ISSN : 2085-3688; e-ISSN : 2460-0997

Analisis Performansi Perutingan *Link State* Menggunakan Algoritma *Djikstra* Pada Platform *Software Defined Network* (SDN)

Abu Riza Sudiyatmoko[1], Sofia Naning Hertiana[2], Ridha Muldina Negara[3]
[1,2,3] Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[1,2,3]Jl. Telekomunikasi No. 01, Terusan Buah Batu, Bandung, Jawa Barat 40257
Email Korespondensi : sudiyatmoko92@gmail.com

Dikirim 13 Februari 2016, Diperbaiki 16 April 2016, Diterima 21 April 2016

Abstrak – *Software Defined Network* (SDN) merupakan paradigma baru dalam sistem jaringan. Konsep dasar yang diusung oleh SDN adalah pemisahan antara layer *control* dan *forward* dalam perangkat yang berbeda. Konsep inilah yang menjadi perbedaan anatar SDN dan jaringan konvensional. Selain itu SDN memberikan konsep *network topology virtualisation* dan memungkinkan administrator untuk melakukan *customize* pada *control plane*. Dengan diterapkannya protokol *OpenFlow* pada SDN maka terdapat peluang untuk menerapkan perutingan *flow based* pada jaringan SDN dalam pendistribusian data dari *source* sampai ke *destination. Link state* IS-IS merupakan protokol routing yang menggunakan algoritma djikstra untuk menentukan jalur terbaik dalam pendistribusian paket. Dalam penelitian ini dilakukan analisis terhadap implementasi *Link State* IS-IS pada *paltform* SDN dengan menggunakan arsitektur *RouteFlow*. Parameter yang digunakan adalah *throughput*, *delay*, *jitter* dan *packet* loss serta performansi perangkat *controller*. Hasil pengujian pada kondisi overload yaitu dengan *background traffic* 125 Mb nilai *packet loss* mencapai 1,23%, nilai *throughput* 47,6 Mbp/s dan *jitter* 2.012 ms. Nilai *delay* terbesar adalah pada topology 11 *switch* 11 *host* yaitu berkisar diangka 553 ms. Sedangkan performansi perangkat controller dengan konsumsi *memory* pada saat menjalankan mengontrol jaringan berkisar diantara 25,638% sampai 39,04%.

Kata kunci - *Software Defined Network; OpenFlow; RouteFlow; QoS; Linkstate; IS-IS, Background Traffic*

Abstract - Software Defined Network (SDN) is a new paradigm in network system. The basic consept used by SDN is separation beetwen control and forward layer. This concept makes SDN different from todays trditional network. In addition SDN provides network topology virtualization concept and allows administrators to customize the control plane [1]. With the implementation of OpenFlow protocol on SDN, then there is an opportunity to apply a routing in the SDN network from source to destination. Link State IS-IS is a routing protocol use djikstra alghoritm to determine the best path. This research analyse the implementation of Link State IS-IS use RouteFlow architecture. Some parameter are used such as throughput, delay, jitter and packet loss also controller device performance. Results of the analysis showed that based on the above parameters , the IS- IS link state can be implemented on the platform SDN. The result in overload condition is with 125 Mb background traffic, packet loss value reach 1.23% and throughput is 47.6 MB/s and jitter is 2.012ms. The biggest value of delay is happened when network use 11 switch and 11 host topology. In that condition delay value reach 553 ms. While controller device performance consume the memory between 25.638% until 39.04% when it run the topologies.

Keywords - Software Defined Network; OpenFlow; RouteFlow; QoS; Linkstate; IS-IS, Background Traffic

I. PENDAHULUAN

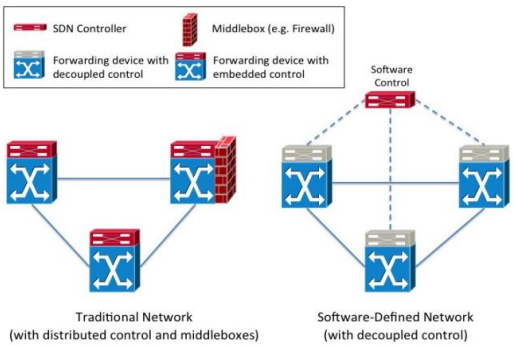
Kondisi jaringan yang mulai jenuh menjadi alasan mulai banyaknya penelitian dan percobaan *platform* SDN (*Software defined network*) yang merupakan salah satu evolusi teknologi jaringan sesuai dengan tuntutan yang berkembang. Dibandingkan dengan jaringan konvensional, *Software defined networking* (SDN) memberikan kemudahan kepada pengguna dalam mengembangkan aplikasi pengontrol jaringan dengan memisahkan fungsi *data plane* dari *control plane*. Pemisahan ini juga memudahkan administrator mengontrol secara langsung paket yang berjalan melalui jaringan [5].

