



UNIVERSITAS ESA UNGGUL

**DETEKSI DAN MITIGASI DOS PADA *SOFTWARE DEFINED*
NETWORK DENGAN MENGGUNAKAN OPENFLOW DAN
SFLOW**

TUGAS AKHIR

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana

NAMA : RANDY MUKTI

NIM : 20180801243

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS ESA UNGGUL
TAHUN 2021**

HALAMAN PERNYATAAN KEASLIAN

Tugas Akhir ini adalah hasil karya saya sendiri, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar.

Nama : Randy Mukti
NIM : 20180801243
Tanda Tangan :

Tanggal : 21 Agustus 2021

HALAMAN PENGESAHAN TUGAS AKHIR

Tugas Akhir ini diajukan oleh

Nama : Randy Mukti
NIM : 20180801243
Program Studi : Teknik Informatika
Fakultas Ilmu Komputer - Universitas Esa Unggul
Judul Tugas Akhir : Deteksi dan Mitigasi DOS pada *Software Defined Network*
dengan menggunakan Openflow dan sFlow

Telah berhasil dipertahankan di hadapan Tim Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Komputer pada Program Studi Teknik Informatika Fakultas Ilmu Komputer, Universitas Esa Unggul.

TIM PENGUJI

Pembimbing : Habibullah Akbar, S.Si., M.Sc., Ph.D
Penguji I : Harry Kurniawan, S.T, M.T
Penguji II : Agung Mulyo Widodo, ST, M.Sc

(*Habibullah Akbar*)
(*Harry Kurniawan*)
(*Agung Mulyo Widodo*)

Ditetapkan di : Jakarta
Ketua Program Studi : M. Bahrul Ulum, S.Kom, M.Kom
Tanggal : 21 Agustus 2021

(*M. Bahrul Ulum*)



HALAMAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademika Universitas Esa Unggul, saya yang bertanda tangan di bawah ini:

Nama : Randy Mukti
NIM : 20180801243
Program Studi : Teknik Informatika
Fakultas : Ilmu Komputer
Jenis Karya Ilmiah : Tugas Akhir

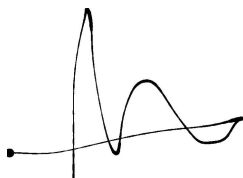
demi pengembangan ilmu pengetahuan, teknologi, dan seni, menyetujui untuk memberikan kepada Universitas Esa Unggul Hak Bebas Royalti Noneksklusif atas karya ilmiah saya yang berjudul:

Deteksi dan Mitigasi DOS pada *Software Defined Network* dengan menggunakan Openflow dan sFlow.

beserta perangkat yang ada (apabila diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini, Universitas Esa Unggul berhak menyimpan, mengalihmediakan, mengelola dalam bentuk pangkalan data, merawat, dan mempublikasikan Tugas Akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Jakarta
Pada tanggal : 21 Agustus 2021
Yang menyatakan



(Randy Mukti)

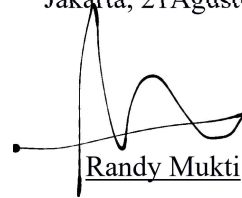
KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadiran Allah SWT, karena telah memberikan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan tugas akhir yang berjudul ” Deteksi dan Mitigasi DOS pada *Software Defined Network* dengan menggunakan Openflow dan sFlow“. Dalam kesempatan ini penulis juga mengucapkan terima kasih kepada berbagai pihak yang telah memberikan dukungan, bimbingan, dan kerja samanya dalam menyusun Tugas Akhir ini, karena tanpa dukungan mereka penyusunan Tugas Akhir ini tidak akan tercapai. Karena itu penulis mengucapkan terima kasih kepada :

1. Bapak Habibullah Akbar, S.Si, M.Sc, Ph.D selaku Dekan Fakultas Ilmu Komputer dan Dosen Pembimbing Materi yang telah meluangkan banyak waktunya untuk memberikan bimbingan dan arahan kepada penulis dalam menyelesaikan Laporan Tugas Akhir.
2. Bapak M. Bahrul Ulum, S.Kom, M.Kom selaku Ka. Prodi Teknik Informatika Fakultas Ilmu Komputer.
3. Kepada teman-teman Keluarga Besar Fakultas Ilmu Komputer seluruh angkatan, khususnya Angkatan 2018. Terima kasih atas dukungan dan bantuannya.
4. Kepada saya sendiri yang tidak menyerah sampai saat ini.

