Terakreditasi SINTA Peringkat 2
Surat Keputusan Dirjen Penguatan Riset dan Pengembangan Ristek Dikti No. 10/E/KPT/2019
masa berlaku mulai Vol. 1 No. 1 tahun 2017 s.d Vol. 5 No. 3 tahun 2021

Terbit online pada laman web jurnal: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 4 No. 5 (2020) 970 - 977 ISSN Media Elektronik: 2580-0760

MiTE: Program Penyunting Topologi Jaringan untuk Pembelajaran SDN

Muhammad Fajar Sidiq[1], Akbari Indra Basuki[2], Didi Rosiyadi[3]
[1]Fakultas Informatika, Institut Teknologi Telkom Purwokerto
[2,3]Pusat Penelitian Informatika, Lembaga Ilmu Pengetahuan Indonesia
[1]fajar@ittelkom-pwt.ac.id, [2]akbari@informatika.lipi.go.id*, [3]didi016@lipi.go.id

Abstract

Software-defined networking (SDN) is a network programmability concept that separates the control plane from the data plane by proposing a centralized control plane called a controller. Thus, network administrators are able to program the entire networks and their components via the controller. However, learning SDN is challenging due to its complex network setup and the different types of SDN networks such as OpenFlow, and P4. To ease the learning curve, the use of network emulation and a graphical-based network editor is necessary. This paper discusses the implementation of such an application, called MiTE. It satisfies both requirements: a visual network editor enforced with a configuration generator for emulation purpose. We evaluate the program by implementing IP routing cases for both, OpenFlow-based and P4-based networks. The result shows that both cases can be created easily by using a mouse command. The program has more interactive user interface while the created topologies are more informative. Compared to similar applications, our proposed application has better support for a wider range of SDN networks (Openflow and P4), fine-grain network configuration, and a more informative user interface.

Keywords: Topology, Editor, OpenFlow, P4, Emulation

Abstrak

Software-defined networking (SDN) merupakan paradigma pemograman jaringan secara terpusat yang berbasis pada perangkat *controller* dengan skema pemisahan anatara komponen pengirim paket (*data plane*) dan penentu arah *routing* paket (*control plane*). Pengelola jaringan dapat mengubah perilaku jaringan khususnya perilaku dari setiap perangkat jaringan melalui perangkat *controller*. Kendala utama dalam mempelajari SDN adalah rumitnya konfigurasi jaringan SDN dan begitu berbedanya jenis-jenis jaringan SDN, seperti protokol OpenFlow dan bahasa P4. Dalam mempermudah pembelajaran SDN, dibutuhkan sebuah pengujian berbasis emulasi jaringan. Oleh karena itu, dibutuhkan sebuah program berbasis visual yang mampu mempermudah pembuatan topologi jaringan secara cepat dan sederhana, seperti kemampuan konfigurasi jaringan secara otomatis. Penelitian ini membahas implementasi program yang dimaksud dengan nama MiTE yang memiliki dua fungsi utama: 1) Penyunting topologi jaringan berbasis visual, dan 2) Penghasil konfigurasi jaringan SDN untuk tujuan emulasi jaringan. Sebagai pengujian, penelitian ini menyajikan cara penerapan studi kasus IP *routing* pada jaringan SDN berjenis OpenFlow dan P4. Hasil pengujian menunjukkan bahwa studi kasus tersebut, dapat mudah dibuat dengan menggunakan perangkat *mouse*. Berdasarkan perbandingan dengan penelitian sebelumnya, program MiTE memiliki keunggulan dalam hal dukungan jaringan SDN yang lebih beragam (OpenFlow dan P4), kemampuan konfigurasi yang lebih terperinci, dan tampilan yang lebih informatif.

Kata kunci: Topologi, Penyunting, OpenFlow, P4, Emulasi

1. Pendahuluan

Software-Defined Networking (SDN) mengusung konsep pemrograman jaringan secara tersentralisasi dengan cara memisahkan komponen pengontrol (*control plane*) dan pengirim (*data plane*) dari setiap perangkat jaringan. Pada jaringan SDN, sebuah perangkat pengontrol yang disebut *controller* bertugas untuk menentukan arah routing dan aturan-aturan pemrosesan paket (*flow rules*) dari setiap perangkat jaringan. Pada sisi lain, parangkat jaringan hanya bertindak sebagai *data plane* dengan memproses paket berdasarkan aturan yang telah ditentukan oleh perangkat *controller* [1].

Keunggulan jaringan SDN adalah terdapatnya pengawasan dan pemrograman terpusat akan seluruh komponen jaringan. Generasi awal SDN dipelopori oleh protokol OpenFlow [2]. OpenFlow bertujuan untuk mengatasi masalah interoperabilitas dari perangkat jaringan yang memiliki protokol dan konfigurasi yang

Diterima Redaksi : 16-09-2020 | Selesai Revisi : 16-10-2020 | Diterbitkan Online : 30-10-2020

berbeda-beda. Dengan protokol yang terstandarisasi, pengelola jaringan dapat mengintegrasikan perangkat jaringan dari berbagai vendor dan memprogramnya secara terpusat.

