Telecommunications, Networks, Electronics, and Computer Technologies

Vol.1 (2), 2021, 99-107

JURNAL UPI

TELNECT

Journal homepage: http://ejournal.upi.edu/index.php/TELNECT/

Simulasi Equal Cost MultiPath Selection (ECMP) pada Jaringan Berbasis Software Defined Networking

Banda Subagja[1], Nova Nurul Putri[2], Pratiwi Sanada[3], Tarisa Citra Dewi[4], Galura Muhammad Suranegara[5]*

[1,2,3,4] Program Studi Sistem Telekomunikasi, Universitas Pendidikan Indonesia, Indonesia

*Corresponding Author: E-mail: esanoerfadhila@upi.edu

ABSTRACTS

Software Defined Networking adalah arsitektur jaringan yang memisahkan control plane dan data plane. Penelitian ini bertujuan untuk mendistribusikan lalu lintas jaringan komputer secara seimbang dengan beberapa jalur agar lalu lintas jaringan komputer berjalan secara optimal dengan biaya yang sama menggunakan Equal Cost Multpath Routing. Dalam penelitian ini metode yang digunakan adalah melakukan uji coba, menggunakan tiga bandwidth yang berbeda dan parameter yang diuji adalah delay dan packet rate. Hasil pengujian delay terkecil diperoleh bandwidth 50 Mbps dengan selisih besar delay 2,19% dengan bandwidth 100 Mbps dan selisih besar delay 10,02% dengan bandwidth 150 Mbps. Pengujian packet rate terbesar diperoleh pada bandwidth 50 Mbps dengan perbedaan besar pada packet rate sebesar 0,30% dengan bandwidth 100 Mbps dan perbedaan besar pada packet rate sebesar 9,19% dengan bandwidth 150 Mbps. Delay terkecil diperoleh bandwidth 50 Mbps disebabkan jalur transfer data berjalan lebih lancar karena tidak ada banyak waktu akumulasi lalu lintas pengiriman. Packet Rate tertinggi diperoleh bandwidth 50 Mbps karena delay yang didapat lebih kecil dari 100 Mbps dan 150 Mbps sehingga tingkat paket yang dihasilkan akan lebih besar dari 100 Mbps dan 150 Mbps. Diharapkan hasil dari penelitian ini dapat dijadikan gambaran untuk pembaca maupun teknisi untuk merancang topologi sebelum membangun jaringan.

ARTICLE INFO

Article History:
Received 15 Oktober 2021
Revised 22 November 2021
Accepted 11 Desember 2021
Available online 15 Desember 2021

Keyword:
Control Plane, Data Plane, Controller, SDN, Delay, Packet Rate, Bandwith

1. INTRODUCTION

Equal Cost Multipath Routing (ECMP) merupakan proses pemilihan rute yang menggunakan segala opsi jalur yang ada antara satu node dengan node yang lain dalam jaringan [1]. *Controller* adalah bagian dari perangkat jaringan komputer yang akan menentukan bagaimana perangkat merespon aliran atau paket yang datang. Sedangkan *data-plane* adalah bagian yang akan melakukan *forwarding outflow* fungsi. Komponen kunci dalam SDN adalah *controller*, di mana *controller* sebagai otak *Software Defined Networking* yang dirancang untuk mempermudah dan mengatur jaringan. Pada infrastruktur jaringan konvensional, konfigurasi, pemecahan masalah, dan proses implementasi jaringan melibatkan administrator sehingga kurang efisien karena tidak adanya pemisahan keputusan.

Software Defined Networking (SDN) dengan membuat bidang kontrol terpusat, SDN menyediakan jaringan yang lebih mudah dikelola, manajemen yang fleksibel, pengelolaan keamanan yang mudah, optimasi sumber energi, serta pengaturan jaringan dapat dikelola sendiri tanpa menunggu persetujuan administrator untuk optimasi jaringan. Pada Simulasi Defined Networking (SDN), tiap *switch* tersambung dengan *controller* serta isi dari *routing table* pada *switch-switch* tersebut harus ditetapkan oleh *controller* [1].