Penulis hanyalah manusia biasa yang tidak luput dari kesalahan, begitu juga Tugas Akhir ini, untuk itu penulis sangat berharap adanya respon dari pembaca dalam bentuk saran ataupun kritik yang dapat membantu penulis untuk lebih baik lagi kedepannya. Akhir kata, semoga Tugas Akhir ini dapat memberikan manfaat dan menambah wawasan pengetahuan bagi berbagai pihak yang membaca Tugas Akhir ini.

Jakarta, 21 Agustus 2021

A handwritten signature in black ink, consisting of a series of loops and curves, positioned above the printed name.

Randy Mukti

NIM. 20180801243

ABSTRAK

Judul : Deteksi dan Mitigasi DOS pada *Software Defined Network* dengan menggunakan Openflow dan sFlow

Nama : Randy Mukti

Program Studi : Teknik Informatika

Software Defined Network (SDN) adalah arsitektur jaringan yang memisahkan control plane dan data plane. Fokus penelitian ini membahas tentang penerapan sFlow dan OpenFlow pada *Software Defined Network* untuk mendeteksi dan melakukan pencegahan serta packet capture terhadap packet besar atau DOS yang masuk ke dalam jaringan SDN. sFlow dan OpenFlow diimplementasikan pada sistem operasi Ubuntu 20.04, mininet sebagai virtual node, ONOS sebagai *OpenFlow Controller* dan network topology, sFlow-RT sebagai monitor route atau traffic dalam network, NodeJS dan python sebagai bahasa pemrograman yang digunakan untuk aplikasi SDN ini. Skenario pengujian dilakukan dengan cara switch mengirim sample packet dan mengirimkan header dari sample packet tersebut ke sFlow-RT. sFlow-RT kemudian memetakan packet yang sudah diterima kedalam bahasa atau flow yang lebih terstruktur. Jika paket sudah melewati batas *threshold* yang sudah ditentukan kemudian akan mentrigger event. Event tersebut kemudian yang akan dapat diakses dari aplikasi external melalui REST-API. Adapun hasil dari simulasi ini menunjukkan bahwa ONOS dan Mininet dapat digunakan untuk pengembangan dan pengujian serangan DOS terhadap jaringan yang mengadopsi paradigma SDN.

Kata kunci : SDN, Deteksi dan Mitigasi DOS, OpenFlow

Title : *DOS Attack Detection and Mitigation with OpenFlow*

Name : Randy Mukti

Study Program : *Information Technology*

ABSTRACT

A Software Defined Network (SDN) is a network architecture that separates the control plane and data plane. The focus of this research discusses the

application of sFlow and OpenFlow in the Software Defined Network to detect and prevent as well as packet capture of large packets or DOS that enter the SDN network. sFlow and OpenFlow are implemented on the Ubuntu 20.04 operating system, Mininet as a virtual node, ONOS as an OpenFlow Controller and network topology, sFlow-RT as a route or traffic monitor in the network, NodeJS and Python as programming languages used for this SDN application. The test scenario is done by means of the switch sending sample packets and sending the header of the sample packet to sFlow-RT. sFlow-RT then maps the received packets to a more structured language or flow. If the packet has passed the predetermined threshold then it will trigger an event. The event will then be accessible from external applications via the REST-API. In the result from following simulation showing that ONOS and Mininet able to be used for development and testing DOS attack for SDN Networks.

Keywords: *SDN, DOS Detection and Mitigation, OpenFlow.*

DAFTAR ISI

Halaman

HALAMAN PERNYATAAN KEASLIAN.....	i
HALAMAN PENGESAHAN TUGAS AKHIR	ii
HALAMAN PERSETUJUAN PUBLIKASI KARYA ILMIAH	iii
KATA PENGANTAR.....	ii
ABSTRAK	iv
DAFTAR ISI.....	vi
DAFTAR TABEL	viii
DAFTAR GAMBAR.....	ix
DAFTAR LAMPIRAN	x
BAB 1 PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah.....	2
1.3 Tujuan Tugas Akhir.....	2
1.4 Manfaat Tugas Akhir.....	3
1.5 Lingkup Tugas Akhir.....	3
1.6 Metodologi Penelitian.....	4
1.7 Sistematika Penulisan	5
BAB 2 TINJAUAN PUSTAKA	7
2.1 <i>Software Defined Network</i>	7
2.2 Protocol OpenFlow	9
2.3 Mininet.....	11
2.4 sFlow	11
2.4.1 Agent sFlow.....	12

