ANALISIS QOS DENGAN VIRTUAL TENANT NETWORK PADA SOFTWARE DEFINE NETWORKING

Rakhmat Purnomo[1],
Teknik Informatika
Universitas Bhayangkara Jakarta Raya
rakhmat.purnomo@dsn.ubharajaya.ac.id

Prilly Rizky Arisandi[2]
Teknik Informatika
Universitas Bhayangkara Jakarta Raya
prilly@gmail.com

Abstract
The purpose of this study is to analyze the application of the Virtual Tenant Network function in OpenDaylight in the Software-Define Networking technology architecture. The main problem in building computer network infrastructure is that it is very dependent on network device vendors. The price of network devices is also relatively expensive and the compatibility of each device is also part of the problem that needs a solution. Open-source Software-Define Networking technology allows network infrastructure developers not to depend on network device vendors. The research method uses simulation. The service quality indicators (Quality of Services) tested include delay, jitter, throughput, and packet lost. The results of the study show that the quality of service is in the good category because the value is above the standard set by the International Telecommunication Union (ITU-T).

Keywords : QoS, Virtual Tenant Network, SDN

I. PENDAHULUAN

Sudah lebih dari 47 tahun internet digunakan untuk menghubungkan perangkat jaringan guna berkomunikasi dan berbagi informasi (Comer, 2009). Setiap tahun lebih dari 1 juta orang terhubung ke internet. Lahirnya teknologi *Internet of Things* (IoT) yang menyediakan konektifitas dan pengelolaan jarak jauh ke hampir semua perangkat melalui internet mengakibatkan peningkatan lalu lintas data (Zimmermann et al., 2015). Data ini dibagi menjadi 2 yaitu kontrol lalu lintas data dan data itu sendiri. Mengkonfigurasi router dan switch secara konvensional menjadi masalah yang kompleks. Dasar inilah yang mendorong untuk melakukan inovasi dalam mengelola perangkat jaringan.

Software-define Networking (SDN) merupakan pendekatan baru dalam bidang jaringan komputer yang merubah arsitektur perangkat jaringan seperti saklar dengan menyederhanakan struktur node yang kompleks (Ummah & Abdillah, 2016). SDN memberikan satu pengatur jaringan terpusat yang berbasis perangkat lunak dan dipisahkan dengan data nya. Banyak Controller yang digunakan dalam SDN. Penelitian ini menggunakan *Open Daylight.*

Salah satu penelitian yang dilakukan (Ummah & Abdillah, 2016) adalah dengan membangun simulasi jaringan virtual berbasis SDN. Tools yang digunakan Mininet. Skenario yang dibuat meliputi 2 switch, 4-switch, 8-switch, dan 16-switch. Hasil penelitiannya adalah simulasi jaringan virtual SDN telah bekerja dengan baik dengan parameter uji meliputi packet *loss*, *delay*, *jitter*, dan *throughput*.

Penelitian yang dilakukan (Kaur, Singh, & Ghumman, 2014) menunjukan emulator yang paling banyak digunakan untuk membangun SDN adalah mininet. Dengan *mininet*, pembuatan prorotype jairngan berskala besar secara mudah dilakukan seperti membuat *virtual host*, *switch controller*, dan *link*. *Mininet* mendukung untuk penelitian, pengembangan eksperimen, pembelajaran, *prototyping*, pengetesan, pencarian kesalahan, dan masih banyak pengembangan eksperimen yang bisa dilakukan di mininet dengan hanya menggunakan PC atau laptop.

Penelitian yang dilakukan oleh (Zoher Bholebawa & Dalal, 2016) membandingkan kinerja antara topologi bawaan pada mininet yaitu topologi *linier*, *single*, dan *tree*. Penelitian ini melakukan pengujian bandwidth pada setiap topologi yang kemudian dibandingkan. Hasil penelitian ini menunjukan topologi *tree* lebih baik dibandingkan dengan topologi lainnya.

Penelitian yang dilakukan (Asadollahi, Goswami, & Gonsai, 2017) menggunakan *OpenDaylight* sebagai kontroler. Penelitiannya membahas tahapan dalam penerapan kontroler Opendaylight dalam SDN. Versi yang digunakan adalah *Beryllium*-SR4 yang dirilis pada 26 Oktober 2016.

Berdasarkan kajian penelitian sebelumnya yang berkaitan dengan SDN, perlu di analisis salah satu fitur yang ada di SDN yaitu *Virtual Tenant Network* (VTN). Penerapan VTN pada SDN akan disesuaikan dengan topologi jaringan yang ada di Universitas Bhayangkara Jaya.

A. OpenDaylight

OpenDaylight merupakan sebuah kontroler SDN yang berlisensi terbuka atau *opensource* dan menggunakan bahasa pemrograman Java dan dikelola Linux Foundation dan didukung oleh lebih dari 40 perusahaan seperti IBM, Cisco, Juniper, VMWare, dan sejumlah vendor jaringan besar (Asadollahi et al., 2017).