SDN mengusung sebuah paradigma baru dalam jaringan yang awalnya bersifat terdistribusi dan cenderung tertutup menjadi jaringan yang bersifat terpusat, *programmable* dan terbuka (*open source*). Dengan demikian memungkinkan seorang peneliti

atau bahkan administrator untuk mengatur jaringan sesuai dengan kebutuhan. Tidak harus terpaku pada *protocol*, API dan standar yang ditetapkan oleh para vendor terhadap *device*/perangkat mereka. SDN memberikan konsep *network topology virtualisation*dan memungkinkan *administrator* untuk melakukan *customize* pada *control plane* [1].

Dikutip dari *white paper* yang diterbitkan oleh *Open Network Foundation* yang berjudul "*Software defined networking : The New Norm for Networks*" [6], *software defined networking* (SDN) merupakan sebuah arsitektur jaringan yang sedang berkembang di mana *control plane* dipisahkan dari *forwarding plane* dan dapat secara langsung diprogram. Pemindahahan pengendali ,yang sebelumnya terikat pada setiap perangkat jaringan, menjadi dapat diakses oleh perangkat komputasi memungkinkan infrastruktur jaringan dapat diringkas sebagai aplikasi dan layanan jaringan, yang dapat memperlakukan jaringan sebagai sebuah entitas logika atau maya.

Dengan pemisahan fungsi DP (*Datapath Plane*) dan CP (*Control Plane*), maka dibutuhkan suatu protokol yang berguna untuk menghubungkan dua elemen tersebut. Saat ini terdapat beberapa protokol yang dikembangkan oleh para pengembang antara lain : *OpenFlow* dan ForCES. *OpenFlow* merupakan suatu kerangka kerja yang dikembangkan oleh Universitas Stanford kemudian pada periode selanjutnya dikelola oleh *Open Network Foundation* (ONF). Sementara itu ForCES dikembangkan oleh grup riset IETF yang mendefinisikan sebuah arsitektur yang terdiri atas protokol transport dan model. ForCES memberikan API yang berguna untuk memisahkan elemen *forwarding* dan *control* [8]. Perbedaan mendasar pada jaringan SDN dengan tradisional adalah penempatan fungsi *control* dan *forward.* Pada jaringan tradisional fungsi *control* dan *forward* ditempatkan pada *device* yang sama yaitu *router*. Sedangkan pada jaringan SDN fungsi *control* ditempatkan pada software terpusat (*controller*) dan fungsi *forward* ditempatkan pada suatu perangkat kosong berupa *switch* (*forwarding decive*).



Gambar 1. Perbedaan SDN dengan Jaringan Tradisional [5]

Sementara untuk elemen *control*, terdapat beberapa jenis *controller* yang saat ini berkembang, antara lain [7].

- Ryu
- POX
- Opendaylight

Tabel 1. Perbedaan Controllers SDN [7]

	Opendaylight	**POX**	**Ryu**
Bahasa	Java	Python	Python
GUI	Tersedia	Tersedia	Tersedia
Versi OF	v.1.0	v.1.0	v.1.0, v.1.1, v.1.2, v.1.3
REST API	Tersedia	Tidak Tersedia	Tersedia
Platform	Linux, MAC, Windows	Linux, MAC, Windows	Linux

POX dan Ryu merupakan perangkat lunak *controller* bersumber terbuka yang menggunakan bahasa python sementara *Opendaylight* menggunakan bahasa java. *Opendaylight* merupakan proyek yang memiliki dukungan yang cukup luas, proyek ini didukung oleh beberapa perusahaan antara lain : Cisco, Brocade, IBM. [7]

Penelitian ini menggunakan emulator *mininet* dan *RouteFlow* untuk merancang topologi jaringan dan sebuah protokol pada kontroler (*POX*) dalam jaringan SDN. Penggunaan *POX* dilakukan karena merupakan perangkat lunak bersumber terbuka yang menyediakan jalan untuk mengimplementasikan protokol *OpenFlow*. *POX* memberikan kemudahan pengguna untuk membuat program yang fungsinya berbeda-beda.

