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

Simulasi *Virtual Local Area Network (VLAN)* Berbasis *Software Defined Network* (SDN) Menggunakan *POX Controller*

Rohmat Tulloh[1], Ridha Muldina Negara[2], Arif Nur Hidayat[3]
[1,2,3] Telkom University
[1,2,3] Jalan Telekomunikasi Terusan Buah Batu Bandung 40257
[1]rohmatth@telkomuniversity.ac.id, [2]ridhanegara@telkomuniversity.ac.id,
[3]arifnurhidayatoke@gmail.com

Abstrak – VLAN (Virtual LAN) merupakan sebuah teknologi yang dapat mengkonfigurasi jaringan logis independen dari struktur jaringan fisik. Hasil dari penelitian sebelumnya sudah diprediksi bahwa dibutuhkan *Virtual Network* yang akhirnya terciptalah VLAN. Namun paradigma jaringan saat ini tidak flexible, ketergantungan terhadap vendor sangat besar karena fungsi *data plane* dan *control plane* berada dalam satu paket *device*. SDN (*Software defined network*) yang merupakan salahsatu evolusi teknologi jaringan sesuai dengan tuntutan yang berkembang dimana memisahkan fungsi *data plane* dan *control plane* pada suatu perangkat. POX *Controller* digunakan untuk men-simulasikan dan menguji *Platform* SDN (*Software defined network*). Pada penelitian ini menggunakan *Openflow versi 1.0* untuk memasang *header* VLAN sehingga penelitian ini difokuskan untuk mengevaluasi performa *forwarding VLAN* yang memanfaatkan *Openflow* sebagai *control plane* dapat berfungsi dengan baik. Hasil penelitian ini mengusulkan penerapan karakteristik teknologi VLAN pada SDN karena telah berjalan dengan benar sesuai hasil pengujian konektifitas, verifikasi dan keamanan. Kemudian hasil pengujian lanjutan untuk melihat pengaruh SDN dengan skenario penambahan jumlah VLAN ID didapatkan bahwa *set-up time* akan bertambah seiring meningkatnya jumlah *host* dan dengan menggunakan protokol *OpenFlow, latency* yang terjadi di jaringan dapat dipantau dengan parameter *round trip time* (RTT) yang stabil direntang 0,2 sampai 6 *second* walaupun jumlah vlan_id dan *background traffic* bertambah.

Kata Kunci - VLAN, Software defined network, POX, Mininet. OpenFlow

Abstract - VLAN (Virtual LAN) is a technology that can configure logical networks independent of the physical network structure. Results from previous studies have predicted that it takes Virtual Network which eventually creates a VLAN. However, the current network paradigm is not flexible, very large dependency on vendors as a function of *data plane* and *control plane* in the same device package. SDN (*Software defined network*) which is one of the main evolution of network technology in accordance with the growing demands which separates the functions of *data plane* and *control plane* on a device. The author tries simulates and tests Platform SDN (*Software defined network*) using POX Controller. In this study, using OpenFlow version 1.0 to install the VLAN header so that this study focused on evaluating the performance of forwarding VLAN that utilizes OpenFlow as a *control plane* function properly. Results of this study proposes the application of the characteristics of the SDN VLAN technology because it has been running correctly according to the results of testing connectivity, and security verification. Then the results of further testing to see the effect of SDN with a scenario of increasing the number of VLAN ID was found that set-up time will increase with increasing number of *host* and using protocol OpenFlow we can monitor the latency that occurs on the network with parameter round trip time (RTT) stable extended 0.2 to 6 second, although the number vlan_id and background traffic increases.

Keywords - VLAN, *Software defined network, POX, Mininet, OpenFlow*

I. PENDAHULUAN

Di awal penelitian, VLAN (Virtual LAN) adalah sebuah teknologi yang dapat mengkonfigurasi jaringan logis independen dari struktur jaringan fisik. Dengan VLAN, pengguna yang memiliki kesamaan ruang (seperti ruang rapat) dapat mengakses ke jaringan departemen mereka karena perubahan struktur jaringan logis dicapai hanya dengan konfigurasi switch VLAN.

Dalam metode konfigurasi umum VLAN, ada masalah yang akan muncul: (1) sejak VLAN dikelola statis oleh administrator, biaya administrasi untuk mengelola VLAN sementara adalah cukup tinggi, (2) karena VLAN *identifier* (VLAN-ID) yang

didefinisikan oleh IEEE802.1q [7] memiliki ruang hanya 12bits, insufisiensi dari VLAN-ID mungkin terjadi, (3) terutama dalam organisasi skala besar di mana VLAN dikelola oleh masing-masing departemen independen, konflik VLAN-ID dialokasikan oleh masing-masing departemen dapat terjadi.

Untuk mengatasi masalah ini, penelitian yang dilakukan Kiyohiko Okayama [8], mengusulkan Metode yang menyediakan interkoneksi dinamis antara VLAN Configured sementara di ruang umum dan VLAN pengguna dari departemen mereka.

