 Letak *Controller* dan *Average Latency* untuk Jumlah *Cluster* = 2

Pada gambar 5 dengan jumlah *cluster* = 2 didapatkan letak *controller* yang optimal untuk jaringan yaitu pada *node* 8 dan *node* 29. Dengan *average latency* = 0.22ms, dan SSE = 278.3089. Untuk anggota *cluster* dapat dilihat pada tabel 1.

Table 1 Anggota setiap *cluster* dari *node* 8 dan *node* 29

node 8	*node* 29
2, 3, 4, 7, 10	1, 5, 6, 9, 14
11, 12, 13, 16, 17	15, 21, 23, 29
18, 19, 20, 22, 24	32, 33
25, 26, 27, 28, 30	
31, 34	

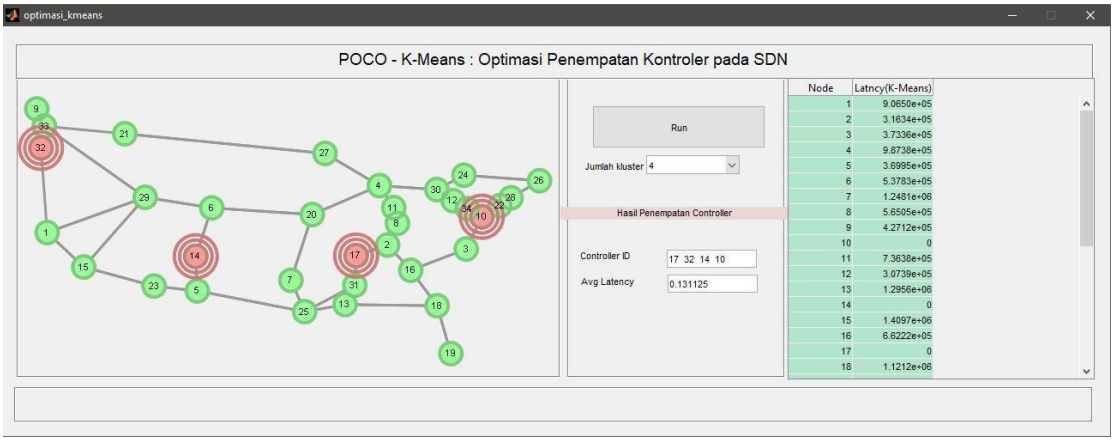


Gambar 6 Letak *Controller* dan *Average Latency* untuk Jumlah *Cluster* = 3

Pada gambar 6 dengan jumlah *cluster* = 3 didapatkan letak *controller* yang optimal untuk jaringan yaitu pada *node* 10, *node* 17, dan *node* 29. Dengan *average latency* = 0.17ms, dan SSE = 235.0761. Untuk anggota *cluster* dapat dilihat pada tabel 2.

Table 2 Anggota setiap *cluster* dari *node* 10, *node* 17, dan *node* 29

node 10	*node* 17	*node 29*
3, 12, 22, 24, 26	2, 4, 7, 8, 11	1, 5, 6, 9, 14
28, 30, 34	13, 16, 17, 18, 19	15, 21, 23, 32, 33
	20, 25, 27, 31	



Gambar 7 Letak *Controller* dan *Average Latency* untuk Jumlah *Cluster* = 4

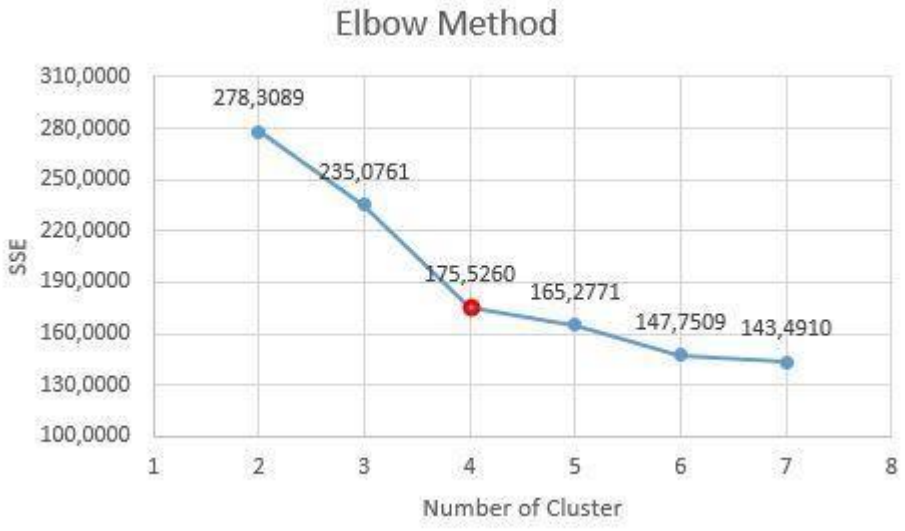
Pada gambar 7 dengan jumlah *cluster* = 4 didapatkan letak *controller* yang optimal untuk jaringan yaitu pada *node* 10, *node* 14, *node* 17, dan *node* 32. Dengan *average latency* = 0.13ms, dan SSE = 175.5260. Untuk anggota *cluster* dapat dilihat pada tabel 3.