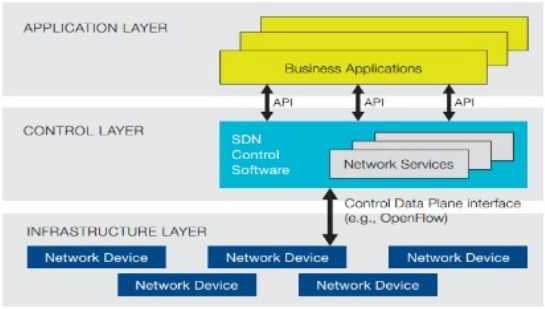
Pada penelitian sebelumnya yang berjudul "On The Resilence of Software Defined Routing Platform" [9][10], telah dilakukan penerapan OSPF pada jaringan SDN dan dihasilkan nilai *failover time* yang lebih cepat dibandingkan pada jaringan tradisional. Namun belum dilakukan penelitian mengenai pengaruh protokol routing lainnya yaitu *Linkstate Intemediate System Intermediate System (IS-IS)* pada SDN. Oleh karena itu pada penelitian ini dilakukan analisis apakah *protocol routing Linkstate IS-IS* dapat diterapkan pada *platform* SDN berdasarkan parameter *throughput*, *delay*, *jitter* dan *packet* loss serta performansi perangkat *controller*.

II. METODOLOGI PENELITIAN

A. Software Defined Network (SDN)

Software Defined Networking (SDN) adalah arsitektur jaringan yang muncul di mana kontrol jaringan dipisahkan dari forwarding dan dapat diprogram secara langsung [2]. Migrasi dari *control logic* yang digunakan pada *networking devices* misalkan *ethernet switches* yang bersifat *tightly*

integrated menjadi jaringan yang *accessible* dan *logically centralized controller*, sehingga mampu menyediakan jaringan yang *fleksibel*, *programable*, *vendor-agnostic*, *cost efficient*, dan arsitektur jaringan yang lebih inovatif.

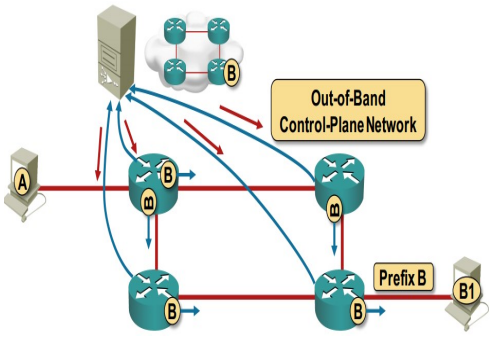


Gambar 2. Arsitektur SDN [2]

Pada Gambar 2, Arsitektur SDN membagi jaringan menjadi 3 layer yaitu *application layer, control layer* dan *insfrastructure*/data layer. *Application layer* merupakan *interface* terhadap seorang admin atau peneliti dalam mengelola atau mengembangkan jaringan SDN. *Control plane berisi* suatu *controller* bersifat terpusat dan *based on software*. *Subordinate hardware* dikontrol sepenuhnya oleh *controlling plane* atau *controller* dalam melakukan *forwarding decission*. Semua *subordinate* terhubung ke *controller*.

B. *Protokol Openflow*

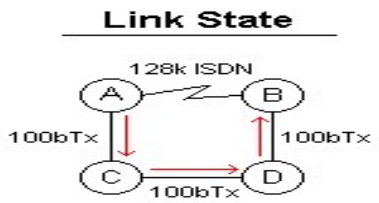
Openflow adalah sebuah *control interface* yang memungkinkan untuk memprogram *switch* pada *data plan*. Sehingga administrator dapat mengontrol secara langsung lalu lintas paket pada *forward plan* atau *data plan* melalui *interface OpenFlow* ini. OpenFlow mendefinisikan infrastruktur *flow-based forwarding* dan *Application Programmatic interface* (API) standar yang memungkinkan *controller* untuk mengarahkan fungsi dari *switch* melalui saluran yang aman (*secure chanel*) [2]. Pada Gambar 3 dapat dilihat bahwa *controller* yang telah diprogram sesuai dengan konfigurasi jaringan yang diinginkan dapat terhubung dengan *forward plane*. Dimana *scure channel* ini merupakan perantara bagi *controller* untuk mengontrol jaringan pada *forward plane*.



