Jurnal Informatika Terpadu

https://journal.nurulfikri.ac.id/index.php/JIT
ISSN ONLINE : 2460-8998

PENERAPAN EFEKTIFITAS FIREWALL PADA *SOFTWARE DEFINED NETWORK* BERBASIS OPENFLOW

Fathul Muiin[1] , Henry Saptono[2]

[1, 2]Teknik Informatika, Sekolah Tinggi Teknologi Terpadu Nurul Fikri
Jakarta Selatan, DKI Jakarta, Indonesia 12640
fmuiin14@gmail.com, henry@nurulfikri.co.id

Abstract

The use of internet access in the world is growing and in line with increasingly complex computer network technology. Therefore, data security on a computer becomes a crucial part of a network. Thus, SDN is a solution to provide computer network needs today. Software-Defined Network (SDN) is an approach to network technology that simplifies network control and management. This network will use the OpenFlow protocol, whose primary principle is to separate the control plane and data plane functions on the device. The network control on a controller is programmable, so with the SDN, the network will be easily managed and more flexible. This firewall implementation and analysis uses a mininet emulator to create a simple network topology. In firewall, testing uses the XML language for data flow implementation, then uses the postman application as a tool to add a new flow table to the switch, and the controller used is opendaylight.

Keywords: *Software Defined Network*; *Openflow*; *Firewall*; *Controller*; *Opendaylight*; *Postman*

Abstrak

Penggunaan akses internet di dunia semakin berkembang, dan selaras dengan perkembangan teknologi jaringan komputer yang semakin kompleks. Oleh karena itu, keamanan data pada sebuah komputer menjadi salah satu bagian yang sangat penting dalam sebuah jaringan. Dan SDN merupakan sebuah solusiuntuk menyediakan kebutuhan jaringan komputer saat ini. *Software Defined Network* (SDN) merupakan pendekatan pada teknologi jaringan yang melakukan penyederhanaan terhadap kontrol dan manajemen jaringan. Pada jaringan ini nantinya akan menggunakan protokol *openflow*, yang prinsip utamanya memisahkan fungsi *control plane* dan *data plane* pada perangkat. Kontrol jaringan pada sebuah *controller* bersifat *programmable*, jadi dengan adanya SDN maka jaringan akan mudah diatur dan lebih fleksibel. Implementasi dan analisis *firewall* ini menggunakan *emulator mininet* untuk membuat topologi jaringan yang sederhana. Dalam pengujian *firewall* menggunakan bahasa XML untuk implementasi aliran data, lalu menggunakan aplikasi *postman* sebagai alat untuk menambahkan *flow table* baru pada *switch*, dan*controller* yang digunakan adalah *Opendaylight*.

Kata kunci: *Software Defined Network*, *Openflow*, *Firewall*, *Controller*, *Opendaylight*, *Postman*

1. PENDAHULUAN

Layanan internet dan teknologi informasi berkembang dengan berbagai kompleksitas, desain, manajemen, dan operasional yang menyebabkan jumlah perangkat yang terhubung dan volume aliran data dalam jaringan telah meningkat dengan sangat pesat dalam beberapa tahun terakhir. Dalam halpenerapan teknologi perangkat jaringan, sangat dibutuhkan metode baru untuk melakukan kontrol dan pengelolaan jaringan yang efisien dan efektif. Disamping itu, kita juga perlu memperhatikan aspek keamanan dalam perangkat jaringan. Perusahaan dan penyedia layanan paham akan keterbatasan arsitektur jaringan konvensional yang tidak fleksibel dan membutuhkan biaya banyak.

Software Defined Network (SDN) adalah sebuah konsep pendekatan jaringan komputer dimana sistem pengontrol dari arus data (*data plane*) dipisahkan dari perangkat kerasnya (*control plane)*. Pemisahan *data plane* dan *control plane* pada perangkat jaringan komputer seperti *router* dan *switch* memungkinkan kita memprogram perangkat tersebut sesuai dengan yang diinginkan secara terpusat [1].

Analisis efektivitas *firewall* pada SDN akan diteliti dengan cara mengatur *flow table* pada suatu topologi jaringan dengan menggunakan *tools postman* sebagai metode untuk *push* menuju *web service* yang disediakan oleh *controller opendaylight*, dan menggunakan bahasa XML. Cara

Diterima 07 Juli 2018, Direvisi 12 Agustus 2018, Diterima untuk publikasi 15 September 2018

mengatur *flow table* adalah dengan menerapkan sistem *drop* pada sebuah topologi jaringan, karena pada setiap topologi jaringan, *default* kontrol akses dari jaringannya adalah *accept.* Dan peneliti menguji kontrol akses dengan cara *drop* menggunakan *IP Address, MAC Address, TCP, UDP,* dan *ICMP*. Sehingga pada nantinya akan dibuatkesimpulan dari hasil pengujian dan implementasi. Dalam pelaksanaannya, akan ada 3 topologi yang nanti di ujikan yaitu topologi *single,* topologi *linear,* dan topologi *tree.* Peneliti menggunakan macam-macam topologi karena pada penelitian ini ingin mengetahui pola dari tingkah laku jaringan SDN dan kinerja yang dimiliki terhadap jaringan yang semakin besar dansemakin kompleks.

2. METODOLOGI PENELITIAN

2.1 Studi Literatur

Tahapan ini dilakukan dengan mencari, mengumpulkan dan membaca artikel, jurnal ilmiah, buku elektronik (*e-book*), website maupun beberapa skripsi peneliti lainnya untuk mengkaji mengenai implementasi teknologi SDN.

Hasil dari studi literatur adalah pembuatan acuan rancangan penelitian serta acuan bagaimana penelitian harus dilakukan dan data apa saja yang diperlukan sehingga tujuan pada penelitian ini dapat tercapai.

2.2 Analisis Kebutuhan Sistem

Tahapan ini dilakukan untuk menganalisis apa saja kebutuhan yang diperlukan sebuah sistem dalam pengimplementasiannya pada *firewall*. *Firewall* yang diterapkan *software defined network*, kemudian dievaluasi apakah sudah optimal atau belum dalam penerapannya pada sebuah SDN.

2.3 Perancangan Sistem

Pada tahap ini akan dilakukan perancangan sistem yang dibuat dan diujikan, yang meliputi perancangan topologi untuk pengujian *firewall.*

2.4 Implementasi

Setelah dilakukan analisis tujuan dan perancangan sistem maka masuk ke fase implementasi, dimana pada fase ini dilakukan proses konfigurasi dari topologi yang telah dibuat ke simulasi jaringan hingga siap untuk dilakukan pengujian *firewall.*

2.5 Pengujian

Pada tahap ini akan dilakukan pengujian terhadap sistem yang akan dianalisis untuk mengukur bagaimana dampak *software defined network* yang diimplementasikan pada *firewall* dan seberapa baik perintah yang diberi berjalan dengan semestinya.

2.6 Analisis Hasil

Pada tahap ini berisi sebuah analisa hasil pengujian dari simulasi dan implementasi yang telah dibuat, apakah sesuai dengan perumusan masalah dan tujuan yang telah dibuat oleh peneliti atau tidak.

2.7 Penarikan Kesimpulan dan Saran

Dari hasil pengujian dapat ditarik kesimpulan apakah teknologi yang diterapkan dapat menjadi inovasi baru bagi keamanan sebuah *firewall* pada komputer atau tidak, dan menjadi solusi bagi banyak pihak serta memerlukan saran-saran untuk optimalisasi serta kelanjutan penelitian berikutnya.