Dalam website resminya, *OpenDaylight Foundation* selaku perusahaan yang mempromosikan *OpenDaylight Platform* menyatakan bahwa *OpenDaylight* (ODL) merupakan sebuah *platform* SDN terbuka untuk segala ukuran dan skalabilitas jaringan. ODL mengaktifkan layanan jaringan melalui sebuah *spectrum* perangkat keras dalam banyak *vendor*. Arsitekturnya membuat pengguna dapat mengendalikan aplikasi, protokol, dan berbagai macam *plugin*. ODL juga menyediakan koneksi antara pemakai luar dan penyedia layanan. Pengembangan ODL di kerjakan oleh sebuah komunitas yang besar yang memperbarui *platform* tersebut kurang lebih setiap enam bulan dan secara terus menerus dan menyesuaikan untuk mendukung rangkaian kasus penggunaan SDN dan *Network Funtions Virtualization* (NFV) yang paling luas di industri.

OpenDaylight memiliki banyak fitur atau *plugin* yang mudah untuk diaktifkan dan dinonaktifkan. Diantaranya ialah:

- AAA
- ALTO
- Border Gateway Protocol
- Border Gateway Monitoring Protocol (BMP)
- Control and Provisioning of Wireless Access Points (CAPWAP)
- Controller Shield
- Device Identification and Driver Management (DIDM)
- DLUX
- Fabric as a Service (FaaS)
- Group Based Policy
- Internet of Things Data Management
- Link Aggregation Control Protocol (LACP)
- Location Identifier Separation Protocol (LISP) Flow Mapping Service (LISP)
- NEMO
- NETCONF
- NetIDE
- OVSDB-based Network Virtualization Services
- OpenFlow plugin
- Path Computation Element Protocol (PCEP)
- Secure Network Bootstraping Interface (SNBi)
- Service Function Chaingin (SFC)
- SNMP Plugin
- SNMP4SDN
- Source-Group Tag Exchange Protocol (SXP)
- Topology Processing Framework
- Time Series Data Repository (TSDR)
- Unified Secure Channel
- Virtual Tenant Network (VTN)