Gambar 3. OpenFlow [11]

C. *Link State Intermediate System Intermediate System (IS-IS)*

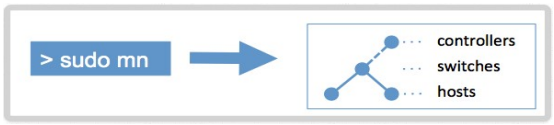
Intermediate System Intermediate System adalah sebuah protokol *International Organization for Standarization* (ISO) untuk *Connectionless-mode Network Service* (CLNS), protokol network layer dari *Operating System Interconnection* (OSI). ISIS menggunakan ISO *Packet Data Unit* (PDU) untuk saling berkomunikasi antar *router*. Beberapa tipe PDU yang digunakan oleh ISIS adalah *Hello* PDU, *Link State* PDU (LSP), dan *Sequence Number* PDU (SNP). *Hello* PDU digunakan oleh *router* yang berdampingan satu sama lain, selain itu *router* juga dapat menjelajah *router* baru dalam jaringan serta menetapkan *router* yang tidak lagi tercapai. *Router* ISIS saling bertukar informasi melalui *Link State* PDU (LSP) untu membangun dan mempertahankan *data base link state*. Adapun cara kerja Link State dalam memilih jalur dapat dilihat pada Gambar 4.



Gambar 4. *Routing Link State* [12]

D. Emulator *(mininet)*

Emulator yang digunakan dalam penelitian ini adalah *mininet*. *Mininet* adalah suatu *software emulator* yang memungkinkan untuk melakukan *prototyping* pada jaringan yang luas dengan hanya menggunakan satu mesin [3]. Pada *mininet* ini dilakukan perancangan jaringan dengan topologi yang diinginkan. Secara sederhana mininet ini berfungsi untuk emulasi pada bagian data path untuk mengetes konfigurasi jaringan SDN. Sedangkan untuk melakukan testing pada mininet dapat dilakukan dengan *command* "sudo mn". Dengan *command* ini mininet akan mengemulasikan konfigurasi jaringan SDN yang terdiri dari 1 *controller*, 1 *switch* dan 2 *host*.



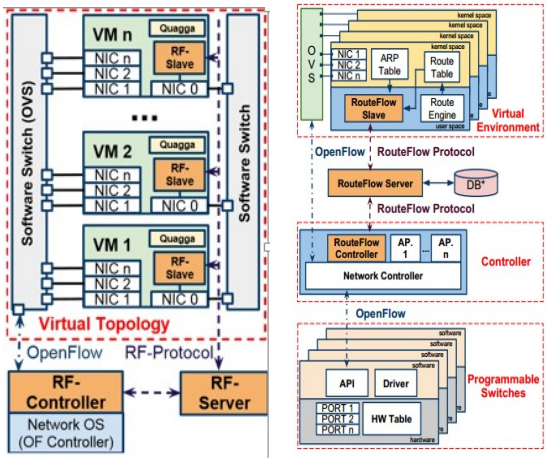
Gambar 5. *Single command* MININET [3]

E. *RouteFlow*

RouteFlow adalah suatu proyek *open source* untuk menyediakan layanan virtualisasi IP *routing* pada *OpenFlow enabled hardware (switch). Routeflow* mengikuti paradigma dari *Software Defined Network* yaitu berdasarkan pada *progmatic* and *logically network control* [4]. *RouteFlow* terdiri dari *Openflow.*

1. *Controller Application* (RF Proxy).
2. *Independent RouteFlow* Server (RF Server).
3. *Virtual Network Environment* (*physical infrastructure*) dan IP *Routing Engine* (*Quagga*).

RouteFlow Controller (RFC) bekerja sebagai aplikasi di atas *Network Controller*. *Network Controller* bertanggung jawab dalam *interface* terhadap *switch openflow*, berkomunikasi dengan *Routeflow* dengan API, dan melakukan peninjauan pada topologi jaringan. *Core control logic* berada pada *RF-Server* yang diberitahu tentang *event* yang relevan pada jaringan dan mempertahankan yang diperlukan. Untuk setiap *OpenFlow switch* yang ditemukan, *RF-Server* menginisiasi atau menetapkannya sebagai satu VM. Pada Gambar 6 dapat dilihat *RouteFlow* terbagi menjadi 3 layer yaitu *Virtual Environment*, *Controller* (*POX*) dan *Programmable switches*. Pada penelitian ini, *programmable switches* diemulasikan dengan menggunakan *mininet*.