Analisis percobaan menggunakan pengukuran parameter-parameter Quality of Service (QoS) diantaranya *delay* dan *packet rate* ECMP berbasis SDN dengan skema yang diajukan pada saat pengiriman *traffic-traffic* yang dapat memicu terjadinya tabrakan data.

2. MATERIALS AND METHOD

Penelitian ini bertujuan untuk mengetahui *traffic* jaringan komputer secara seimbang dengan beberapa jalur agar *traffic* jaringan komputer berjalan secara optimal melalui Equal Cost Multipath Routing (ECMP). Pada mininet dengan uji coba sebanyak 10 kali menggunakan tiga bandwidth yang berbeda, yaitu 50 Mbps, 100 Mbps, dan 150 Mbps. Parameter yang diuji adalah delay dan packet rate dengan mengirimkan 1000 paket dengan ukuran tiap paket 1 Kb selama 60 detik.

2.1 SDN

SDN merupakan sebuah arsitektur baru dalam dunia jaringan yang berfungsi untuk memisahkan *control plane* dan *informasi plane*, baik secara logical maupun secara physical. *Control Plane* merupakan bagian yang paling penting karena berperan sebagai pusat dari sebuah jaringan melakukan proses kalkulasi untuk fungsi seperti *routing, load balancer, intrusion detection system, topology learning,* dan lain sebagainya [3].

2.2 Controller

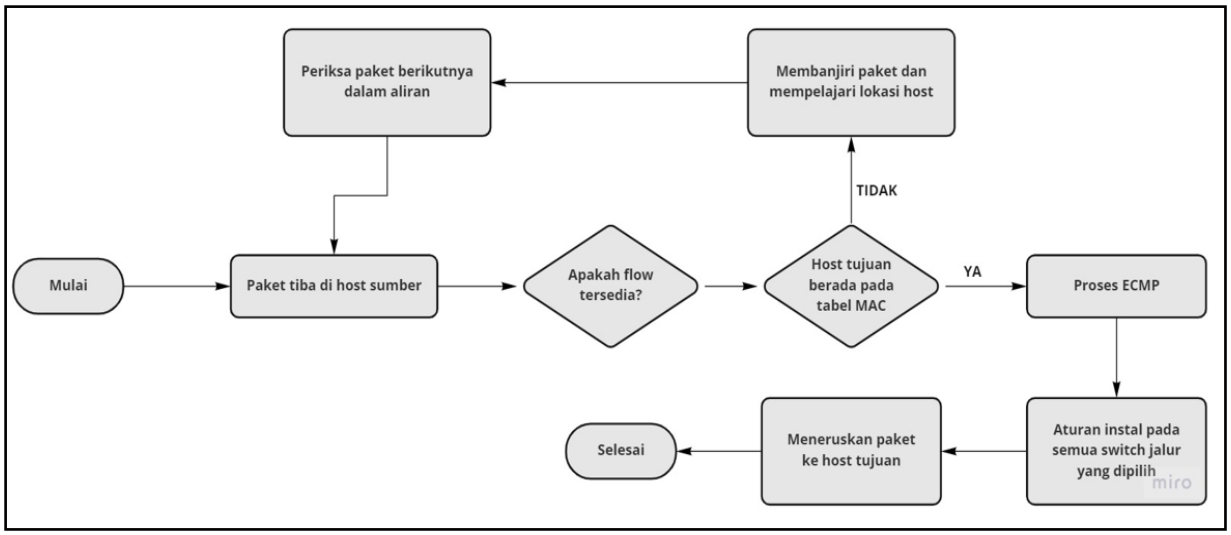
Controller yang digunakan akan digunakan pada percobaan ini adalah RYU *controller.* RYU merupakan implementasi python *open source* dari SDN *controller* dengan arsitektur berbasis komponen [4]. *Controller* ini menggunakan pemrograman bahasa python yang mudah digunakan dan memiliki lisensi administrasi untuk memudahkan penggunanya selain itu, RYU mendukung berbagai protokol perangkat lunak yang ditentukan oleh jaringan *OpenFlow*. *Controller* RYU juga dapat mendukung komponen lama sesuai kebutuhan dan diterapkan ke elemen baru dan dapat memfasilitasi fitur untuk membangun jaringan [3].