Dalam metode yang mereka usulkan, pengguna di ruang umum dapat mengakses ke jaringan/ departemen nya dengan mulus (yaitu data-link komunikasi tingkat) dengan mengubah sementara VLAN-ID di ruang umum dan VLAN-ID yang digunakan dalam / nya departemen masing-masing lainnya. Selanjutnya, metode yang diusulkan memberikan Alokasi VLAN-ID dinamis dan remote otomatis VLAN beralih konfigurasi. Hasil dari penelitian sebelumnya sudah diprediksi bahwa pekerjaan di masa depan mencakup pertimbangan metode pencarian otomatis untuk switch VLAN. Hal ini terlihat pada kondisi jaringan saat ini yang ternyata juga memiliki kekurangan karena ketergantungan terhadap perangkat dari vendor tertentu. Paradigma jaringan saat ini tidak flexible, ketergantungan terhadap vendor sangat besar karena fungsi *data plane* dan *control plane* berada dalam satu paket device [1].

Kondisi jaringan yang mulai jenuh menjadi alas an mulai banyaknya penelitian dan percobaan *platform* SDN (*Software defined network*) yang merupakan salahsatu 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 [1]. Penelitian yang dilakukan ini adalah mencoba men-simulasikan konsep teknologi VLAN sesuai dengan *Platform* SDN (Sofware) menggunakan *OpenFlow* versi 1.0. *Openflow* digunakan untuk memasang *header* VLAN sehingga penelitian ini mengevaluasi apakah performa *forwarding* VLAN yang memanfaatkan *openflow* dapat berfungsi.

Pada konsep SDN, jaringan yang ada dikelola secara tersentralisasi sehingga servis yang diberikan seragam untuk setiap node (*uniform service*) dan *cost effective* karena mengurangi biaya operasional *multi network element* [2]. Hal ini juga mempertemukan teknologi dengan keinginan *business* IT saat ini karena konsep SDN memberikan kemudahan pengaturan trafik jaringan sesuai keinginan yang dikehendaki tanpa menunggu pengembangan dari vendor. Sehingga kostumasi koneksi ke *data center* dapat dibuat sebaik mungkin dengan pengaturan pada *controller*-nya [3].

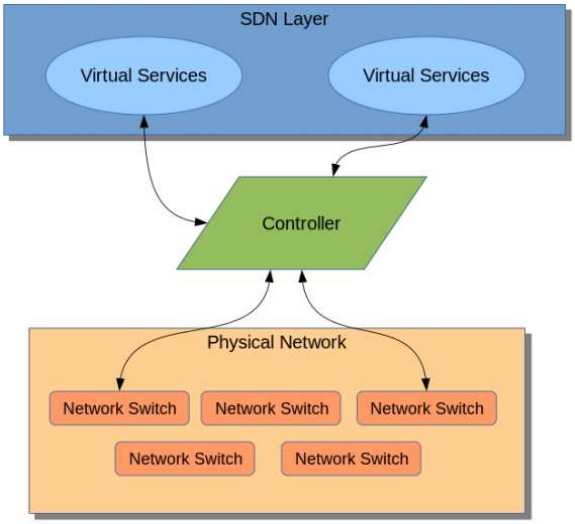
Konsep SDN mendukung multiple API sehingga memudahkan IT Network Engineer memprogram sesuai Bahasa pemrograman yang dikehendaki. SDN juga memberikan dukungan yang luas karena bersifat *open source* sehingga perkembangannya cepat [3].

Penjelasan pada penelitian ini adalah sebagai berikut. Dimulai dengan membahas beberapa asumsi. Dalam Bagian 2, dijelaskan metode VLAN berbasis SDN yang diusulkan. Kemudian, di bagian 3, di evaluasi kinerja dari metode yang diusulkan. Akhirnya, di bagian 4, dijelaskan kesimpulan dan pekerjaan lebih lanjut. *Emulator* mininet dan *controller* POX digunakan untuk mengimplementasikan jaringan VLAN berbasis SDN di lingkungan Telkom University. Pengujian dimaksudkan untuk mengukur dan menganalisis parameter konektifitas VLAN, verifikasi perilaku VLAN, pengujian keamanan pada VLAN dan analisis set-up time maupun *round trip time* dengan scenario perubahan jumlah VLAN.

Dikutip dari white paper yang diterbitkan oleh *Open Network Foundation* yang berjudul "*Software defined networking : The New Norm for Networks*"[5], *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 dan CP, 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* [2].



Gambar 1. Controller berfungsi menghubungkanPhysical Layer dan SDN Layer[1]

Sementara untuk elemen *control*, terdapat beberapa jenis *controller* yang saat ini berkembang, antara lain :

- Ryu
- POX
- Opendaylight

Tabel 1. Perbedaan Controllers SDN

	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.