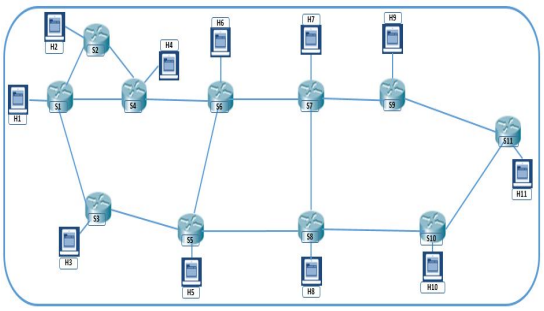


Gambar 6. Arsitektur RouteFlow dan Virtual plane [4]

Sebuah protokol *RouteFlow* didefinisikan untuk interaksi antara *RF-Server*, *RF-Controller* dan *RF-Slave*. Pesan pada RF-protokol dapat berupa *type command* atau *event* dan dapat dilihat sebagai *subset* sederhana dari pesan protokol *OpenFlow* ditambah sejumlah pesan baru untuk tujuan konfigurasi dan manajemen *VM*/*RF*-Slave misalnya untuk menerima atau menolak *VM*, konfigurasi *RF-Slave* dan *send update*. [3]

III. HASIL PENELITIAN

Pada penelitian ini, protokol disimulasikan menggunakan *emulator mininet* dan menggunakan *Pox* sebagai kontroler yang telah dienkapsulisasi pada arsitektur *RouteFlow*. Sistem dimodelkan sebagai sebuah jaringan yang terdiri dari sebuah kontroler dengan sebelas *switch* dimana setiap port pada *switch* dan PC memiliki IP masing-masing.



Gambar 7. Topologi Jaringan (Abilene)

Link yang menghubungkan suatu *switch* dengan *switch* lain mempunyai *cost* masing-masing. *Cost* inilah yang nantinya akan menentukan pemilihan rute dalam pengiriman data dari *source* sampai dengan *destination* sesuai dengan *routing protocol* yang digunakan.

A. *Proses Simulasi*

Topologi jaringan dilsimulasikan dalam *emulator mininet*. Jumlah *switch* yang digunakan berjumlah 11 (sebelas) buah. Setiap *link* yang menghubungkan *switch* mempunyai *cost* masing-masing yang digunakan sebagai parameter dalam menentukan rute. Setelah itu akan dilakukan pemrograman pada kontroler *Pox* (*RF Proxy* pada *RouteFlow*). Selain tu skenario topologi 7sw 7 h, 9sw 9h juga akan diujikan. Iperf yang digunakan sebagai *background traffic* untuk membanjiri jaringan adalah 50 Mb, 75 mb, 100 Mb dan 125 Mb.

Tabel 2. Alokasi *Background Traffic*

No	Topologi	*Server-Client*	*Background Traffic*
1.	*7 switch* 7 *host*	h1-h7	50 Mb 75 Mb 100 Mb 125 Mb
2.	9 *switch* 9 *host*	*h1 - h9*	50 Mb 75 Mb 100 Mb 125 Mb
3.	11 *switch* 11 *host*	*h1 - h11*	50 Mb 75 Mb 100 Mb 125 Mb

B. *Parameter Pengujian*

Dalam penelitian ini dilakukan pengujian pengiriman beberapa *bit* data dalam jaringan SDN. Setelah itu diukur performansi yang dihasilkan pada jaringan SDN dengan menggunakan parameter QoS

yaitu *throughput*, *delay*, *jitter*, *packet loss* serta *performnce device* pada *controller*.

C. *Perangkat*

1. *Hardware*

Dalam penelitian ini diperlukan beberapa perangkat keras untuk menunjang pengujian terhadap jaringan SDN yang telah dirancang. Perangkat keras yang akan digunakan dalam penelitian ini seperti yang ditunjukkan pada Tabel 3.

Tabel 3. Data *Hardware*

No	Perangkat	Rincian
1.	PC I	*Laptop HP Probook 4420s* *Processor Intel core i3* *OS Ubuntu 12.04.4* *RAM 4 GB*
2.	PC II	*Desktop Gigabyte 78LMT-S2P* *Processor AMD 8320* *OS Ubuntu 12.04.4* *RAM 8 GB*
3.	*Switch*	*D-Link DES-1024A*
4.	Kabel LAN	*RJ 45*

2. *Software*

Software yang dibutuhkan dalam penelitian ini digunakan untuk proses perancangan maupun pengambilan data. Semua software di bawah ini dijalankan dalam OS Ubuntu 12.04.4 baik pada PC I maupun PC II. Adapun *software-software* tersebut seperti yang ditunjukkan pada Tabel 4.