Bahasa P4 [3] merupakan perkembangan dari teknologi SDN yang bertujuan untuk mengatasi masalah fleksibilitas dari protokol OpenFlow. Dengan keterbatasan spesifikasi yang dimiliki, protokol OpenFlow tidak dapat digunakan untuk keperluan-keperluan khusus seperti untuk mendeteksi kongesti dan pengawasan jeda pada pengiriman paket. Dengan bahasa P4, pengelola dapat membuat protokol-protkol baru tanpa terpaku pada protokol-protokol yang sudah ada seperti IP, TCP, UDP, dll. Dengan kemampuan tersebut, P4 lebih fleksibel dan potensial untuk menyelesaikan masalah-masalah jaringan yang sebelumnya tidak dapat diselesaikan oleh protokol yang sudah ada. Beberapa contoh masalah tersebut adalah penapisan paket secara terperinci [4], pendeteksian kemacetan jalur *routing* [5], dan sistem telemetri *in-band* [6].

Meskipun memiliki banyak keunggulan, penggunaan jaringan SDN pada aplikasi riil masih sangat minim. Hal tersebut dikarenakan SDN sangat sulit dipelajari. Konfigurasi jaringan SDN sangat rumit dan jaringan SDN sendiri terdiri dari berbagai jenis. Sebelum memulai memprogram jaringan, pengguna harus terlebih dahulu memiliki jaringan uji yang siap diprogram. Sebagai konsekuensi dari hal tersebut, pengguna harus melakukan konfigurasi berbagai hal berikut:

1. Menentukan konfigurasi dari setiap perangkat jaringan dan *host*/PC, seperti alamat IP dan MAC.
2. Menentukan konfigurasi setiap *link*, seperti nilai latensi.
3. Mengatur konfigurasi *controller* untuk setiap perangkat *switch* sesuai jenis jaringan SDN yang dijalankan (OpenFlow atau P4).

Dengan kerumitan konfigurasi jaringan, banyak pengguna yang menganggap memprogram jaringan SDN sangat susah. Kebanyakan pengguna tidak fokus pada bagaimana cara memprogram jaringan, tetapi lebih fokus pada bagaimana cara mengkonfigurasi jaringan uji. Disamping itu, kegagalan sistem yang dibuat tidak dapat ditelusuri dengan baik. Kegagalan bisa dikarenakan kesalahan program yang dibuat atau kesalahan dari konfigurasi jaringan yang dijadikan bahan uji.

Penelitian ini mengajukan sebuah program yang dapat mempermudah pembuatan konfigurasi jaringan SDN, baik jenis OpenFlow maupun P4. Tujuan dari program ini adalah agar pengguna dapat fokus pada studi kasus yang akan diuji tanpa terbebani untuk mengkonfigurasi jaringan uji. Dengan program ini, pengguna dapat menguji kebenaran dari program *controller* dengan cara menjalankannya secara emulasi menggunakan program Mininet [7].

Program memiliki fitur yang lebih lengkap dan terperinci dibandingkan dengan program MiniEdit [8]. Program MiniEdit hanya mendukung jaringan SDN berjenis OpenFlow, dan tidak mendukung konfigurasi secara terperinci untuk setiap antarmuka (*interface*). Konfigurasi terperinci tersebut seperti menentukan nomor dari antarmuka mana yang terhubung ke perangkat lainnya dan berapa alamat IP dan MAC dari setiap antarmuka tersebut.

Program kami memungkinkan pengguna untuk menentukan antarmuka yang akan digunakan untuk menghubungkan satu perangkat dengan perangkat lain. Hal ini memungkinkan pengguna untuk melakukan pengujian jaringan berbasis emulasi yang sesuai kondisi riil di lapangan.

Keunggulan program kami dibandingkan dengan program yang sudah ada adalah dukungan terhadap bahasa P4. Pengguna dapat membuat topologi jaringan yang dapat mengemulasikan *switch* berbasis P4 dan menjalankan program P4 pada perangkat tersebut.

2. Metode Penelitian

Pengembangan program pada penelitian ini dapat dibagi ke dalam tiga bagian, yaitu: 1) analisis konfigurasi sistem, 2) metode pembuatan program berbasis konsep *Model View Control* (MVC), dan 3) Mekanisme penghasilan konfigurasi jaringan untuk keperluan emulasi jaringan.

2.1. Analisis konfigurasi sistem

Batasan dari jenis jaringan SDN yang dapat dibuat oleh program adalah jenis SDN OpenFlow berbasis *switch* OpenVSwitch [9] dan SDN P4 berbasis *switch* BMv2. Sub bab ini menjelaskan secara lebih terperinci mengenai persyaratan konfigurasi dari kedua jenis jaringan SDN tersebut. Tabel 1 menjelaskan ringkasan konfigurasi untuk jaringan SDN OpenFlow sedangkan Tabel 2 menjelaskan ringkasan konfigurasi untuk jaringan SDN P4.