2.5	IPERF	13
2.6	DOS	13
2.7	Penelitian Sebelumnya	14
BAB 3 METODE PENELITIAN		18
3.1	Pendahuluan.....	18
3.2	Kerangka Simulasi.....	19
3.3	Peralatan Yang Diperlukan.....	21
3.4	Perancangan.....	21
3.5	Penentuan Skenario Pengujian	27
BAB 4 HASIL DAN PEMBAHASAN		31
4.1	Kebutuhan Simulasi.....	31
4.2	Topologi Jaringan	31
4.3	Construction.....	32
4.4	Hasil simulasi serangan DNS	37
BAB 5 KESIMPULAN DAN SARAN		40
5.1	Kesimpulan	40
5.2	Saran	40
DAFTAR REFERENSI		41

DAFTAR TABEL

Tabel 1 Informasi Perangkat Keras.....	23
--	----

DAFTAR GAMBAR

Gambar 1-1 Metodologi Penelitian.....	4
Gambar 2-1 Arsitektur SDN	8
Gambar 2-2 Arsitektur ONOS <i>Controller</i>	10
Gambar 2-3 Mininet Program	11
Gambar 2-4 Aplikasi sFlow	12
Gambar 2-5 Fungsi sFlow	13
Gambar 3-1 Tahapan Simulasi.....	19
Gambar 3-2 Kerangka Simulasi	20
Gambar 3-3 Topologi Jaringan Mininet.....	22
Gambar 3-4 Integrasi ONOS dan Mininet	24
Gambar 3-5 Integrasi sFlow	25
Gambar 3-6 Flowchart Aplikasi.....	26
Gambar 3-7 Aplikasi sFlow dan OpenFlow	28
Gambar 3-8 Flow Script.....	29
Gambar 4-1 Topologi Jaringan	32
Gambar 4-2 sFlow-RT Mininet Dashboard	34
Gambar 4-3 Topologi Jaringan ONOS-SDN	35
Gambar 4-4 Daftar Perangkat pada ONOS-SDN	35
Gambar 4-5 Pengujian Traffic	36
Gambar 4-6 <i>Traffic Flow Peace Time</i>	36
Gambar 4-7 <i>Traffic</i> Serangan DNS tanpa script	37
Gambar 4-8 <i>Traffic</i> Serangan DNS dengan script	38
Gambar 4-9 <i>Traffic Flow War Time</i>	39

DAFTAR LAMPIRAN

Lampiran 1. Kode Program – Deteksi dan Mitigasi DoS	43
Lampiran 2. Daftar Riwayat Hidup.....	46

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Jaringan tradisional adalah perangkat jaringan yang memiliki *control plane* dan *data plane* didalam perangkat yang sama. Sedangkan *Software Defined Networks* merupakan arsitektur yang memisahkan antara control plane dan data plane, serta kemudian melakukan abstraksi sistem dan meng-isolasi kompleksitas yang ada pada komponen atau sub-sistem dengan mendefinisikan antar-muka (*interfaces*) yang standard. Dengan menerapkan arsitektur SDN bertujuan untuk membuat jaringan menjadi lebih fleksibel dan mempermudah dalam mengontrol jaringan apabila terdapat perubahan dalam *business requirement*. Pada *Software Defined Network*, network administrator atau network engineer dapat membentuk lalu lintas jaringan melalui sebuah central console, sehingga tidak perlu mengkonfigurasi masing-masing switch atau perangkat yang terdapat pada topologi.

Dalam konsep SDN, di perkenalkan dengan adanya *open interface* yang memungkinkan sebuah entitas software atau aplikasi dapat mengelola konektivitas yang disediakan oleh sejumlah sumber-daya jaringan, mengendalikan aliran traffic yang melewatinya serta melakukan inspeksi terhadap atau memodifikasi traffic tersebut. Sedangkan OpenFlow merupakan salah satu komponen dari arsitektur SDN. OpenFlow merupakan pionir standard terbuka yang di definisikan oleh *Open Networking Foundation* (ONF) untuk protocol komunikasi antara *control* dan *forwarding* atau *data plane*. OpenFlow memungkinkan pengaturan *routing* dan pengiriman paket melalui sebuah switch. Dalam sebuah jaringan tradisional, setiap switch hanya meneruskan paket data yang diterima kemudian dikirimkan ke tujuan tanpa mampu membedakan tipe protocol data yang dikirimkan. Dengan menggunakan *OpenFlow* memungkinkan untuk dapat mengakses dan memanipulasi *forwarding plane* secara langsung dari perangkat-perangkat jaringan seperti switch dan router baik secara fisik maupun virtual.

sFlow merupakan salah satu protocol yang digunakan untuk monitoring perangkat dalam jaringan. sFlow memiliki kemampuan untuk dapat melihat penggunaan jaringan, kinerja dan beserta informasi-informasi lain mengenai kondisi jaringan yang sedang di monitoring. Selain itu, sFlow memiliki dukungan multivendor lebih banyak dibanding protokol monitoring lain seperti NetFlow, RMON dan RMON II.