3. ANALISIS DAN PERANCANGAN

3.1 Analisis Kebutuhan *Hardware*

Dalam melakukan penelitian ini, kebutuhan *hardware* minimal yang dapat digunakan untuk melakukan penelitian adalah sesuai dengan spesifikasi dari *Linux Ubuntu* 16.04 LTS. Dan spesifikasi minimal*hardware*nya adalah sebagai berikut:

Prosesor : Dual Core 2 GHz
RAM : 2GB
Hardisk : 25GB
VGA : 1024 x 768

Dalam penelitian kali ini, peneliti menggunakan laptop dengan spesifikasi sebagai berikut:

Prosesor : Intel® Core™ i5-3320M CPU@2.60GHz
RAM : 8GB
Hardisk : 125GB SSD
VGA : 1024 x 768

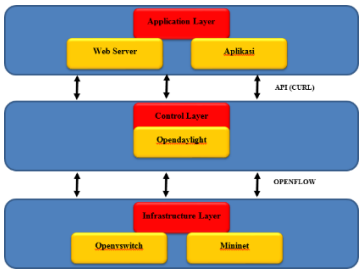
3.2 Analisis Kebutuhan *Software*

Software yang digunakan dalam penelitian ini adalah sebagai berikut:

Host : Mininet 2.3.0d1
Switch : Openvswitch ovs-vswitchd 2.5.4
Controller : Opendaylight Nitrogen SR3(seventh release)
OS : Linux Ubuntu 16.04 LTS Xenial Xerus

3.3 Perancangan Arsitektur Sistem

Berikut adalah arsitektur sistem dalam pengujian efektivitas *firewall* pada SDN.



Gambar 1. Arsitektur Sistem

Ada tiga lapisan dalam perancangan arsitektur sistem yang akan dibuat, berikut penjelasan masing- masing dari tiga lapisan tersebut:

1. *Infrastructure Layer*

Pada lapisan ini, peneliti menggunakan *openvswitch* sebagai *switch* virtual yang digunakan untuk melakukan penelitian. Sedangkan *Mininet* adalah emulator berbasis CLI (*Command Line Interface*) yang digunakan untuk membuat sebuah topologi jaringan sesuai dengan kebutuhan yang dapat disesuaikan dengan penelitian.

2. *Control Layer*

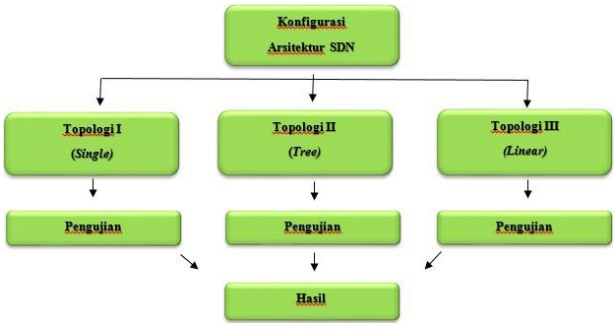
Pada lapisan *Control Layer*, penelitimenggunakan *software opendaylight* sebagai pengontrol data dan menjadi penghubung antara *Infrastructure Layer* (lapisan bawah) dan *Application Layer* (lapisan atas).

3. *Application Layer*

Merupakan tempat yang menjadi antar muka untuk memudahkan peneliti dalam melakukan fungsi konfigurasi, kontrol, dan evaluasi.

3.4 Rancangan Pengujian

Penelitian kali ini menggunakan 3 topologi yang dibuat menggunakan *software mininet.* Topologi yang dibuat yaitu topologi *single*, topologi *tree*, dan topologi l*inear*. Hal yang di uji dalam *firewall* pada *software defined network* adalah *blocking* berdasarkan *mac address, blocking* koneksi *TCP*, *UDP*, *ICMP*, dan *blocking* berdasarkan *IP address*. Untuk lebih leTngkapnya, silahkan perhatikan gambar di bawah:

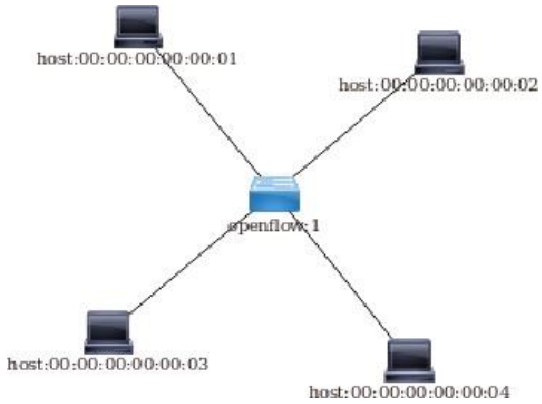


Gambar 2. Rancangan Pengujian dengan Topologi

1. Konfigurasi Arsitektur SDN

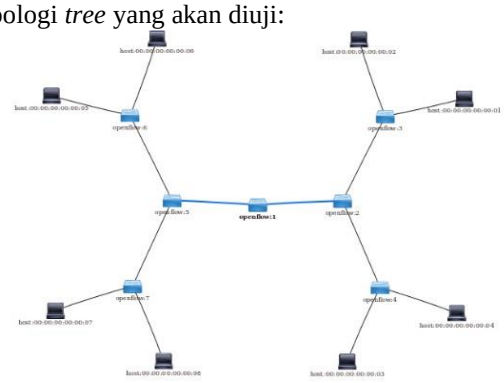
Pada bagian ini, peneliti akan melakukan pengaturan terlebih dahulu untuk konfigurasi *Software Defined Network*, implementasi dengan menggunakan simulasi *mininet*, dan nantinya akan dilakukan penelitian dengan 3 jenis topologi:

a. Topologi 1 (*Single Topology)*
Topologi jaringan yang pertama dibuatmenggunakan *software mininet*, dan menggunakan topologi *single* dengan 1 buah *switch* dan 4 *host*. Berikut gambar topologi *single* yang akan di uji:



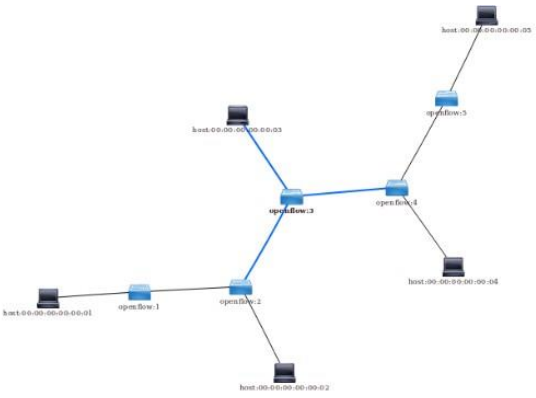
Gambar 3. *Single Topology*

b. Topologi 2 (*Tree Topology)*
Topologi jaringan yang kedua dibuat menggunakan *software mininet*, dan menggunakan topologi *tree* dengan 7 buah *switch* dan 8 *host*. Berikut gambar topologi *tree* yang akan diuji:



Gambar 4. *Tree Topology*

c. Topologi 3 (*Linear Topology)*
Topologi jaringan yang ketiga dibuat menggunakan *software mininet*, dan menggunakan *topologi linear* dengan 5 buah *switch* dan 5 *host*. Berikut gambar topologi linear yang akan diuji:



Gambar 5. *Linear Topology*

2. Pengujian

Secara *default*, topologi jaringan yang dibangun dengan mininet ini tidak menerapkan akses kontrol (*firewall*). Untuk itu, dalam tahap pengujian ini peneliti akan melakukan pengaturan dan penerapan akses kontrol (*firewall*) dalam jaringan berbasis SDN yang dibangun dengan simulasi menggunakan *mininet*. Skenario pengaturan akses kontrol (*firewall*) tersebut bisa dilihat dari tabel berikut ini.