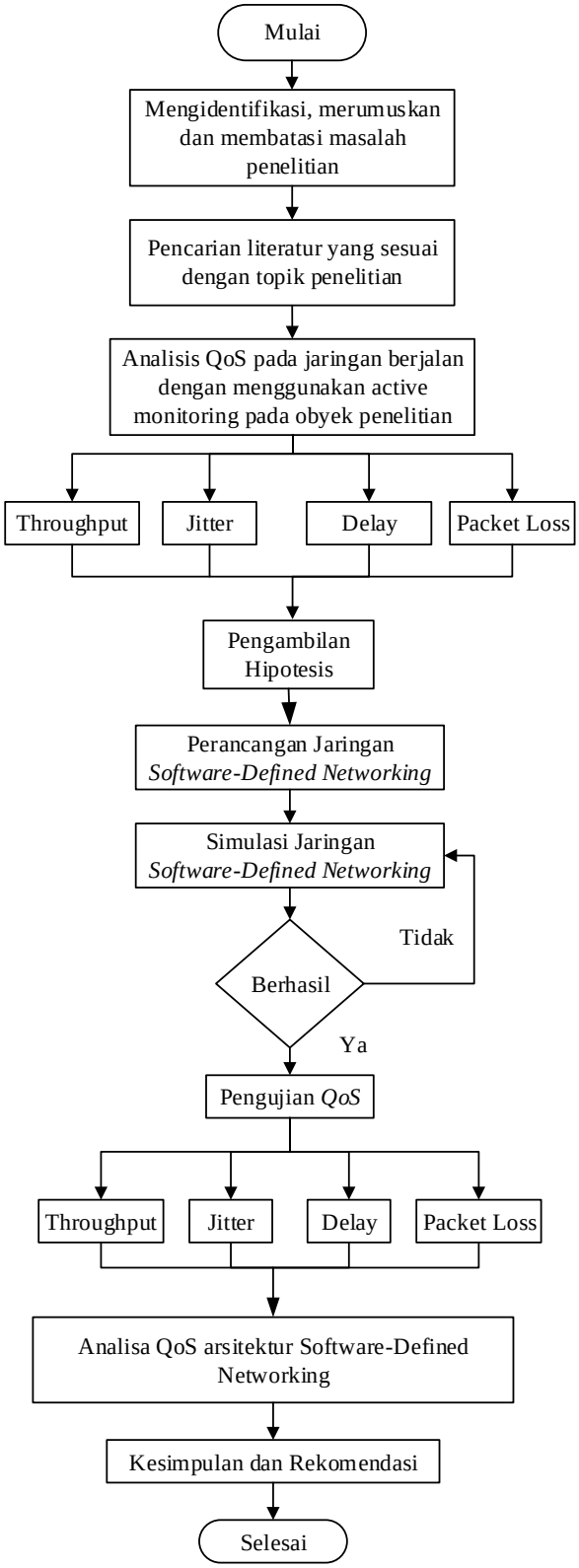
B. Virtual Tenant Network

Virtual Tenant Network (VTN) merupakan salah satu fitur dari *OpenDaylight* (Shin, Kang, Kwak, Lee, & Yang, 2014). VTN ini merupakan sebuah aplikasi yang menyediakan berbagai macam jaringan *tenant* pada sebuah kontroler SDN. Keunikan dari VTN ini ialah abstraksi secara logika. Yang membuat pemisahan antara topologi secara logika dan topologi secara fisik. VTN juga dapat mendefinisikan jaringan yang terlihat seperti jaringan konvensional L2/L3. Ketika jaringan didesain pada VTN, maka akan secara otomatis di mapping ke dalam jaringan fisik dan kemudian dikonfigurasikan pada setiap switch yang terhubung dengan kontroler SDN. Pendefinisian dari *logical plane* tidak hanya memungkinkan disembunyikannya kompleksitas jaringan tapi juga dapat lebih baik dalam memanajemen sumber daya jaringan. Hal ini akan mengurangi waktu pengkonfigurasian ulang layanan jaringan dan meminimalkan eror dalam konfigurasi. Ada dua komponen dari VTN yaitu VTN *Manager* dan VTN *Coordinator*. VTN *manager* adalah fitur yang berinteraksi dengan modul lain untuk mengimplementasikan model komponen VTN. VTN *Manager* juga menyediakan antarmuka *REST API* untuk mengkonfigurasi komponen VTN di *OpenDaylight*. Diantaranya membuat, memperbarui, dan menghapus komponen VTN. Untuk mengaktifkan fitur ini memerlukan plugin *odl-vtn-manager* dan *odl-vtn-manager-rest.*

II. METODOLOGI

A. Objek Penelitian

Metode penelitian dengan melakukan simulasi penerapan arsitektur *Software-Defined Networking* dengan protokol *OpenFlow* pada Universitas Bhayangkara Jakarta Raya beserta analisis kinerja jaringan Universitas Bhayangkara Jakarta Raya (UBJ) dan kinerja Arsitektur *Software-Defined Networking*. Gambar 1 menunjukan tahapan penelitian yang dilakukan.



Gambar 1. Tahapan Penelitian

Langkah – langkah dalam alur penelitiannya yaitu dimulai dari pengidentifikasian masalah dengan melakukan wawancara dan observasi pada tempat penelitian. Observasi dilakukan dengan melakukan percobaan kualitas layanan jaringan. Hasil dari observasi berupa gambar topologi jaringan pada UBJ yang masih menggunakan jaringan konvensional.

Kemudian peneliti melakukan kajian pustaka untuk mencari teori pendukung berdasarkan topik yang dikaji. Tahap berikutnya adalah melakukan *active monitoring* untuk mengetahui seberapa besar kualitas layanannya.

Berikutnya dilakukan perancangan usulan skema jaringan yang akan dibuat. Topologi yang dibuat sudah menggunakan arsitektur berbasis SDN. Rancangan ini kemudian diterapkan pada simulasi menggunakan komputer. Selanjutnya pengujian dilakukan dengan parameter *throughput, jitter, delay* dan *packet loss*. Hasilnya dibandinkan dengan arsitektur jaringan konvensional.

B. Peralatan penelitian

Peralatan hardware yang digunakan untuk membangun simulasi SDN dapat dilihat dari Tabel 1 berikut :

Tabel 1. Peralatan Penelitian

Spek	Mininet	Controller
Prosesor	Intel® Core™ I3 CPU 3.3 GHz	Intel® Core™ I3 CPU 3.3 GHz
RAM	4 GB	4 GB
Sistem Operasi	Ubuntu Desktop 16.04 64 bit	Ubuntu Server 14.04 LTS 64 bit
Fungsi	Untuk simulasi jaringan *Software-Defined Networking*	komputer untuk kontroler *Software-Defined Networking*

Sedangkan Perangkat Lunak yang digunakan antara lain :

1. OpenDaylight Boron 0.5.3 sebagai *SDN Controller.*
2. Mininet 2.2.1 sebagai emulator jaringan *Software-Defined Networking.*

Wireshark 2.0 sebagai program analisa paket data.

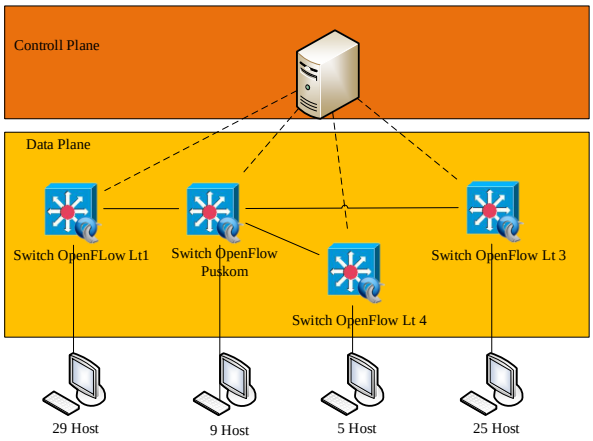
III. HASIL DAN PEMBAHASAN

A. Perancangan Topologi Arsitektur SDN

Topologi menggunakan arsitektur *Software-Defined Networking*, dalam penerapannya arsitektur ini menggunakan *Switch OpenFlow* sebagai *data plane* dan *controller* sebagai *control plane*. Topologi yang akan

diterapkan yaitu dengan spesifikasi sebagai berikut:

1. Jumlah Switch OpenFlow : 4 buah
2. *Controller* : 1 buah
3. Jumlah Host : 68 buah
4. Jumlah link : 71 link



Gambar 2. Topologi Skema Arsitektur SDN

B. Topologi Simulasi

Implementasi jaringan dengan arsitektur *Software-Defined Networking* (SDN) ini menggunakan metode simulasi. Jadi studi kasus penerapan arsitektur SDN ini menggunakan metode simulasi.

Skenario simulasi jaringan terlihat pada gambar 3.