Pada makalah ini, POX *controller* digunakan untuk mensimulasikan jaringan VLAN karena POX merupakan perangkat lunak bersumber terbuka yang menyediakan jalan untuk mengimplementasikan protokol OpenFlow. POX memberikan kemudahan pengguna untuk membuat program yang fungsinya berbeda-beda. Oleh karena fungsi VLAN disimulasikan dengan memilih POX *controller*.

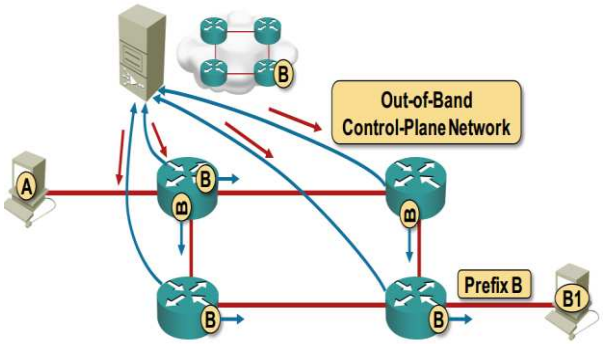
II. METODOLOGI PENELITIAN

A. *VLAN (Virtual Local Access Network)*

Teknologi VLAN (Virtual Local Area Network) bekerja dengan cara melakukan pembagian jaringan secara logik ke dalam beberapa subnet. VLAN adalah kelompok device dalam sebuah LAN yang dikonfigurasi (menggunakan software manajemen) sehingga mereka dapat saling berkomunikasi seakan-akan dihubungkan dengan jaringan yang sama walaupun secara fisikal mereka berada pada segmen LAN yang berbeda. Jadi VLAN dibuat bukan berdasarkan koneksi fisikal namun lebih pada koneksi logikal, yang tentunya lebih fleksibel. Secara logika, VLAN membagi jaringan ke dalam beberapa subnetwork. VLAN mengijinkan banyak subnet dalam jaringan yang menggunakan switch yang sama [9].

B. *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) [11]. OpenFlow dapat dikatakan sebagai pemisah antara control plan dan data plan [10].

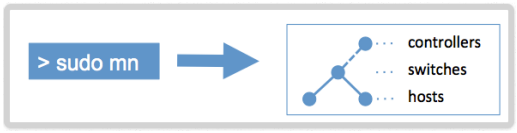


Gambar 2. OpenFlow [10]

C. *Komponen Perangkat Penelitian*

1. Emulator (*Mininet)*

   Mininet adalah suatu software emulator yang memungkinkan untuk melakukan *prototyping* pada jaringan yang luas dengan hanya menggunakan satu mesin [12]. Mininet digunakan untuk mengemulasi data plan atau *infrastructure* layer dalam perancangan jaringan SDN.



Gambar 3. *Single Command Mininet* [12]

2. *Contoller*

   POX adalah *platform* yang digunakan untuk pengembangan dan pemodelan pada *network control software*. POX menggunakan *python* dalam bahasa pemrogramannya. POX bekerja pada layer control plan sebagai sebuah network *contoller*. Dalam arsitektur *RouteFlow POX* berada pada RFProxy yang bertanggung jawab untuk interaksi dengan *OpenFlow switch* (*datapath*) melalui protokol *OpenFlow*.

Gambar 4. *POX*

3. *Ubuntu* Versi 12.04

   Ubuntu merupakan salah satu distribusi Linux yang bersifat *user friendly* dan didistribusikan sebagai perangkat lunak bebas. Dalam penelitian ini dibutuhkan *Operating System* (OS) *Linux Ubuntu* 12.04 untuk menjalankan *RouteFlow* maupun *mininet*. Karena kedua software ini hanya dapat dijalankan langsung di platform Linux tanpa menggunakan *Virtual Box* (VB). *Ubuntu* versi 12.04 direkomendasikan untuk dapat menjalankan *RouteFlow* serta *mininet* vesi 2.0.0.

Dikutip dari paper berjudul ”A Network in a Laptop: Rapid Prototyping for Software-Defined Networks” [6], Mininet merupakan sebuah sistem untuk mempurwarupakan jaringan besar secara cepat di atas sumber daya terbatas sebuah laptop. Prototyping Envir onment yang ada saat ini mempunyai kelebihan dan kekurangan. Perangkat uji yang dibuat khusus sangatlah mahal. Simulator seperti ns-2 atau opnet sangatlah menarik karena dapat dijalankan di atas sumber daya yang terbatas, tetapi kurang realistis.

Terdapat beberapa alasan menggunakan Mininet, antara lain [4].

1. Dapat membuat topologi sesuai keinginan pengguna
2. Dapat menggunakan segala program yang berjalan pada Linux
3. Dapat menyesuaikan *forwarding* paket sesuai pengguna berdasarkan protokol OpenFlow
4. Mininet di desain sederhana sehingga pengguna dapat menggunakan walaupun program yang dibuat sederhana.