Namun, prosesnya sebagian besar masih rentan terhadap gangguan. Program pengguna dieksekusi dengan konteks proses desain, karena dapat merusak tabel aliran di *switch* dengan mengirimkan pesan paket masuk dan pesan paket keluar (*flow mod*). *Controller* RYU tidak memiliki mekanisme yang dapat digunakan untuk memeriksa otorisasi pengguna atau aplikasi sebelum mengirim pesan yang disebutkan di atas untuk beralih [4].

2.3 ECMP

Equal Cost Multipath Routing (ECMP) merupakan salah satu teknik *routing* yang digunakan untuk memastikan seluruh jalur yang tersedia dengan *cost* sepadan antara satu node dengan node lainnya. Pada penerapan ECMP, setiap *router* akan menggunakan *hash function* untuk setiap *header trafik* seperti IP pengirim dan IP tujuan untuk memetakan setiap trafik ke beberapa jalur yang tersedia [5].

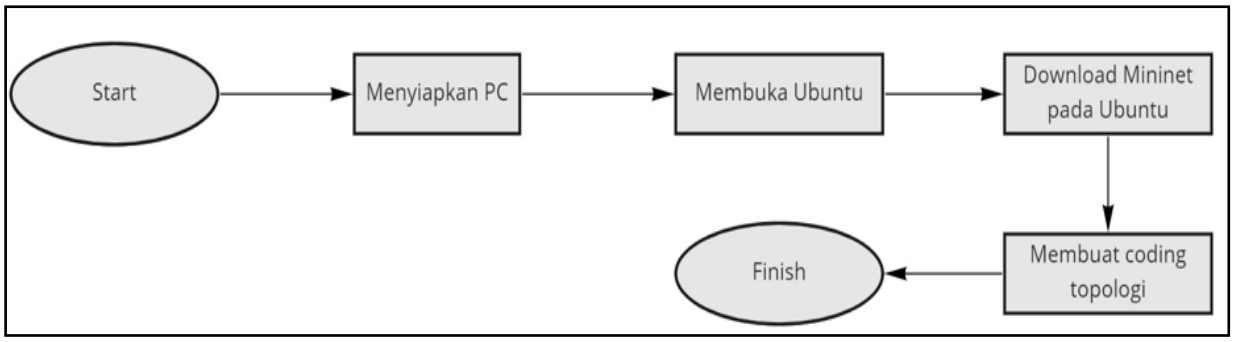
Dikarenakan jumlah jalur yang terbatas dalam sebuah topologi, sangat memungkinkan dua atau lebih topologi dipasangkan dengan jalur - jalur yang memiliki link beririsan. Ketika trafik - trafik dikirimkan pada saat yang bersamaan, maka kongesti akan terjadi. Jalur-jalur lain yang seharusnya bisa digunakan untuk menghindari kongesti, tidak dipilih karena sudah dipasangkan dengan kemungkinan trafik lain oleh *hash function* [5].