Dengan menggabungkan sFlow dan OpenFlow memberikan kemampuan yang lebih baik dalam SDN. OpenFlow memungkinkan sebuah aplikasi *Controller* untuk dapat melakukan konfigurasi terhadap *forwarding tables* didalam sebuah jaringan yang ada pada perangkat switch. Sementara *sFlow* dapat memberikan instrument standard kedalam *forwarding tables* perangkat switch tersebut yang kemudian dapat di akses melalui API untuk dapat memberikan gambaran *real-time* jaringan yang ada. Dengan menggabungkan keduanya, artinya kita dapat membuat sebuah aplikasi yang sifatnya *feedback control system* atau dapat melakukan adaptasi seketika bila dibutuhkan suatu perubahan kepada suatu jaringan.

Berdasarkan ulasan hasil sebelumnya, maka penelitian ini akan melakukan analisis dan penerapan OpenFlow dan sFlow untuk deteksi dan mitigasi DOS pada *Software Defined Network*.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dikemukakan sebelumnya, maka didapatkan perumusan masalah yaitu :

- a. Bagaimana mengimplementasikan OpenFlow dan sFlow untuk melakukan deteksi terhadap serangan DOS pada *Software Defined Network*.
- b. Bagaimana mitigasi serangan DOS pada *Software Defined Network* dengan menggunakan OpenFlow dan sFlow.

1.3 Tujuan Tugas Akhir

Adapun tujuan yang hendak dicapai dalam penelitian tugas akhir adalah sebagai berikut :

- a. Merancang jaringan *Software Defined Network* menggunakan Mininet.
- b. Implementasi ONOS sebagai *OpenFlow Controller* dan *Network Topology*.
- c. Menggunakan sFlow-RT untuk monitor route dan traffic dalam jaringan.
- d. Melakukan pengujian dan analisa terhadap deteksi dan mitigasi bila terjadi serangan DOS pada *Software Defined Network*.

1.4 Manfaat Tugas Akhir

Adapun manfaat yang dapat diambil dari penelitian tugas akhir adalah :

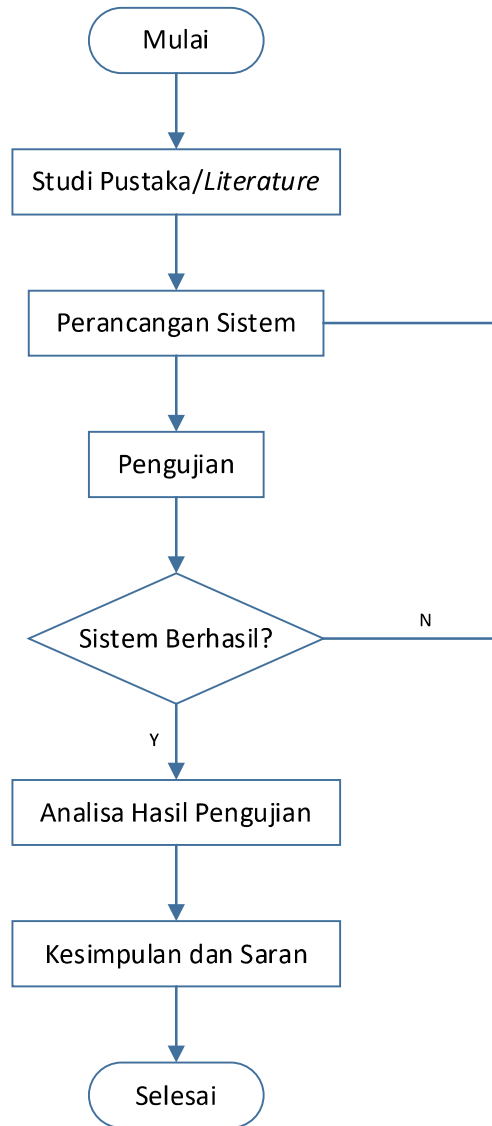
- a. Dapat menjadi solusi pencegahan DOS terhadap *Software Defined Network* yang lebih mudah dan terjangkau.
- b. Dapat menjadi referensi untuk penelitian atau pengembangan selanjutnya.