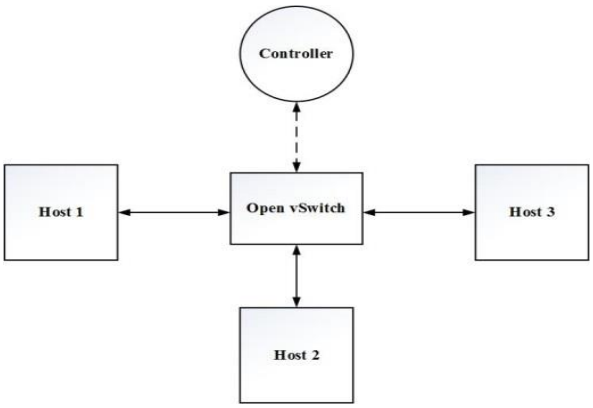
Terdapat beberapa *command* yang biasa digunakan untuk membuat topologi pada mininet.

Tabel 2. Command Mininet

Commands	Deskripsi
mn	Menjalankan mininet
--topo single, 4	Membuat 1 switch, 4 *host*
--mac	Membuat alamt mac node
--arp	Membuat arp
--switch ovsk	Menggunakan Openvswitch
--controller remote	Menggunakan controller
--ip	Alamat ip controller

Berikut diberikan contoh penggunaan program mininet.

#mn –mac –topo single,3 –switch ovsk –controller remote.

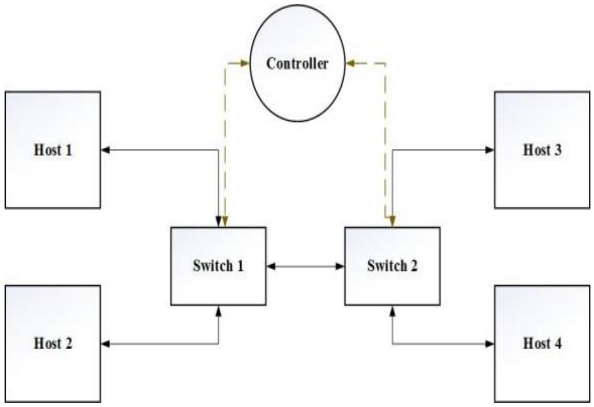


Gambar 5. Skenario Percobaan dengan Satu Switch dan Tiga *Host*

III. HASIL PENELITIAN

Pada pengujian ini menggunakan skenario jaringan VLAN menggunakan controller POX. Pada dasarnya VLAN merupakan bagian dari network virtualization yang membagi *host* berdasarkan suatu persamaan dan perbedaan. VLAN pada umumnya membagi *host* berdasarkan berdasarkan input port ke switch atau alamat mac. Tujuan utama membuat jaringan VLAN adalah membatasi broadcast domain.

Pada skenario ini, akan dibuat topologi yang terdiri dari dua switch, empat *host*, serta satu controller POX.



Gambar 6. Topologi VLAN

Terdapat *built-in functions* yang disediakan POX untuk membuat fungsi VLAN bekerja. Fungsi tersebut adalah.

a) of.ofp_action_vlan_vid(vlan_vid)

➔ berfungsi memberikan *header* vlan pada paket yang masuk

b) of.ofp_action_strip_vlan

➔ berfungsi melepas *header* vlan jika suatu kondisi memenuhi

c) match.dl_vlan

➔ berfungsi melakukan cek terhadap nilai *header* vlan

IV. PENGUJIAN DAN ANALISIS

Dari simulasi yang sudah dilakukan, didapatkan hasil analisis kinerja teknologi VLAN berbasis SDN dengan parameter *ping test*, verifikasi perilaku VLAN serta *setup time*. Pada pengujian kali ini digunakan standar VLAN 802.1Q. Pada pengujian *time setup* dilakukan beberapa skenario pengujian dengan penambahan beberapa *user* serta penambahan jumlah VLAN_ID. Tujuan dari pengujian *setup time* adalah mengetahui waktu *setup paket openflow mod* yang dikirimkan dari controller ke *switch* ketika *switch* tidak tahu *action* apa yang harus dilakukan terhadap suatu paket.

A. Pengujian Konektivitas VLAN

Untuk menganalisis perilaku konektivitas VLAN terlebih dahulu diaktifkan topologi jaringan di mininet serta mengaktifkan *controller.* Pada pengujian konektifitas dilakukan dengan menggunakan protocol ICMP dimana dari setiap *host* akan mencoba me-*request* paket menuju *host* lainnya. Pada pengujian ini, dilakukan ping dari *host* 1 menuju *host* 9 dimana kedua *host* tersebut tergabung dalam satu domain VLAN ID yang sama. Dari gambar 4 dibawah dilakukan 18 kali pengujian, didapatkan info bahwa 18 paket sudah terkirim dan berhasil diterima semua dengan persentase *packet loss* 0%.