Gambar 1. Diagram Alir Proses ECMP

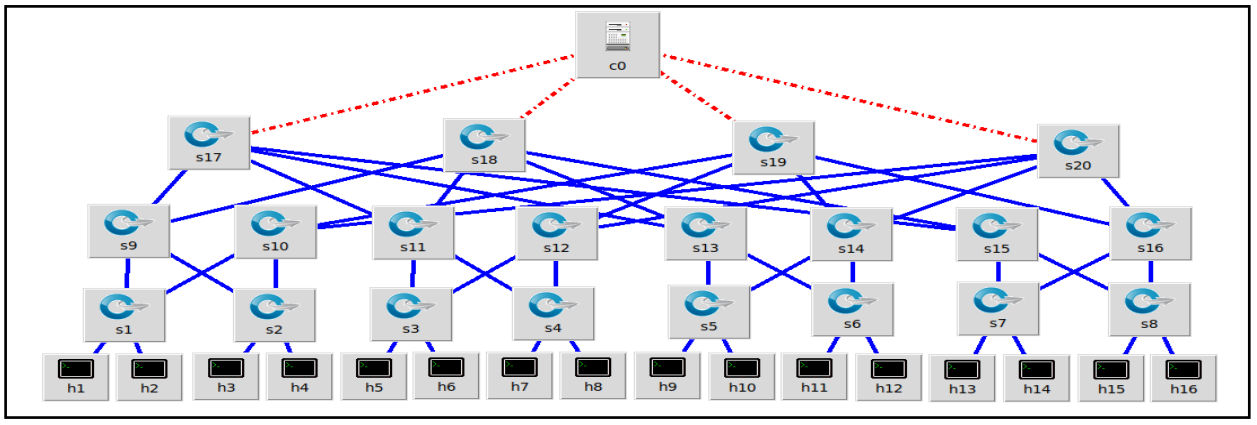
Flowchart ini menjelaskan bahwa di dalam arsitektur sistem jaringan yang ditetapkan fitur lunak yang dibuat, terdapat *host*. *Host* pada sistem ini bisa berbentuk pc klien, server, ataupun fitur pengguna akhir lainnya. *Host* akan memperoleh Alamat IP dinamis yang diresmikan oleh DHCP (*Dynamic Host Configuration Protocol),* yang dijalankan oleh *controller. Switch* dalam sistem ini mempunyai posisi sebagai *relay* antara *host* serta *controller* pada DHCP *discover- offer- request*. Alamat IP pada sistem ini hanya mempunyai satu *subnet* dalam satu jaringan. Dalam flowchart diatas dimulai dari menjalankan topologi, lalu mengirimkan paket sampai tiba di *host* sumber dan jika *flow* tersedia maka *host* tujuan ada di tabel mac dan dapat melanjutkan ke proses ECMP lalu membuat aturan install pada semua *switch* jalur yang dipilih lalu meneruskan paket ke *host* tujuan lalu selesai, tetapi jika pada *flow* tidak tersedia maka akan membanjiri paket dan mempelajari kembali lokasi *host.* Paket akan kembali diperiksa dalam aliran kemudian kembali ke step proses ECMP.

2.4 Method



Gambar 2. Diagram Alir Proses Method

Pada flowchart diatas menjelaskan untuk melakukan percobaan yaitu diawali dengan membuka ubuntu lalu install mininet jika mininet sudah terinstal memasukan codingan topologi. Lalu uji coba dan melihat hasil parameternya.



Gambar 3. Topologi Jaringan ECMP

Pada gambar diatas merupakan topologi jaringan ECMP dengan 16 *host*, 8 switch yang terhubung ke host, dan 12 switch yang terhubung antar switch. Topologi ini dibangun dengan Bahasa pemrograman Python dan dijalankan melalui aplikasi mininet pada sistem operasi linux ubuntu [1]. Banyaknya jalur pada topologi yang digunakan berguna untuk mengetahui rute paket yang dikirim dari satu host ke host lain untuk mendapatkan hasil yang efisien dari delay dan packet rate. Pada arsitektur konvensional, aktivitas pengiriman data pada jaringan yang banyak menyebabkan terjadinya tabrakan data, sehingga data bisa hilang atau paket tidak terkirim kepada host penerima [1].

