

**PEMANFAATAN MININET UNTUK MEDIA PEMBELAJARAN
SOFTWARE-DEFINED NETWORKING**

Laporan Tugas Akhir

Disusun sebagai syarat kelulusan Mata Kuliah Tugas Akhir II

Oleh

RIFKIANSYAH MEIDIAN CAHYAATMAJA

NIM : 13511084



**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO & INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG**

Agustus 2017

**PEMANFAATAN MININET UNTUK MEDIA PEMBELAJARAN
SOFTWARE DEFINED NETWORKING**

Laporan Tugas Akhir

Oleh

RIFKIANSYAH MEIDIAN CAHYAATMAJA

NIM : 13501157

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

Telah disetujui dan disahkan sebagai Laporan Tugas Akhir
di Bandung, pada tanggal 22 Agustus 2017

Tim Pembimbing,

Pembimbing I,

Ir. Afwarman Manaf, M.Sc., Ph.D.

NIP. 131803257

LEMBAR PERNYATAAN

Dengan ini saya menyatakan bahwa:

1. Pengerjaan dan penulisan Laporan Tugas Akhir ini dilakukan tanpa menggunakan bantuan yang tidak dibenarkan.
2. Segala bentuk kutipan dan acuan terhadap tulisan orang lain yang digunakan di dalam penyusunan laporan tugas akhir ini telah dituliskan dengan baik dan benar.
3. Laporan Tugas Akhir ini belum pernah diajukan pada program pendidikan di perguruan tinggi mana pun.

Jika terbukti melanggar hal-hal di atas, saya bersedia dikenakan sanksi sesuai dengan Peraturan Akademik dan Kemahasiswaan Institut Teknologi Bandung bagian Penegakan Norma Akademik dan Kemahasiswaan khususnya Pasal 2.1 dan Pasal 2.2.

Bandung, 22 Agustus 2017

Rifkiansyah Meidian C.
NIM 13512602

ABSTRAK

Pemanfaatan Mininet Untuk Media Pembelajaran Software Defined Networking

Oleh

Rifkiansyah Meidian C.

NIM: 13511084

Jaringan komputer tentunya sudah tidak asing bagi para pengguna internet maupun pengguna komputer yang menggunakan jaringan lokal tanpa tersambung ke internet. Sayangnya, jaringan komputer adalah hal yang rentan akan perubahan. Perubahan sedikit saja dalam sistem jaringan dapat menghasilkan kerusakan beruntun yang mengakibatkan lumpuhnya seluruh jaringan atau setidaknya menurunnya kinerja jaringan secara keseluruhan. Beberapa metoda telah dibuat untuk meringankan kinerja jaringan, seperti *routing*, *load balancing*, dan *backup server* untuk memungkinkan jaringan tetap berjalan ketika dilakukan perubahan terhadap jaringan. Meski begitu, hal tersebut tidak menutup kelemahan yaitu ketika jaringan diubah, tidak ada jaminan bahwa jaringan dapat bekerja seperti teori yang telah dibuat sebelumnya ketika dijalankan. Jika terjadi kesalahan setelah perubahan diaplikasikan, maka bisa jadi kinerja jaringan akan lebih buruk dari sebelumnya. Pada tugas akhir ini dibahas mengenai aplikasi *virtual machine* **Mininet** untuk **Software Defined Networking** dan penggunaannya dalam menciptakan purwarupa jaringan untuk mensimulasikan perilaku jaringan. Dalam penggunaan *Software Defined Networking* secara umum, pengguna perlu membuat beberapa *host*, menggunakan *controller*, dan kemudian menjalankan jaringan. Bagaimanapun, pembelajaran perangkat Software-Defined Networking tidaklah mudah. Banyak tutorial yang telah dibuat, namun tidak semuanya dapat secara efektif mengajarkan penggunaan perangkat dan konsep Software-Defined Networking. Dalam tugas akhir ini akan dibuat implementasi sederhana dari *Mininet* yang dapat membantu pengguna pemula untuk mempelajari konsep-konsep *Software Defined Network*.

KATA PENGANTAR

Puji Syukur saya panjatkan kepada Allah SWT karena rahmat dan bimbingan-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul “Pemanfaatan Virtual Machine untuk Pembelajaran Software Defined Networking”.

Terima kasih yang sebanyak-banyaknya saya ucapkan untuk pak Ir. Afwarman Manaf, MsC. PhD. selaku pembimbing Tugas Akhir saya. Tanpa beliau saya tidak akan dapat menyelesaikan Tugas Akhir ini. Terima kasih juga saya haturkan kepada Dr. Ir Rinaldi Munir, MT. selaku dosen wali saya selama menjadi mahasiswa di Teknik Informatika.

Tidak lupa saya haturkan terima kasih yang sebanyak-banyaknya kepada kedua orang tua saya yang selalu mempercayai keputusan putra sulungnya.

Meski saya membutuhkan waktu lebih lama dari sebagian besar rekan-rekan saya, namun akhirnya saya dapat menyelesaikan studi ini bersama dengan rekan-rekan saya yang tersisa. Sungguh sebuah kehormatan berjuang bersama mereka.

Tugas akhir ini dibuat sebagai syarat lulus tingkat sarjana di Program Studi Teknik Informatika, Institut Teknologi Bandung.

DAFTAR ISI

Abstrak.....	v
KATA PENGANTAR.....	i
DAFTAR ISI.....	ii
DAFTAR GAMBAR.....	v
DAFTAR TABEL.....	vi
DAFTAR ISTILAH.....	vii
Bab I PENDAHULUAN.....	1
I.1. Latar Belakang.....	1
I.2. Rumusan Masalah.....	2
I.3. Tujuan.....	3
I.4. Batasan Masalah.....	3
I.5. Metodologi.....	3
Bab II STUDI LITERATUR.....	5
II.1 Software Defined Network.....	5
II.1.1 Arsitektur Software Defined Network.....	6
II.1.2 Controller.....	7
II.2 MiniNet.....	8
II.2.1 Alur Kerja Mininet.....	9
II.2.1.1 Penciptaan jaringan.....	9
II.3 OpenFlow.....	14
II.3.1 Arsitektur OpenFlow.....	14
II.3.1.1 <i>Flow Modification</i>	15

II.3.1.2 Instruksi.....	15
II.4 Open vSwitch.....	16
II.4.2 Arsitektur Open vSwitch.....	16
II.4.2 Konfigurasi.....	17
II.4.3 Fungsi Dasar Open vSwitch.....	17
II.5 Tkinter.....	17
II.6 Pendekatan Sejenis.....	19
II.6.1 Ivan Kupalov SDN Case Study.....	19
II.6.2 Miniedit.....	19
Bab III ANALISIS DAN PERANCANGAN.....	21
III.1 Analisis Masalah.....	21
III.1.1 Pertimbangan Solusi.....	21
III.2 Perancangan Solusi.....	22
III.2.1 Arsitektur Solusi.....	23
III.3 Tampilan Program.....	24
III.3.1 Library Tkinter.....	25
BAB IV IMPLEMENTASI DAN PENGUJIAN.....	26
IV.1.1 Menu Topologi.....	26
IV.1.2 Menu Tutorial.....	27
IV.1.3 Toolbox.....	27
IV.1.4 Menu Bantuan.....	27
IV.4 Tujuan Pengujian.....	28
IV.5 Metode Uji.....	29
IV.6 Hasil.....	30
IV.6.1 Uji Kasus I.....	30

IV.6.1 Uji Kasus II.....	32
IV.7 Evaluasi.....	33
IV.7.1 Uji Kasus 1.....	33
IV.7.1 Uji Kasus 2.....	34
BAB V KESIMPULAN DAN SARAN.....	35
V.1 Kesimpulan.....	35
V.2. Saran.....	35
Lampiran I: Workflow Program.....	37
Lampiran 2: Daftar Kelas dan Fungsi.....	38
Lampiran 3: Langkah Kasus Uji I dan Form Pertanyaan.....	42

DAFTAR GAMBAR

Gambar 1: Arsitektur umum Software Defined Network (A Survey on Software Defined Network, 2015).....	6
Gambar 2: Virtual Network mininet menggunakan OpenFlow dan virtual Ethernet (veth).....	9
Gambar 3: Pengaktifan Mininet dengan topologi minimal, 2 host dan 1 switch.	10
Gambar 4: Informasi node h1 yang ditampilkan Mininet.....	10
Gambar 5: Ping dari h1 menuju h2.....	11
Gambar 6: Penggunaan opsi '-v debug' pada pengaktifan Mininet.....	11
Gambar 7: Contoh kode Mininet Python API, menggunakan OVSController.....	12
Gambar 8: Contoh spesifikasi jaringan Python API.....	13
Gambar 9, Arsitektur umum OpenFlow(SDN: Software Defined Network, 2013)	15
Gambar 10, Arsitektur umum OpenVSwitch (Extending Networking Into the Virtualization Layer, 2009).....	16
Gambar 11, Komponen Open vSwitch (Contributing to OpenFlow Support, 2012)	17
Gambar 12: Tampilan Miniedit, http://www.brianlinkletter.com/how-to-use-miniedit-mininets-graphical-user-interface/	20
Gambar 13: Implementasi Mininet API dengan UI.....	26
Gambar 14: Menu Topologi.....	26
Gambar 15: Menu Tutorial.....	27
Gambar 16: Menu Toolbox.....	27
Gambar 17: Menu Bantuan.....	27

DAFTAR TABEL

Tabel 1: Daftar Widget Tkinter[10].....	18
Tabel 1: Langkah Uji Kasus 2.....	33

DAFTAR ISTILAH

Istilah	Arti
<i>Node</i>	Obyek dalam sebuah jaringan. Dapat berupa <i>Host</i> , <i>Switch</i> , atau <i>Controller</i> dan terhubung satu sama lain di dalam jaringan.
<i>Host</i>	Sebuah komputer atau alat lainnya yang terhubung ke sebuah jaringan komputer.
<i>Switch</i>	Sebuah alat jaringan komputer yang menghubungkan alat-alat lainnya dalam jaringan komputer menggunakan protokol-protokol jaringan.
<i>Switching</i>	<i>Switching</i> atau <i>packet switching</i> adalah metode komunikasi jaringan digital. Data yang dikirimkan dikelompokkan kedalam bagian berukuran tertentu yang disebut <i>packet</i> yang dapat dibagi lebih dari satu sesi komunikasi secara bersamaan.
<i>Forwarding</i>	<i>Forwarding</i> atau <i>packet forwarding</i> adalah penyampaian paket dari satu segmen jaringan ke segmen lainnya melewati simpul dalam jaringan komputer.
<i>Hypervisor</i>	Sebuah bagian dari peranti lunak komputer yang menciptakan dan menjalankan <i>virtual machine</i>
<i>Northbound</i>	Sebuah antarmuka dimana konsep detil tingkat rendah seperti data atau fungsi digunakan oleh atau didalam komponen.
<i>Southbound</i>	Bagian antarmuka yang mengatur protokol untuk komponen jaringan yang berupa perangkat keras

<i>Datapath</i>	Lapisan yang berisi kumpulan unit-unit fungsional yang melakukan operasi, <i>register</i> , dan penyambungan proses data untuk pengiriman paket data.
<i>Controller</i>	Lapisan yang membuat keputusan mengenai bagaimana paket harus dikirimkan ke tujuan.

BAB I

PENDAHULUAN

I.1. Latar Belakang

Jaringan komputer adalah sebuah hal yang rumit. Untuk membuat sebuah jaringan komputer dibutuhkan banyak sumber daya termasuk perangkat keras, perangkat lunak, serta berbagai macam protokol. Satu jaringan kecil saja dapat memiliki belasan protokol yang harus diikuti oleh setiap perangkat didalam jaringan tersebut. Hal ini membuat efisiensi kerja jaringan selalu lebih rendah daripada seharusnya, ditambah dengan kualitas dari perangkat keras yang juga tidak selalu berada pada kondisi prima.