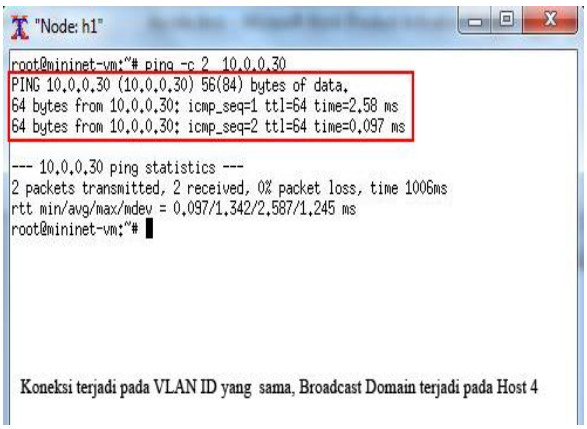
```
PING 10.0.0.9 (10.0.0.9) 56(84) bytes of data.
64 bytes from 10.0.0.9: icmp_seq=1 ttl=64 time=2.61 ms
64 bytes from 10.0.0.9: icmp_seq=2 ttl=64 time=0.226 ms
64 bytes from 10.0.0.9: icmp_seq=3 ttl=64 time=0.123 ms
64 bytes from 10.0.0.9: icmp_seq=4 ttl=64 time=0.127 ms
64 bytes from 10.0.0.9: icmp_seq=5 ttl=64 time=0.121 ms
64 bytes from 10.0.0.9: icmp_seq=6 ttl=64 time=0.249 ms
64 bytes from 10.0.0.9: icmp_seq=7 ttl=64 time=0.138 ms
64 bytes from 10.0.0.9: icmp_seq=8 ttl=64 time=0.121 ms
64 bytes from 10.0.0.9: icmp_seq=9 ttl=64 time=0.120 ms
64 bytes from 10.0.0.9: icmp_seq=10 ttl=64 time=0.137 ms
64 bytes from 10.0.0.9: icmp_seq=11 ttl=64 time=0.264 ms
64 bytes from 10.0.0.9: icmp_seq=12 ttl=64 time=0.218 ms
64 bytes from 10.0.0.9: icmp_seq=13 ttl=64 time=0.221 ms
64 bytes from 10.0.0.9: icmp_seq=14 ttl=64 time=0.244 ms
64 bytes from 10.0.0.9: icmp_seq=15 ttl=64 time=0.130 ms
64 bytes from 10.0.0.9: icmp_seq=16 ttl=64 time=0.135 ms
64 bytes from 10.0.0.9: icmp_seq=17 ttl=64 time=0.226 ms
64 bytes from 10.0.0.9: icmp_seq=18 ttl=64 time=0.255 ms
^C
--- 10.0.0.9 ping statistics ---
18 packets transmitted, 18 received, 0% packet loss, time 17032ms
rtt min/avg/max/mdev = 0.120/0.315/2.615/0.560 ms
```

Gambar 7. Pengujian Konektivitas LAN

B. Verifikasi VLAN

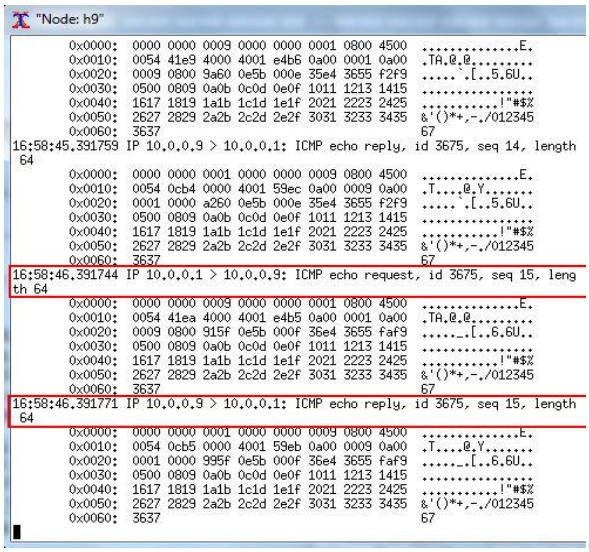
Untuk menganalisis perilaku konektivitas VLAN terlebih dahulu diaktifkan topologi jaringan di mininet serta mengaktifkan *controller*. Pengujian konektifitas dilakukan dengan menggunakan protokol ICMP dimana dari setiap *host* akan mencoba meminta paket menuju *host* lainnya.

Pada pengujian ini, dilakukan ping dari *host* 1 menuju *host* 4 dimana kedua *host* tersebut tergabung dalam satu domain VLAN ID yang sama. Selain itu untuk memastikan skenario VLAN yang dibuat berhasil, pengujian juga dilakukan dengan *host* yang berbeda domain VLAN.



Gambar 8. Percobaan Ping H1 ke H4 Sukses

Selain *ping request*, proses terjadinya komunikasi antar vlan juga dianalisis dalam domain yang sama.