Table 3 Anggota setiap *cluster* dari *node* 10, *node* 14, *node* 17, dan *node* 32

node 10	*node* 14	*node 17*	*node 32*
3, 12, 22, 24, 26,	5, 6, 14, 15, 23	2, 4, 7, 8, 11	9, 21, 32, 33
28, 30, 34	29	13, 16, 17, 18, 19	
		20, 25, 27, 31	

4.2 Analisis Hasil Pengujian

Berdasarkan hasil pengujian diatas, dapat dilihat bahwa, semakin banyak jumlah *cluster* yang ada, nilai *latency* semakin kecil. Hal ini disebabkan karena *latency* berhubungan dengan jarak antara *client* dan *server*. Semakin banyak *cluster*, maka jarak antara *client* dan *server* semakin tersusun berdasarkan kelompok *cluster* yang didapatkan dari algoritma *K-Means*, sehingga mempersingkat *latency* antara *client* dan *server*. Dan jumlah *cluster* yang paling optimal digunakan adalah jumlah *cluster* = 4 dengan nilai SSE 175.5260 yang dapat dilihat pada gambar 8.



Gambar 8 Grafik Metode Elbow

5. Kesimpulan

Pada optimasi penempatan *controller* ini didapatkan letak *controller* yang akan ditempatkan pada jaringan. Berdasarkan hasil pengujian yang dilakukan metode POCO - K-Means dapat diterapkan pada penempatan *controller* sesuai dengan skenario dan topologi yang digunakan dalam penelitian ini. Selain itu, penggunaan jumlah *cluster* dapat ditentukan sesuai jumlah yang diperlukan. Dan metode Elbow digunakan untuk mendapatkan jumlah *cluster* yang optimal pada jaringan sehingga dapat mengoptimalkan biaya pada pembangunan sistem.

Untuk penelitian yang akan datang, K-Means sensitif pada pemilihan *centroid* awal sehingga disarankan untuk memilih *centroid* awal yang optimal berbasis median. Selain itu, parameter dapat ditambahkan dari segi beban *controller* atau biasa disebut dengan ketahanan, karena beban *controller* juga merupakan faktor penting yang dapat membantu kinerja jaringan pada saat terjadi kegagalan *controller*.

Daftar Pustaka

[1] Y. Jimenez, C. Cervello-Pastor, and A. J. Garcia, "On the controller placement for designing a distributed SDN control layer," in *2014 IFIP Networking Conference*, Trondheim, Norway, 2014, pp. 1–9.
[2] ESOA, "LATENCY IN COMMUNICATIONS NETWORKS." ESOA (EMEA SATELLITE OPERATORS ASSOCIATION).
[3] B. P. R. Killi and S. V. Rao, "Capacitated Next Controller Placement in Software Defined Networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, no. 3, pp. 514–527, Sep. 2017.
[4] L. Han, Z. Li, W. Liu, K. Dai, and W. Qu, "Minimum Control Latency of SDN Controller Placement," p. 6.

[5] sdxcentral, "What are SDN Controllers (or SDN Controllers Platforms) ?," *SDX Central*. .
[6] S.-K. Yoon, Z. Khalib, N. Yaakob, and A. Amir, "Controller Placement Algorithms in Software Defined Network - A Review of Trends and Challenges," *MATEC Web Conf.*, vol. 140, p. 01014, 2017.
[7] G. Wang, Y. Zhao, J. Huang, Q. Duan, and J. Li, "A K-means-based network partition algorithm for controller placement in software defined network," in *2016 IEEE International Conference on Communications (ICC)*, Kuala Lumpur, Malaysia, 2016, pp. 1–6.
[8] D. A. Popescu, N. Zilberman, and A. W. Moore, "Characterizing the impact of network latency on cloud-based applications' performance," p. 20.
[9] H. Kuang, Y. Qiu, R. Li, and X. Liu, "A Hierarchical K-Means Algorithm for Controller Placement in SDN-Based WAN Architecture," in *2018 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, Changsha, 2018, pp. 263–267.
[10] Amir Tjolleng, M.Sc., *Pengantar Pemrograman MATLAB*. PT Elex Media Komputindo, 2017.
[11] Dr. Suyanto, S.T., M.Sc., *Data Mining untuk Klasifikasi dan Klasterisasi Data*. Penerbit Informatika, 2017.
[12] Elly Muningsih, "OPTIMASI JUMLAH CLUSTER K-MEANS DENGAN METODE ELBOW UNTUK PEMETAAN PELANGGAN," pp. 105–114, Sep. 2017.
[13] Shruti Kapil, Meenu Chawla, and Mohd Dilshad Ansari, "On K-means Data Clustering Algorithm with Genetic Algorithm," *2016 Fourth Int. Conf. Parallel Distrib. Grid Comput. PDGC*, 2016.
[14] Arif Indra Irawan, Maya Rahayu, Fidyatun Nisa, and Nana Rachmana Syambas, "Network Migration to SDN Using Pareto Optimal Resilience Controller (POCO) : Case Study in the UPI Network," *IEEE*, 2015.
[15] S. I. Harned, "POCO-MOEA: Using Evolutionary Algorithms to Solve the Controller Placement Problem," p. 133.