3. RESULTS AND DISCUSSION

3.1 Result

TABEL 1. Hasil Pengujian Delay

Percobaan	Delay (s)		
	50 Mbps	100 Mbps	150 Mbps
1	13.9	14.23	15.31
2	13.53	14.31	15.34
3	13.34	14.31	15.37
4	13.82	14.43	15.23
5	13.73	14.33	15.45
6	13.69	14.53	15.43
7	13.97	14.42	15.32
8	13.79	14.34	15.52
9	14.11	14.54	15.42
10	13.73	14.34	15.42

Pengujian *delay* dilakukan sebanyak 10 kali, dapat dilihat pada Tabel 1. Pengujian dilakukan dengan cara mengirimkan 3 *traffic* data yang berbeda yaitu 50 Mbps, 100 Mbps, dan 150 Mbps. Setiap *trafik* mengirimkan 1000 paket setiap detik dengan ukuran setiap paket 1 Kb selama 60 detik.

TABEL 2. Hasil Pengujian *Packet Rate*

Percobaan	Packet Rate (Pkt/s)		
	50 Mbps	100 Mbps	150 Mbps
1	79	14.82	15.13
2	79	14.92	15.15
3	80.93	14.43	15.52
4	80.73	14.41	15.56
5	79	14.42	15.63
6	79	14.33	15.26
7	79	14.53	15.72
8	79	14.63	15.74
9	79	14.26	15.47
10	79	14.61	15.23

Pengujian *packet rate* dilakukan sebanyak 10 kali, dapat dilihat pada Tabel 2. Pengujian dilakukan dengan cara mengirimkan 3 buah *traffic* data berukuran 50 Mbps, 100 Mbps, dan 150 Mbps. Setiap *trafik* mengirimkan 1000 paket setiap detik dengan ukuran setiap paket 1 Kb selama 60 detik.

3.2 Disscussion

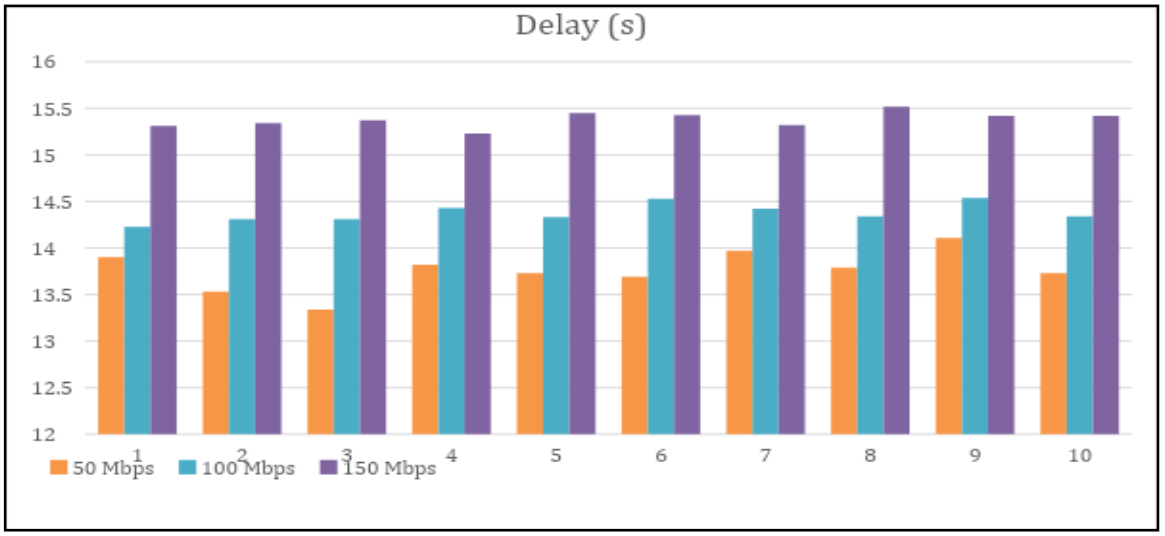
1. Delay

Delay adalah waktu tunda yang terjadi pada suatu paket untuk menempuh jarak dari proses mengirimnya paket hingga terkirimnya paket [7]. Satuan *delay* adalah ms dengan rumus sebagai berikut.