Tabel 1.Rincian konfigurasi Jaringan SDN OpenFlow

Komponen	Parameter konfigurasi	Nilai
Controller	Jumlah perangkat	≥ 1 (*multi-controller*)
	Setiap antarmuka	Alamat IP dan port TCP
	Protokol komunikasi	OpenFlow API
	Jumlah koneksi	≤ Jumlah perangkat *switch*
	Jenis koneksi	Kabel terdedikasi ke *switch (*koneksi otomatis sesuai konfigurasi *switch)*
Switch OpenFlow	Jumlah perangkat	≥ 1
	Setiap antarmuka	Alamat IP dan MAC
	Protokol komunikasi	Sesuai OpenFlow versi 1.3 (Eth, IP, MPLS, dll)
	Jumlah koneksi	1 buah *controller*, koneksi ke *switch* dan *host* sesuai konfigurasi pengguna

Komponen	Parameter konfigurasi	Nilai
	Jenis koneksi	Kabel terdedikasi ke *controller* (otomatis), dan kabel Ethernet antar ke *switch* dan *host* (diatur oleh pengguna)
	Fitur lain	Pemilihan *controller* oleh pengguna
Host/PC	Jumlah perangkat	Minimal = 0, ≥ 2 buah untuk pengujian koneksi
	Setiap antarmuka	Alamat IP dan MAC
	Protokol komunikasi	Sesuai OpenFlow versi 1.3 (Eth, IP, MPLS, dll)
	Jumlah koneksi	koneksi ke *switch* dan *host* sesuai pengaturan pengguna
	Jenis koneksi	Kabel Ethernet antar ke *switch* dan *host* (dapat ditentukan oleh pengguna)

Tabel 2.Rincian konfigurasi Jaringan SDN P4

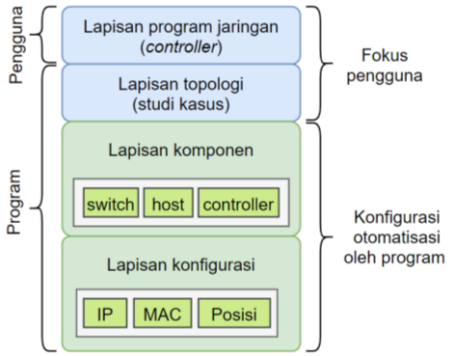
Komponen	Parameter konfigurasi	Nilai
Controller	Jumlah perangkat	1 (*single-controller*)
	Setiap antarmuka	*port* Thrift
	Protokol komunikasi	Thrift
	Jumlah koneksi	= Jumlah perangkat *switch*
	Jenis koneksi	*Remote procedure call* ke *switch*
Switch BMv2	Jumlah perangkat	≥ 1
	Setiap antarmuka	Alamat IP dan MAC
	Protokol komunikasi	Ditentukan oleh pengguna (diprogram dengan P4)
	Jumlah koneksi	1 buah *controller*, koneksi ke *switch* dan *host* sesuai konfigurasi pengguna
	Jenis koneksi	*Controller*: Inter-process communication (IPC). Kabel *Ethernet* antar ke *switch* dan *host* (diatur oleh pengguna)
Host/PC	Jumlah perangkat	Minimal = 0, ≥ 2 buah untuk pengujian koneksi
	Setiap antarmuka	Alamat IP dan MAC
	Protokol komunikasi	Ditentukan oleh pengguna (diprogram dengan scapy)
	Jumlah koneksi	koneksi ke *switch* dan *host* sesuai konfigurasi pengguna
	Jenis koneksi	Kabel *Ethernet* antar ke *switch* dan *host* (diatur oleh pengguna)

Perbedaan utama dari emulasi jaringan SDN berbasis OpenFlow dan P4 adalah pada manjemen komunikasi antara *controller* dan perangkat *switch*. Pada jaringan berbasis OpenFlow, *controller* diemulasikan sebagai sebuah *node* komputer terpisah dan dihubungkan ke *switch* dengan menggunakan kabel *Ethernet* (Tabel 1). Sebagai konsekuensi program yang dibuat harus mampu merekam konfigurasi koneksi antara *controller* dan *switch*. Bagian *controller* perlu diperhatikan mengenai informasi nilai alamat IP *controller* dan *port* TCP yang digunakan. Sedangkan pada bagian *switch*, perlu dicatat antarmuka yang merupakan jalur terpendek atau jalur khusus yang menghubungkan *switch* dengan *controller*.

Pada jaringan SDN berbasis P4, *controller* diemulasi sebagai sebuah proses internal yang terhubung ke *switch* BMV2 melalaui *inter-process communication* (IPC) berbasis protokol Thrift (Tabel 2). Skema ini menuntut pencatatan nomor *port* untuk protokol Thrift dari setiap *switch*, sehigga *controller* dapat berkomunikasi dengan *switch* untuk mengubah aturan pengiriman paket (*flow rules*). Pada emulasi jaringan berbasis P4 tidak diperlukan alamat IP karena *controller* dan perangkat *switch* tidak dihubungkan dengan protokol TCP/IP.

2.2. Metode pembuatan program

Pembuatan program menggunakan konsep MVC atau struktur koordinasi antara Data, Tampilan dan Kontrol. Bahasa pemrograman yang dipakai adalah Python 3 dengan komponen penyusun berupa *Class* dan modul *tkinter*. Komponen data dan tampilan direalisasikan menggunakan *Class method*, sedangkan komponen kontrol diimplementasikan menggunakan konsep *event-drivent* berbasis masukan dari pengguna.



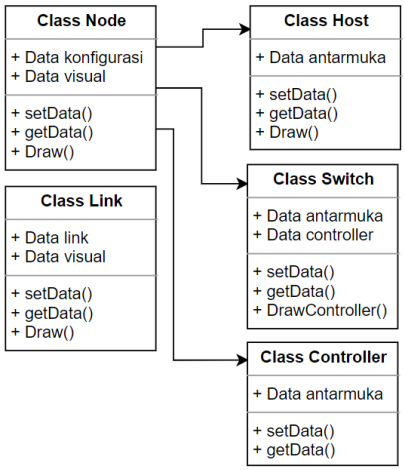
Gambar 1. Susunan abstraksi program

Struktur data atau komponen model diabstraksikan kedalam empat lapisan sebagaimana ditunjukkan oleh Gambar 1. Lapisan aplikasi adalah lapisan yang terpasang di perangkat *controller* dan merupakan komponen siap pakai yang disediakan oleh pengguna. Lapisan topologi atau struktur jaringan merupakan realsiasi dari studi kasus yang dipelajari. Lapisan komponen berisi deskripsi dari setiap komponen jaringan seperti perangkat *switch*, *controller*, dan *host*/PC. Lapisan konfigurasi menjabarkan konfigurasi secara terperinci dari setiap komponen jaringan.