1.5 Lingkup Tugas Akhir

Selain perumusan masalah diatas, juga terdapat batasan masalah pada tugas akhir ini, antara lain :

- a. Topologi *Software Defined Network* dibangun secara simulasi
- b. *Tools* yang digunakan untuk membangun Topologi *Software Defined Network*, yaitu Mininet emulator yang bertindak sebagai data plane, ONOS sebagai *OpenFlow Controller*, sFlow digunakan untuk monitor route atau traffic dalam jaringan serta Nodejs dan Python digunakan sebagai bahasa pemrograman yang digunakan dalam pembuatan script untuk deteksi dan mitigasi serangan DOS.
- c. Protokol yang digunakan untuk komunikasi antara *Controller plane* dan *data plane* pada jaringan *Software Defined Network* adalah OpenFlow.
- d. Pengujian serangan DOS dilakukan dengan metode serangan *DNS Amplification*.
- e. Dalam pengujian saat ini Mininet belum mendukung multiple session dalam melakukan pengujian, sehingga pengujian potensi kesalahan (*false positive*) terhadap *traffic DNS legitimate* tidak dapat dilakukan.

1.6 Metodologi Penelitian



Gambar 1-1 Metodologi Penelitian

Pada Gambar 1-1, berikut ditampilkan metodologi penelitian secara visual dalam bentuk diagram alir yang merepresentasikan proses pelaksanaan penelitian. Metodologi yang akan digunakan dalam penelitian tugas akhir akan melewati beberapa tahap, yaitu :

1. Tahap Pertama (Studi Pustaka atau *Literature*)

Tahap ini dilakukan setelah masalah yang akan dibahas telah sesuai dan relevan untuk dijadikan sebagai penelitian, dengan membaca artikel atau makalah penelitian yang berhubungan langsung dengan tugas akhir.

2. Tahap Kedua (Perancangan Sistem)

Pada tahapan ini merupakan tahapan mengenai bagaimana proses membangun system dengan menggunakan metode atau pendekatan tertentu. Selain itu, apa saja perangkat keras ataupun perangkat lunak yang digunakan, kemudian bagaimana proses perancangan system, selanjutnya bagaimana pula penerapan metode pada penelitian tugas akhir.

3. Tahap Ketiga (Pengujian)

Tahap ini merupakan tahap lanjutan dari proses perancangan yang telah dilakukan. Dengan melakukan pengujian berdasarkan metodologi penelitian sehingga didapatkan data hasil uji yang sesuai dan tepat secara konsep ataupun praktis.

4. Tahap Keempat (Analisa)

Pada tahap ini adalah menganalisa data hasil pengujian berdasarkan parameter yang telah ditentukan, sehingga didapat hasil yang objektif dimana data diperoleh dari proses pengujian.

5. Tahap Kelima (Kesimpulan dan Saran)

Pada tahap ini akan dirumuskan suatu kesimpulan berdasarkan permasalahan, studi pustaka, metodologi penelitian dan analisis hasil pengujian. Kemudian beberapa saran yang dapat dijadikan landasan untuk lanjutan.

1.7 Sistematika Penulisan

Untuk memudahkan dalam proses penyusunan tugas akhir dan memperjelas konten dari setiap bab, maka dibuat suatu sistematika penulisan sebagai berikut :

BAB I PENDAHULUAN

Bab ini berisi latar belakang, identifikasi masalah, tujuan tugas akhir, dan manfaat tugas akhir, lingkup tugas akhir, metodologi penelitian serta sistematika penulisan.

BAB II TINJUAN PUSTAKA

Bab ini berisi mengenai dasar teori dari penelitian tugas akhir tentang sejarah *Software Defined Network*, definisi *Software Defined Network*, arsitektur *Software Defined Network*, konsep protokol OpenFlow, ONOS sebagai *OpenFlow Controller*, mininet sebagai perangkat lunak untuk simulasi *Software Defined Network*, *sFlow-RT* sebagai monitor route dan traffic dalam jaringan, iperf sebagai *traffic generator* serta parameter pengujian deteksi dan mitigasi DOS.

BAB III METODE

Bab ini berisi penjelasan sistematis, mengenai bagaimana penelitian dilakukan. Penjelasan pada bab ini tentang tahapan perancangan sistem dan penerapan metode penelitian.