Tabel 1. Pengujian

Rule No	SRC MAC	DST MAC	SRC IP	DST IP	PROTOCOL	DST PORT	ACTION	RESULT
1	00:00:00:00:00:01	00:00:00:00:00:04	ANY	ANY	ICMP	ANY	DROP	SUCCESS
2	00:00:00:00:00:02	00:00:00:00:00:03	10.0.0.4	10.0.0.5	TCP	ANY	DROP	SUCCESS
3	00:00:00:00:00:03	00:00:00:00:00:02	10.0.0.1	10.0.0.3	UDP	ANY	DROP	SUCCESS
4	00:00:00:00:00:04	00:00:00:00:00:01	ANY	10.0.0.6	UDP	ANY	DROP	SUCCESS

Dari table di atas, peneliti akan menerapkan akseskontrol dengan kriteria-kriteria sebagai berikut:

a. SRC MAC : Source Mac Address
b. DST MAC: Destination Mac Address
c. SRC IP: Source IP Address
d. DST IP: Destination IP Address
e. PROTOCOL: TCP/ UDP
f. DST PORT: Destination Port
g. ACTION: DROP/ ALLOW
h. RESULT: SUCCESS/ DENY

3. Hasil

Hasil pengujian dari penerapan akses kontrol (*firewall*) yang dilakukan oleh peneliti dalam tahapan pengujian adalah suatu kondisi atau status keberhasilan dari skenario *rule* yang diterapkan, yang nantinya akan dibuat laporan sehingga menjadi referensi bagi penelitian pengujian *firewall* menggunakan *opendaylight* untuk penelitian selanjutnya.

4. HASIL PENELITIAN

A. *Blocking IP Address Topologi 1 (Single)*

Pengujian pertama melakukan pengalamatan *rule* adalah dengan *blocking* aliran data berdasarkan *IP Address.* Pengujian ini mencoba menambahkan *flow*pada tabel *flow openflow:4* (*s1*), sehingga saat ada aliran data menuju *host* 4 dari *host* 2, koneksinya akan di blok. Berikut perintahnya dengan menggunakan *POSTMAN.*

Pada *Authorization*:

- Type : Basic Auth
- Username : admin
- Password : admin

Pada *Headers*:

- Key : Content-Type
- Value : application/xml

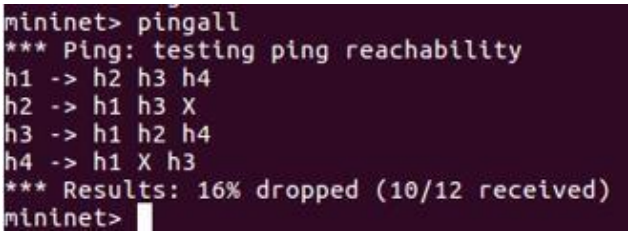
Pada *URL Bar*:

- PUT *http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:1/table/0/flow/1*

Pada *Body (raw)*

```
<flow xmlns="urn:opendaylight:flow:inventory">
        <id>1</id>
        <table_id>0</table_id>
        <match>
                <ethernet-match>
                <ethernet-type>
<type>2048</type> </ethernet-type>
                </ethernet-match>
                <ipv4-
destination>10.0.0.4/32</ipv4-destination>
                <in-port>2</in-port>
        </match>
        <priority>15</priority>
</flow>
```

Hasil dari *flow* yang baru ditambahkan sebagai berikut. Antara h2 dan h4 tidak bisa salingberkomunikasi, karena aliran datanya telah di blok.



Gambar 6. Hasil dari *Flow Berdasarkan IP Address Topologi 1*

Peneliti juga melakukan beberapa pengujian lainnya, dan hasilnya dapat di lihat pada tabel berikut.

Tabel 2. Pengujian Berdasarkan *IP Address Topologi 1*

Rule No	SRC MAC	DST MAC	SRC IP	DST IP	PROTOCOL	DST PORT	ACTION	RESULT
1	00:00:00:00:00:02	00:00:00:00:00:04	10.0.0.2	10.0.0.4	ANY	ANY	DROP	SUCCESS
2	00:00:00:00:00:01	00:00:00:00:00:03	10.0.0.1	10.0.0.3	ANY	ANY	DROP	SUCCESS
3	00:00:00:00:00:01	00:00:00:00:00:04	10.0.0.1	10.0.0.4	ANY	ANY	DROP	SUCCESS
4	00:00:00:00:00:04	00:00:00:00:00:01	10.0.0.4	10.0.0.1	ANY	ANY	DROP	SUCCESS

Dapat terlihat bahwa hasil dari pengujian yangdi lakukan dalam kontrol akses *firewall* pada *rule flow* berhasil dilakukan sesuai dengan perintah yang di jalankan.

B. *Blocking IP Address Topologi 2 (Tree)*

Selanjutnya *blocking* aliran data berdasarkan *IP Address* Topologi 2. Kali ini akan mencobamenambahkan *flow* pada *table flow openflow:5*, dengan aliran data membiarkan semua jaringan dapat berkomunikasi dengan h8, kecuali h5 dan h6 yang *rule*nya ditambahkan pada *openflow:5*, dengan menggunakan perintah *PUT* pada *address bar*, dan alamat yang ditujukan adalah sebagai berikut.

http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:5/table/0/flow/1

Pada *Body (raw)*

```
<flow xmlns="urn:opendaylight:flow:inventory">
        <id>1</id>
        <table_id>0</table_id>
                <match>
                        <ethernet-match>
                        <ethernet-type>
<type>2048</type> </ethernet-type>
                        </ethernet-match>
                        <ipv4-
destination>10.0.0.8/32</ipv4-destination>
                        <in-port>1</in-port>
                </match>
        <priority>20</priority>
</flow>
```

Hasilnya sebagai berikut:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 h5 h6 h7 h8
h3 -> h1 h2 h4 h5 h6 h7 h8
h4 -> h1 h2 h3 h5 h6 h7 h8
h5 -> h1 h2 h3 h4 h6 h7 X
h6 -> h1 h2 h3 h4 h5 h7 X
h7 -> h1 h2 h3 h4 h5 h6 h8
h8 -> h1 h2 h3 h4 X X h7
*** Results: 7% dropped (52/56 received)
mininet>
```

Gambar 7. Hasil dari *Flow Berdasarkan IP Address Topologi 2*

Peneliti juga melakukan beberapa pengujian lainnya, dan hasilnya dapat di lihat pada tabel berikut.

Tabel 3. Pengujian Berdasarkan *IP Address Topologi 2*

Rule No	SRC MAC	DST MAC	SRC IP	DST IP	PROTOCOL	DST PORT	ACTION	RESULT
1	00:00:00:00:00:05	00:00:00:00:00:08	10.0.0.5	10.0.0.8	ANY	ANY	DROP	SUCCESS
2	00:00:00:00:00:01	ANY	10.0.0.1	ANY	ANY	ANY	DROP	SUCCESS
3	00:00:00:00:00:08	00:00:00:00:00:06	10.0.0.8	10.0.0.6	ANY	ANY	DROP	SUCCESS
4	00:00:00:00:00:04	ANY	10.0.0.4	ANY	ANY	ANY	DROP	SUCCESS

C. *Blocking IP Address Topologi 3 (Linear)*

Selanjutnya *blocking* aliran data berdasarkan *IP Address* Topologi 3. Kali ini akan mencoba untuk menambahkan *flow* pada *table flow openflow:3*, dengan aturan h3 tidak bisa melewati *destination port openflow:3:2*, namun jaringan selain *destination port* 2 bisa menjalankan aliran data. Berikut perintahnya dengan menggunakan *POSTMAN*.

Pada *Authorization*:

- Type : Basic Auth
- Username : admin
- Password : admin

Pada *Headers*:

- Key : Content-Type
- Value : application/xml

Pada *URL Bar*:

- PUT *http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:3/table/0/flow/2*

Pada *Body (raw)*