Hingga beberapa tahun yang lalu, penyimpanan, komputasi, serta sumber daya jaringan dijaga terpisah satu sama lainnya. Sekitar 10 tahun yang lalu muncul sebuah teknologi yang memungkinkan sebuah sistem operasi *host* untuk mengeksekusi satu atau lebih sistem operasi klien. Program pencipta *virtual machine* ini disebut sebagai *hypervisor* atau *virtual machine monitor*.

Saat ini, perkembangan dari program di atas hari ini telah menjelma menjadi apa yang disebut *Software Defined Network*. Pendukung awal dari SDN melihat bahwa perangkat jaringan yang ada tidak memenuhi kebutuhan mereka di bagian pengembangan fitur dan ruang inovasi^[1]. Peralatan *switching* tingkat tinggi juga terlampau mahal pada saat itu.

Sebelum kemunculan SDN telah dibuat beberapa kemajuan dalam bidang *programmable network*. Salah satu contohnya adalah konsep *active networking* yang mencoba untuk mengendalikan jaringan secara *real-time* menggunakan piranti lunak. Beberapa perangkat *routing* dalam piranti keras PC konvensional juga mencoba untuk menciptakan *software router* dengan membuat peralatan jaringan dapat diprogram^[2].

Kemajuan lainnya adalah *decoupling* antara kendali dan bidang data, dimana modul kendali dan modul data dibuat tidak terlalu tergantung satu sama lain, meningkat dalam 10 tahun terakhir. Dengan saling ketidak-terkaitan antara kendali jaringan dan bidang data, SDN memiliki kendali yang lebih besar terhadap jaringan lewat programming. Kombinasi dari kedua kemajuan diatas dapat memberikan konfigurasi dan kinerja yang lebih baik serta inovasi dalam arsitektur dan operasi jaringan.

Dalam SDN, diketahui ada tiga tantangan utama^[2]. Pertama, kurangnya driver untuk memudahkan penggunaan *platform* SDN yang sudah ada. Hal ini menyulitkan penggunaan berbagai *platform* SDN dan memaksa pengguna untuk menggunakan platform SDN dengan menggunakan fungsi dasar *platform* SDN yang memakan waktu lama untuk dipelajari.

Tantangan kedua adalah kurangnya API *northbound* untuk pengembangan SDN. *Northbound* dalam hal ini adalah bagian dari SDN yang berinteraksi dengan lapisan aplikasi dalam arsitektur SDN. API ini menyediakan antarmuka yang memudahkan pengguna untuk menggunakan SDN.

Tantangan ketiga adalah kurangnya sebuah bahasa pemrograman tingkat tinggi yang dapat digunakan untuk pengembangan SDN. Hal ini memaksa pengguna menggunakan SDN secara manual hanya dengan perintah-perintah yang disediakan oleh platform SDN dan mengurangi kemungkinan pengembangan lebih lanjut.

Dalam menghadapi ketiga tantangan tersebut, diperlukan lebih banyak pengetahuan mengenai SDN dimana prinsip-prinsip dasar SDN perlu dipelajari oleh siapapun dengan minat baik untuk sekadar mengetahui kulit luar SDN maupun sebagai jalan masuk untuk mempelajari SDN lebih lanjut.

I.2. Rumusan Masalah

Berdasarkan latar belakang diatas, maka diperlukan sebuah media pembelajaran SDN dimana pengguna dapat bereksperimen dengan prinsip-prinsip dasar SDN yang diketahui tanpa secara langsung berinteraksi dengan perangkat-perangkat SDN seperti ***Controller*** dan ***Hypervisor***.

Saat ini ada beberapa platform serta perangkat SDN yang berkembang, seperti **Mininet** dan **Open vSwitch**. Belum ada SDN yang dapat dikonfigurasi dengan cukup mudah untuk menghasilkan sebuah generator praktis yang dapat menciptakan sebuah jaringan virtual. Oleh karena itu, maka rumusan masalah dari tugas akhir ini adalah sebagai berikut:

1. Bagaimana cara mengkonfigurasi peralatan SDN yang tersedia secara umum? Dalam hal ini, peralatan yang digunakan adalah Mininet dan Open vSwitch
2. Apa saja prinsip dasar SDN yang perlu dipelajari untuk mulai masuk kedalam ranah SDN?

3. Dapatkah dibuat sebuah media untuk pembelajaran SDN tingkat dasar untuk mengetahui prinsip dasar serta konfigurasi peralatan SDN pada umumnya?

I.3. Tujuan

1. Membuat sebuah antarmuka untuk mempermudah pembelajaran konsep *Software-Defined Network*.
2. Membandingkan efektivitas pembelajaran antara penggunaan antarmuka dengan penggunaan perangkat Software-Defined Networking secara alami.

I.4. Batasan Masalah

Topik Tugas Akhir akan dibatasi dalam pemanfaatan Mininet API sebagai platform pembelajaran konsep-konsep *Software-Defined Network*. Program yang bekerjasama dengan Mininet seperti ovs-ofctl tidak akan secara luas dibahas.

I.5. Metodologi

Langkah-langkah dalam pelaksanaan tugas akhir ini adalah sebagai berikut.

1. Analisis Pembangunan Model Lingkungan Pembelajaran

Pada tahap ini dilakukan proses analisis untuk mendefinisikan solusi tiap permasalahan yang diangkat. Dilakukan pembahasan masalah lebih detil dan solusi tiap permasalahan. Didapatkan solusi sebagai berikut:

1. Mekanisme penciptaan *virtual physical host*.
2. Mekanisme jaringan antar *physical host*.
3. Integrasi penjelasan prinsip dasar SDN dalam media pembelajaran

2. Pembuatan Lingkungan Pembelajaran

Pada tahap ini dilakukan pembuatan lingkungan pembelajaran. Memanfaatkan Mininet Python API dibuat sebuah program yang mendukung penyederhanaan pembelajaran konsep Software Defined Networking.

3. Pembuatan Model Pembelajaran

Pada tahap ini dilakukan pembuatan model pembelajaran. Bentuk dari lingkungan pembelajaran ini berupa instruksi dan tutorial untuk memahami konsep Software Defined Network lewat praktek menggunakan Mininet.

4. Pengujian Purwarupa

Lingkungan Pembelajaran yang telah dibuat akan diuji relevansinya dengan pembelajaran konsep Software Defined Network

BAB II

STUDI LITERATUR

II.1 Software Defined Network

Software Defined Network adalah sebuah arsitektur jaringan yang tengah muncul dimana kendali jaringan terpisah dari *forwarding* dan dapat diprogram langsung. Sebuah bidang kendali mengendalikan beberapa perangkat. SDN bersifat dinamis, dapat diatur, efektif biaya, serta dapat beradaptasi yang membuatnya ideal untuk sifat padat *bandwidth* serta dinamis aplikasi masa kini. SDN memiliki sifat: dapat langsung diprogram; *agile*; manajemen terpusat; terkonfigurasi program; dan standar terbuka.(Xia *et al*, 2015). Secara sederhana, konsep SDN adalah memungkinkan konfigurasi *behaviour* jaringan menggunakan perangkat lunak

SDN membuat sebuah abstraksi yang jelas antara lapisan yang membuat keputusan bagaimana sebuah paket harus dikirimkan (*controller*) dan lapisan yang mengirimkan paket (*datapath*).

Controller mengawasi informasi hubungan jaringan; membuat sebuah peta jaringan dengan mengumpulkan informasi tentang perangkat di sekitar, kemudian membangun sebuah kelompok rute optimal untuk mengirimkan paket. *Controller* kemudian membuat sebuah tuple berisi klasifikasi paket dan bagaimana cara mengirimkan paket yang disebut ***flow entry***. *Flow entry* disimpan didalam sebuah tabel yang digunakan oleh *datapath*: ***Forwarding Information Base*** atau FIB.

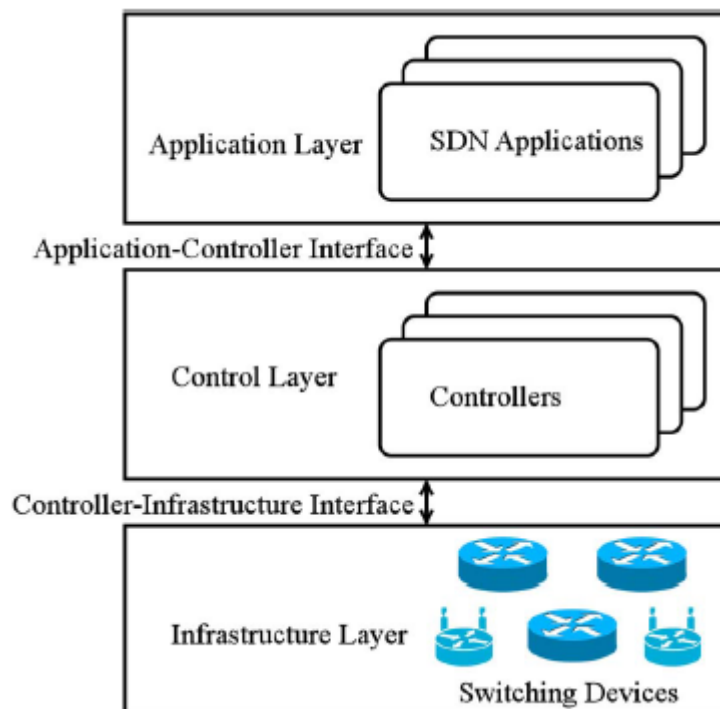
Datapath adalah lapisan yang berurusan dengan mengirimkan paket individual. Datapath mengacu pada tabel FIB. Tiap kali sebuah paket masuk kedalam datapath, paket tersebut diklasifikasikan dan ditentukan cara pengirimannya berdasarkan FIB.

Keuntungan dari SDN terutama ada dua. Pertama adalah memungkinkan inovasi dalam bidang jaringan untuk dilakukan eksperimen protokol jaringan baru yang belum pernah dicoba sebelumnya tanpa perlu melakukan implementasi fisik. Kedua adalah memperbaiki konfigurasi dan kinerja. Dalam hal ini, konfigurasi yang dimaksud adalah konfigurasi dari sebuah jaringan dimana konfigurasi ini mengatur bagaimana perangkat-perangkat dalam jaringan bekerja. Konfigurasi baru perlu ditambahkan ketika perangkat baru ditambahkan, dimana dengan SDN konfigurasi dapat diujicoba terlebih dahulu

sebelum ditambahkan perangkat baru sehingga perangkat baru dapat segera bekerja dan mengurangi waktu transisi dari sebelum adanya perangkat baru.

Kinerja dalam hal ini adalah optimasi penggunaan sumberdaya yang dimiliki oleh infrastruktur jaringan. Penggunaan optimasi dengan pendekatan masa kini umumnya terkendala oleh perbedaan teknologi dan pemegang kepentingan di sebuah jaringan. SDN mengatasi hal ini dengan memberikan kendali tersentralisasi untuk jaringan secara keseluruhan. Masalah yang ada dapat dikurangi dengan algoritma tersentralisasi yang tepat.

II.1.1 Arsitektur Software Defined Network



Gambar 1: Arsitektur umum Software Defined Network (A Survey on Software Defined Network, 2015)

Menurut Open Networking Foundation, Model SDN terdiri atas tiga lapisan: lapisan infrastruktur, lapisan kendali, dan lapisan aplikasi. Lapisan infrastruktur terdiri atas peralatan *switching* dalam bidang data. Peralatan-peralatan tersebut berfungsi ganda. Pertama untuk menyimpan status jaringan, kedua untuk proses paket berdasarkan aturan yang disediakan oleh pengendali.

Lapisan kendali menyambung lapisan aplikasi dan infrastruktur melewati kedua antarmukanya. Untuk interaksi dengan lapisan infrastruktur (*southbound*), lapisan ini

mengatur fungsi kendali terhadap perangkat *switching*. Untuk interaksi dengan lapisan aplikasi (*northbound*), lapisan ini mengatur poin akses *service* dalam berbagai bentuk, misalnya sebuah API.