Gambar 3. Skema Simulasi Jaringan

C. Konfigurasi OpenDaylight

Tahapan instal dan konfigurasi OpenDaylight diawalai dengan menentukan versi yang akan digunakan yaitu Boron versi 0.5.3 yang diinstal pada sistem operasi Ubuntu Server Desktop LTS 16.04. OpenDaylight ini menggunakan fitur kafar untuk menjalankan Opendaylight tanpa harus instalasi. Fitur kafar OpenDaylight dapat kapan saja dijalankan maupun dihentikan programnya.

Tahap 1 : Persiapan

Unduh OpenDaylight Boron-SR3. Langkah-langkah unduh pada website OpenDaylight adalah sebagai berikut :

1) Ketik url http://www.opendaylight.org/start pada browser. Terdapat banyak versi yang disediakan oleh OpenDaylight. Peneliti menggunakan versi Boron-SR3.
2) Kemudian klik Pre-BuiltZip.
3) Unggah berkas *OpenDaylight* tersebut ke server kontroler menggunakan aplikasi *ftp client*. Disini peneliti menggunakan *WinSCP FTP Client*. Selanjutnya buka aplikasi WinSCP tersebut.
4) Isi ***Hostname*** yaitu IP Address dari server kontroler dan **Username** serta **password**. Proses autentikasi server berlangsung. Kemudian pada aplikasi WinSCP cari file OpenDayligh yang telah terunduh.
5) Ungga ke dalam server kontroler
6) Setelah *OpenDaylight* berhasil terunggah. Selanjutnya ekstrak file zip *OpenDaylight* tersebut. Caranya ketikan perintah pada *console*. Proses ini dilakukan pada server kontroler.
   unzip distribution-karaf-0.5.3-Boron-SR3.zip
7) Selanjutnya install Java JDK versi 8 yang mendukung versi OpenDaylight tersebut.
   apt-get install oracle-java8-installer
8) Kemudian konfigurasi letak *JAVA_HOME*.
   export JAVA_HOME=/usr/lib/jvm/java-8-oracle

Tahap 2. Menjalankan OpenDaylight

Setelah mengunduh, mengekstrak *OpenDaylight* dan menyiapkan paket pendukung lainnya. Langkah selanjutnya adalah menjalankan *OpenDaylight.*

1) Caranya masuk ke dalam *direktori OpenDaylight.*
   cd distribution-karaf-0.5.3-Boron-SR3
2) Kemudian ketikkan perintah dibawah ini:
   bin/karaf clean -ofl3
3) Setelah berhasil menjalankan *OpenDaylight* selanjutnya masuk ke dalam *console OpenDaylight.*

Tahap 3. Mengaktifkan fitur-fitur yang dibutuhkan

Secara default setelah *OpenDaylight* dijalankan belum ada plugin atau fitur yang aktif. Fitur atau plugin tersebut diaktifkan secara manual. Pada penelitian ini fitur *OpenDaylight* yang akan diaktifkan adalah *dlux*, *l2switch* dan *vtn-manager*.

Caranya ketik pada *console OpenDaylight* : feature:install odl-dlux-all odl-l2switch-switch-ui odl-vtn-manager odl-vtn-manager-rest

OpenDaylight telah siap digunakan langkah selanjutnya akses Kontroler SDN melalui *browser*

dengan mengakses
http://192.168.5.2:8181/index.html

D. Konfigurasi *Mininet*

Mininet merupakan sebuah emulator yang bisa mensimulasikan jaringan dengan arsitektur jaringan berbasis *Software-Defined Networking* yang bisa menciptakan *host* seperti nyatanya, dan komponen perangkat jaringan lainnya seperti *Open vSwitch* yang mendukung *Switch Openflow* (Pambudi & Wibowo, 2015). Dengan mininet ini kita bisa mempunyai arsitektur jaringan seprti aslinya. Proses ini dilakukan pada PC *client* yang akan dijadikan tempat untuk *mininet*.

Tahap 1. Persiapan

Untuk memulai pengunduhan mininet dari sumber utama, langkah – langkahnya sebagai berikut :

1) ketik command dibawah berikut pada terminal.
   git clone git://github.com/mininet/mininet
2) Selanjutnya masuk ke dalam *direktori* mininet. Dengan mengetikan:
   cd mininet
3) Kemudian cek versi mininet yang tersedia.
   git tag
4) Setelah itu kita keluar dari *direktori mininet*. Dan kita jalankan perintah installasi mininet.
   mininet/util/install.sh -a
5) Setelah berhasil terinstall mininet siap dijalankan.

Tahap 2. Menjalankan Mininet

Setelah melakukan proses installasi *mininet* dan sebelum menjalankan *mininet*, peneliti membuat *script* yang telah menyesuaikan topologi jaringan UBJ. *Script* tersebut digunakan untuk menyesuakan topologi yang digunakan. Tahap selanjutnya adalah menjalankan *script* mininet tersebut.

```
mn –topo mytopo –custom=topo_ubj.py –controller remote,ip=192.168.5.2 –switch ovsk,protocols=OpenFlow13
```

Setelah *mininet* dijalankan maka akan tampil semua host dan switch berdasarkan yang telah dibuat dalam script mininet.