```
<flow xmlns="urn:opendaylight:flow:inventory">
        <id>2</id>
        <table_id>0</table_id>
                <match> <ethernet-match>
                        <ethernet-type>
<type>2048</type>
                        </ethernet-type>
                        </ethernet-match>
                        <ipv4-
destination>10.0.0.3/32</ipv4-destination>
                        <in-port>2</in-port>
                </match>
        <priority>29</priority>
</flow>
```

Hasilnya sebagai berikut:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X h4 h5
h2 -> h1 X h4 h5
h3 -> X X h4 h5
h4 -> h1 h2 h3 h5
h5 -> h1 h2 h3 h4
*** Results: 20% dropped (16/20 received)
mininet>
```

Gambar 8. Hasil dari *Flow Berdasarkan IP Address Topologi 3*

Peneliti juga melakukan beberapa pengujian lainnya, dan hasilnya dapat di lihat pada tabel berikut:

Tabel 4. Pengujian Berdasarkan *IP Address Topologi 3*

Rule No	SRC MAC	DST MAC	SRC IP	DST IP	PROTOCOL	DST PORT	ACTION	RESULT
1	00:00:00:00:00:01	00:00:00:00:00:03	10.0.0.1	10.0.0.3	ANY	ANY	DROP	SUCCESS
2	00:00:00:00:00:05	ANY	10.0.0.5	ANY	ANY	ANY	DROP	SUCCESS
3	00:00:00:00:00:04	ANY	10.0.0.4	ANY	ANY	ANY	DROP	SUCCESS
4	00:00:00:00:00:02	00:00:00:00:00:03	10.0.0.2	10.0.0.3	ANY	ANY	DROP	SUCCESS

D. *Blocking MAC Address Topologi 1 (Single)*

Sealnjutnya *blocking* aliran data berdasarkan *MAC Address* Topologi 1. Dan aturan yang akan di tambahkan adalah semua aliran data yang menuju atau dari h1 akan di *blok*, dengan menggunakan perintah *PUT* pada *address bar*, dan alamat yang ditujukan adalah sebagai berikut.

```
http://localhost:8181/restconf/config/opendayl
ight-
inventory:nodes/node/openflow:1/table/0/flow/1
```

Berikut teks *XML* yang perlu ditambahkan menggunakan *POSTMAN*:

```
<flow xmlns="urn:opendaylight:flow:inventory">
    <table_id>0</table_id>
    <id>1</id>
    <cookie_mask>255</cookie_mask>
    <installHw>false</installHw>
    <match>
        <ethernet-match>
            <ethernet-source>

<address>00:00:00:00:00:01</address>
            </ethernet-source>
        </ethernet-match>
    </match>
    <priority>15</priority>
</flow>
```

Hasilnya sebagai berikut.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X
h2 -> X h3 h4
h3 -> X h2 h4
h4 -> X h2 h3
*** Results: 50% dropped (6/12 received)
mininet>
```

Gambar 9. Hasil dari *Flow Berdasarkan MAC Address Topologi 1*

Peneliti juga melakukan beberapa pengujian lainnya, dan hasilnya dapat di lihat pada tabel berikut:

Tabel 5. Pengujian Berdasarkan *MAC Address Topologi 1*

Rule No	SRC MAC	DST MAC	SRC IP	DST IP	PROTOCOL	DST PORT	ACTION	RESULT
1	00:00:00:00:00:01	00:00:00:00:00:04	10.0.0.1	10.0.0.4	ANY	ANY	DROP	SUCCESS
2	00:00:00:00:00:02	00:00:00:00:00:05	10.0.0.2	10.0.0.5	ANY	ANY	DROP	SUCCESS
3	00:00:00:00:00:04	00:00:00:00:00:02	10.0.0.4	10.0.0.2	ANY	ANY	DROP	SUCCESS
4	00:00:00:00:00:05	ANY	10.0.0.5	ANY	ANY	ANY	DROP	SUCCESS

E. *Blocking MAC Address Topologi 2 (Tree)*

Selanjutnya *blocking* aliran data berdasarkan *MAC Address* Topologi 2. Dan aturannya adalah menambahkan aturan *flow firewall* pada *openflow:5* (s5) untuk *blocking* ke h8, sehingga nantinya semua aliran data yang menuju/ dari h8 tidak bisa berkomunikasi dengan h8, kecuali h7, dengan menggunakan perintah *PUT* pada *address bar*, dan alamat yang ditujukan adalah sebagai berikut.

```
http://localhost:8181/restconf/config/opendayl
ight-
inventory:nodes/node/openflow:5/table/0/flow/1
```

Berikut teks *XML* yang perlu ditambahkan menggunakan *POSTMAN*.

```
<flow xmlns="urn:opendaylight:flow:inventory">
    <table_id>0</table_id>
    <id>1</id>
    <cookie_mask>255</cookie_mask>
    <installHw>false</installHw>
    <match>
        <ethernet-match>
            <ethernet-source>

<address>00:00:00:00:00:08</address>
            </ethernet-source>
        </ethernet-match>
    </match>
    <priority>15</priority>
</flow>
```

Hasilnya sebagai berikut.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 X
h2 -> h1 h3 h4 h5 h6 h7 X
h3 -> h1 h2 h4 h5 h6 h7 X
h4 -> h1 h2 h3 h5 h6 h7 X
h5 -> h1 h2 h3 h4 h6 h7 X
h6 -> h1 h2 h3 h4 h5 h7 X
h7 -> h1 h2 h3 h4 h5 h6 h8
h8 -> X X X X X X h7
*** Results: 21% dropped (44/56 received)
mininet>
```

Gambar 10. Hasil dari *Flow Berdasarkan MAC Address Topologi 2*

Peneliti juga melakukan beberapa pengujian lainnya, dan hasilnya dapat di lihat pada tabel berikut.

Tabel 6. Pengujian Berdasarkan *MAC Address Topologi 2*

Rule No	SRC MAC	DST MAC	SRC IP	DST IP	PROTOCOL	DST PORT	ACTION	RESULT
1	00:00:00:00:00:01	ANY	10.0.0.1	ANY	ICMP	ANY	DROP	SUCCESS
2	00:00:00:00:00:02	00:00:00:00:00:01	10.0.0.2	10.0.0.1	ICMP	ANY	DROP	SUCCESS
3	00:00:00:00:00:03	00:00:00:00:00:04	10.0.0.3	10.0.0.4	ICMP	ANY	DROP	SUCCESS
4	00:00:00:00:00:05	00:00:00:00:00:01	10.0.0.5	10.0.0.1	ICMP	ANY	DROP	SUCCESS

F. *Blocking MAC Address Topologi 3 (Linear)*

Selanjutnya *blocking* aliran data berdasarkan *MAC Address* Topologi 3. Dan aturan yang akan di tambahkan adalah semua aliran data yang menuju/ dari h5 akan di blok, dengan menggunakan perintah *PUT* pada *address bar*, dan alamat yang ditujukan adalahsebagai berikut.