Dengan mengkapsulasi rincian konfigurasi jaringan ke lapisan yang lebih rendah, kesalahan konfigurasi jaringan dapat diminimalkan. Selain itu, fokus pengguna dapat dapat lebih terarah untuk mempelajari program jaringan dan penerapannya pada topologi jaringan yang dibuat.

Meskipun program telah melakukan otomasi pengaturan jaringan, pada beberapa kasus tetap diperlukan konfigurasi khusus. Sebagai contoh, jika pengguna ingin melakukan analisis efisiensi algoritma *routing* suatu jaringan riil, maka sudah pasti konfigurasi jaringan harus disesuaikan dengan konfigurasi jaringan yang dijadikan rujukan. Implementasi program mengizinkan pengguna untuk mengubah konfigurasi tersebut secara manual melalui jendela konfigurasi komponen.

Diagram kelas untuk komponen data dan tampilan ditunjukkan oleh Gambar 2. Sedangkan daftar pendeteksian *event* untuk komponen kontrol ditunjukkan oleh Tabel 3.



Gambar 2. Digram struktur kelas dari program

Tabel 3.Jenis e*vent* dan perintah untuk komponen Kontrol

Event	Komponen	Perintah kontrol
Klik kiri *mouse*	Program	Tutup semua jendela konfigurasi
Klik kiri dan geser *mouse*	*Switch, host*	Pindahkan posisi komponen yang bersangkutan
Klik kanan *mouse*	*Switch, host, controller*	Buka jendela konfigurasi (tambah koneksi, atur konfigurasi secara manual, pilih *controller* untuk *switch*)
	link	Hapus *link*/koneksi dan sinkronisasi data
Geser *mouse*	Program	Gambar *link* jika sedang membuat koneksi antar komponen
Pemilihan menu	Ekspor ke OpenFlow	Menghasilkan konfigurasi mininet untuk jaringan OpenFlow
	Ekspor ke P4	Menghasilkan konfigurasi mininet untuk jaringan P4
	Simpan	Menyimpan proyek yang dibuat ke dalam *file*
	Buka	Membuka proyek yang dibuat
	Tutup	Menutup aplikasi

2.3. Penghasilan konfigurasi jaringan

Listing program berikut menunjukkan cara kerja program dalam menghasilkan konfigurasi jaringan berdasarkan topologi jaringan yang dibuat.

```
Program penghasil konfigurasi
Input: type, nodes, links
Output: cfg
  Initialization cfg = ""
  Get controller, switch, host, link
    controller, switch, host = extract(nodes)
    link = extract(links)
    if type = "OpenFlow"
      for item in controller do
        link_cfg = genConfig(link, item)
        cfg = cfg + getConfig(item) + link_cfg
      end for
    else:
      cfg = cfg + getConfig(controller)
    end if
    for item in switch do
      link_cfg = genConfig(link, item)
      cfg = cfg + getConfig(item) + link_cfg
    end for
    for item in host do
      link_cfg = genConfig(link, item)
      cfg = cfg + getConfig(item)+ link_cfg
    end for
```

Pembuatan konfigurasi jaringan ditentukan oleh jenis jaringan SDN (OpenFlow atau P4). Ditinjau dari konektivitas perangkat *controller* dengan *switch*, hanya jaringan OpenFlow yang memerlukan data konfigurasi konektivitas dengan controller. Jaringan P4 tidak menggunakan rujukan data konektivitas karena program controller terhubung dengan perangkat *switch* melalui *inter-process communication* (IPC). Ditinjau dari konektivitas antar perangakt *switch* dan antara perangkat *switch* dan *host*, kedua jenis jaringan SDN tersebut sama-sama memerlukan data konfigurasi konektivitas antara perangakat.

3. Hasil dan Pembahasan

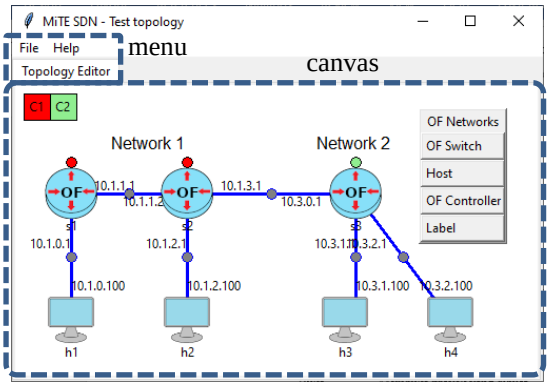
Penjelasan mengenai hasil pembuatan program penyunting topologi (MiTE) dibagi kedalam 4 bagian dengan rincian sebagai berikut. Bagian pertama menjelaskan hasil tampilan program berikut komponen-komponen program. Bagian kedua dan ketiga menyajikan contoh studi kasus *routing* paket berbasis IPv4 untuk jaringan OpenFlow dan P4. Bagian keempat menyajikan perbandingan program MiTE yang diusulkan pada penelitian ini dengan program-program penyunting topologi jaringan SDN yang sudah ada.