Konfigurasi Virtual Tenant Network

Peneliti menerapkan *Virtual Tenant Network* yaitu pada *Switch OpenFlow* LT1 dengan *switch OpenFlow* LT4. Peneliti mencoba membuat topologi virtual. Sehingga kedua *host* antara sisi *switch OpenFlow* LT1 dengan *host* di switch OpenFlow LT4 akan terhubung langsung secara logika. Konfigurasi ini dilakukan setelah *OpenDaylight* dan mininet dijalankan.

Langkah-langkah konfigurasi VTN adalah sebagai berikut:

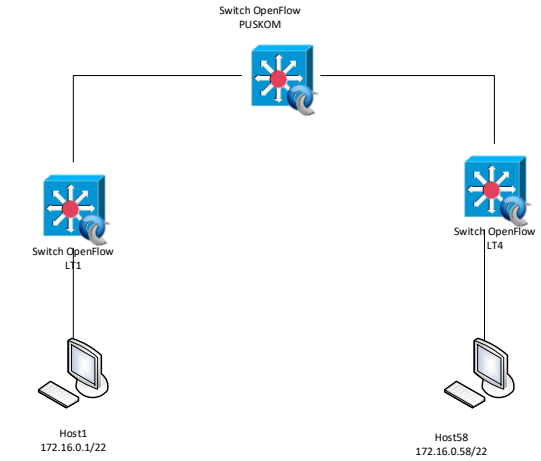
1. Pertama buat sebuah vtn. Langkah konfigurasi VTN ini menggunakan *vtn manager* pada *OpenDaylight*.
2. Selanjutnya buat *virtual bridge* pada VTN.
3. Kemudian buat 2 buah *virtual interface* pada *virtual bridge* dengan nama port1 dan port2.
4. Langkah terakhir adalah mapping port fisik dari *Open vSwitch* pada mininet ke *virtual interface* pada *virtual bridge*

Setelah semua konfigurasi selesai dilakukan selanjutnya verifikasi bahwa VTN telah dibuat. Caranya adalah dengan mengakses link **http://192.168.5.2:8181/restconf/operation/vtn:vtns** pada web browser.

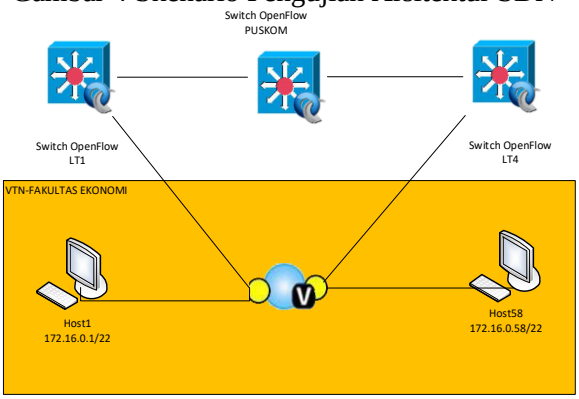
Setelah itu pilih dan klik **Show/Hide**, selanjutnya klik **Try it out** maka akan ditampilkan konfigurasi VTN yang telah dibuat.

E. Pengujian dan Analisis

Pengujian dilakukan dengan 2 skenario yaitu yang pertama tanpa menggunakan VTN dan yang kedua dengan menggunakan VTN.



Gambar 4 Skenario Pengujian Arsitektur SDN

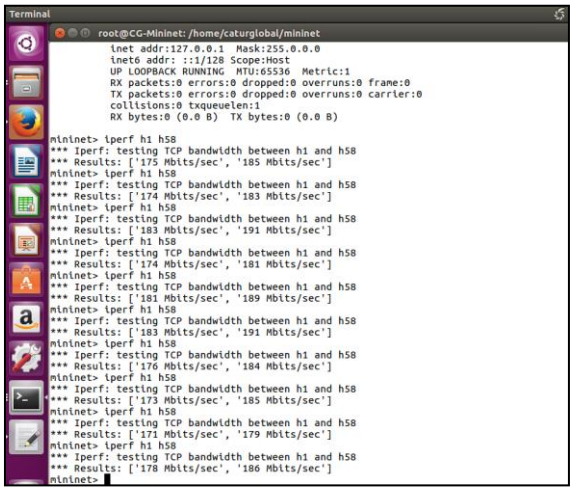


Gambar 5 Skenario Pengujian Arsitektur SDN dengan VTN

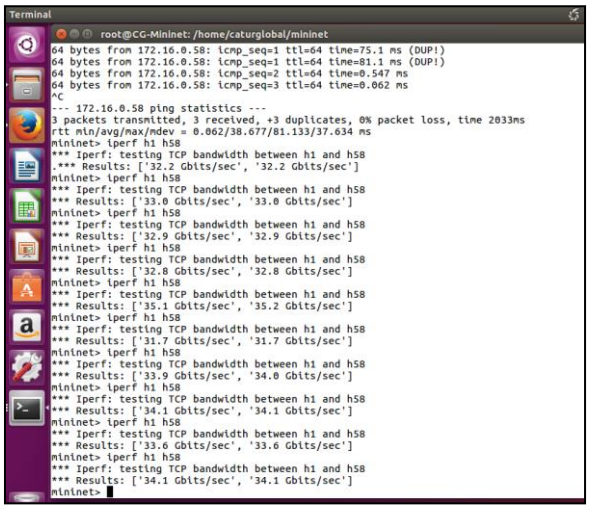
Pengujian kualitas layanan jaringan pada arsitektur *Software-Defined Networking* dengan parameter *throughput*, *delay*, *jitter* dan *packet loss* dengan skema penerapan VTN dan non VTN.