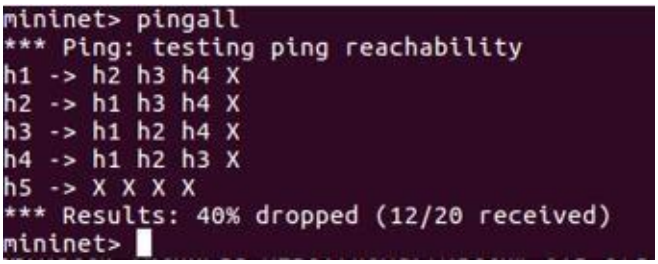
```
http://localhost:8181/restconf/config/opendayl
ight-
inventory:nodes/node/openflow:5/table/0/flow/1
```

Berikut text XML yang perlu ditambahkan menggunakan *POSTMAN*.

```
<flow xmlns="urn:opendaylight:flow:inventory">
    <table_id>0</table_id>
    <id>1</id>
    <cookie_mask>255</cookie_mask>
    <installHw>false</installHw>
    <match>
        <ethernet-match>
            <ethernet-source>

<address>00:00:00:00:00:05</address>
            </ethernet-source>
        </ethernet-match>
    </match>
    <priority>15</priority>
</flow>
```

Hasilnya sebagai berikut.



Gambar 11. Hasil dari *Flow Berdasarkan MAC Address Topologi 3*

Peneliti juga melakukan beberapa pengujian lainnya, dan hasilnya dapat dilihat pada tabel berikut.

Tabel 7. Pengujian Berdasarkan *MAC Address Topologi 3*

Rule No	SRC MAC	DST MAC	SRC IP	DST IP	PROTOCOL	DST PORT	ACTION	RESULT
1	00:00:00:00:00:01	00:00:00:00:00:04	10.0.0.1	10.0.0.4	ANY	ANY	DROP	SUCCESS
2	00:00:00:00:00:02	00:00:00:00:00:05	10.0.0.2	10.0.0.5	ANY	ANY	DROP	SUCCESS
3	00:00:00:00:00:04	00:00:00:00:00:02	10.0.0.4	10.0.0.2	ANY	ANY	DROP	SUCCESS
4	00:00:00:00:00:05	ANY	10.0.0.5	ANY	ANY	ANY	DROP	SUCCESS

G. *Blocking UDP Address Topologi 1 (Single)*

Selanjutnya *blocking* aliran data berdasarkan *Port UDP* Topologi 1. Dan aturan yang akan di tambahkan adalah menutup komunikasi yang akan dilakukan oleh h2 pada *UDP* (Port 17), dengan menggunakan perintah *PUT* pada *address bar*, dan alamat yang ditujukan adalah sebagai berikut.

```
http://localhost:8181/restconf/config/opendayl
ight-
inventory:nodes/node/openflow:1/table/0/flow/1
```

Berikut teks *XML* yang perlu ditambahkan menggunakan *POSTMAN.*

```
<flow xmlns="urn:opendaylight:flow:inventory">
    <table_id>0</table_id>
    <id>1</id>
    <cookie_mask>255</cookie_mask>
    <match>
       <ethernet-match>
                        <ethernet-type>
<type>2048</type>
                        </ethernet-type>
                </ethernet-match>
                <ip-match>
             <ip-protocol>17</ip-protocol>
         </ip-match>
         <ipv4-destination>10.0.0.2/32</ipv4-
destination>
         <in-port>1</in-port>
        </match>
        <priority>50</priority>
        <hard-timeout>1200</hard-timeout>
    <cookie>8</cookie>
    <idle-timeout>3400</idle-timeout>
</flow>
```

Hasilnya sebagai berikut.

Gambar 12. Hasil dari *Flow Berdasarkan UDP Address Topologi 1*

Peneliti juga melakukan beberapa pengujianlainnya, dan hasilnya dapat di lihat pada tabel berikut.

Tabel 8. Pengujian Berdasarkan *UDP Address Topologi 1*

Rule No	SRC MAC	DST MAC	SRC IP	DST IP	PROTOCOL	DST PORT	ACTION	RESULT
1	00:00:00:00:00:01	00:00:00:00:00:04	10.0.0.1	10.0.0.4	UDP	ANY	DROP	SUCCESS
2	00:00:00:00:00:02	00:00:00:00:00:04	10.0.0.2	10.0.0.4	UDP	ANY	DROP	SUCCESS
3	00:00:00:00:00:04	00:00:00:00:00:01	10.0.0.4	10.0.0.1	UDP	ANY	DROP	SUCCESS
4	00:00:00:00:00:02	00:00:00:00:00:03	10.0.0.2	10.0.0.3	UDP	ANY	DROP	SUCCESS

H. *Blocking UDP Address Topologi 2 (Tree)*

Selanjutnya *blocking* aliran data berdasarkan *Port UDP* Topologi 2. Dan aturan yang akan di tambahkan adalah menutup komunikasi yang akan dilakukan oleh h2 pada *UDP* (Port 17), dengan menggunakan perintah *PUT* pada *address bar*, dan alamat yang ditujukan adalah sebagai berikut:

```
http://localhost:8181/restconf/config/opendayl
ight-
inventory:nodes/node/openflow:1/table/0/flow/1
```

Berikut teks *XML* yang perlu ditambahkan menggunakan *POSTMAN.*

```
<flow xmlns="urn:opendaylight:flow:inventory">
    <table_id>0</table_id>
    <id>1</id>
    <cookie_mask>255</cookie_mask>
    <match>
       <ethernet-match>
                        <ethernet-type>
<type>2048</type>
                        </ethernet-type>
                </ethernet-match>
                <ip-match>
             <ip-protocol>17</ip-protocol>
        </ip-match>
        <ipv4-destination>10.0.0.1/32</ipv4-
destination>
        <in-port>2</in-port>
       </match>
       <priority>50</priority>
       <hard-timeout>1200</hard-timeout>
    <cookie>8</cookie>
    <idle-timeout>3400</idle-timeout>
</flow>
```

Saat coba untuk mengirim pesan menggunakan *port UDP* ke h1, maka hanya h2, h3, dan h4 saja yang bisa berkomunikasi dengan h1. Karena *port* 2 pada s1 telah ditambahkan *rule* baru, sehingga h5, h6, h7, dan h8 tidak bisa berkomunikasi menggunakan protokol *UDP*.

Peneliti juga melakukan beberapa pengujian lainnya, dan hasilnya dapat di lihat pada tabel berikut.