BAB IV PENGUJIAN DAN ANALISIS

Bab ini menjelaskan tentang hasil pengujian yang dilakukan serta analisis dari data yang didapat dari hasil pengujian. Analisis data yang akan dilakukan yaitu berdasarkan parameter yang telah ditentukan sebelumnya yaitu *delay*, *packet counter* dan *packet trend*.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan hasil penulisan dan saran yang dituliskan untuk penulisan dan penyempurnaan selanjutnya pada penelitian.

BAB 2

TINJAUAN PUSTAKA

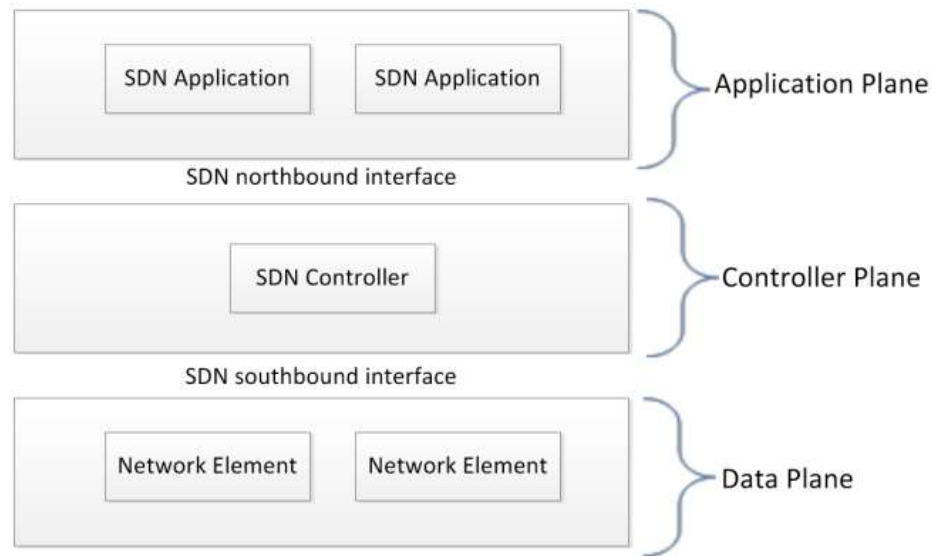
2.1 *Software Defined Network*

Software Defined Network (SDN) adalah istilah yang merujuk pada konsep/paradigma baru dalam mendisain, mengelola dan mengimplementasikan jaringan, terutama untuk mendukung kebutuhan dan inovasi di bidang ini yang semakin lama semakin kompleks. Konsep dasar SDN adalah dengan melakukan pemisahan eksplisit antara control dan forwarding plane, serta kemudian melakukan abstraksi sistem dan meng-isolasi kompleksitas yang ada pada komponen atau sub-sistem dengan mendefinisikan antar-muka (*interface*) yang standard. Dalam hal ini dapat memberikan fleksibilitas terhadap implementasi dan manajemen perangkat jaringan.

Beberapa aspek penting dari SDN adalah :

- Adanya pemisahan secara fisik/eksplisit antara forwarding/data-plane dan control-plane
- Antarmuka standard (*vendor-agnostic*) untuk memprogram perangkat jaringan
- Control-plane yang terpusat (secara logika) atau adanya sistem operasi jaringan yang mampu membentuk peta logika (*logical map*) dari seluruh jaringan dan kemudian memrepresentasikannya melalui (sejenis) API (*Application Programming Interface*)
- Virtualisasi dimana beberapa sistem operasi jaringan dapat mengontrol bagian-bagian (slices atau substrates) dari perangkat yang sama.

Arsitektur SDN terdiri dari tiga layer yaitu sebagai berikut, seperti tampak pada Gambar 2-1



Gambar 2-1 Arsitektur SDN

1. *Application Plane*: berada pada lapisan teratas, berupa aplikasi yang dapat secara langsung dan eksplisit mendefinisikan network requirement dan network behavior yang diinginkan. Layer ini berkomunikasi dengan *Control Layer* melalui *NorthBound Interface* (NBI)
2. *Controller Plane*: yaitu entitas kontrol yang memiliki tugas yaitu mentranslasikan network requirement yang telah didefinisikan oleh *Application Layer* menjadi instruksi-instruksi yang sesuai untuk *Infrastructure Layer*, dan memberikan abstract view yang dibutuhkan bagi *Application Layer* (*abstract view* meliputi informasi statistik dan event yang terjadi di jaringan).
3. *Data Plane*: terdiri dari elemen jaringan yang dapat menerima instruksi dari *Control Layer*. *Interface* antara *Controller Plane* dan *Data Plane* disebut *SouthBound Interface* (SBI), atau *Control-To-Data-Plane Interface* (CDPI).