3.1. Tampilan program

Tampilan program yang berjalan pada sistem operasi Windows 10 ditujukkan oleh Gambar 3. Pada pengujian tersebut, jaringan SDN yang dibuat adalah jenis OpenFlow dengan 3 buah *switch*, 2 buah *controller*, dan 4 buah *host*/PC.

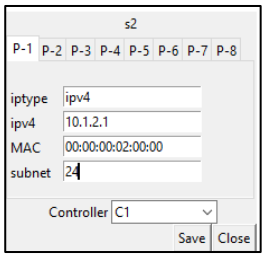
Penambahan komponen baru, baik *controller*, *switch*, maupun *host* dapat dilakukan dengan cara melakukan klik kanan pada bagian *canvas*. Terdapat empat buah elemen yang dapat ditambahkan dalam jaringan jenis OpenFlow oleh pengguna, yaitu: *switch*, *host*, *controller*, dan *label*. Komponen *label* digunakan untuk memberikan keterangan tambahan pada topologi yang dibuat. Program MiTE hanya mendukung satu buah

controller untuk jaringan jenis P4. Pengguna tidak perlu menambahkan *controller* karena program akan secara otomatis membuat dan mengkonfigurasi *controller* tersebut.



Gambar 3. Tampilan program MiTE SDN pada Windows 10

Secara umum, program MiTE secara otomatis menentukan alamat IP, MAC, dan *subnet* kepada setiap antarmuka *switch* dan *host*. Program MiTE juga mengizinkan pengubahan konfigurasi secara manual untuk parameter alamat IP, MAC, dan *subnet*. Pengguna dapat merubah konfigurasi ini melalui jendela konfigurasi dari setiap *switch* atau *host* dengan cara melakukan klik kanan pada masing-masing komponen. Gambar 4 menunjukkan tampilan jendela konfigurasi untuk *switch* 2 (s2).



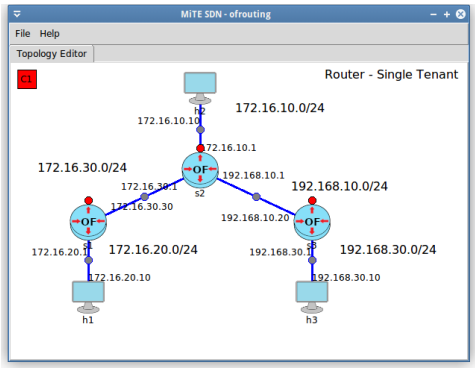
Gambar 4. Tampilan jendela konfigurasi pada *switch*

3.2. Studi kasus IP *Routing* pada jaringan OpenFlow

Studi kasus IP *routing* untuk OpenFlow merujuk pada Bab 11.1 dari dokumentasi resmi *controller* Ryu [10]. Sedangkan *virtual machine* (VM) yang digunakan untuk menjalankan emulasi dapat diunduh melalui [11].

Studi kasus berkenaan dengan pembuatan jaringan single tenant berbasis *controller* Ryu [12] dan protokol OpenFlow. Perangkat *controller* mengimplementasikan program REST API untuk menerina aturan *routing* dan memasangnya pada OpenFlow *switch*. Pengguna dapat mengirimkan aturan *routing* statis ke perangkat *controller* melalui aplikasi *web browser* atau program Curl.

Kelebihan program MiTE dibanding aplikasi lain adalah tampilan yang lebih informatif dan terdapatnya fitur konfigurasi manual untuk penyesuaian dengan studi kasus. Sebagaimana ditunjukkan oleh Gambar 5, selain dapat menampilkan alamat IP dari setiap antarmuka, pengguna juga dapat menambahkan keterangan tambahan seperti nilai *subnet* untuk memperjelas topologi yang dibuat. Dengan fitur konfigurasi manual, pengguna dapat menyesuaikan konfigurasi dari topologi yang dibuat, seperti alamat IP dan nilai *default routing* untuk setiap *host*. Hal ini dapat mempercepat proses pengujian dan pemahanan yang lebih baik terhadap skenario pengujian.



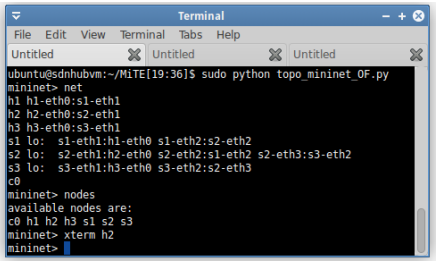
Gambar 5. Tampilan studi kasus OpenFlow sesuai [10]

Langkah pengujian pada studi kasus jaringan OpenFlow adalah sebagai berikut.