Tabel 9. Pengujian Berdasarkan *UDP Address Topologi 2*

Rule No	SRC MAC	DST MAC	SRC IP	DST IP	PROTOCOL	DST PORT	ACTION	RESULT
1	00:00:00:00:00:05	00:00:00:00:00:01	10.0.0.5	10.0.0.1	UDP	ANY	DROP	SUCCESS
2	00:00:00:00:00:01	00:00:00:00:00:04	10.0.0.1	10.0.0.4	UDP	ANY	DROP	SUCCESS
3	00:00:00:00:00:07	00:00:00:00:00:08	10.0.0.7	10.0.0.8	UDP	ANY	DROP	SUCCESS
4	00:00:00:00:00:04	00:00:00:00:00:07	10.0.0.4	10.0.0.7	UDP	ANY	DROP	SUCCESS

I. *Blocking UDP Address Topologi 3 (Linear)*

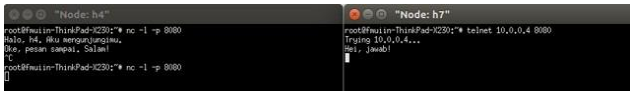
Selanjutnya *blocking* aliran data berdasarkan *Port TCP* Topologi 2. Aturan yang akan di tambahkan adalah menutup jalur komunikasi untuk h4 pada *port* 2 di s1, dengan menggunakan perintah *PUT* pada *address bar*, dan alamat yang ditujukan adalah sebagai berikut.

```
http://localhost:8181/restconf/config/opendayl
ight-
inventory:nodes/node/openflow:1/table/0/flow/1
```

Berikut teks *XML* yang perlu ditambahkan menggunakan *POSTMAN*.

```
<flow xmlns="urn:opendaylight:flow:inventory">
    <table_id>0</table_id>
    <id>1</id>
    <cookie_mask>255</cookie_mask>
    <match>
       <ethernet-match>
                        <ethernet-type>
<type>2048</type>
                        </ethernet-type>
                </ethernet-match>
                <ip-match>
             <ip-protocol>6</ip-protocol>
        </ip-match>
        <ipv4-destination>10.0.0.4/32</ipv4-
destination>
        <in-port>2</in-port>
       </match>
       <priority>20</priority>
       <hard-timeout>1200</hard-timeout>
    <cookie>8</cookie>
    <idle-timeout>3400</idle-timeout>
</flow>
```

Hasilnya sebagai berikut.



Gambar 13. Hasil dari *Flow Berdasarkan UDP Address Topologi 3*

Peneliti juga melakukan beberapa pengujian lainnya, dan hasilnya dapat di lihat pada tabel berikut.

Tabel 10. Pengujian Berdasarkan *UDP Address Topologi 3*

Rule No	SRC MAC	DST MAC	SRC IP	DST IP	PROTOCOL	DST PORT	ACTION	RESULT
1	00:00:00:00:00:05	00:00:00:00:00:02	10.0.0.5	10.0.0.2	UDP	ANY	DROP	SUCCESS
2	00:00:00:00:00:02	00:00:00:00:00:01	10.0.0.2	10.0.0.1	UDP	ANY	DROP	SUCCESS
3	00:00:00:00:00:05	00:00:00:00:00:01	10.0.0.5	10.0.0.1	UDP	ANY	DROP	SUCCESS
4	00:00:00:00:00:03	00:00:00:00:00:01	10.0.0.3	10.0.0.1	UDP	ANY	DROP	SUCCESS

J. *Blocking TCP Address Topologi 1 (Single)*

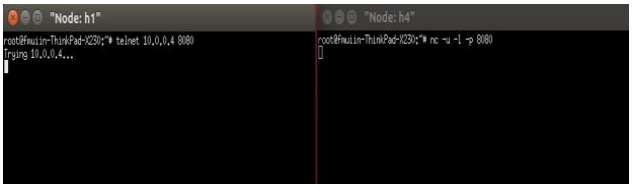
Selanjutnya *blocking* aliran data berdasarkan *Port TCP* Topologi 1. Dan aturan yang akan di tambahkan adalah menutup komunikasi yang akan dilakukan oleh h2 pada *TCP* (*Port* 6), dengan menggunakan perintah *PUT* pada *address bar*, dan alamat yang ditujukan adalah sebagai berikut.

```
http://localhost:8181/restconf/config/opendayl
ight-
inventory:nodes/node/openflow:1/table/0/flow/1
```

Berikut teks *XML* yang perlu ditambahkan menggunakan *POSTMAN*.

```
<flow xmlns="urn:opendaylight:flow:inventory">
    <table_id>0</table_id>
    <id>1</id>
    <cookie_mask>255</cookie_mask>
    <match>
       <ethernet-match>
                        <ethernet-type>
<type>2048</type>
                        </ethernet-type>
                </ethernet-match>
                <ip-match>
             <ip-protocol>6</ip-protocol>
        </ip-match>
        <ipv4-destination>10.0.0.4/32</ipv4-
destination>
        <in-port>1</in-port>
       </match>
       <priority>20</priority>
       <hard-timeout>1200</hard-timeout>
    <cookie>8</cookie>
    <idle-timeout>3400</idle-timeout>
</flow>
```

Hasilnya sebagai berikut.



Gambar 14. Hasil dari *Flow Berdasarkan TCP Address Topologi 1*

Peneliti juga melakukan beberapa pengujian lainnya, dan hasilnya dapat di lihat pada tabel berikut.

Tabel 11. Pengujian Berdasarkan *TCP Address Topologi 1*

Rule No	SRC MAC	DST MAC	SRC IP	DST IP	PROTOCOL	DST PORT	ACTION	RESULT
1	00:00:00:00:00:01	00:00:00:00:00:04	10.0.0.1	10.0.0.4	TCP	ANY	DROP	SUCCESS
2	00:00:00:00:00:02	00:00:00:00:00:04	10.0.0.2	10.0.0.4	TCP	ANY	DROP	SUCCESS
3	00:00:00:00:00:04	00:00:00:00:00:02	10.0.0.4	10.0.0.2	TCP	ANY	DROP	SUCCESS
4	00:00:00:00:00:02	00:00:00:00:00:03	10.0.0.2	10.0.0.3	TCP	ANY	DROP	SUCCESS

K. *Blocking TCP Address Topologi 2 (Tree)*

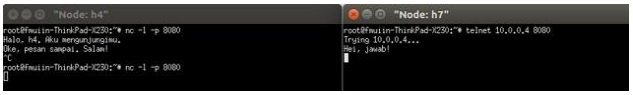
Selanjutnya *blocking* aliran data berdasarkan *Port TCP* Topologi 2. Aturan yang akan di tambahkan adalah menutup jalur komunikasi untuk h4 pada *port* 2 di s1, dengan menggunakan perintah *PUT* pada *address bar*, dan alamat yang ditujukan adalah sebagai berikut.

```
http://localhost:8181/restconf/config/opendayl
ight-
inventory:nodes/node/openflow:1/table/0/flow/1
```

Berikut teks *XML* yang perlu ditambahkan menggunakan *POSTMAN*.

```
<flow xmlns="urn:opendaylight:flow:inventory">
    <table_id>0</table_id>
    <id>1</id>
    <cookie_mask>255</cookie_mask>
    <match>
       <ethernet-match>
                        <ethernet-type>
<type>2048</type>
                        </ethernet-type>
                </ethernet-match>
                <ip-match>
             <ip-protocol>6</ip-protocol>
        </ip-match>
        <ipv4-destination>10.0.0.4/32</ipv4-
destination>
        <in-port>2</in-port>
       </match>
       <priority>20</priority>
       <hard-timeout>1200</hard-timeout>
    <cookie>8</cookie>
    <idle-timeout>3400</idle-timeout>
</flow>
```

Hasilnya sebagai berikut.



Gambar 15. Hasil dari *Flow Berdasarkan TCP Address Topologi 2*

Peneliti juga melakukan beberapa pengujian lainnya, dan hasilnya dapat di lihat pada tabel berikut.