Lapisan Aplikasi mengandung aplikasi SDN yang didesain untuk memenuhi kebutuhan pengguna. Melewati platform yang disediakan oleh lapisan kendali, aplikasi SDN dapat mengakses dan mengendalikan perangkat *switching* pada lapisan infrastruktur. Aplikasi ini dapat berupa *dynamic access control*, *server load balancing*, dan virtualisasi jaringan.

II.1.2 Controller

Controller mengawasi informasi hubungan jaringan. Saat ini telah tersedia banyak *controller* baik open-source maupun berbayar yang memiliki fungsi berbeda-beda sesuai dengan kebutuhan pengguna. Berikut beberapa *controller* yang diketahui

II.1.2.1 Open vSwitch Controller

Open vSwitch Controller atau **ovs-controller** adalah sebuah *controller* standar untuk mengatur *switch* jarak jauh menggunakan protokol OpenFlow. Ovs-controller dapat mengendalikan *switch* OpenFlow menggunakan metode koneksi yang berbeda-beda seperti SSL, TCP, dan Unix domain.

II.1.2.2 Ryu

Ryu adalah sebuah framework berbasis komponen untuk Software Defined Networking. Ryu menyediakan komponen perangkat lunak dengan API terdefinisi untuk memudahkan penciptaan manajemen jaringan baru dan aplikasi kendali jaringan. Ryu mendukung berbagai protokol untuk mengatur perangkat jaringan misalnya OpenFlow, Netconf, OF-config, dan sebagainya.^[3]

Filosofi yang menjadi dasar dibuatnya Ryu adalah *agile* dan *flexible*.

Sebagai *controller* dengan framework berbasis komponen, Ryu menyediakan komponen-komponen yang dapat dimodifikasi sesuai kebutuhan. Komponen baru juga dapat diimplementasikan bila dibutuhkan.

II.1.2.3 Floodlight

Floodlight adalah sebuah *controller* Software Defined Network tingkat perusahaan berbasis Java dan berlisensi Apache. Beberapa fitur yang dimiliki Floodlight diantaranya:

1. Sistem pemuatan modul untuk menyederhanakan ekstensi dan perbaikan
2. Mudah diatur dengan dependensi minimal.
3. Mendukung banyak *switch* OpenFlow baik virtual maupun fisik.
4. Dapat mengurus jaringan gabungan OpenFlow dan non-OpenFlow
5. Didesain untuk performa menggunakan *multithreading*.
6. Mendukung platform cloud OpenStack.

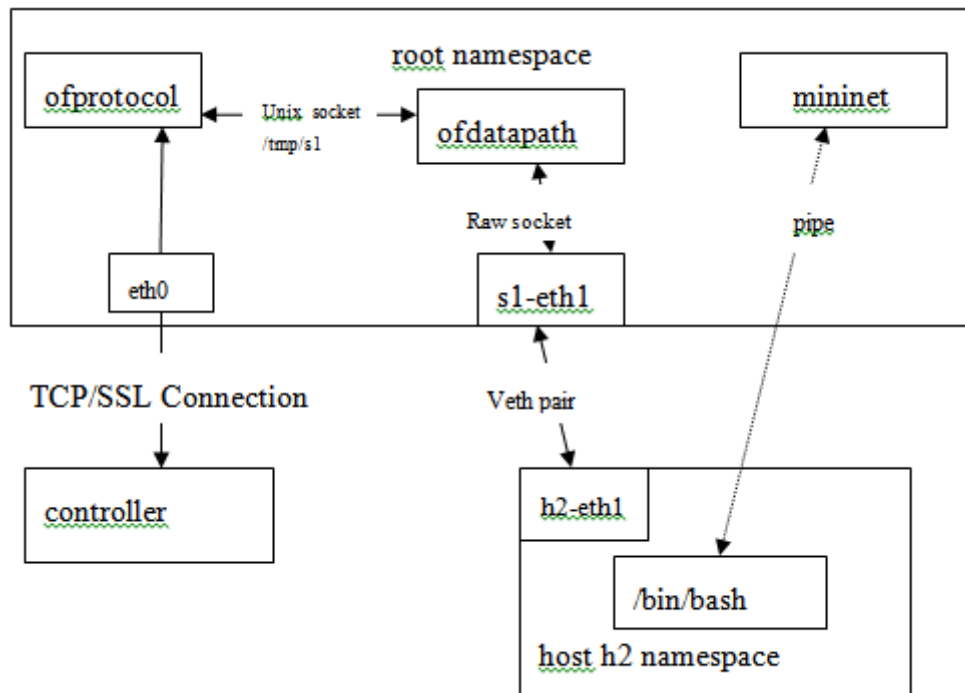
II.1.2.4 Open Daylight

Open Daylight adalah sebuah platform modular Software Defined Network terbuka untuk jaringan dengan ukuran dan skala apapun.

Open Daylight memungkinkan jaringan melewati perangkat keras yang berbeda dari vendor yang berbeda dalam satu jaringan.

II.2 MiniNet

Mininet adalah sebuah sistem untuk membuat purwarupa jaringan besar menggunakan sedikit sumberdaya, misalnya sebuah laptop. Pendekatan ringan dilakukan menggunakan virtualisasi tingkat OS termasuk proses dan nama jaringan memungkinkan ratusan *node* dari jaringan untuk dibuat^[4]. Mininet dibuat sebagai sebuah lingkungan purwarupa memanfaatkan fitur-fitur Linux diantaranya proses dan pasangan Ethernet virtual dalam nama jaringan untuk membuat jaringan dengan bandwidth besar dan ratusan *node*. Pembuatan jaringan oleh Mininet tetap dibatasi oleh sumber daya dari sistem yang digunakan.



Gambar 2: Virtual Network mininet menggunakan OpenFlow dan virtual Ethernet (veth)

II.2.1 Alur Kerja Mininet

Mininet bekerja menggunakan kombinasi CLI dan API serta virtualisasi ringan.

Langkah kerja Mininet adalah sebagai berikut:

1. Mininet menciptakan sebuah topologi jaringan dengan virtualisasi ringan, menciptakan *host*, *switch*, dan *controller* virtual dan jaringan yang menghubungkannya.
2. Mininet melakukan perubahan atau pengambilan informasi pada topologi jaringan menggunakan fungsi yang dimilikinya.
3. Mininet terhubung dengan program Software-Defined Networking lainnya untuk mensimulasikan manajemen dan pengujian jaringan.

II.2.1.1 Penciptaan jaringan

Menggunakan command-line, dapat dibuat sebuah jaringan dalam *virtual machine*. Beberapa perintah yang dapat diberikan pada Mininet diantaranya:

1. *switch*: menentukan jenis *switch* yang akan digunakan (misal: Open vSwitch)
2. *controller*: menentukan jenis *controller* yang akan digunakan (misal: OVSController, Ryu)

3. topo: menentukan topologi dari jaringan (misal: tree, single.)

```
sheizan@sheizan:~$ sudo mn
[sudo] password for sheizan:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> █
```

Gambar 3: Pengaktifan Mininet dengan topologi minimal, 2 host dan 1 switch.

II.2.1.2 Pengujian Jaringan

Setelah terbentuk jaringan yang diinginkan, maka dapat diujikan pengiriman paket data antara node yang telah terbentuk dan dihubungkan satu sama lain.

Mininet dapat menampilkan informasi setiap node yang telah dibuat, dan opsi seperti ‘--mac’ dapat digunakan pada pengaktifan Mininet untuk memberikan informasi yang lebih mudah dibaca ketika mencari informasi terkait sebuah node.

```
mininet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.1 netmask 255.0.0.0 broadcast 10.255.255.255
    inet6 fe80::947e:8ff:feed:aa7a prefixlen 64 scopeid 0x20<link>
    ether 96:7e:08:ed:aa:7a txqueuelen 1000 (Ethernet)
    RX packets 52 bytes 6343 (6.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 24 bytes 1992 (1.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Gambar 4: Informasi node h1 yang ditampilkan Mininet

Tanpa perlu mengetahui langsung informasi sebuah node pun sudah dapat dilakukan pengujian terhadap hubungan antara dua node.

```
mininet> h1 ping -c10 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=9.84 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=4.56 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.314 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.073 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.064 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.049 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.074 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.064 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.070 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.060 ms
```

Gambar 5: Ping dari h1 menuju h2

Selain pengujian dasar dan menciptakan jaringan secara default, dapat juga diberikan opsi-opsi tertentu pada jaringan, misalnya mengatur delay, melakukan uji regresi, menampilkan informasi lebih banyak, dan sebagainya.

```
sheizan@sheizan:~$ sudo mn -v debug
[sudo] password for sheizan:
*** errRun: ['which', 'controller']
/usr/local/bin/controller
0*** errRun: ['grep', '-c', 'processor', '/proc/cpuinfo']
4
0*** Setting resource limits
*** Creating network
*** Adding controller
*** errRun: ['which', 'mnexec']
/usr/bin/mnexec
0*** errRun: ['which', 'ifconfig']
/sbin/ifconfig
0*** c0 : ('unset HISTFILE; stty -echo; set +m',)
unset HISTFILE; stty -echo; set +m
*** errRun: ['which', 'telnet']
/usr/bin/telnet
0*** c0 : ('echo A | telnet -e A 127.0.0.1 6653',)
Telnet escape character is 'A'.
Trying 127.0.0.1...
telnet: Unable to connect to remote host: Connection refused
*** Adding hosts:
*** errRun: ['which', 'mnexec']
/usr/bin/mnexec
0*** errRun: ['which', 'ifconfig']
/sbin/ifconfig
0*** h1 : ('unset HISTFILE; stty -echo; set +m',)
unset HISTFILE; stty -echo; set +m
h1 *** h2 : ('unset HISTFILE; stty -echo; set +m',)
unset HISTFILE; stty -echo; set +m
h2
*** Adding switches:
*** errRun: ['which', 'ovs-vsctl']
/usr/bin/ovs-vsctl
0*** errRun: ['ovs-vsctl', '-t', '1', 'show']
cf09e16a-bd53-4366-9c6d-bc90f147711c
ovs_version: "2.6.0"
0*** errRun: ['ovs-vsctl', '--version']
ovs-vsctl (Open vSwitch) 2.6.0
DB Schema 7.14.0
0*** s1 : ('unset HISTFILE; stty -echo; set +m',)
unset HISTFILE; stty -echo; set +m
```

Gambar 6: Penggunaan opsi '-v debug' pada pengaktifan Mininet

II.2.2 Mininet Python API

Mininet memiliki **Python API** yang dapat digunakan sesuai kebutuhan pengguna. Fungsi-fungsi yang terdapat dalam *library* Mininet API mencakup fungsi-fungsi yang digunakan dalam Mininet lewat *command line*^[5].

```
1  from mininet.topo import Topo
2  from mininet.net import Mininet
3  from mininet.util import dumpNodeConnections
4  from mininet.log import setLogLevel
5  from mininet.node import OVSController
6
7  class CustomSwitchHostTopo(Topo):
8      "x switch connected to n hosts."
9      def build(self, n=2):
10         int countSwitch, countHost
11         countSwitch = input('Enter number of switch: ')
12         countHost = input('Enter number of host: ')
13         for i in range(countSwitch):
14             switch = self.addSwitch('s%s' % (countSwitch +
15 # Python's range(N) generates 0..N-1
16         for h in range(countHost):
17             host = self.addHost('h%s' % (h + 1))
18             self.addLink(host, switch)
19
20
21 def simpleTest():
22     "Create and test a simple mininet+ryu"
23     topo = SingleSwitchTopo(n=4)
24     net = Mininet(topo=topo, controller=OVSController)#, c
25     #print "Initializing Ryu simple switch."
26     net.start()
27     print "Dumping host connections"
28     dumpNodeConnections(net.hosts)
29     print "Testing network connectivity"
30
31     net.pingAll()
32     net.stop()
```

Gambar 7: Contoh kode Mininet Python API, menggunakan OVSController

Python API ini memberikan pengembang sebuah opsi untuk menciptakan jaringan menggunakan opsi-opsi tertentu yang dapat langsung digunakan tanpa perlu mengatur opsi dalam *command line interface* Mininet. Program yang dibuat juga dapat segera didistribusikan apabila diperlukan pengujian dalam kondisi yang berbeda-beda.

Kelebihan lain Python API adalah kemampuan menciptakan spesifikasi khusus bagi sebuah jaringan yang akan dibuat, dimana fleksibilitas dari kode Python dapat digunakan untuk penciptaan jaringan yang lebih rumit dari penyambungan *default* yang digunakan dalam aktivasi Mininet lewat terminal.