$$Delay = waktu\ penerimaan\ paket - waktu\ pengiriman\ paket$$

Dengan demikian, delay merupakan selisih waktu penerimaan dan pengiriman paket. Rumus rata-rata delay adalah sebagai berikut [7].

$$Rata - rata\ Delay = \frac{total\ delay}{total\ paket\ yang\ diterima}$$

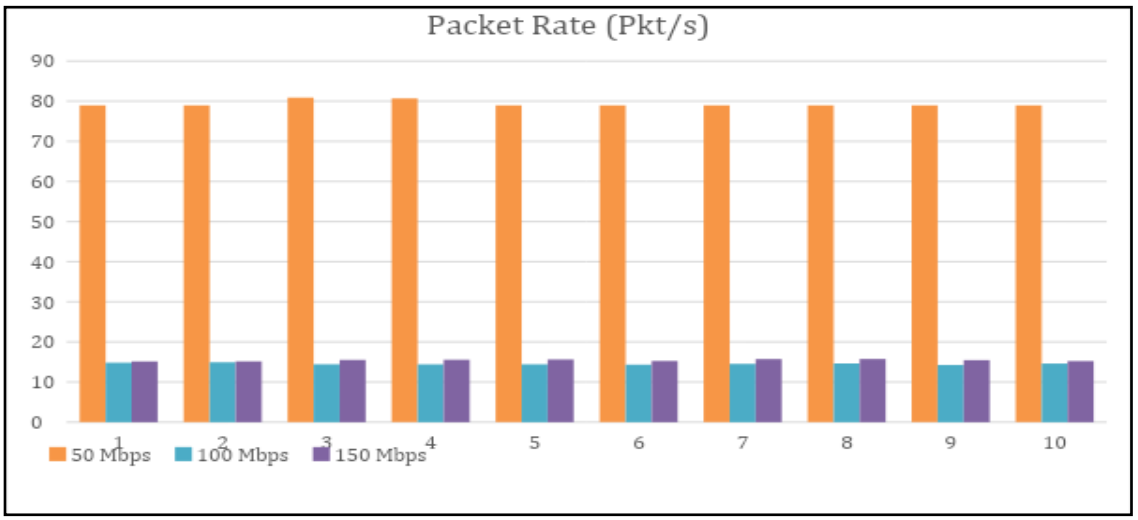


Gambar 4. Grafik Hasil Pengujian Delay

Pada grafik diatas terlihat perbedaan antara pengujian dengan *traffic* data 50 Mbps, 100 Mbps, dan 150 Mbps. Delay terendah dalam percobaan ini diperoleh bandwidth 50 Mbps. Dibandingkan dengan bandwidth 100 Mbps dan bandwidth 150 Mbps, bandwidth 50 Mbps memiliki jalur transfer data yang berjalan lebih lancar karena tidak banyak waktu akumulasi *traffic* pengiriman. Sedangkan bandwidth 100 Mbps memiliki jalur transfer data yang berjalan cukup lancar, dan bandwidth 150 Mbps memiliki jalur transfer data yang berjalan kurang lancar.

2. Packet Rate

Paket rate adalah nilai kecepatan paket yang dikirim per satuan waktu. Setiap paket yang dikirim berbeda-beda kecepatannya tergantung dari topologi dan bandwidth yang diberikan.