1. Membuat topologi jaringan
2. Mengkonversi menjadi file Mininet
3. Menjalankan emulasi Mininet
4. Menjalankan Program *controller*
5. Memasang *routing* statis melalui *controller*
6. Menguji jalur *routing* statis dengan perintah Ping

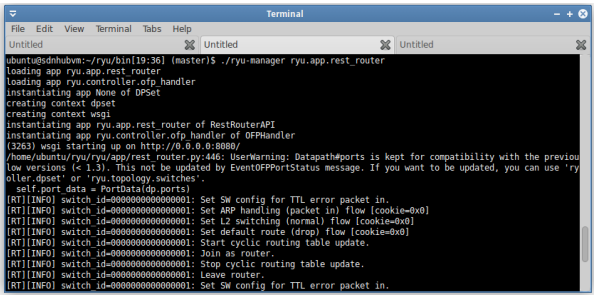
Pembuatan topologi sesuai gambar 5 dapat dilakukan dengan cepat. Pengguna dapat menambahkan 3 *switch*, 3 *host*, dan sebuah *controller* yang diikuti dengan menghubungkan setiap komponen dengan *link*. Secara otomatis, program menghasilkan alamat IP untuk setiap antarmuka *switch* dan *host*. Pengguna dapat mengubah alamat IP sesuai dengan studi kasus melalui jendela konfigurasi dari komponen tersebut (Gambar 4).

Pengguna dapat membuat file konfigurasi untuk Mininet dengan cara memilih menu *File*, *Export*, dan "*Export* to *Ryu Mininet (OF)*". Pengguna dapat menentukan *file* tujuan *export* yang kemudian dapat dijalankan menggunakan perintah *python* dengan status *adminstrator* (*sudo python*). Gambar 6 menunjukkan cara menjalankan emulasi Mininet untuk topologi jaringan yang telah dibuat.

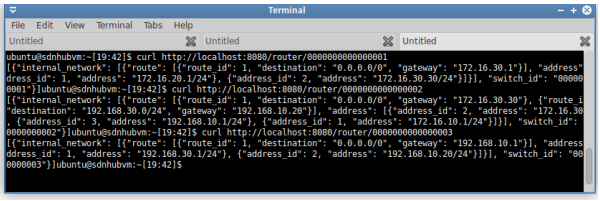


Gambar 6. Emulasi jaringan yang dibuat dengan program Mininet

Pengguna harus menjalankan program di perangkat *controller* untuk memasang daftar *routing statis* ke setiap *switch* (Gambar 7). Selanjutnya, pengguna dapat memasang aturan *routing statis* menggunakan program Curl ke alamat IP dari perangkat *controller* (Gambar 8).

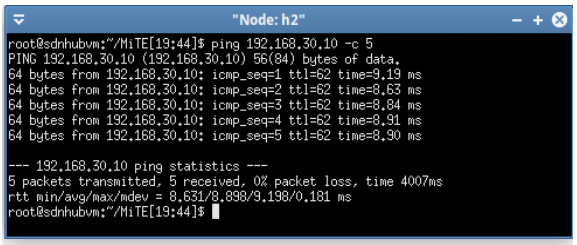


Gambar 7. Program *controller* berbasis Ryu



Gambar 8. Instalasi *routing statis* menggunakan program Curl

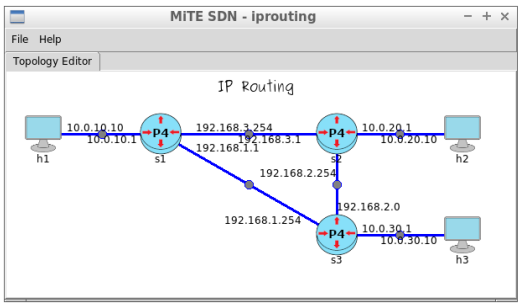
Hasil pengujian ditunjukkan oleh Gambar 9. Dikarenakan semua perangkat *switch* sudah memiliki aturan *routing* statis yang sesuai, paket dapat dikirim dari *host* 1 ke *host* 2 melalui perangkat *switch* 1 dan *switch* 2.



Gambar 9. Pengujian *routing* statis

3.3. Studi kasus IP *Routing* pada jaringan P4

Studi kasus IP *routing* untuk P4 merujuk pada pelatihan yang terdapat di [13]. Sedangkan *virtual machine* (VM) untuk menjalankan emulasi dapat diunduh melalui [14]. Topologi jaringan dari studi kasus dapat dibuat dengan program MiTE sebagaimana ditunjukkan oleh gambar 10.

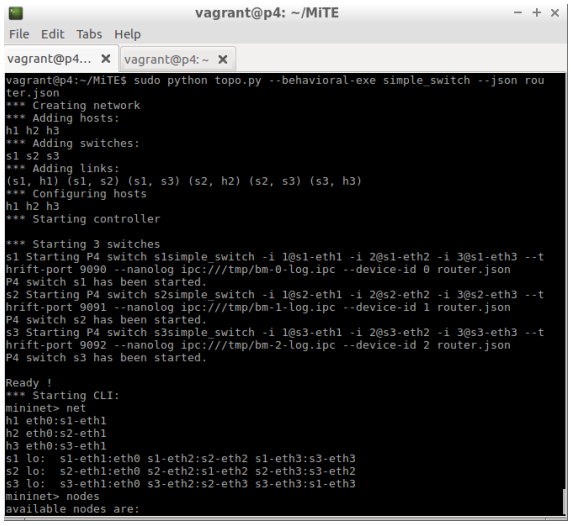


Gambar 20. Tampilan studi kasus sesuai [13]

Langkah pengujian studi kasus IP *routing* pada jaringan P4 dapat dijabarkan sebagai berikut.