```
net = Mininet()
s1 = net.addSwitch('s1')
s2 = net.addSwitch('s2')
s3 = net.addSwitch('s3')
c0 = net.addController('c0')
h1 = net.addHost('h1')
h2 = net.addHost('h2')
h3 = net.addHost('h3')
h4 = net.addHost('h4')
h5 = net.addHost('h5')
h6 = net.addHost('h6')
h7 = net.addHost('h7')
h8 = net.addHost('h8')
h9 = net.addHost('h9')
net.addLink(h1, s1)
net.addLink(h2, s1)
net.addLink(h3, s2)
net.addLink(h4, s3)
net.addLink(h5, s3)
net.addLink(h6, s3)
net.addLink(h7, s3)
net.addLink(h8, s3)
net.addLink(h9, s3)
net.addLink(s1, s2)
net.addLink(s1, s3)
```

Gambar 8: Contoh spesifikasi jaringan Python API

Penggunaan Python API memungkinkan jaringan untuk dibuat sesuai dengan keinginan pengguna hingga tingkat detail menggunakan opsi-opsi yang diinginkan terkait jaringan yang dirancang.

II.3 OpenFlow

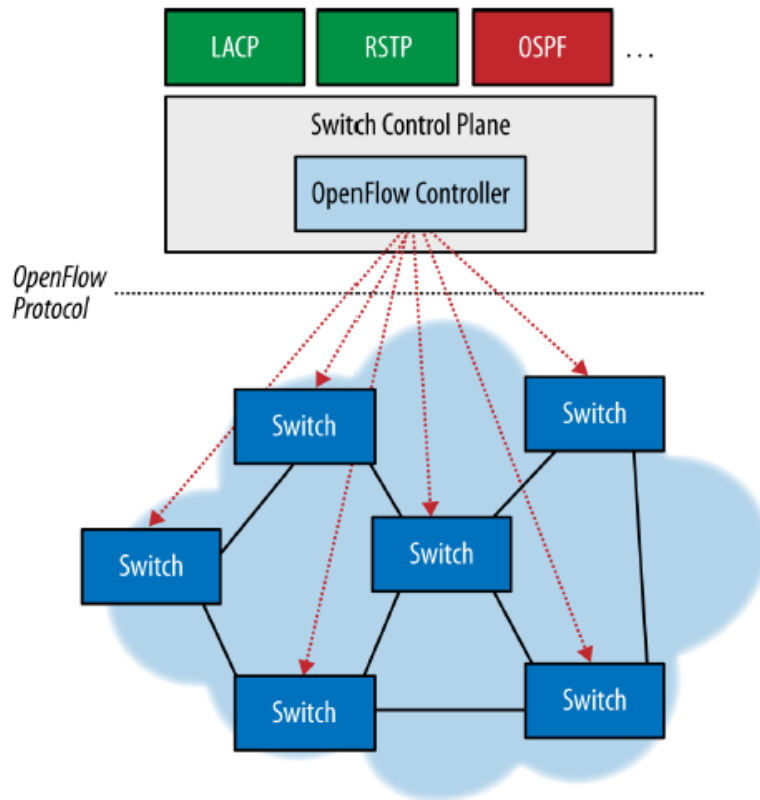
OpenFlow adalah sebuah kumpulan protokol dan API yang awalnya diimplementasikan sebagai bagian dari riset jaringan di Universitas Stanford. Tujuan awalnya adalah memungkinkan penciptaan protokol eksperimental pada jaringan kampus yang dapat digunakan untuk riset dan eksperimen. Sebelum adanya OpenFlow, universitas harus menciptakan platform eksperimen dari nol^[6]. McKeown et al. mendeskripsikan OpenFlow sebagai sebuah API untuk *datapath* dari perangkat keras jaringan.

Dari OpenFlow berkembanglah ide dimana OpenFlow dapat menggantikan fungsi dari protokol lapisan 2 dan 3 sepenuhnya pada *switch* dan *router* komersil. OpenFlow memanfaatkan fakta bahwa sebagian besar *switch* dan *router* Ethernet memiliki tabel aliran untuk fungsi penting jaringan, seperti *routing*, *subnetting*, perlindungan *firewall*, dan analisa statistik aliran data.

II.3.1 Arsitektur OpenFlow

Komponen-komponen utama OpenFlow yang ditunjukkan dalam Gambar 9 menjadi definisi umum dari SDN. Terutama pemisahan antara bidang kendali dan bidang data, penggunaan protokol terstandarisasi antara pengendali dan agen dalam elemen jaringan untuk instansiasi *state*, dan penyediaan kemampuan pemrograman jaringan dari sebuah *view* tersentralisasi lewat API modern. Paket data lazimnya dibagi-bagi kedalam *flow* atau aliran data berdasarkan atribut data tersebut, misalnya asal, tujuan, dan protokol yang digunakan.

Saat ini, protokol OpenFlow dibagi menjadi dua: *wire protocol* untuk menyiapkan *session* kendali, mendefinisikan struktur pesan untuk bertukar *flow modifications* dan mendefinisikan struktur sebuah *switch*; dan protokol konfigurasi dan manajemen berdasarkan NETCONF untuk mengalokasikan *switch port* fisik ke sebuah pengendali tertentu, mendefinisikan *availability*, dan sikap ketika terjadi kegagalan pada hubungan pengendali.



Gambar 9, Arsitektur umum OpenFlow(SDN: Software Defined Network, 2013)

II.3.1.1 *Flow Modification*

Pada OpenFlow 1.0, FIB dari *datapath* direpresentasikan sebuah tabel logika yang memiliki daftar *flow entry* yang menjelaskan *flow* yang perlu dicocokkan dan aksi yang perlu dilakukan kepada paket yang cocok. Sebuah *switch* OpenFlow setidaknya harus mendukung aksi untuk mengirimkan atau melepaskan paket. Aksi umum yang dilakukan jika tidak ada *flow entry* adalah mengirimkan paket kepada *controller* sehingga dapat diciptakan sebuah *flow entry* yang cocok dengan paket.

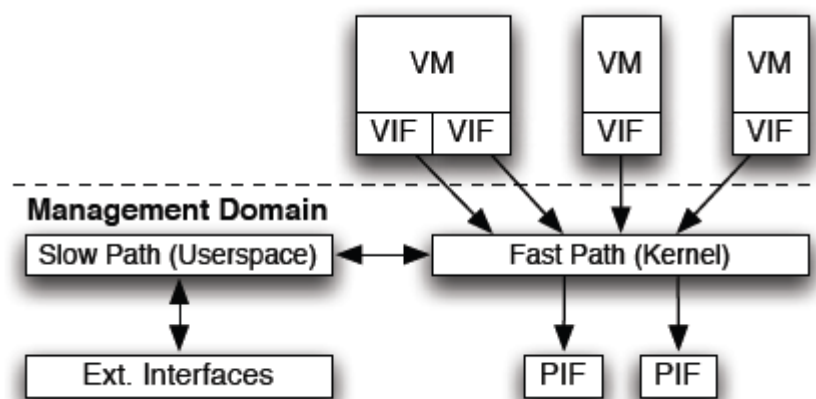
II.3.1.2 Instruksi

Pada OpenFlow 1.1, *datapath* diperluas untuk menyokong beberapa tabel. Proses dimulai pada tabel pertama dan dapat selesai setelah satu tabel atau dikonfigurasi untuk diproses pada tabel lainnya.

II.4 Open vSwitch

Open vSwitch adalah sebuah *multilayer virtual switch* sekaligus *datapath* didesain dengan fleksibilitas dan portabilitas untuk menyokong beberapa protokol besar termasuk OpenFlow^[7]. Open vSwitch berada di lapisan *hypervisor* atau domain manajemen dan menyediakan konektivitas antar perangkat maya dan antarmuka fisik. Open vSwitch berbeda dengan pendekatan tradisional dimana ia mengeluarkan sebuah antarmuka untuk kendali halus dari *forwarding* yang dapat digunakan untuk mendukung *Quality of Service*, *tunneling*, dan aturan *filtering*^[8]. Open vSwitch juga mendukung fitur-fitur yang dibutuhkan sebuah *switch* canggih: visibilitas kedalam aliran lalu lintas jaringan; *Access Control List* dan *Quality of Service* yang tersaring dengan baik; dan dukungan terhadap kendali tersentralisasi.

II.4.2 Arsitektur Open vSwitch

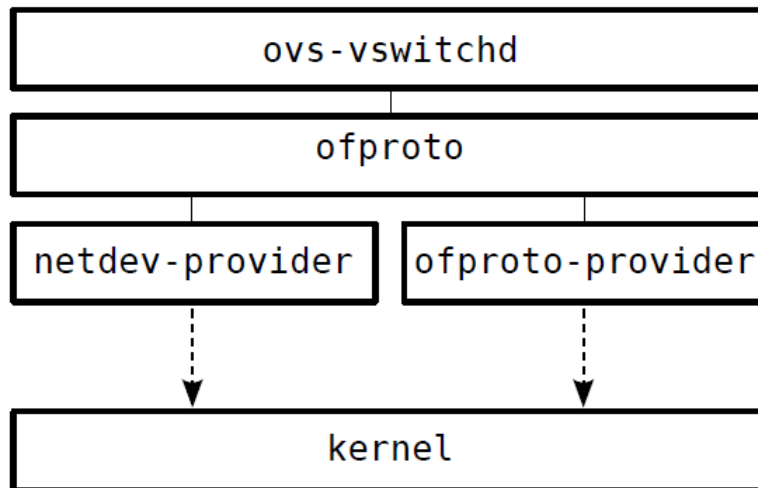


Gambar 10, Arsitektur umum OpenVSwitch (Extending Networking Into the Virtualization Layer, 2009)

Open vSwitch mengimplementasi *switching* Ethernet umum seperti VLAN, RSPAN, dan ACL dasar. Struktur internal Open vSwitch terdiri atas komponen-komponen berikut:

- ovs-vcstd: Proses *daemon* yang menyimpan state.
- netdev-provider: Menyediakan informasi perangkat jaringan.
- ofproto: menyatukan koneksi jaringan OpenFlow.

- ofproto-provider: mengimplementasi fungsionalitas OpenFlow.



Gambar 11, Komponen Open vSwitch (Contributing to OpenFlow Support, 2012)

II.4.2 Konfigurasi

Lewat sebuah antarmuka konfigurasi, sebuah proses terpencil dapat melakukan baca dan tulis status konfigurasi dan mengatur trigger untuk menerima *event asynchronous* mengenai perubahan status konfigurasi. Menggunakan `ovs-ofctl`, Open vSwitch dapat melakukan *parsing* terhadap pesan OpenFlow dan mencetak teks sesuai dengan tujuan dari pesan tersebut.

II.4.3 Fungsi Dasar Open vSwitch

Open vSwitch memiliki fungsi dasar untuk mengatur *flow* dari paket antara node dalam jaringan. Dengan akses terhadap controller maka Open vSwitch dapat mengubah tabel *forwarding information base* yang dimiliki oleh sebuah controller.

II.5 Tkinter

Tkinter adalah *graphical user interface* standar milik Python^[9]. Tkinter memiliki kemampuan untuk menciptakan sebuah antarmuka untuk program-program yang menggunakan Python. Tkinter memiliki beberapa *widget* yang menjadi basis dari sebuah antarmuka standar.

Antarmuka Tkinter menggunakan root untuk membuat sebuah instansi antarmuka yang dapat kemudian dikembangkan dengan *widget* lainnya

Nama Widget	Fungsi
Button	Tombol sederhana untuk melakukan sebuah perintah atau operasi lainnya.
Canvas	Grafik terstruktur untuk menggambar, menciptakan editor grafik, dan mengimplementasikan <i>widget</i> buatan.
Checkbutton	Merepresentasikan variabel dengan dua nilai berbeda. Menekan tombol mengubah antara kedua nilai.
Entry	Menerima masukan teks
Frame	<i>Widget</i> kontainer untuk menyimpan atau mengelompokkan beberapa <i>widget</i> lain ketika menciptakan sebuah aplikasi.
Label	Menampilkan teks atau gambar
Listbox	Menampilkan sebuah daftar pilihan. Listbox dapat diatur menjadi <i>checklist</i> atau <i>radiobutton</i>
Menu	Panel menu untuk implementasi menu <i>pop-up</i> atau <i>pulldown</i>
Menubutton	Tombol menu untuk mengimplementasi menu <i>pulldown</i>
Message	Menampilkan teks, dapat secara otomatis mengatur teks untuk ukuran <i>window</i> tertentu.
Radiobutton	Merepresentasikan satu nilai dari sebuah variabel dengan banyak nilai. Nilai terpilih akan mengisi variabel.
Scale	Memungkinkan pengaturan nilai angka menggunakan sebuah slider
Scrollbar	<i>Scrollbar</i> standar untuk widget yang dapat digeser.
Text	Tampilan teks dengan format yang memungkinkan tampilan dan pengubahan teks dengan berbagai atribut.
Toplevel	Sebuah kontainer <i>widget</i> yang ditampilkan sebagai <i>window</i> yang terpisah

Tabel 1: Daftar Widget Tkinter^[10]

II.6 Pendekatan Sejenis