Gambar 5. Grafik Hasil Pengujian Packet Rate

Pada grafik diatas terlihat perbedaan antara pengujian dengan *traffic* data 50 Mbps, 100 Mbps, dan 150 Mbps. Tingkat *packet rate* tertinggi dalam percobaan ini diperoleh bandwidth 50 Mbps. Karena *delay* yang didapat lebih kecil dari 100 Mbps bandwidth dan 150 Mbps bandwidth maka tingkat paket yang dihasilkan akan lebih besar dari 100 Mbps dan bandwidth 150 Mbps. Sedangkan *packet rate* dengan bandwidth 100 Mbps mendapatkan *delay* yang cukup lancar sehingga packet rate yang didapat cukup tinggi, dan *packet rate* dengan bandwidth 150 Mbps mendapatkan *delay* yang kurang lancar sehingga *packet rate* yang didapat kurang tinggi.

4. CONCLUSION

Simulasi topologi ECMP dengan 16 *host*, 8 *switch* yang terhubung ke *host*, dan 12 *switch* yang terhubung antar *switch* menggunakan *controller* Ryu telah dilakukan. Melalui pengujian yang telah dilakukan, dapat disimpulkan bahwa penggunaan Ryu sebagai *controller* SDN dengan bandwidth 50 Mbps menghasilkan delay paling baik dengan perbedaan besar delay yaitu 2,19 % dari bandwidth 100 Mbps dan selisih besar delay 10,2% dari bandwidth 150 Mbps. Pada hasil pengukuran *packet* rate dengan bandwidth 50 Mbps menghasilkan *packet rate* paling baik dengan perbedaan sebesar 0,3% dari bandwidth 100 Mbps dan selisih *packet rate* 9,19% dari bandwidth 150 Mbps. Sehingga delay dan *packet rate* diantara skema bandwidth 50 Mbps, 100 Mbps, dan 150 Mbps didapat bandwidth terbaik dan efisien adalah bandwidth 50 Mbps pada penelitian ini, karena mendapatkan nilai *delay* paling rendah dan *packet rate* paling besar sehingga paket dikirim secara cepat dan stabil.

5. REFERENCES

[1] H. Endah Wahanani, M. Idhom, E. P. Mandyartha, “Equal Cost Multipath RYU Controller Analysis in Software Defined Networking,” 2020.
[2] N. McKeown , T. Anderson, H. Balakrishnan, G. Parukar, L. Peterson, J. Rexford, S. Shenker and J. Turner, “OpenFlow : Enabling Innovation in Campus Network,” vol. 38, no. 2, pp. 69-74, 2008.
[3] R. Dewanto, R. Munadi, R. M Negara, “Minimalization of Overlopped Path Selection on Equal Cost Multipath Routing (ECMP),” 2018.[4]ec
[4] M. Al-Fares, S. Radhakrishnan, B.Raghavan, “Hedera Dynamic Trafik Scheduling for Data Center Networks,” 2019. [2]SDN
[5] Arbettu, R.K, “Security Analysis of Open Daylight, ONOS, Rosemary and RYU SDN Controllers,” September 2016.[3]
[6] S. Asadollahi, B. Goswami, “Implementation of SDN Using Open Daylight Controller",” 2017.
[7] S. Asadollahi, B. Goswami, “Software Defined System Network,Controller Comparison,” vol. 5, 2017.
[8] Al, Gude, “NOX : Towards an Operating System for Network,” vol. 38, no. 33, pp. 105-110.
[9] M. Chiesa, G. Kindler, and M. Schapira , “Trafic Engineering With Equal-Cost_Multipath : An Communication Perspective,” vol. 25, no. 2, pp. 779-792, April 2017.
[10] D. Erickson, “The Beacon OpenFlow Controller,” 2013.
[11] P. Floodlight, “ Project Floodlight,” 2012. [Online]. Available: http://floodlight.openflowhub.org/. [Accessed 20 November 2021].
[12] A. Martins, “Critical Ethernet Base On Openflow,” 2014.
[13] M.McCauley, “POX,” 2012. [Online]. Available: http://www.noxrepo.org/, . [Accessed 30 Oktober 2021].