Controller Plane merupakan otak dari jaringan SDN, dapat dijalankan secara terpisah dari *Data Plane*. Sedangkan *Data Plane* merupakan perangkat-perangkat keras jaringan yang terprogram secara khusus dan dikendalikan penuh oleh *Control Plane*. Pada SDN tersedia *open interface* yang memungkinkan sebuah entitas

software atau aplikasi untuk mengendalikan konektivitas dari sumber daya jaringan, mengendalikan aliran traffic, serta melakukan inspeksi atau memodifikasi traffic tersebut. Berikut beberapa keunggulan SDN antara lain :

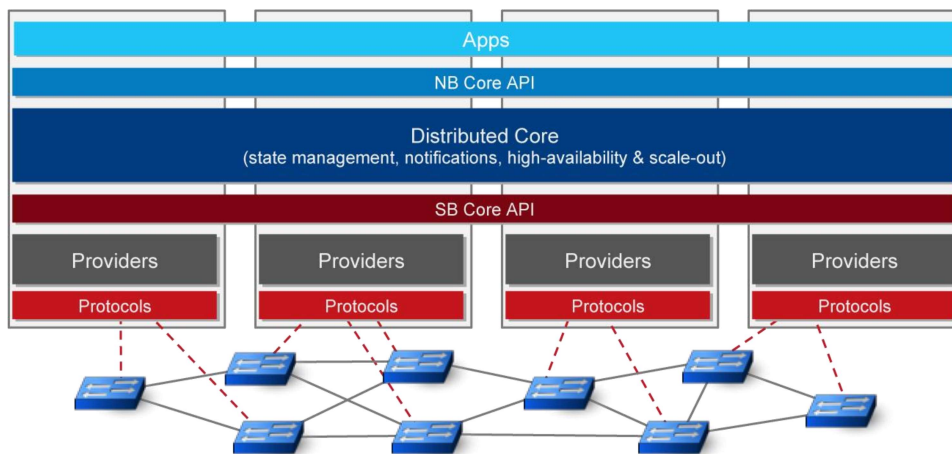
- *Virtualisasi dan Cloud*: Komponen dan entitas jaringan hybrid - antara fisik bare metal dan yang virtual
- *Orchestration dan Scalability*: Kemampuan untuk mengatur dan mengelola ribuan perangkat melalui sebuah point of management
- *Programmability dan Automation*: Kemampuan untuk mengubah behaviour (perilaku) jaringan serta untuk dapat melakukan perubahan tersebut secara otomatis (sebagai contoh adalah kemampuan troubleshooting, perubahan policy dan lain-lain)
- *Visibility*: Kemampuan untuk dapat memonitor jaringan, baik dari sisi sumber daya, konektivitas dan lain-lain.
- *Kinerja*: Kemampuan untuk memaksimalkan penggunaan perangkat jaringan, misalnya optimasi *bandwidth*, *load balancing*, *traffic engineering* dan lain-lain (berhubungan dengan *Programmability* dan *Scalability*)

2.2 Protocol OpenFlow

OpenFlow adalah protokol paling utama pada SDN. Posisinya berada diantara *Controller* dan *forwarding (data plane)*. OpenFlow memungkinkan pengaturan routing dan pengiriman paket ketika melalui sebuah switch. Dalam sebuah jaringan, setiap switch hanya berfungsi meneruskan paket yang melalui suatu port tanpa mampu membedakan tipe protokol data yang dikirimkan. OpenFlow memungkinkan untuk mengakses dan memanipulasi forwarding plane secara langsung dari perangkat-perangkat jaringan seperti switch dan router baik secara fisik maupun virtual. Pada penelitian ini penulis menggunakan ONOS sebagai *OpenFlow Controller*.

ONOS (*Open Network Operating System*) merupakan sistem operasi jaringan sumber terbuka dengan teknologi SDN yang berorientasi pada jaringan operator (carrier-grade service provider). ONOS didisain dengan filosofi HA (*high*

availability), kinerja tinggi (HP - *high performance*) dan kemampuan scaleout yang baik. ONOS diprakarsai dan dibuat oleh ON.Lab (sekarang sudah merger dengan ONF) dengan kerja-sama lintas sektor, baik operator, vendor maupun universitas. ONOS versi pertama (1.0.0) dirilis tanggal 5 Desember 2014. Rilis baru setiap kuartal, saat ini (01.2021) sudah dalam versi 2.5.0. ONOS ditulis dalam bahasa Java dan menggunakan OSGi untuk manajemen fungsionalitas. Setiap fitur dalam ONOS diaktifkan melalui Apache Karaf (OSGi runtime).