Pembelajaran Software-Defined Network bukanlah sesuatu yang jelas. Hingga pada saat tugas akhir ini dibuat masih banyak cara untuk mulai mempelajari Software-Defined Network diakibatkan pendekatan yang berbeda-beda terhadap Software-Defined Network serta jenis peralatan yang beragam.

II.6.1 Ivan Kupalov SDN Case Study

Dalam tesisnya, Ivan Kupalov(2016) membuat sebuah kasus studi pembelajaran Software-Defined Network memanfaatkan Mininet sebagai pencipta simulasi jaringan dan OpenDaylight sebagai Controller, serta Wireshark untuk analisis paket yang berjalan dan peramban Chromium dengan tambahan Postman.^[11]

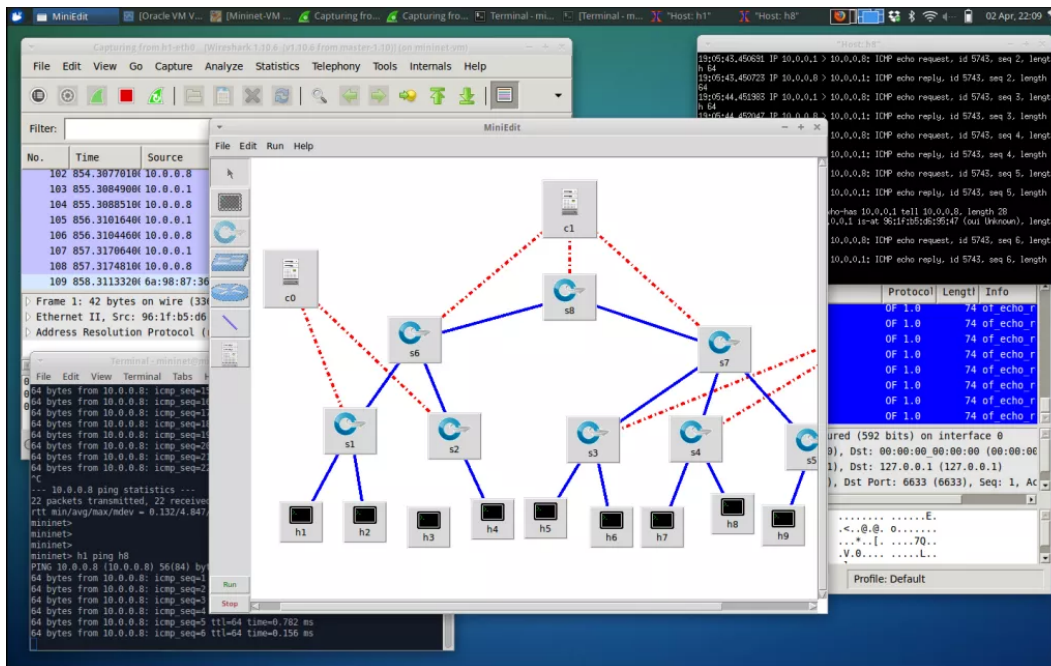
Studi kasus ini dibentuk menggunakan dua buah *host* dan tiga buah *switch*, dimana setiap *switch* terhubung satu sama lain sementara kedua *host* hanya terhubung pada dua dari tiga *switch*.

Ivan Kupalov merancang sebuah kasus uji pembelajaran mulai dari instalasi fitur, menyalakan Controller OpenDaylight, lalu menyalakan topologi dan memeriksa paket yang berjalan saat CLI Mininet diberikan perintah semisal pingall, link down dan sebagainya.

II.6.2 Miniedit

Miniedit adalah GUI Mininet yang dibuat oleh Brian Linkletter(2015) untuk mendemonstrasikan jaringan yang dibuat Mininet dengan tampilan yang memudahkan pembelajaran. Miniedit perlu berjalan dalam *Virtual Machine* Mininet yang didistribusikan langsung oleh Mininet^[12].

Miniedit memiliki kelebihan berupa desain yang intuitif dimana pembelajaran dapat dilakukan dengan lebih mudah karena pengguna dapat melihat langsung bentuk dari topologi yang sedang berjalan, hubungan antar node serta sifat dan konfigurasi dari hubungan tersebut dan juga dengan cepat mengubah jaringan dengan menu yang tersedia.



Gambar 12: Tampilan Miniedit, <http://www.brianlinkletter.com/how-to-use-miniedit-mininets-graphical-user-interface/>

BAB III

ANALISIS DAN PERANCANGAN

III.1 Analisis Masalah

Software Defined Network adalah sebuah bidang yang tergolong baru dan maka dari itu masih terdapat beberapa permasalahan, salah satunya adalah kurangnya media pembelajaran untuk peminat SDN ataupun mereka yang ingin belajar SDN secara umum.

Dengan tidak adanya media pembelajaran khusus, maka untuk memasuki lebih dalam dunia SDN agak sulit dilakukan. Pada bagian ini akan dibahas mengenai media pembelajaran yang dan apa saja yang diperlukan dalam sebuah media pembelajaran SDN.

Dalam pembelajaran, penting adanya instruksi dan *feedback* untuk menunjang perkembangan pembelajar. Instruksi yang dimaksud adalah langkah-langkah yang diperlukan dalam mempelajari sebuah perangkat Software-Defined Networking. Langkah-langkah ini setiap dijalankan perlu memberikan sebuah *feedback* yang membantu pembelajaran pengguna dan menjelaskan informasi mengenai apa saja yang telah dilakukan pengguna.

Penting juga bahwa sebuah media pembelajaran memfasilitasi pengajar untuk membuat kasus-kasus sesuai dengan kebutuhan pengajaran. Kasus-kasus tersebut sebaiknya mudah untuk dibuat, agar pengajar tidak habis waktunya dalam menciptakan kasus tersebut dan dapat lebih banyak meluangkan waktu dalam membantu pembelajar mempelajari kasus-kasus yang telah dibuat.

III.1.1 Pertimbangan Solusi

Dalam media pembelajaran tentunya diperlukan sebuah pengenalan, yaitu terhadap istilah-istilah yang sering ditemukan dalam *Software-Defined Networking*. Selain itu diperlukan juga percobaan untuk mengetahui secara langsung cara kerja SDN, dan langkah-langkah yang berjalan ketika SDN diaplikasikan dalam sebuah jaringan.

III.1.1.1 Tutorial Software-Defined Networking

Dalam tutorial SDN maka akan dibuat sebuah program yang menjelaskan langkah-langkah dalam membuat jaringan SDN dasar serta istilah-istilah yang sering digunakan dalam *controller* seperti *datapath*, *controller*, *northbound* dan sebagainya.

Kelebihan dari tutorial ini adalah pengguna tinggal mengikuti perintah yang ada untuk mempelajari Mininet dan Software-Defined Network

Kekurangan dari tutorial adalah fungsi-fungsi lain terkait Software-Defined Network belum tentu semuanya tercakup dalam sebuah tutorial

III.1.1.2 Antarmuka Software-Defined Network

Dapat dibuat sebuah antarmuka yang memudahkan pengguna untuk mempelajari dasar-dasar *Software-Defined Networking*. Dengan desain yang intuitif dan menu bantuan, pengguna dapat mempelajari fungsi mininet dengan lebih mudah dan membaca topologi yang tengah berjalan dengan lebih mudah.

Kelebihan dari antarmuka adalah tampilan intuitif yang dapat membantu pengguna mempelajari fungsi-fungsi yang dimiliki oleh Mininet dan Software-Defined Network.

Kekurangan dari antarmuka adalah integrasi dengan sistem eksternal lain dan perlunya penggunaan API secara ekstensif.

III.1.1.3 Sandbox

Dapat dibuat sebuah peralatan dimana pengguna dapat secara bebas mengutak-atik sifat dan rancangan dari SDN sehingga pengguna dapat mensimulasikan SDN sesuai dengan kebutuhannya.

Kelebihan dari Sandbox adalah pengguna dapat membuat topologi dan mengatur jaringan yang dibuatnya dengan sebarang mungkin

Kekurangan dari Sandbox adalah kurangnya informasi dari fungsi yang ditawarkan oleh Mininet dan informasi prinsip Software-Defined Network sehingga sulit digunakan untuk pembelajaran.

III.2 Perancangan Solusi

Dari ketiga hal pada bab III.1.1, pertimbangan yang diberikan adalah sistem antarmuka memberikan nilai interaktivitas dan kemudahan lebih bagi pengguna. Dilakukan fokus

ke Antarmuka Software-Defined Network. Untuk memenuhi kriteria utama berupa antarmuka dan kriteria lainnya yaitu Tutorial dan Sandbox dibuat antarmuka menggunakan **API Python Mininet** dengan menggunakan **Tkinter Python** untuk membentuk tampilan antarmuka.

Mempertimbangkan proyek antarmuka serupa yaitu Miniedit, akan dibuat sebuah program yang lebih sederhana. Miniedit menggunakan sebuah Custom Constructor yang perlu dicari terlebih dahulu sebelum dapat digunakan. Beberapa distribusi Mininet *virtual machine* belum tentu membawa class Custom Constructor ini. Program akan dibuat tanpa perlu menggunakan *virtual machine* Mininet, dengan syarat Mininet terinstalasi dalam komputer.

Kemudian, dengan mempertimbangkan studi kasus Ivan Kupalov, program akan dibuat untuk dapat menjalankan kasus uji pembelajaran tertentu dengan langkah-langkah yang telah ditentukan. Selain itu pengguna dapat berkreasi dengan topologi untuk memenuhi kriteria Sandbox.

Controller yang digunakan adalah bawaan *default* Mininet, Open vSwitch sebagai *virtual switch* dan OpenFlow sebagai protokol SDN.

Untuk implementasi, SDN umumnya memerlukan lebih dari satu *physical host* sebagai simulasi beberapa *host* dalam sebuah jaringan dimana biasanya digunakan *virtual machine* sebagai *physical host* tambahan untuk simulasi. Dengan program ini, maka pengguna dapat mensimulasikan topologi dasar yang disediakan di Mininet serta mengubahnya menggunakan fungsi yang disediakan dari Mininet API.

III.2.1 Arsitektur Solusi

Solusi dibuat dalam satu modul utama yang dikembangkan menggunakan Python 2.7.

Modul Mininet Python API memanfaatkan API Mininet untuk membangun sebuah jaringan Mininet tanpa perlu membuka *virtual machine* atau terminal untuk menjalankan Mininet. Modul ini akan membuat sebuah jaringan sesuai dengan spesifikasi yang diinginkan pengguna selama masih didalam jangkauan kemampuan mininet.

Digunakan *library* untuk penggunaan Mininet dari Python API, *controller* Open vSwitch, *Command-Line Interface*, serta tingkat log yang diperlukan untuk menjelaskan informasi.