Tabel 12. Pengujian Berdasarkan *TCP Address Topologi 2*

Rule No	SRC MAC	DST MAC	SRC IP	DST IP	PROTOCOL	DST PORT	ACTION	RESULT
1	00:00:00:00:00:01	00:00:00:00:00:02	10.0.0.1	10.0.0.2	TCP	ANY	DROP	SUCCESS
2	00:00:00:00:00:02	00:00:00:00:00:07	10.0.0.2	10.0.0.7	TCP	ANY	DROP	SUCCESS
3	00:00:00:00:00:04	00:00:00:00:00:08	10.0.0.4	10.0.0.8	TCP	ANY	DROP	SUCCESS
4	00:00:00:00:00:02	00:00:00:00:00:05	10.0.0.2	10.0.0.5	TCP	ANY	DROP	SUCCESS

L. *Blocking TCP Address Topologi 3 (Linear)*

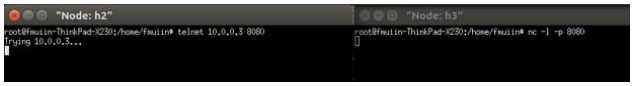
Selanjutnya *blocking* aliran data berdasarkan *Port TCP* Topologi 3. Dan aturan yang akan di tambahkan adalah menutup komunikasi TCP pada h3 dengan *destination port* 2 pada s3, sehingga nantinya h4 dan h5 masih bisa komunikasi dengan h3, dengan menggunakan perintah *PUT* pada *address bar*, dan alamat yang ditujukan adalah sebagai berikut.

```
http://localhost:8181/restconf/config/opendayl
ight-
inventory:nodes/node/openflow:3/table/0/flow/1
```

Berikut teks *XML* yang perlu ditambahkan menggunakan *POSTMAN*.

```
<flow xmlns="urn:opendaylight:flow:inventory">
    <table_id>0</table_id>
    <id>1</id>
    <cookie_mask>255</cookie_mask>
    <match>
       <ethernet-match>
                        <ethernet-type>
<type>2048</type>
                        </ethernet-type>
                </ethernet-match>
                <ip-match>
             <ip-protocol>6</ip-protocol>
         </ip-match>
         <ipv4-destination>10.0.0.3/32</ipv4-
destination>
         <in-port>2</in-port>
        </match>
        <priority>20</priority>
        <hard-timeout>1200</hard-timeout>
    <cookie>8</cookie>
    <idle-timeout>3400</idle-timeout>
</flow>
```

Hasilnya sebagai berikut.



Gambar 16. Hasil dari *Flow Berdasarkan TCP Address Topologi 3*

Peneliti juga melakukan beberapa pengujian lainnya, dan hasilnya dapat di lihat pada tabel berikut.

Tabel 13. Pengujian Berdasarkan *TCP Address Topologi 3*

Rule No	SRC MAC	DST MAC	SRC IP	DST IP	PROTOCOL	DST PORT	ACTION	RESULT
1	00:00:00:00:00:01	00:00:00:00:00:02	10.0.0.1	10.0.0.2	TCP	ANY	DROP	SUCCESS
2	00:00:00:00:00:03	00:00:00:00:00:05	10.0.0.3	10.0.0.5	TCP	ANY	DROP	SUCCESS
3	00:00:00:00:00:05	00:00:00:00:00:02	10.0.0.5	10.0.0.2	TCP	ANY	DROP	SUCCESS
4	00:00:00:00:00:02	00:00:00:00:00:05	10.0.0.2	10.0.0.5	TCP	ANY	DROP	SUCCESS

M. *Blocking ICMP Address Topologi 1 (Single)*

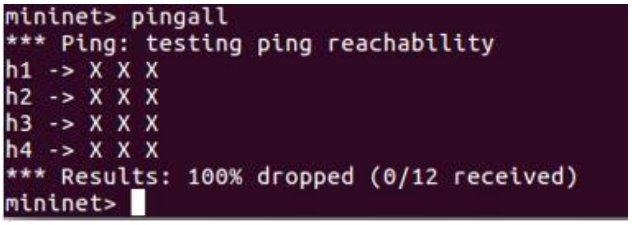
Selanjutnya *blocking* aliran data berdasarkan *ICMP* pada Topologi 1. Dan aturan yang akan di tambahkan adalah menutup semua komunikasi yangakan dilakukan, dengan menggunakan perintah *PUT*pada *address bar*, dan alamat yang ditujukan adalah sebagai berikut.

```
http://localhost:8181/restconf/config/opendayl
ight-
inventory:nodes/node/openflow:1/table/0/flow/1
```

Berikut teks *XML* yang perlu ditambahkan menggunakan *POSTMAN.*

```
<flow xmlns="urn:opendaylight:flow:inventory">
    <priority>11</priority>
    <flow-name>Dropl3</flow-name>
    <match>
        <ethernet-match>
            <ethernet-type>
                <type>2048</type>
            </ethernet-type>
        </ethernet-match>
        <ip-match>
            <ip-protocol>1</ip-protocol>
        </ip-match>
    </match>
    <id>1</id>
    <table_id>0</table_id>
    <instructions>
        <instruction>
            <order>0</order>
            <apply-actions>
                <action>
                  <order>0</order>
                  <drop-action/>
                </action>
            </apply-actions>
        </instruction>
    </instructions>
</flow>
```

Hasilnya sebagai berikut.



Gambar 17. Hasil dari *Flow Berdasarkan ICMP Address Topologi 1*

Peneliti juga melakukan beberapa pengujian lainnya, dan hasilnya dapat di lihat pada tabel berikut.

Tabel 14. Pengujian Berdasarkan *ICMP Address Topologi 1*

Rule No	SRC MAC	DST MAC	SRC IP	DST IP	PROTOCOL	DST PORT	ACTION	RESULT
1	00:00:00:00:00:01	ANY	10.0.0.1	ANY	ICMP	ANY	DROP	SUCCESS
2	00:00:00:00:00:02	ANY	10.0.0.2	ANY	ICMP	ANY	DROP	SUCCESS
3	00:00:00:00:00:03	ANY	10.0.0.3	ANY	ICMP	ANY	DROP	SUCCESS
4	00:00:00:00:00:04	ANY	10.0.0.4	ANY	ICMP	ANY	DROP	SUCCESS

N. *Blocking ICMP Address Topologi 2 (Tree)*

Selanjutnya *blocking* aliran data berdasarkan *ICMP* pada Topologi 2. Dan aturan yang akan di tambahkan adalah blok protokol *ICMP* pada s2,sehingga nantinya h1 h2 h3 h4 tidak bisa melakukan aliran data karena telah di *drop* saat melalui s2, dengan m*enggunakan perintah* PUT *pada* address bar, *dan alamat yang ditujukan adalah sebagai berikut.*

```
http://localhost:8181/restconf/config/opendayl
ight-
inventory:nodes/node/openflow:1/table/0/flow/1
```

Berikut teks *XML* yang perlu ditambahkan menggunakan *POSTMAN.*

```
            <flow
xmlns="urn:opendaylight:flow:inventory">
    <priority>25</priority>
    <flow-name>Dropl3</flow-name>
    <match>
        <ethernet-match>
            <ethernet-type>
                <type>2048</type>
            </ethernet-type>
        </ethernet-match>
        <ip-match>
            <ip-protocol>1</ip-protocol>
        </ip-match>
    </match>
    <id>1</id>
    <table_id>0</table_id>
    <instructions>
        <instruction>
            <order>0</order>
            <apply-actions>
                <action>
                  <order>0</order>
                  <drop-action/>
                </action>
            </apply-actions>
```