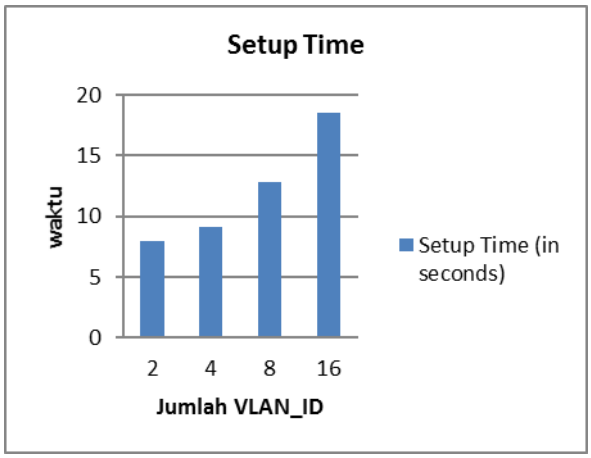


Gambar 9. Verifikasi VLAN

Dari Gambar 9 dapat dijelaskan bahwa, agar komunikasi vlan dapat terbentuk maka dua *host* yang saling berhubungan mengirimkan paket echo request serta echo reply sebelum kedua *host* tersebut melakukan komunikasi ICMP.

C. *Analisa Set up Time Dengan Skenario Perubahan Jumlah VLAN_ID*

Pada bagian ini disajikan analisis hasil simulasi setup time dengan skenario perubahan jumlah VLAN. Topologi yang diuji mula-mula, terdapat empat buah *host* dengan dua *switch*, kemudian jumlah *host* ditambahkan dengan VLAN ID yang bertambah pula. Tujuan utama dari pengujian setup time ini adalah untuk mengetahui seberapa lama waktu setup paket *openflow* mod yang dikirimkan dari *controller* ke *switch* sampai kondisi *switch* stabil yaitu kondisi ketika *switch* tidak lagi meminta paket action terhadap suatu paket.



Gambar 10. Kinerja *Setup Time Controller*

Dari pengujian yang dilakukan bahwa *setup time* untuk jumlah *host* yang semakin bertambah maka *setup time* akan semakin besar. Hal ini dikarenakan *controller* menggunakan mode *proactive* sehingga semakin banyak *host* maka *rule* yang dibutuhkan juga semakin banyak sehingga kerja dari *controller* untuk memberikan informasi kepada *switch* semakin banyak.

Hal ini memberikan keuntungan dan kekurangan ketika *controller* menggunakan mode *proactive*. Kekurangan dari mode *proactive* ketika menggunakan mode ini maka *rule* yang diberikan akan semakin banyak bergantung pada kondisi jaringan yang ada. Sementara itu kelebihan dari mode ini dibandingkan mode *reactive* adalah *controller* hanya memberikan pengaturan sekali diawal saja, sehingga jaringan cepat stabil sedangkan pada mode *reactive,* controller akan memberikan rule hanya ketika ada permintaan paket "*in*" dari suatu *switch* yang sebelumnya belum ada sehingga jika digunakan mode reactive maka harus dibuat controller adaptif terhadap suatu permintaan selain itu juga harus didesain *resource* untuk *buffer* paket, baik disisi *switch* ataupun disisi *controller* sehingga menambah kompleksitas coding.

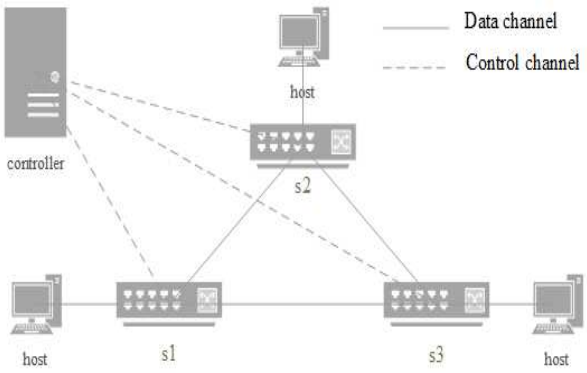
D. *Analisa Round Trip Time Dengan Skenario Perubahan Jumlah VLAN_ID*

Latency adalah metrik penting untuk dipertimbangkan dalam hari ke hari pengoperasian jaringan, terutama jika digunakan untuk pengiriman data dari aplikasi sensitif. Seperti sambungan VoIP yang baik membutuhkan *latency* kurang dari 50ms. Sementara persyaratan layanan Video Streaming, beberapa *loss* (kurang dari 5%) masih diterima untuk sebagian besar codec. Nilai Latency rata-rata ~150ms dinilai cukup memadai dan *delay* 5 detik juga mungkin diterima jika frame didalam *buffer*. Namun untuk layanan Video interaktif memiliki persyaratan yang lebih tinggi dan kerentanan lebih besar terhadap frame hilang. Meskipun beberapa kerugian (kurang dari 1%) masih dapat diterima, jitter memiliki lebih banyak dampak pada layanan ini.