III.2.1.1 Library Mininet API

Library Mininet API terdiri atas sembilan kelas utama

- Clean: kelas untuk membersihkan sisa dari sesi sebelumnya.
- CLI: kelas untuk menggunakan command-line interface dalam Mininet
- link: kelas yang berisi Interface dan pengaturan interface node.
- log: kelas yang berfungsi menampilkan informasi dan tingkat informasi yang diinginkan.
- net: kelas yang menjalankan program utama Mininet
- node: kelas yang berisi obyek utama dari *virtual networking* yaitu node yang dapat berupa *host* atau *switch*
- nodelib: kelas yang berisi bridge dan Network Access Translation(NAT)
- topo: kelas yang berisi topologi dasar yang dapat digunakan dalam mininet.
- topolib: kelas yang berisi topologi tingkat lanjut yang didukung mininet.

III.2.1.2 Pembagian Kelas

Aplikasi dibangun dalam satu kelas utama yaitu kelas App yang di dalamnya terdiri atas beberapa subkelas dan fungsi serta beberapa kelas topologi yang

1. Subkelas Init

Terdiri atas fungsi topologi utama yang memulai sebuah instansi Net untuk menjalankan mininet. Terdapat juga fungsi untuk keluar dan fungsi penjelasan untuk setiap tombol.

2. Subkelas Minimal Topo

III.3 Tampilan Program

Menggunakan library Tkinter Python 2.7 dibuat sebuah antarmuka sederhana untuk pengguna berinteraksi dengan Mininet.

III.3.1 Library Tkinter

Library Tkinter memberikan kemampuan grafik dasar untuk program Python. Kelas yang digunakan dalam program ini sebagai berikut:

- Frame: Menampilkan sebuah bagian persegi empat dalam layar.
- Top Level: Menampilkan window frame yang terpisah dari layar utama.
- Button: Menampilkan tombol dan mengatur fungsi tombol
- Message Box: Menampilkan pesan ketika melakukan aksi tertentu.

BAB IV

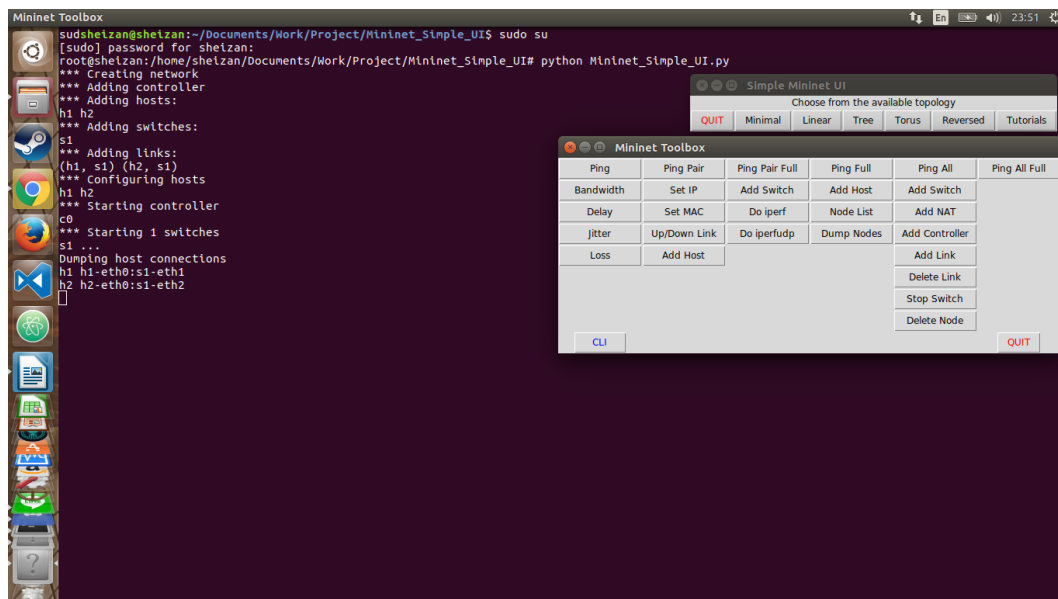
IMPLEMENTASI DAN PENGUJIAN

IV.1 Implementasi Modul Mininet API

Mininet Python API memberikan kemampuan bagi developer untuk menggunakan Mininet dalam program tanpa perlu membuka terminal terpisah untuk mengaktifkan mininet.

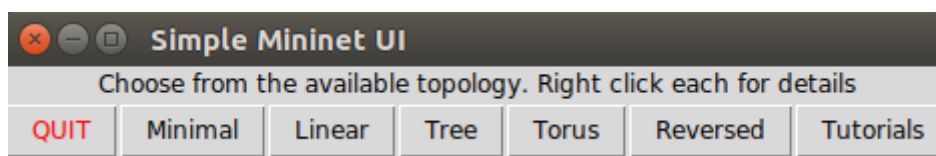
Implementasi dilakukan dengan menggunakan beberapa library Python API dari program Mininet agar dapat membaca input pengguna pada modul main untuk menciptakan jaringan maya yang diinginkan pengguna.

Digunakan modul Tkinter Python untuk memberikan tampilan agar memudahkan pengguna untuk menggunakan antarmuka sederhana Mininet.



Gambar 13: Implementasi Mininet API dengan UI.

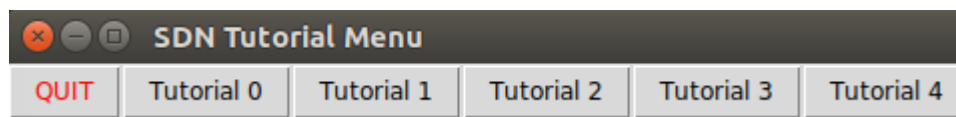
IV.1.1 Menu Topologi



Gambar 14: Menu Topologi

Menu yang pertama muncul adalah menu topologi. Setiap tombol dapat dilakukan klik kanan untuk mengetahui fungsi dari setiap tombol.

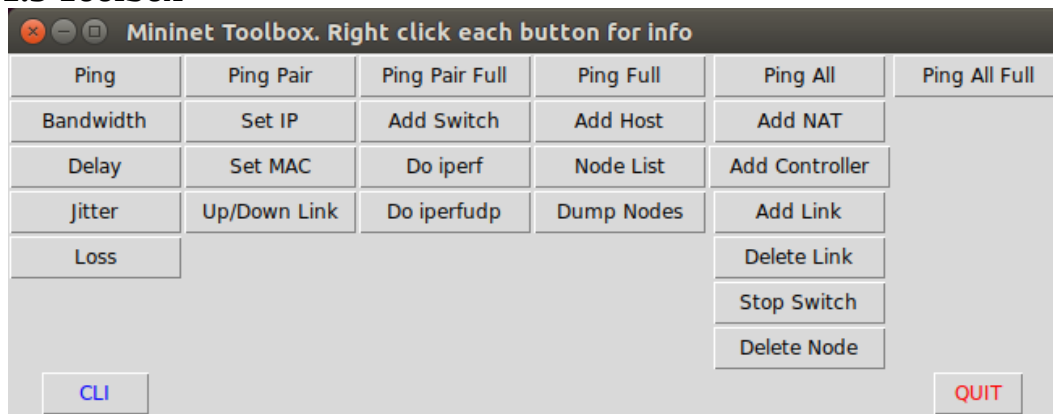
IV.1.2 Menu Tutorial



Gambar 15: Menu Tutorial

Menu tutorial berisi beberapa kasus uji yang dapat digunakan untuk eksperimen. Klik kanan tidak dapat dilakukan pada menu ini.

IV.1.3 Toolbox

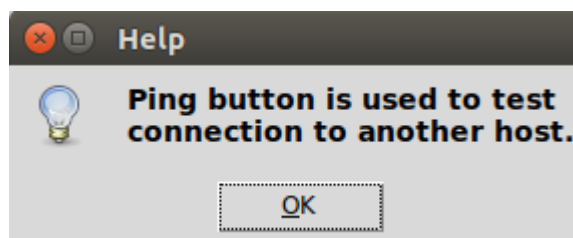


Gambar 16: Menu Toolbox

Menu toolbox berisi tools yang sering digunakan dalam Mininet. Klik kanan dapat dilakukan untuk mendapatkan info lebih lanjut pada tiap tombol.

IV.1.4 Menu Bantuan

Menu bantuan berisi info dari sebuah topologi atau fungsi yang dipilih. Dengan melakukan klik kanan pada sebuah fungsi, maka akan memunculkan bantuan.



Gambar 17: Menu Bantuan

IV.2 Lingkungan Implementasi

Lingkungan yang digunakan untuk implementasi antarmuka sederhana Mininet adalah sebagai berikut:

Hardware

- Prosesor: Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz x 4
- Sistem Operasi: Linux Ubuntu 17.04
- RAM: 8 GB (7,6 GB efektif)
- VGA Card: GeForce GT630M/PCIe/SSE2

Software

- Editor: Atom, Visual Studio Code
- Bahasa Pemrograman: Python

IV.3 Lingkungan Pengujian

- Prosesor: Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz x 4
- Sistem Operasi: Linux Ubuntu 17.04
- RAM: 8 GB (7,6 GB efektif)
- VGA Card: GeForce GT630M/PCIe/SSE2

IV.4 Tujuan Pengujian

Tujuan dari pengujian ini adalah mendapatkan informasi mengenai efektivitas pembelajaran menggunakan antarmuka Mininet dibandingkan dengan menggunakan Mininet secara langsung.

Secara spesifik, tujuan dari pengujian adalah sebagai berikut:

1. Mengetahui sejauh apa efek antarmuka dalam pembelajaran Software-Defined Networking menggunakan Mininet.
2. Mengetahui sejauh apa kemampuan API Mininet jika dibandingkan dengan Mininet secara langsung pada pengguna awam.

IV.5 Metode Uji

Pengujian dilakukan menggunakan kasus uji yang dapat digunakan baik dengan Mininet secara normal maupun antarmuka Mininet yang dibuat.

Digunakan dua skenario uji kasus dengan fokus yang berbeda.

IV.5.1 Uji Kasus I

Skenario ini menguji fungsionalitas dasar dari Mininet API yang telah dibuat dan membandingkannya dengan fungsionalitas Mininet tanpa menggunakan API dan juga bertujuan agar pengguna dapat memahami dasar penggunaan Mininet terhadap topologi yang diciptakan Mininet.

Koresponden akan melakukan pengujian menggunakan Mininet terlebih dahulu sebelum menggunakan program antarmuka Mininet yang telah dibuat.

Langkah-langkah yang ditempuh dalam skenario ini sebagai berikut:

1. Buka Mininet dan gunakan topologi Minimal
2. Lakukan Ping dari *host* 1 terhadap *host* 2
3. Tambah sebuah *host* baru
4. Hubungkan *host* baru dengan *switch* yang sudah ada.
5. Assign IP untuk *host* baru.
6. Lakukan dump node untuk mendapat detail seluruh node.
7. Tambahkan flow pada *switch* untuk *host* baru.
8. Lakukan ping antara *host* baru dengan kedua *host* sebelumnya.
9. Lakukan penghentian *switch* yang bekerja
10. Lakukan pengecekan bandwidth TCP antara dua *host* setelah *switch* berhenti bekerja.

IV.5.2 Uji Kasus II

Skenario ini menguji fungsionalitas Mininet API jika digunakan dengan program lain yang bekerja sama dengan Mininet, *ovs-ofctl* dan *xterm*.

Pengujian dilakukan tanpa menggunakan sambungan dari Mininet terhadap jaringan internal komputer dan *virtual machine*

1. Buka antarmuka Mininet dan masuk ke dalam tutorial.
2. Pilih tutorial 0 untuk menjalankan topologi twinpair.
3. Gunakan Command-line Interface.
4. Gunakan `sh ovs-ofctl dump-flows` dari CLI untuk mengetahui flow dari keempat *switch*.
5. Tambahkan sebuah link dari s1 ke s2 dan s3 ke s4.
6. Lakukan ping all.
7. Lakukan Down link s1 ke s4.
8. Lakukan Ping h1 ke h4.

IV.6 Hasil

Berikut adalah hasil pengujian dari kedua kasus uji. Kasus uji pertama diberikan kepada koresponden, sementara kasus uji kedua dilakukan sendiri.

IV.6.1 Uji Kasus I

Uji kasus pertama dilakukan terhadap 5 orang koresponden pembelajar IT. Uji coba menggunakan nilai sebagai berikut:

- 1: Sangat buruk
- 2: Buruk
3. Cukup Buruk
4. Cukup Baik
5. Baik
6. Sangat Baik

Nilai yang didapat dari hasil pengujian kemudian dilakukan rerata untuk mendapat nilai yang pasti.

Selain nilai, penguji juga diberikan tempat untuk memberikan kesan terhadap Mininet dan Antarmuka Mininet.

Ada empat poin pengujian:

1. Kemudahan Penggunaan: Poin ini mencakup kemudahan penggunaan perangkat *Software-Defined Network* pada umumnya serta tingkat kesulitan dalam menggunakan perangkat *Software-Defined Network*.
2. Kejelasan Program dan Fungsinya: Poin ini mencakup kejelasan penggunaan program dan fungsi yang ditawarkan oleh program tanpa menggunakan menu bantuan apapun.
3. Kemudahan Pembelajaran: Poin ini mencakup tingkat fasilitasi perangkat *Software-Defined Network* untuk pembelajaran dan sejauh apa perangkat dapat membantu pengguna belajar.