Tabel 4. Data *Software*

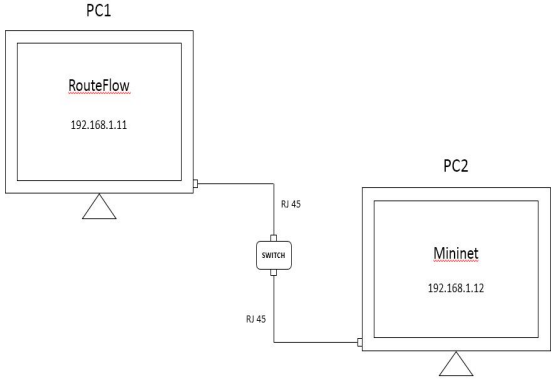
No	*Software*	*Runned in*	*Function*
1.	*RouteFlow (POX)*	*PC I*	*Controller*
2.	*Mininet*	*PC II*	*Emulator data plan*
3.	*Iperf*	*PC II*	*Background traffic*
4.	*D-ITG*	*PC II*	*Analysis throughput, delay, jitter and packet loss*

D. *Skenario Pengambilan Data*

Setelah skenario berhasil dilakukan dalam arti *host* dan *switch* pada jaringan diatas bisa terkoneksi satu dengan yang lain maka akan dilakukan pengambilan data dari jaringan tersebut. Pengambilan data disini dilakukan dengan menguji performansi pada jaringan dengan menggunakan parameter QoS yaitu *throughput*, *delay*, *jitter* dan *packet loss*.

Untuk meningkatkan kinerja perangkat, maka akan digunakan perangkat PC2 sebagai perangkat tambahan. RouteFlow akan dijalankan pada PC1 dengan OS *Ubuntu* 12.04.4 sedangkan mininet akan dijalankan pada PC2 dengan OS yang sama. Untuk *software* penguji akan digunakan

Iperf dan *D-ITG* untuk menguji performansi dari jaringan yang telah dirancang.

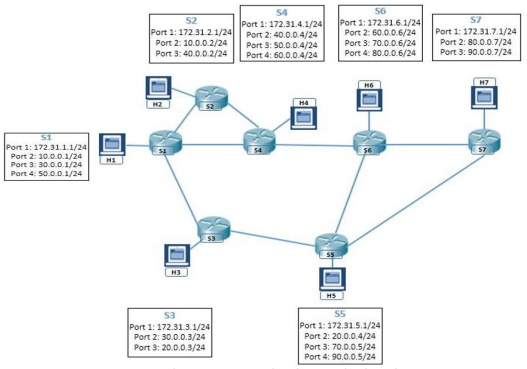


Gambar 8. Skenario Pengambilan Data

Pengambilan data akan menggunakan jumlah *switch* dan *host* sebagai variabel peubah. Selain itu juga akan digunakan *bandwidth* per *link* dan juga *bandwidth background traffic*. Ada tiga topologi yang dilakukan yaitu.

1. Menggunakan 7 *switch* dan 7 *host*.
2. Menggunakan 9 *switch* dan 9 *host*.
3. Menggunakan 11 *switch* dan 11 *host*.

Disini akan dilakukan pengamatan pada beberapa *link*.



Gambar 9. Topologi 7 switch 7 host

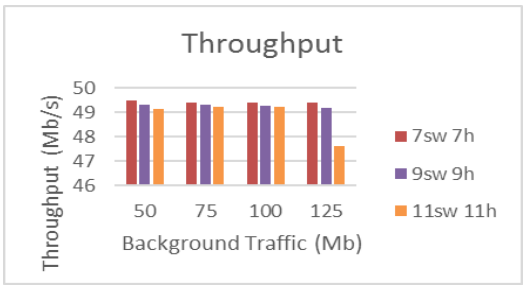
IV. HASIL DAN PEMBAHASAN

Trafik *data*, *video* dan *VoIP* disalurkan melalui *server* sampai ke *client* yang telah ditentukan. Dari pengukuran QoS dengan menggunakan *D-ITG* dapat dilakukan pencatatan dan analisis *throughput*, *delay, jitter* dan *packet loss* serta *perfomance device* pada *controller*.

Pengukuran performansi telah dilakukan sebanyak 30 kali setiap skenario (*background traffic*). Jumlah *host* dan *switch* digunakan sebagai variable pengubah. Acuan standar QoS yang digunakan adalah *ETSI TS* 101 329-2 (*European Telecommunications Standard Institute*, 2000) tetang *Telecommunications and*

Internet Protocol Harmonization Over Network (*TIPHON*); *End to End Quality* of *Service* in *TIPHON Systems*.