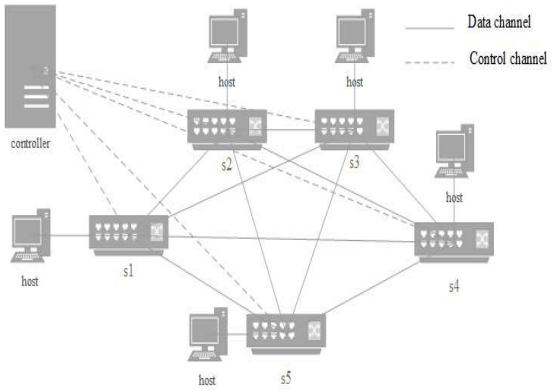
Software defined networking, khususnya melalui protokol seperti OpenFlow, hadir dalam jaringan. Hal ini bertujuan untuk memisahkan *Data plane* dari *control plane* di banyak pemrograman jaringan, layanan, heterogenitas dan pemeliharaan. Bahkan jika aplikasi mobile dan multimedia sering menunjuk menunjukkan kegagalan pada arsitektur jaringan saat ini, maka tidak ada cara yang secara efisien dinamis mendapatkan nilai latency dalam jaringan.

OpenFlow secara efisien menerapkan kebijakan QoS. Dalam tulisan ini, diusulkan mekanisme untuk mengukur latency link dari controller OpenFlow. Parameter yang diukur adalah RTT. Nilai RTT sangat tergantung dari kondisi trafik. Semakin padat

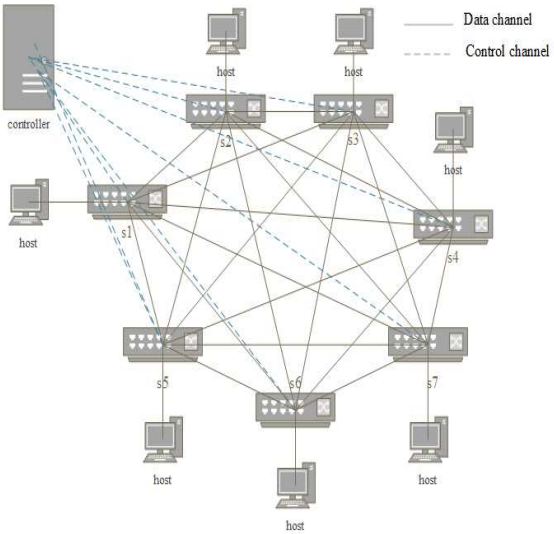
trafik maka semakin besar RTT nya, semakin besar ukuran paket data, maka semakin besar pula RTT nya jika kondisi trafik jaringan sedang bagus dan stabil. Dalam simulasi ini dibuat tiga buah *scenario* dengan jumlah penambahan jumlah *switch* seperti pada Gambar 11,12, dan 13.



Gambar 11. Topologi 1 (3 buah Switch)



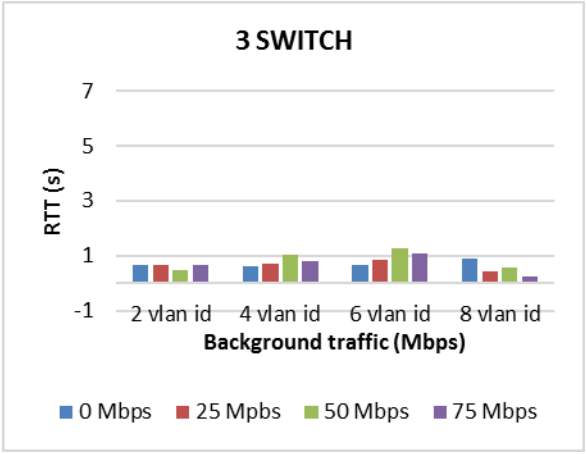
Gambar 12. Topologi 2 (5 buah Switch)



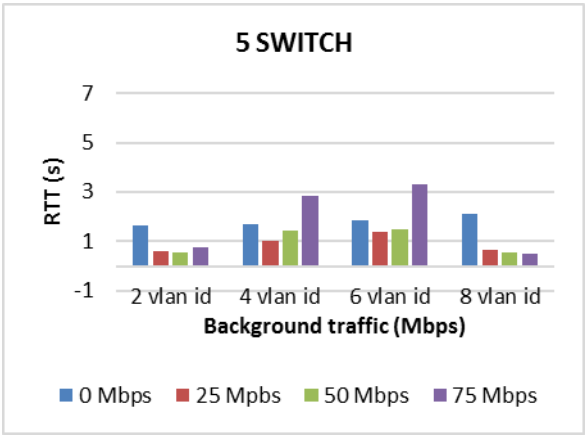
Gambar 13. Topologi 3 (7 buah switch)

Pada penelitian ini diamati pengaruh penambahan *switch,* VLAN_ID dan *background traffic,* yaitu.