4. Menu Bantuan: Poin ini mencakup jenis bantuan yang ditawarkan oleh perangkat *Software-Defined Network* untuk memahami lebih jauh fungsi perangkat dan penjelasan proses *Software-Defined Network* yang terjadi.

Dari hasil pengujian, didapat data sebagai berikut:

1. Kemudahan Penggunaan

Mininet cenderung mendapat nilai rerata 2.8, yaitu kurang baik. Menurut para penguji, nilai ini karena penggunaan CLI yang cenderung membingungkan untuk orang awam dan bantuan yang kurang eksplisit.

Antarmuka Mininet mendapat nilai rerata 4.8, yaitu di antara baik. Menurut para penguji, nilai ini karena antarmuka lebih interaktif dan output serta help lebih jelas dibanding dengan Mininet yang dijalankan secara langsung.

2. Kejelasan Program dan Fungsinya

Mininet mendapat nilai rerata 3, yaitu kurang baik. Menurut para penguji, nilai ini karena menu bantuan yang kurang menjelaskan fungsi yang dimiliki oleh Mininet, dan beberapa perintah fungsi yang redundan.

Antarmuka Mininet mendapat nilai rerata 5, yaitu baik. Menurut para penguji, nilai ini karena antarmuka memiliki menu help yang menjelaskan sebuah fungsi mininet dengan lebih detil dan memiliki alur yang jelas.

3. Kemudahan Pembelajaran

Mininet mendapat nilai rerata 2.4, yaitu kurang baik. Menurut para penguji, nilai ini karena Mininet tidak memiliki pengarahannya yang jelas dan penggunaan Mininet sendiri perlu mengetahui fungsi-fungsi apa saja yang dapat digunakan.

Antarmuka Mininet mendapat nilai rerata 4,2, yaitu cukup baik. Menurut para penguji, nilai ini karena antarmuka memberikan feedback dan penjelasan terhadap fungsi apa saja yang dapat dijalankan.

4. Menu Bantuan

Mininet mendapat nilai rerata 2.2, yaitu cukup buruk. Menurut para penguji, hal ini karena menu bantuan Mininet tidak secara eksplisit menjelaskan fungsi dari setiap perintah yang ditunjukkan pada menu bantuan.

Antarmuka Mininet mendapat nilai rerata 5.4, yaitu baik. Menurut para penguji, hal ini karena antarmuka memberikan penjelasan dari fungsi Mininet secara lebih eksplisit dan dapat diakses secara intuitif.

IV.6.1 Uji Kasus II

Uji Kasus kedua dilakukan untuk menguji fungsionalitas dari program.

Dari hasil pengujian, ditemukan bahwa beberapa API Python dari Mininet tidak berjalan sebagaimana semestinya. Beberapa *behavior* dari fungsi berbeda dengan fungsi yang dijalankan langsung dari Mininet.

Selain itu, ditemukan juga bahwa fungsi yang perlu dijalankan menggunakan Python Script dari Mininet secara alami dapat langsung digunakan dalam antarmuka sehingga memudahkan penggunaan.

Berikut adalah rincian dari langkah-langkah pengujian yang didapat menggunakan Antarmuka Mininet.

No.	Langkah	Hasil
1.	Buka antarmuka dan masuk ke menu tutorial	Masuk dalam menu tutorial
2.	Pilih menu tutorial 0	Topologi Twinpair dijalankan

3.	Buka Command-line Interface	Membuka CLI Mininet lewat tombol CLI yang disediakan di antarmuka
4.	Gunakan <code>sh ovs-ofctl dump-flows</code> dari CLI untuk mengetahui flow dari keempat <i>switch</i> .	Flow dari flow table <i>controller</i> ditampilkan di terminal.
5.	Tambahkan sebuah link dari s1 ke s2 dan s3 ke s4.	Link ditambahkan menggunakan antarmuka. Interface muncul dan diset default.
6.	Lakukan ping all.	Ping gagal dilakukan dari h1 ke h2, h1 ke h3, h2 ke h1, h3 ke h1, h4 ke h2, h4 ke h3, h2 ke h4, h3 ke h4.
7.	Lakukan Down link s1 ke s4.	Link antara s1 dan s4 berhasil dimatikan.
8.	Lakukan Ping h1 ke h4.	Ping gagal dilakukan dari h1 ke h4.

Tabel 1: Langkah Uji Kasus 2

IV.7 Evaluasi

IV.7.1 Uji Kasus 1

Dari pengujian kasus 1 diketahui penggunaan antarmuka memanfaatkan Python API memberikan interaktivitas yang lebih baik dibandingkan menggunakan Mininet secara alami. Dengan penggunaan API serta antarmuka, menggunakan Mininet yang dokumentasi bantuannya kurang baik dan masih perlu menjalankan *script* Python untuk beberapa fungsinya menjadi lebih cepat dan intuitif.

Menu Bantuan yang dihadirkan membuat pembelajaran lebih mudah untuk pengguna awam, karena fungsi-fungsi yang ada dapat ditampilkan penjelasannya kapanpun.

Pengguna juga tidak perlu memasukkan *script* Python yang memerlukan pembelajaran Python terlebih dahulu sebelum mempelajari Mininet untuk beberapa fungsi yang diimplementasi dari API Mininet.

Tutorial yang diimplementasi juga dapat membantu untuk kasus pembelajaran sesuai kebutuhan. Bagaimanapun, belum adanya otomatisasi untuk membuat kasus pembelajaran membuat pengajar perlu membuat tutorial sendiri dengan mengubah kode program.

IV.7.1 Uji Kasus 2

Dari pengujian diketahui bahwa fungsi manajemen jaringan sendiri tidak berjalan sepenuhnya hanya dengan penggunaan Mininet saja. Untuk penggunaan manajemen jaringan secara penuh, masih diperlukan integrasi setidaknya dengan *ovs-ofctl*, karena *switch* yang digunakan dalam program ini adalah Open vSwitch menggunakan protokol OpenFlow untuk mengatur paket yang mengalir antara *host*.

Langkah 1, 2, dan 3 yang menggunakan hanya implementasi Mininet API dapat dijalankan tanpa masalah, begitu juga dengan langkah 5 dan 7

Langkah 4, yaitu penggunaan *dump-flows ovs-ofctl* juga berhasil, yaitu *flow controller* berhasil ditampilkan.

Pada langkah 6, ping masih gagal dilakukan karena walaupun link telah dibuat, aturan flow dari satu *switch* ke *switch* lainnya belum terbentuk. Mininet memerlukan *ovs-ofctl add-flow* untuk melakukan hal tersebut karena tidak adanya integrasi dengan *ovs-ofctl* ketika menambahkan link.

Pada langkah 8, link yang dilakukan *down* yaitu sementara dimatikan menggagalkan ping antara h1 dan h4.

BAB V

KESIMPULAN DAN SARAN

V.1 Kesimpulan

Kesimpulan dari tugas akhir ini adalah sebagai berikut:

1. Dapat dibuat sebuah antarmuka untuk membantu pembelajaran Software-Defined Networking menggunakan Mininet API.
2. Pembelajaran menggunakan antarmuka memberikan hasil yang lebih baik. Pengguna lebih nyaman dalam menggunakan antarmuka Mininet dibandingkan Mininet secara alami untuk pembelajaran Software-Defined Networking

V.2. Saran

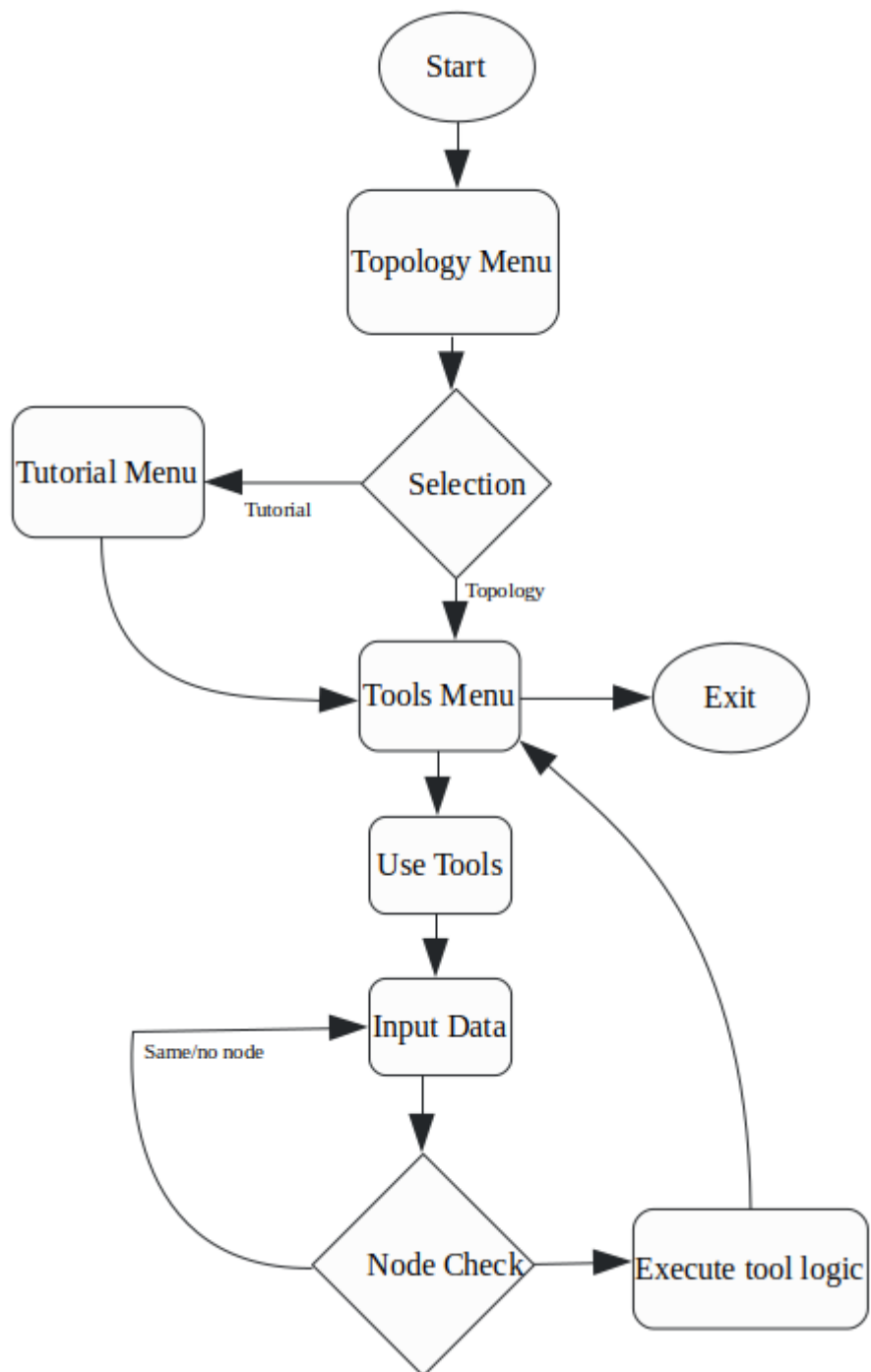
Beberapa saran yang perlu untuk memperbaiki Tugas Akhir ini adalah sebagai berikut:

1. Perlunya integrasi antara API Mininet dan program lain seperti ovs-ofctl dan wireshark atau tcpdump untuk secara akurat melakukan manajemen jaringan seperti penambahan flow jaringan atau pembelajaran *controller*.
2. Tampilan dari program perlu dibuat lebih intuitif lagi. Program purwarupa yang dibuat untuk Tugas Akhir ini masih dapat dibuat lebih jelas lagi dalam penyampaian setiap fungsi tombolnya serta feedback yang diberikan oleh setiap fungsi.
3. Beberapa fungsi yang tidak berjalan baik dari Mininet maupun error dari luar Mininet perlu perbaikan. Jika perlu dilakukan perbaikan dengan melakukan override terhadap API fungsi tersebut.

TABEL REFERENSI

- 1: Nadeau, Thomas D - Gray, Ken, SDN: Software Defined Networks, 2013
- 2: Xia, Wenfeng - Wen, Yonggang - Foh, Chuan Heng - Niyato, Dusit - Xie, Haiyong, A Survey Paper on Software-Defined Networking, 2015
- 3: RYU SDN Framework Community, Ryu: Build SDN Agilely, 2017,
- 3: Lantz, Bob, A Network in a Laptop: Rapid Prototyping for Software-Defined Networks, 2010
- 4: Lantz, Bob - O'Connor, Brian - Heller, Brandon - Handigol, Nikhil - Jeykumar, Vimal , Mininet API Reference Manual, 2017,
- 5: Stringer, Joe, Contributing to OpenFlow 1.1 Support in Open vSwitch, 2012
- 6: , What is Open vSwitch?, 2017,
- 7: Petit, Justin - Gross, Jesse - Pfaff, Ben - Casado, Martin - Crosby, Simon, Virtual Switching in an Era of Advanced Edges, 2010