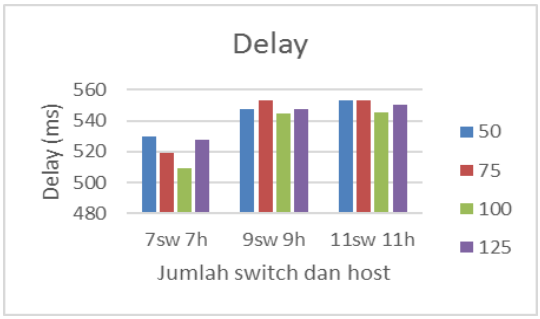
A. Throughput



Gambar 10. *Throughput*

Dari hasil pengukuran hampir semua *bandwidth link* terpakai. Nilai *Throughput* semakin kecil seiring dengan semakin kecil seiring dengan semakin besarnya *traffic* UDP (*background traffic*) yang membanjiri jaringan. Nilai *throughput* masuk dalam kategori "sangat bagus" dengan indeks 4 menurut *ETSI TS* 101 329-2 TIPHON.

B. Analisis Delay

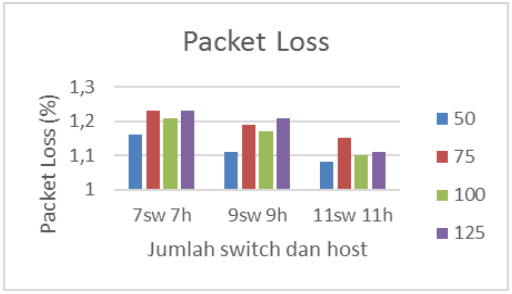


Gambar 11. *Delay*

Pengukuran dengan *background traffic* 100 Mb dan 125 Mb menunjukkan nilai *delay* yang tidak terlalu berbeda dengan pengukuran sebelumnya dengan menggunakan *background traffic* 50 Mb dan 75 Mb. Nilai *delay* tertinggi terjadi pada topologi 11 host 11 *switch* yaitu 550,377 ms. Hal ini disebabkan karena *delay link* sebesar 10 ms sedangkan *link* yang harus dilalui oleh paket dari h1 sampai h11 merupakan jalur terjauh dibandingkan skenario pengiriman yang lain dalam penelitian ini. Menurut *ETSI TS* 101 329-2, hasil pengujian untuk semua *background traffic* masuk dalam kategori "jelek".

C. Analisis Packet Loss

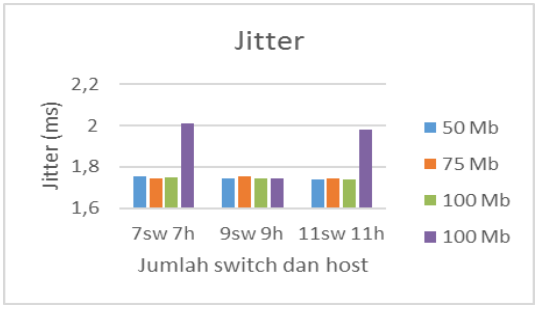
Dapat dilihat pada grafik pada Gambar 12 bahwa nilai *packet loss* semakin menurun seiring semakin besarnya topologi yang diujikan. Nilai *packet loss* terkecil terjadi pada topologi 11 switch 11 *host* dengan *background traffic* 50 Mb yaitu sebesar 1,08%. Menurut *ETSI TS* 101 329-2, hasil pengujian dengan *background traffic* 50 Mb, 75 Mb, 100 Mb dan 125 Mb masuk dalam kategori "Sangat Bagus" dengan indeks 4.



Gambar *12. Packet Loss*

D. Analisis Jitter

Dari hasil pengukuran nilai *jitter* tidak terlalu berbeda jauh antara satu topologi dengan topologi yang lain. Nilai *jitter* terbesar tejadi pada saat pengiriman data pada topologi 7 host 7 switch dengan *background traffic* 125 Mb yaitu sebesar 2,012 ms. Maka dapat disimpulkan bahwa *variasi delay* dari paket-paket yang ditransmisikan dalam jaringan kecil.

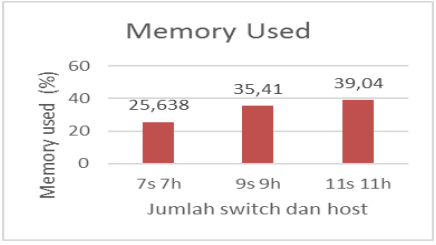


Gambar 13.. Jitter

Menurut nilai jitter dari hasil pengukuran masuk dalam kategori "bagus" dengan indeks 3 untuk semua topologi dan *background traffic* yang digunakan dalam penelitian.