1. Pengujian *Throughput*

Pada pengujian *throughput* peneliti menggunakan *tools iperf* yang merupakan *tool* yang sudah disediakan di dalam mininet. Pada pengujian throughput ini peneliti melakukan iperf dari *host1* ke *host58* untuk mengatahui nilai throughput yang dihasilkan. Untuk proses pengujian bisa dilihat pada gambar 6 dan gambar 7.



Gambar 6 Pengujian *throughput* tanpa VTN



Gambar 7 Pengujian *throughput* dengan VTN

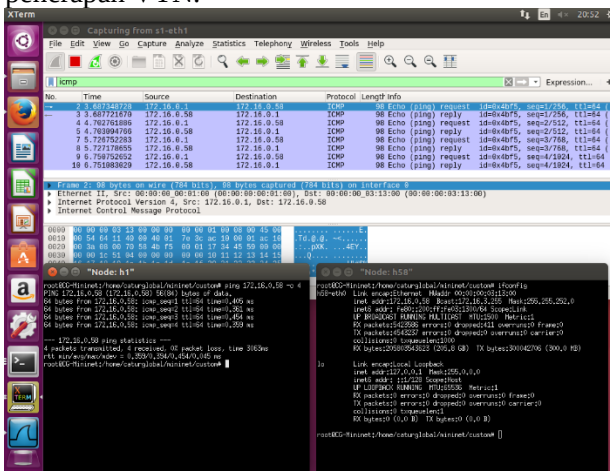
Tabel 2. Perbandingan nilai throughput yang menerapkan VTN dan yang tidak menerapkan VTN

Nomer Per cobaan	Nilai Throughput		
	Jaringan Konvensional (Mbits/sec)	Non VTN (Mbits/ sec)	VTN (Gbits/sec)
P 1	29.47	175	32.2
P 2	47.51	174	33
P 3	65.93	183	32.9
P 4	76.25	174	32.8
P 5	86.8	181	35.1
P 6	92.19	183	31.7
P 7	91.31	176	33.9
P 8	92.01	173	34.1
P 9	91.28	171	33.6
P 10	95.25	178	34.1
Rata-Rata	76.8	176.8	33.34

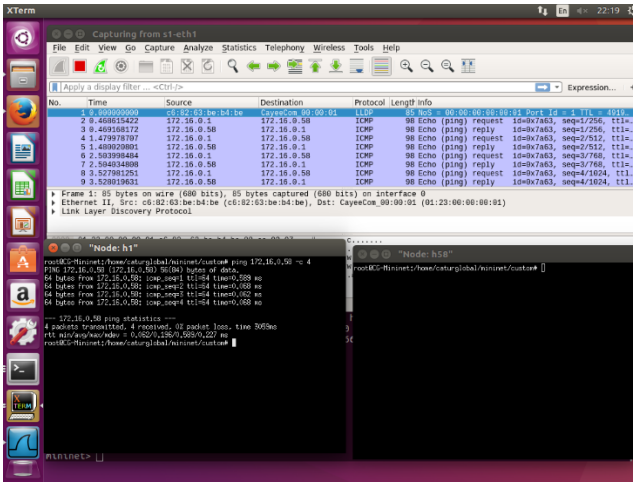
Dari tabel 2 dapat dianalisis bahwa jaringan SDN dengan skenario pengujian menggunakan VTN mampu membawa bandwidth hingga Gbits/sec sedangkan tanpa VTN hanya bisa membawa bandwidth Mbits/sec.

2. Pengujian Delay

Peneliti melakukan uji ping dari *host1* ke *host58*. Untuk mendapatkan nilai *delay*, dilihat pada waktu pengiriman dan waktu penerimaan paket dengan menggunakan aplikasi *wireshark*. Untuk hasil uji coba *delay* tanpa menerapkan VTN bisa dilihat pada tabel 3 dan untuk penerapan VTN bisa dilihat pada tabel 4. Peneliti memberikan sampel proses pengujian delay seperti pada gambar 8 untuk non VTN dan gambar 9 untuk penerapan VTN.



Gambar 8. Proses pengujian delay pada non VTN



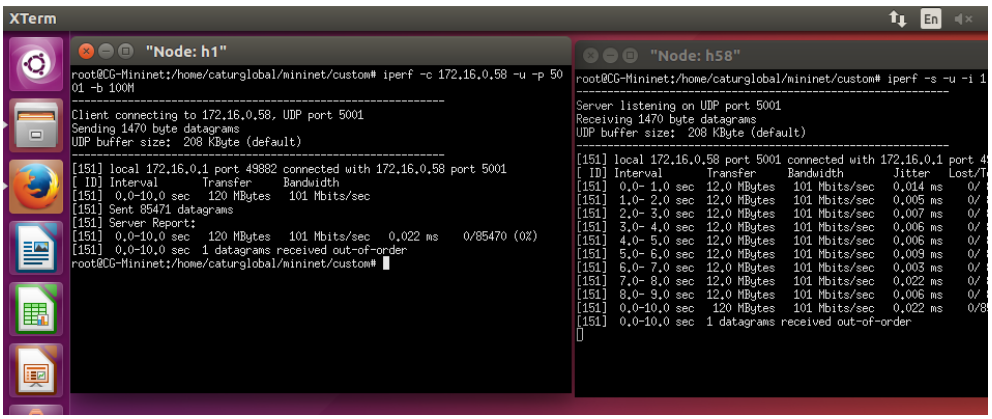
Gambar 9 Proses pengujian *delay* pada VTN