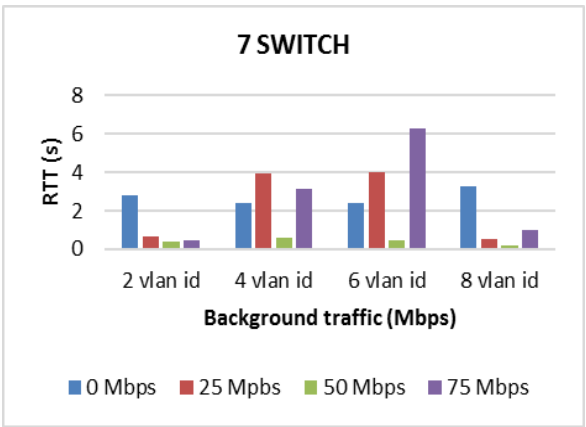
- Penambahan Jumlah VLAN_ID



Gambar 14. Hasil Pengamatan RTT pada Topologi 1 dengan 3 buah Switch, penambahan jumlah VLAN ID dan pemberian *Background Traffic*



Gambar 15. Hasil Pengamatan RTT pada Topologi 2 dengan 5 buah Switch, penambahan jumlah VLAN ID dan pemberian *Background Traffic*



Gambar 16. Hasil Pengamatan RTT pada Topologi 3 dengan 7 buah Switch, penambahan jumlah VLAN ID dan pemberian *Background Traffic*

Hasil berikut ini merupakan nilai RTT yang didapatkan dari percobaan dengan perubahan jumlah switch dan pemberian background trafik. nilai yang didapatkan bernilai stabil dengan rentang berkisar 0.2 s hingga 6 s. hal ini dikarenakan sistem yang dibuat bersifat proaktif. sehingga flow tabel yang terbentuk bersifat statis atau sama untuk semua kondisi yang dilakukan.

V. PENUTUP

A. *Kesimpulan*

1. Pada skenario VLAN yang telah dilakukan, *controller* POX berfungsi dengan baik dimana terlihat dari hasil percobaan bahwa *host* yang berada dalam VLAN_ID yang sama saling berkomunikasi, *host* yang berbeda VLAN ID tidak dapat berkomunikasi walaupun di dalam jaringan berbasis SDN. Hal ini terjadi karena POX mempunyai fitur untuk membuat header VLAN pada suatu paket data dapat masuk menuju switch, serta juga terdapat fungsi untuk melepas header maupun fungsi mencocokan header VLAN sehingga adopsi teknologi VLAN dapat bekerja pada jaringan SDN.
2. Berdasarkan penelitian sebelumnya, pengaturan jaringan secara efisien dan efektif sangat dibutuhkan dengan semakin berkembang dan beragamnya perangkat di jaringan, hal ini dapat teratasi dengan mengaplikasikan *Platform* SDN.
3. Dalam lingkungan yang sebenarnya, waktu *set up* dari VLAN sangat penting bagi pengguna. Dari hasil percobaan ada waktu tambahan yang dibutuhkan untuk *controller* memberikan informasi kepada *switch* yang semakin meningkat. Namun hal ini hanya akan terjadi diawal saja dan selanjutnya tugas switch di jaringan hanya forwarding saja. Hasil setup time yang didapat tidak berbeda jauh dengan hasil penelitian kiyohiko okayama sebelumnya < 20 s.
4. Nilai RTT untuk tiga topologi yang digunakan didapatkan hasil yang stabil berkisar 0,2 – 6 second. Jika melihat standard latency (VoIP <50ms, Streaming < 150ms) memang ini masih tinggi.

B. *Saran*

Pekerjaan di masa depan mencakup pertimbangan metode *controller* menggunakan mode *reactive*.

DAFTAR PUSTAKA

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

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

[3] Van der Pol, R., Boele, S., Dijkstra, F., Barczyk, A., van Malenstein, G., Chen, J. H., & Mambretti, J. (2012,November). Multipathing with MPTCP and OpenFlow. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:* (pp. 1617-1624). IEEE.

[4] Wang, R., Butnariu, D., & Rexford, J. (2011, March). OpenFlow-based server load balancing gone wild. In *Proceedings of the 11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services* (pp. 12-12). USENIX Association.

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

[6] Lantz, Bob, Brandon Heller dan Nick McKeown. (2010). A Network in a Laptop: Rapid Prototyping for Software-defined Networks. Dalam Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. New York: ACM.

[7] IEEE. 802.1Q-1998 IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridge Local Area Networks,IEEE, 1998.

[8] Okayama, Kiyohiko., Yamai, Nariyoshi., Miyashita, Takuya., Kawano, Keita., Okamoto, Takuji., A Method of Dynamic Interconnection of VLANs for Large Scale VLAN Environment., Industrial & Management Engineering fields, Okayama University, 2005.

[9] UBAIDILLAH, AKHMAD FN., Simulasi Perancangan Teknologi VLAN Pada SMA Negeri 4 Yogyakarta Menggunakan Packet Tracer, Sekolah Tinggi Manejemen Informatika Dan Komputer, AMIKOM, Yogyakarta , 2011.

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

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

[12] M. McCauley, "pox," 1 May 2013. [Online]. Available: http://www.noxrepo.org/pox/about-pox/. [Accessed 2014 November 22].

.