- 8: Kupalov, Ivan, Software Defined Networks Case Study, 2016
- 9: Lantz, Bob, Mini Edit App, ,

LAMPIRAN I: WORKFLOW PROGRAM



LAMPIRAN 2: DAFTAR KELAS DAN FUNGSI

Berikut adalah daftar kelas yang dibuat dalam program ini.

Nama Kelas dan Fungsi	Deskripsi
App	Kelas utama program. Mencakup berbagai subkelas dari program.
__init__	Fungsi App. Memulai program dan menampilkan menu utama yaitu tombol-tombol topologi
minimal_topo	Fungsi App. Menciptakan topologi minimal mininet memanfaatkan kelas MinimalTopo dari library Mininet Python API
linear_topo	Fungsi App. Menciptakan topologi dengan x jumlah switch yang masing masing terhubung ke y jumlah host.
tree_topo	Fungsi App. Menciptakan topologi pohon memanfaatkan kelas TreeTopo dari library Mininet Python API
torus_topo	Fungsi App. Menciptakan topologi torus memanfaatkan kelas TorusTopo dari library Mininet Python API
reversed_switch	Fungsi App. Menciptakan topologi dengan nomor host paling rendah terhubung ke nomor port paling tinggi memanfaatkan kelas SingleSwitchReversedTopo dari library Mininet Python API
rectangle_topo	Fungsi App. Menciptakan topologi dengan bentuk geometris persegi empat
twinpair_topo	Fungsi App. Menciptakan topologi dengan bentuk dua pasangan host-switch yang tidak terhubung satu sama lain.
tutorial	Fungsi App. Menampilkan menu tutorial yang berisi

	skenario untuk pembelajaran Software-Defined Networking
options	Fungsi App. Mencakup menu utama untuk melakukan perubahan pada Mininet. Memanfaatkan
ping_node	Subfungsi options. Melakukan ping antara dua node dalam jaringan Mininet yang telah dibuat.
set_bandwidth	Subfungsi options. Mengganti bandwidth antara dua node dalam jaringan Mininet yang telah dibuat
set_delay	Subfungsi options. Mengganti delay antara dua node dalam jaringan Mininet yang telah dibuat
set_jitter	Subfungsi options. Mengganti jitter antara dua node dalam jaringan Mininet yang telah dibuat
set_loss	Subfungsi options. Mengganti loss antara dua node dalam jaringan Mininet yang telah dibuat
set_IP	Subfungsi options. Mengganti IP dari <i>host</i> .
set_MAC	Subfungsi options. Mengganti MAC dari <i>host</i> .
check_IP	Subfungsi options. Memeriksa IP address dari node.
set_link	Subfungsi options. Melakukan up/down link antara dua node dalam jaringan.
add_link	Subfungsi options. Menambahkan sebuah link antara dua node dalam jaringan.
add_host	Subfungsi options. Menambahkan sebuah node baru sebagai virtual <i>host</i> .
add_switch	Subfungsi options. Menambahkan sebuah node baru sebagai virtual <i>switch</i>
add_NAT	Subfungsi options. Menambahkan Network Address

	Translation pada sebuah node.
add_Controller	Subfungsi options. Menambahkan <i>controller</i> Open vSwitch baru.
delete_link	Subfungsi options. Menghapus link yang ada antara dua node dalam jaringan.
stop_switch	Subfungsi options. Menghentikan bekerjanya sebuah <i>switch</i> .
start_iperf	Subfungsi options. Melakukan pemeriksaan bandwidth dengan protokol TCP
start_iperfudp	Subfungsi options. Melakukan pemeriksaan bandwidth dengan protokol UDP dan bandwidth yang ditentukan pengguna
ping_pair	Subfungsi options. Melakukan ping antara dua <i>host</i> pertama
ping_full	Subfungsi options. Melakukan ping dengan menampilkan detail ping.
ping_all	Subfungsi options. Melakukan ping antar semua <i>host</i> yang terhubung dalam jaringan.
ping_all_full	Subfungsi options. Melakukan ping antar semua <i>host</i> yang terhubung dalam jaringan dan menampilkan detail
ping_pair_full	Subfungsi options. Melakukan ping antara dua <i>host</i> pertama dengan menampilkan detail ping.
show_nodes	Subfungsi options. Menampilkan semua node dalam jaringan.
dump_nodes	Subfungsi options. Menampilkan semua node dalam jaringan beserta detail setiap node.

check_node	Fungsi App. Memeriksa input node apakah ada didalam list node.
check_node2	Fungsi App. Memeriksa input node apakah ada didalam list node atau sama dengan input node sebelumnya.
right_(fungsi)	Subfungsi App. Binding fungsi untuk tombol mouse kiri.
left_(fungsi)	Subfungsi App. Binding fungsi untuk tombol mouse kanan.
RectangleTopo	Class RectangleTopo untuk membentuk topologi dengan empat <i>switch</i> terhubung satu sama lain yang masing-masing terhubung ke satu <i>host</i> .
build	Fungsi untuk membuat Rectangle Topology
PairTopo	Class PairTopo untuk membentuk topologi dengan dua pasang <i>switch</i> yang tidak terhubung antar pasangan. Masing-masing terhubung ke satu <i>host</i> .
build	Fungsi untuk membuat Pair Topology
CustomSwitchHostTopo	Class CustomSwitchHostTopo untuk membuat sebuah topologi linear dengan setiap <i>switch</i> terhubung secara berurutan. Setiap <i>switch</i> dapat berisi beberapa <i>host</i> .
build	Fungsi untuk membuat Custom Linear Topology

LAMPIRAN 3: LANGKAH KASUS UJI I DAN FORM PERTANYAAN

Kasus Uji I

Mininet	Antarmuka Mininet
1. Setelah Superuser, buka Mininet dan gunakan topologi Minimal >mn	1. Setelah superuser, buka antarmuka Mininet dan pilih topologi Minimal. (Minimal)
2. Lakukan ping dari <i>host</i> 1 terhadap <i>host</i> 2. >h1 ping h2	2. Lakukan ping dari <i>host</i> 1 terhadap <i>host</i> 2. (Klik ping dan isi h1 lalu h2)
3. Tambah sebuah <i>host</i> baru. >py net.addHost("h3")	3. Tambah sebuah <i>host</i> baru. (klik Add Host dan isi h3)
4. Hubungkan <i>host</i> baru dengan <i>switch</i> yang sudah ada. >py net.addLink("h3","s1")	4. Hubungkan <i>host</i> baru dengan <i>switch</i> yang sudah ada. (klik Add Link , isi h3 dan s1 . Kosongkan bandwidth dan latency untuk nilai default.)
5. Assign IP untuk <i>host</i> baru. >py h3. SetIP("10.0.0.3")	5. Assign IP untuk <i>host</i> baru (klik Set IP , isi h3 , isi 10.0.0.3)
6. Lakukan dump node untuk mendapat detail seluruh node. >dump	6. Lakukan dump node untuk mendapat detail seluruh node (klik Dump Nodes)
7. Tambahkan flow pada <i>switch</i> untuk <i>host</i> baru. >sh ovs-ofctl add-flow s1 in_port=3,actions:output=1	7. Tambahkan flow pada <i>switch</i> untuk <i>host</i> baru. (klik CLI , pada CLI gunakan: >sh ovs-ofctl add-flow s1

<pre>>sh ovs-ofctl add-flow s1 in_port=3,actions:output=2 >sh ovs-ofctl add-flow s1 in_port=1,actions:output=3 >sh ovs-ofctl add-flow s1 in_port=2,actions:output=3</pre>	<pre>in_port=3,actions:output=1 >sh ovs-ofctl add-flow s1 in_port=3,actions:output=2 >sh ovs-ofctl add-flow s1 in_port=1,actions:output=3 >sh ovs-ofctl add-flow s1 in_port=2,actions:output=3</pre>
<p>8. Lakukan ping antara <i>host</i> baru dengan kedua <i>host</i> sebelumnya.</p> <pre>>h3 ping h1</pre> <p>Setelah berjalan, hentikan dengan Ctrl+C, lalu ping</p> <pre>>h3 ping h2</pre> <p>Lakukan lagi Ctrl+C</p>	<p>8. Lakukan ping antara <i>host</i> baru dan kedua <i>host</i> sebelumnya (klik ping, h3 dan h1, lalu h3 dan h2)</p>
<p>9. Lakukan stop <i>switch</i> terhadap s1</p> <pre>>switch s1 stop</pre>	<p>9. Lakukan stop <i>switch</i> terhadap s1 (klik Stop Switch dan isi s1)</p>
<p>10. Lakukan Iperf antara h1 dan h2</p> <pre>>iperf h1 h2</pre>	<p>10. Lakukan Iperf antara h1 dan h2 (klik Do iperf, isi h1 dan h2)</p>

Form Pertanyaan:

Isilah jawaban pertanyaan-pertanyaan berikut dari skala 1 (sangat buruk) sampai 6(sangat baik):

1. Kemudahan penggunaan

Mininet: 
1 2 3 4 5 6

Simple UI: 
1 2 3 4 5 6

2. Kejelasan program dan fungsinya

Mininet: 
1 2 3 4 5 6

Simple UI: 
1 2 3 4 5 6


3. Kemudahan pembelajaran

Mininet: 
1 2 3 4 5 6

Simple UI: 
1 2 3 4 5 6

4. Menu Bantuan

Mininet: 
1 2 3 4 5 6

Antarmuka: 
1 2 3 4 5 6

Saran untuk program:

Mininet	Antarmuka