Tabel 3. Hasil pengujian delay tanpa menerapkan VTN

Nomer Per cobaan	Waktu Kirim (s)	Waktu Terima (s)	Delay (s)	Delay (ms)
P 1	3.687348728	3.68772167	0.000373	0.372942
P 2	0	0.000864378	0.000864	0.864378
P 3	1.632311722	1.632815506	0.000504	0.503784
P 4	1.662460565	1.663448443	0.000988	0.987878
P 5	2.118494402	2.11950914	0.001015	1.014738
P 6	0.502089305	0.502782077	0.000693	0.692772
P 7	1.628222224	1.628999165	0.000777	0.776941
P 8	0.604353737	0.605275029	0.000921	0.921292
P 9	0	0.001374354	0.001374	1.374354
P 10	1.844277731	1.845407692	0.00113	1.129961
Rata-rata				0.863904

Tabel 4 Hasil pengujian delay dengan menerapkan VTN

Nomer Per	Waktu Kirim (s)	Waktu Terima (s)	Delay (s)	Delay (ms)
P 1	0.468615422	0.469168172	0.00055275	0.55275
P 2	0.2398074	0.240417311	0.000609911	0.609911
P 3	0	0.000550951	0.000550951	0.550951
P 4	4.391603886	4.391995865	0.000391979	0.391979
P 5	0	0.000572681	0.000572681	0.572681
P 6	0	0.000421018	0.000421018	0.421018
P 7	5.126731429	5.127309789	0.00057836	0.57836
P 8	0	0.000604257	0.000604257	0.604257
P 9	3.766665757	3.767080451	0.000414694	0.414694
P 10	0	0.000056941	0.000056941	0.056941
Rata-rata				0.475354



Gambar 10. Sampel proses uji coba *jitter* tanpa penerapan VTN

Berdasarkan tabel 3 dan tabel 4 dapat disimpulkan bahwa rata-rata *delay* pada penerapan VTN menghasilkan nilai yang lebih kecil dari non VTN. Meskipun begitu VTN dan non VTN sama-sama masih menghasilkan nilai diatas standar ITU-T.

3. Pengujian Jitter

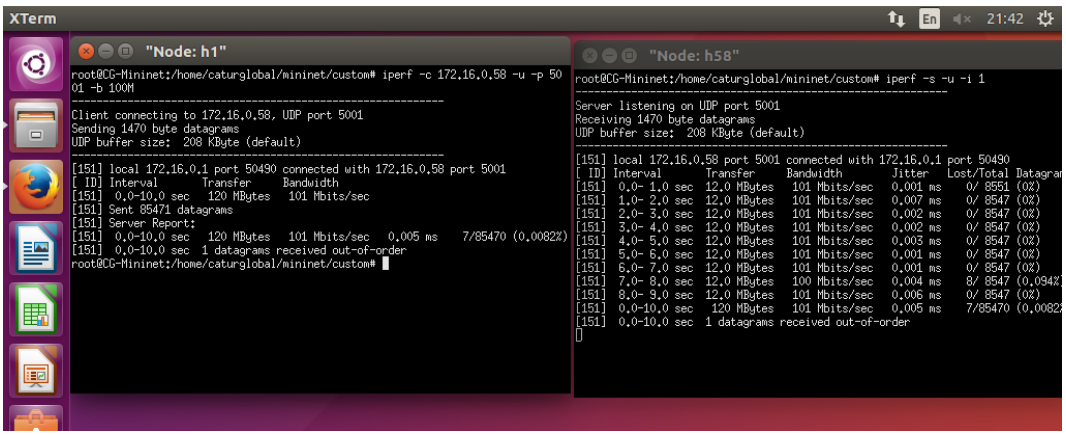
Pengujian *jitter* dilakukan menggunakan *iperf* dengan protocol uji coba adalah protokol UDP dengan maksimal bandwidth 100Mbps. Untuk proses uji coba bisa dilihat pada gambar 10.

Setelah melakukan uji coba *jitter* maka didapatkan nilai jitter pada 10 kali percobaan yang tersusun pada tabel 5.

Tabel 5. Hasil pengujian jitter pada SDN non VTN

Nomer Percobaan	Nilai Jitter (ms)
P 1	0.022
P 2	0.005
P 3	0.006
P 4	0.002
P 5	0.003
P 6	0.008
P 7	0.003
P 8	0.003
P 9	0.008
P 10	0.002
Rata-rata	0.0062

Selanjutnya pada gambar 11 menunjukan proses ujicoba jitter pada SDN yang menerapkan VTN



Gambar 11. Sampel proses uji coba *jitter* dengan penerapan VTN

Tabel 6. Hasil pengujian *jitter* pada SDN VTN

Nomer Percobaan	Nilai Jitter (ms)
P 1	0.005
P 2	0.003
P 3	0.003
P 4	0.004
P 5	0.003
P 6	0
P 7	0.001
P 8	0.006
P 9	0.002
P 10	0.004
Rata-rata	0.0062