E. Analisis Performansi Perangkat

Konsumsi memory (RAM) pada *controller* (PC I) diuji dengan menjalankan 3 skenario topology. *Controller* memegang peran yang sangat penting dalam jaringan SDN. Maka performansi perangkat controller harus baik untuk mengontrol *traffic-traffic* pada jaringan SDN.



Gambar 14. *Memory Used*

Dari Gambar 14 dapat ditarik kesimpulan bahwa semakin besar jaringan dalam hal ini jumlah switch dan host serta link maka *resource* (*memory*) yang terpakai pada controller juga akan semakin besar. Maka dari itu performansi perangkat *controller* sangat menentukan kinerja pada jaringan SDN.

V. PENUTUP

A. Kesimpulan

Dari penelitian yang telah dilakukan dapat ditarik beberapa kesimpulan sebagai berikut.

1) *Link state* ISIS yang sering digunakan sebagai protocol *routing* pada jaringan jaringan konvensional dapat diimplementasikan pada *platform* SDN.
2) Nilai *throughput* mengalami penurunan seiring dengan semakin besarnya *background trafik* (UDP).
3) Dengan BW *link* 50 Mb dan *delay* bahan (*link*) 10 ms dan dialiri *background traffic* 50 Mb, 75 Mb, 100 Mb, 125 Mb menghasilkan nilai *delay* data dengan spesifikasi yang telah dijelaskan sebelumnya masuk dalam kategori "jelek" menurut *ETSI TS* 101 329-2. Semakin besarnya jumlah *switch* dan *link* yang dialalui akan menyebabkan semakin besarnya nilai *delay*.
4) Nilai *packet loss* ketika dialiri *background traffic* 50 Mb, 75 Mb, 100 Mb, 125 Mb masuk dalam kategori "sangat bagus". Mayoritas hasil pengujian menunjukkan penurunan nilai packet loss seiring dengan peningkatan jumlah *switch*.
5) Semakin besar jaringan dalam hal ini jumlah *switch* dan *host* serta *link* pada *infrastructure layer* maka *resource* (*memory*) yang terpakai pada *controller* juga akan semakin besar.

B. Saran

Adapun saran yang diberikan untuk menjadi masukan dalam penelitian ini adalah sebagai berikut.

1) Untuk penelitian selanjutnya disarankan untuk memakai *RouteFlow* yang terbaru agar *support* untuk semua routing protocol terutama *routing* protocol yang masih belum banyak dikenal.
2) Khusus untuk *Link State* IS-IS disarankan untuk menggunakan 2 level untuk penelitian selanjutnya.

DAFTAR PUSTAKA

[1] Cisco, "SDN," 5 April 2014. [Online]. Available: https://www.cisco.com/web/ANZ/cisco-live/attend/hot_topics/sdn.html. [Accessed 5 January 2015].

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

[3] I. Pepelnjak, OpenFlow and SDN: Hype, Useful Tools or Panacea?,IP Space, 2013.

[4] I. Sofana, Cisco CCNP dan Jaringan Komputer, Bandung: Informatika, 2012.

[5] M. Team, Mininet An Instant Virtual Network on your Laptop (or other PC), [Online]. Available: http://mininet.org/. [Accessed 5 Maret 2014].

[6] Nascimento, Marcelo R., The RouteFlow Approach Leveraging Software-Defined, Research and Development Center (CPqD) , 2011.

[7] Open Network Foundation.(2012). Software-Defined Networking: The New Norm for Networks.

[8] S. Azodolmolky, Software Defined Networking with OpenFlow, Birmingham: Packt Publishing, 2013.

[9] The Network and Management Architecture Group (NAM), University of Patras, Greece, "Software defined networking (SDN), [Online]. Available: http://nam.ece.upatra.gr/index.php?q=node/59

[10] Tomovic, S., Radonjic, M., & Radusinovic, I. (n.d.). Quagga Routing Platform: Application and Performance. University of Montenegro, Faculty of Electrical Engineering, Montenegro.

[11] Tulloh. Rohmat, Negara M. Ridha, Hidayat. Arif, Simulasi Virtual Local Area Network (VLAN) Berbasis Software Defined Network (SDN) Menggunakan POX Controller, Jurnal Infotel, 2015.

[12] Zeng, P., Nguyen, K., Shen, Y., & Yamada, S. (2014). On The Resilence of Software Defined Routing Platform. APNOMS.