Hasilnya sebagai berikut.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X X X X X
h2 -> h1 X X X X X
h3 -> X X h4 X X X
h4 -> X X h3 X X X
h5 -> X X X X h6 h7 h8
h6 -> X X X X h5 h7 h8
h7 -> X X X X h5 h6 h8
h8 -> X X X X h5 h6 h7
*** Results: 71% dropped (16/56 received)
mininet>
```

Gambar 18. Hasil dari *Flow Berdasarkan ICMP Address Topologi 2*

Peneliti juga melakukan beberapa pengujian lainnya, dan hasilnya dapat di lihat pada tabel berikut.

Tabel 15. Pengujian Berdasarkan *ICMP Address Topologi 2*

Rule No	SRC MAC	DST MAC	SRC IP	DST IP	PROTOCOL	DST PORT	ACTION	RESULT
1	00:00:00:00:00:05	00:00:00:00:00:08	10.0.0.5	10.0.0.8	ICMP	ANY	DROP	SUCCESS
2	00:00:00:00:00:02	00:00:00:00:00:07	10.0.0.2	10.0.0.7	ICMP	ANY	DROP	SUCCESS
3	00:00:00:00:00:03	00:00:00:00:00:05	10.0.0.3	10.0.0.5	ICMP	ANY	DROP	SUCCESS
4	00:00:00:00:00:07	00:00:00:00:00:01	10.0.0.7	10.0.0.1	ICMP	ANY	DROP	SUCCESS

O. *Blocking ICMP Address Topologi 3 (Linear)*

Selanjutnya *blocking* aliran data berdasarkan *ICMP* pada Topologi 3. Dan aturan yang akan di tambahkan adalah akan blok protokol ICMP pada s1, sehingga semua aliran menuju h1 akan di *drop*, denganmenggunakan perintah *PUT* pada *address bar*, dan alamat yang ditujukan adalah sebagai berikut.

```
http://localhost:8181/restconf/config/opendayl
ight-
inventory:nodes/node/openflow:1/table/0/flow/1
```

Berikut teks *XML* yang perlu ditambahkan menggunakan *POSTMAN*.

```
<flow xmlns="urn:opendaylight:flow:inventory">
    <priority>25</priority>
    <flow-name>Dropl3</flow-name>
    <match>
        <ethernet-match>
            <ethernet-type>
                <type>2048</type>
            </ethernet-type>
        </ethernet-match>
        <ip-match>
            <ip-protocol>1</ip-protocol>
        </ip-match>
    </match>
    <id>1</id>
    <table_id>0</table_id>
    <instructions>
        <instruction>
            <order>0</order>
            <apply-actions>
                <action>
                  <order>0</order>
                  <drop-action/>
                </action>
            </apply-actions>
        </instruction>
    </instructions>
</flow>
```

Hasilnya sebagai berikut.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X X
h2 -> X h3 h4 h5
h3 -> X h2 h4 h5
h4 -> X h2 h3 h5
h5 -> X h2 h3 h4
*** Results: 40% dropped (12/20 received)
mininet>
```

Gambar 19. Hasil dari *Flow Berdasarkan ICMP Address Topologi 3*

Peneliti juga melakukan beberapa pengujian lainnya, dan hasilnya dapat di lihat pada tabel berikut.

Tabel 15. Pengujian Berdasarkan *ICMP Address Topologi 3*

Rule No	SRC MAC	DST MAC	SRC IP	DST IP	PROTOCOL	DST PORT	ACTION	RESULT
1	00:00:00:00:00:01	ANY	10.0.0.1	ANY	ICMP	ANY	DROP	SUCCESS
2	00:00:00:00:00:02	00:00:00:00:00:01	10.0.0.2	10.0.0.1	ICMP	ANY	DROP	SUCCESS
3	00:00:00:00:00:03	00:00:00:00:00:04	10.0.0.3	10.0.0.4	ICMP	ANY	DROP	SUCCESS
4	00:00:00:00:00:05	00:00:00:00:00:01	10.0.0.5	10.0.0.1	ICMP	ANY	DROP	SUCCESS

5. KESIMPULAN

5.1 Kesimpulan

Berdasarkan hasil implementasi dan analisis dari SDN berbasis *openflow* menggunakan *controller opendaylight* sebagai *controller*, maka dapat disimpulkan sebagai berikut:

1. Teknik yang digunakan dalam penerapan*firewall* pada arsitektur *SDN* adalah dengan cara menambahkan aturan *rule* baru pada *flowtable* yang ada di *switch* dan nantinya akan di sesuaikan dengan kebutuhan aliran data yang di inginkan. Dan teknologi yang digunakan adalah dengan menggunakan bahasa XML dan di terapkan pada *switch* virtual pada *mininet*

menggunakan *postman*, dan menerapkan kontrol *drop* (karena *default*nya *accept*).

2. Implementasi *firewall* pada *Software Defined Network* menggunakan *opendaylight* telah berhasil dilakukan karena semua aliran *flow* yang ingin di *drop* telah terlaksana denganbaik. Penerapan akses kontrol dalam mengatur *flow table firewall* pada penelitian ini menggunakan *IP Address, MAC Address, TCP, UDP*, dan *ICMP*. *Rule firewall* di *push* menggunakan *postman.*

5.2 Saran

Berikut adalah beberapa saran untuk kelanjutan penelitian yang dapat dilakukan, diantaranya:

1. Diuji lebih lanjut dengan topologi yang berbeda dengan jaringan yang semakin kompleks.
2. Melakukan penelitian efektivitas *firewall* dengan berbasis teknologi virtualisasi atau dalam lingkungan nyata dan bukan sekedar simulator.
3. Melakukan pengujian dengan metode *push rule firewall* dengan cara yang berbeda dan penambahan *rule* dengan menggunakan teknologi *web service.*
4. Diharapkan penelitian selanjutnya bisa untuk menciptakan antarmuka pengaturan *flow* yang berbasis *web.*

DAFTAR PUSTAKA

[1] B. A. Linuwih, A. V, "Perancangan dan Analisis *Software Defined Network* pada Jaringan LAN: Penerapan dan Analisis Metode Penjaluran *Path Calculating* Menggunakan Algoritma Dijkstra," 2016.

[2] A. R. Sudiyatmoko, S. N. "Analisis Performansi Perutingan *Link State* Menggunakan Algoritma Djikstra pada Platform *Software Defined Network* (SDN)," 2016.

[3] M. Syafrizal, "Pengantar Jaringan Komputer," Yogyakarta, 2005.

[4] J. H. Green, "*Local Area Network A User's Guide For Business Professionals*," London: Scott, Foresman and Company, 1985.

[5] M. Betts, Z. "*Open Networking Foundation*". SDN Architecture, 2014.

[6] A. Kaur, V. S. ,"*Building L2-L4 Firewall using Software Defined Networking*," 2017.

[7] R. Kartadie, E. U., "Prototipe *Infrastruktur Software Defined Network* dengan Protokol Openflow Menggunakan Ubuntu Sebagai Kontroller," 2014.

[8] R. Kartadie, B. S., "Uji Performa Kontroller Floodlight dan Opendaylight Sebagai Komponen Utama Arsitektur *Software Defined Network*," 2015.

[9] R. Tulloh, R. M., "Simulasi Virtual Local Area Network (VLAN) Berbasis *Software Defined Network* (SDN) Menggunakan POX Controller," 2015.

[10] Noor, Juliansyah, "Metodologi Penelitian: Skripsi, Tesis, Disertasi, dan Karya Ilmiah." Jakarta: Kencana, 2017.