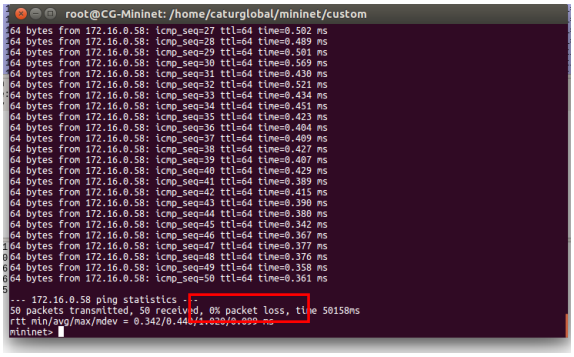
Dari tabel 6. dapat disimpulkan bahwa uji coba *jitter* pada arsitektur SDN yang menerapkan VTN menghasilkan nilai-nilai *jitter* yang lebih kecil dibandingkan yang tidak menerapkan SDN maupun pada jaringan konvensional. Namun hasil *jitter* tersebut sudah diatas standar bagus ITU-T.

4. Pengujian *Packet Loss*

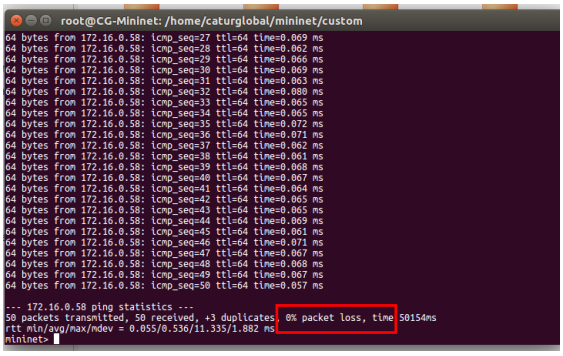
Pengujian ini dilakukan percobaan *ping* dengan parameter *count* sebanyak 50. Nilai *packet loss* yang dihasilkan untuk kedua skenario adalah sama. Sehingga tidak ada perbedaan yang signifikan.

Gambar 12 menunjukan proses pengujian *packet lost* pada arsitektur SDN yang belum menerapkan VTN dimana dihasilkan 0% packet lost. Sedangkan gambar 13 menunjukan pengujian packet lost pada arsitektur SDN yang sudah menerapkan VTN.

Jika pengujian packet lost dilakukan pada dunia nyata maka akan didapat hasil yang signifikan karena banyak faktor yang mempengaruhinya



Gambar 12 Pengujian *packet loss* pada arsitektur SDN yang belum menerapkan VTN



Gambar 13 Pengujian *packet loss* pada arsitektur SDN yang sudah menerapkan VTN

IV. KESIMPULAN

Kesimpulan penelitian ini adalah terdapat peningkatan layangan QoS pada jaringan yang menggunakan arsitektur SDN dibandingkan jika menggunakan arsitektur konvensional, walaupun nilai keduanya masih diatas standar ITU-T. Pengujian *throughput* menghasilkan kemampuan arsitektur SDN dengan VTN mampu membawa bandwidth hingga Gbits/s sedangkan tanpa SDN-VTN hanya bisa membawa bandwidth Mbits/sec. Pengujian *delay* menghasilkan nilai lebih kecil untuk arsitektur SDN-VTN dibandingkan dengan arsitektur konvensional tetapi tetap masih diatas standar ITU-T. Sedangkan pengujian *packet loss* tidak terdapat perbedaan signifikan.

V. DAFTAR PUSTAKA

Asadollahi, S., Goswami, B., & Gonsai, A. M. (2017). Implementation of SDN using OpenDayLight Controller, *5*(2), 218–227.

Comer, D. E. (2009). *Computer Networks and Internets* (Fifth). PEARSON Prentice Hall.

Kaur, K., Singh, J., & Ghumman, N. S. (2014). Mininet as Software Defined Networking Testing Platform. *International Conference on Communication, Computing & Systems (ICCCS–2014)*, 3–6.

Pambudi, W., & Wibowo, F. W. (2015). Uji Throughput Kontroler Floodlight dan Beacon Menggunakan Emulator Mininet. *Universitas Gajah Mada*.

Shin, Y. Y., Kang, S. H., Kwak, J. Y., Lee, B. Y., & Yang, S. H. (2014). The Study on Configuration of Multi-Tenant Networks in SDN Controller. *Electronics and Telecommunications Research Institute*,

3(2), 16–19.

Ummah, I., & Abdillah, D. (2016). Perancangan Simulasi Jaringan Virtual Berbasis Software-Define Networking. *Jounal on Computing*, *1*(1), 95–106. http://doi.org/10.21108/indojc.2016.1.1.20

Zimmermann, A., Schmidt, R., Sandkuhl, K., Wißotzki, M., Jugel, D., & Möhring, M. (2015). Digital Enterprise Architecture - Transformation for the Internet of Things -, 130–138. http://doi.org/10.1109/EDOCW.2015.16

Zoher Bholebawa, I., & Dalal, U. D. (2016). Design and Performance Analysis of OpenFlow-Enabled Network Topologies Using Mininet. *International Journal of Computer and Communication Engineering 5.6*, *5*(6), 419. http://doi.org/10.17706/ijcce.2016.5.6.419-429