1. Membuat topologi jaringan
2. Mengkonversi topologi menjadi file Mininet
3. Menjalankan emulasi Mininet
4. Memasang aturan *routing* dengan menjalanan program *controller*
5. Menguji jalur *routing* dengan perintah Ping

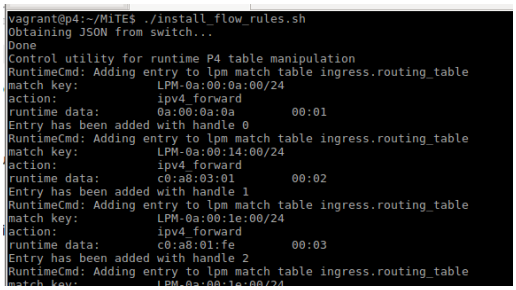
Pembuatan topologi dapat dimulai dengan membuat proyek baru berjenis jaringan P4. Selanjutnya, pengguna dapat menambahkan 3 buah *switch* P4 dan 3 buah *host* ke dalam bagian *canvas*. Pengguna dapat membuka jendela konfigurasi dari setiap komponen *switch* dan *host* untuk melakukan penyesuaian alamat IP.



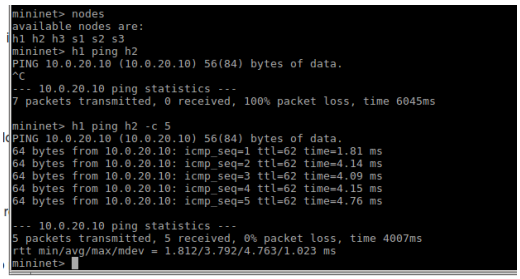
Gambar 31. Emulasi jaringan untuk studi kasus P4 di Mininet

Pengguna dapat mengakses menu *File*, *Export*, dan *Export* to BMv2 Mininet (P4), untuk menghasilkan *file* Mininet dari topologi yang dibuat. Selanjutnya, pengguna dapat menjalankan emulasi jaringan dengan menyertakan program P4 yang sudah dikompilasi (sesuai studi kasus) dan model *switch* P4 yang dipakai. Pada pengujian ini, model *switch* P4 yang dipakai adalah *simple_switch* (Gambar 11).

Aturan *routing* dapat dipasang dengan menjalankan program *controller* seperti yang ditunjukkan oleh Gambar 12. Sebelum program *controller* berjalan, perangkat *switch* tidak dapat mengirim paket karena ketiadaan aturan. Pada tahap ini, struktur pemrosesan paket sudah terpasang di *switch*. Akan tetapi, *switch* belum memiliki aturan yang menentukan paket yang akan diproses. Setelah aturan *routing* dipasang oleh program *controller* paket dari *host* 1 dapat terkirim ke *host* 2 (Gambar 13).



Gambar 42. Pemasangan aturan *routing* oleh program *controller*.



Gambar 53. Pengujian *routing* dengan perintah Ping

3.3. Perbandingan program

Beberapa penelitian telah membuat program penyunting topologi untuk jaringan SDN. Tabel 4 berisi perbandingan antara program yang kami buat (MiTE) dengan program yang dibuat pada penelitian-penelitian sebelumnya. Terdapat 5 aspek perbandingan yang diamati, yaitu: dukungan untuk protokol OpenFlow (OF), bahasa P4 (P4), kemampuan emulasi terintegrasi (*Integrated Emulation*/IE), kemampuan konfigurasi terperinci untuk setiap antarmuka *ethernet* (*Link Configuration*/LC), dan antarmuka grafis yang informatif (*Informative* UI).

Tabel 4. Perbandingan program dengan penelitian yang sudah ada

Program	OF	P4	IE	LC	IUI
Diajukan (MiTE)	V	V	-	V	V
MiniEdit [8]	V	-	V	-	-
Visual network description [15]	V	-	-	?	V
Mininet editor [16]	V	-	-	V	V
NetIDE [17]	V	-	-	?	V
Containernet [18]	V	-	V	-	-

Penelitian sebelum hanya mendukung SDN dengan protokol OpenFlow dan tidak mendukung protokol P4. Dalam hal ini, program yang kami buat memiliki nilai lebih karena dapat mendukung pemrograman jaringan dengan basis bahasa P4.

Fitur emulasi terintegrasi (IE) hanya dimiliki MiniEdit dan containernet. Fitur ini memungkinkan pengguna untuk langsung mengemulasikan topologi jaringan yang dibuat. Akan tetapi, berdasarkan studi pada [16], fitur ini memiliki kelemahan dalam hal kestabilan program dan keamanan sistem dikarenakan memerlukan akses *root*. Dengan demikian, fitur tersebut bukanlah pilihan yang tepat untuk digunakan oleh orang awam.

Fitur untuk memilih pasangan antarmuka jaringan dan *link* merupakan fitur yang penting dalam membuat konfigurasi jaringan agar sesuai dengan studi kasus yang dipelajari. Program yang kami buat (MiTE) dan Mininet editor memiliki keunggulan dalam dalam menentukan koneksi dari setiap antarmuka komponen *switch* dan *host*. Program miniEdit dan Containernet tidak memiliki fitur ini sedangkan dua program lain *Visual Network Desription* dan NetIDE tidak memberikan informasi yang rinci apakah dapat mendukung fitur ini atau tidak.

Tampilan antarmuka grafis (UI) yang informatif memudahkan penggunan dalam mempelajari suatu studi kasus dengan baik. Pengguna lebih mudah memahami apabila program menampilkan informasi yang komprehensif mengenai jaringan yang dibuat, seperti informasi alamat IP dari setiap antarmuka, penjelasan singkat studi kasus, dan informasi *subnet* dari setiap bagian jaringan. Program MiniEdit dan Containernet hanya menampilkan komponen jaringan tanpa informasi pendukung sehingga menyulitkan pengguna awam dalam mengecek konfigurasi jaringan.

Berdasarkan perbandingan dari kelima aspek tersebut, program yang kami buat memiliki kelengkapan yang lebih baik, mulai dari dukungan jaringan SDN yang beragam, pengaturan koneksi yang lebih terperinci, dan tampilan UI yang informatif.

4. Kesimpulan

Pada penelitian ini, telah dikembangkan aplikasi penyunting topologi jaringan SDN bernama MiTE yang mampu untuk mempermudah pembelajaran jaringan SDN. Aplikasi memiliki beberapa keunggulan sebagai berikut. Pertama, aplikasi mendukung jenis jaringan SDN yang lebih beragam yaitu OpenFlow dan P4. Pengguna dapat membuat topologi jaringan SDN dengan mudah sesuai tutorial resmi kedua jenis SDN tersebut. Disamping itu, aplikasi kami mendukung konfigurasi jaringan yang lebih terperinci untuk menyesuaikan kasus riil di lapangan. Tampilan antarmuka aplikasi juga lebih lengkap dengan informasi alamat IP, *subnet*, dan informasi tamabahan lain dari pengguna.

Ucapan Terimakasih

Kami mengucapkan terimakasih untuk Pusat Penelitian Informatika LIPI, atas sarana pendukung yang disediakan dalam melakukan penelitian ini dan LPPM IT Telkom Purwokerto dalam pembiayaan publikasi.

Daftar Rujukan

[1] Nunes, B. A. A., Mendonca, M., Nguyen, X. N., Obraczka, K., & Turletti, T., 2014, A survey of software-defined networking: Past, present, and future of programmable networks. IEEE Communications surveys & tutorials, 16(3), 1617-1634.

[2] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., ... & Turner, J., 2008, OpenFlow:

enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 38(2), 69-74.
[3] Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., ... & Walker, D., 2014, P4: Programming protocol-independent packet rocessors. ACM SIGCOMM Computer Communication Review, 44(3), 87-95.
[4] Datta, R., Choi, S., Chowdhary, A., & Park, Y., 2018, P4guard: Designing p4 based firewall. In MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM) (pp. 1-6). IEEE.
[5] Turkovic, B., Kuipers, F., van Adrichem, N., & Langendoen, K., 2018, Fast network congestion detection and avoidance using P4. In Proceedings of the 2018 Workshop on Networking for Emerging Applications and Technologies (pp. 45-51).
[6] Kim, C., Sivaraman, A., Katta, N., Bas, A., Dixit, A., & Wobker, L. J., 2015, In-band network telemetry via programmable dataplanes. In ACM SIGCOMM.
[7] Lantz, B., Heller, B., & McKeown, N., 2010, A network in a laptop: rapid prototyping for software-defined networks. In Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (pp. 1-6).
[8] Córdoba López, S., 2019, Estudio de redes SDN mediante Mininet y MiniEdit. Doctoral dissertation, Universitat Politècnica de València, 2019.
[9] Pfaff, B., Pettit, J., Koponen, T., Jackson, E., Zhou, A., Rajahalme, J., ... & Amidon, K., 2015, The design and implementation of open vswitch. In 12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15) (pp. 117-130).
[10] RYU project team, Ryu SDN Framework Using OpenFlow 1.3 [Online] (Updated 16 Jan 2005) Tersedia di :https://osrg.github.io/ryu-book/en/Ryubook.pdf [Accessed 9 September 2020]
[11] SDN Hub, 2014, All-in-one SDN App Development Starter VM | SDN Hub [Online] (Updated 16 Jan 2005) Tersedia di :http://sdnhub.org/tutorials/sdn-tutorial-vm/ [Accessed 9 September 2020]
[12] Tomonori, F. U. J. I. T. A, 2013, Introduction to ryu sdn framework. Open Networking Summit, 1-14.
[13] Tomek Osiński, 2019, P4-Research/p4-demos [Online] (Updated 3 Mar 2019) Tersedia di : https://github.com/P4-Research/p4-demos/tree/master/ip-routing [Accessed 10 September 2020]
[14] Stanford, 2019, P4 Tutorial 2019-08-15.ova [Online] (Updated 15 Aug 2019) Tersedia di : http://stanford.edu/~sibanez/docs/P4%20Tutorial%202019-08-15.ova [Accessed 10 September 2020]
[15] Fontes, R. R., Oliveira, A. L., Sampaio, P. N., Pinheiro, T. R., & Figueira, R. A., 2014, Authoring of OpenFlow networks with visual network description (SDN version)(WIP). In Proceedings of the 2014 Summer Simulation Multiconference (pp. 1-6).
[16] Vyčítal, T., 2019, GUI editor pro Mininet. Diploma, Univerzita Pardubice
[17] Doriguzzi-Corin, R., Salvadori, E., Gutiérrez, P. A., Stritzke, C., Leckey, A., Phemius, K., ... & Guerrero, C., 2015, NetIDE: removing vendor lock-in in SDN. In Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft) (pp. 1-2). IEEE.
[18] Peuster, M., Kampmeyer, J., & Karl, H. ,2018, Containernet 2.0: A rapid prototyping platform for hybrid service function chains. In 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft) (pp. 335-337). IEEE.
[19]