Gambar 2-2 Arsitektur ONOS *Controller*

Pada gambar 2-2 merupakan representasi model dari ONOS. ONOS diperkenalkan sebagai sistem operasi jaringan SDN yang multi-entitas dan terdistribusi, serta dirancang untuk HA, kinerja tinggi, skalabel dan dengan abstraksi NB (*Northbound*) /SB (*Southbound*) yang konsisten (dan semakin baik).

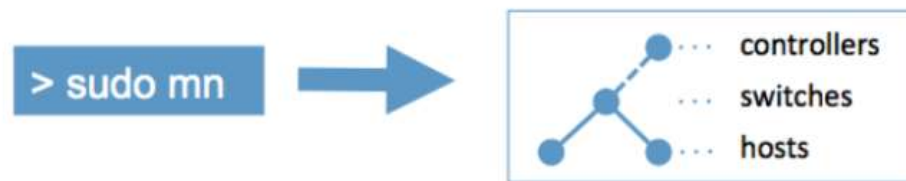
Berikut beberapa attribute terkait dengan ONOS :

- Apps: Menyediakan AIF (*Application Intent Framework*) sebagai NBI untuk aplikasi. AIF adalah *framework (policy-driven)* yang dapat digunakan oleh devs/ops sebagai bahasa high-level tanpa harus memikirkan bagaimana implementasinya di jaringan
- NB Core API: Menyediakan informasi struktur (graph) seluruh jaringan sebagai bagian dari abstraksi NB

- *Distributed Core*: Sistem terdistribusi dengan implikasi HA, HP dan scaleout
- *SB Core API*: Menyediakan abstraksi SB untuk discovery, konfigurasi dan programmability; mendukung OF (OpenFlow) dan sejumlah antar-muka atau perangkat lainnya

2.3 Mininet

Mininet adalah sebuah emulator untuk membuat prototype jaringan berskala besar secara cepat pada sumber daya yang terbatas (seperti pada single komputer atau laptop maupun *Virtual Machine*). Mininet diciptakan dengan tujuan untuk mendukung riset di bidang SDN dan OpenFlow. Emulator Mininet gambar 2-3 memungkinkan kita untuk menjalankan sebuah kode secara interaktif di atas laptop atau di atas virtual hardware, tanpa harus memodifikasi kode tersebut. Artinya kode simulasi sama persis dengan kode pada real network environment.



Gambar 2-3 Mininet Program

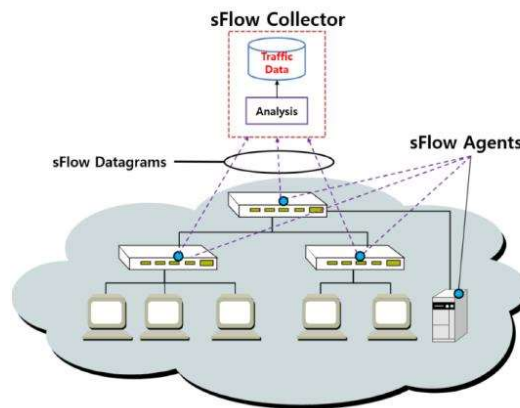
Seperti pada gambar 2-3 mininet menyediakan Command Line interface (CLI) dan *Application Programming Interface* (API) yang kemudian dapat dipanggil oleh *OpenFlow Controller* nantinya.

- Host : mengemulatkan pada level OS process
- Switch : mengemulatkan berdasarkan fungsi dari software pada switch

2.4 sFlow

sFlow merupakan salah satu protocol yang digunakan untuk monitoring perangkat dalam jaringan. sFlow memiliki kemampuan untuk dapat melihat penggunaan jaringan, kinerja dan beserta informasi-informasi lain mengenai kondisi jaringan

yang sedang di monitoring. Selain itu, sFlow memiliki dukungan multivendor lebih banyak dibanding protokol monitoring lain seperti NetFlow, RMON dan RMON II. Pada protocol sFlow terdapat dua system kerja yaitu Agen sFlow (*sFlow Agent*) dan Kolektor sFlow (*sFlow Collector*). Data yang dikirimkan dari Agen sFlow ke kolektor berupa datagram yang terdiri atas informasi-informasi mengenai kondisi jaringan yang sedang dimonitoring, diagram terkait mekanisme cara kerja sFlow merujuk pada gambar 2-4.



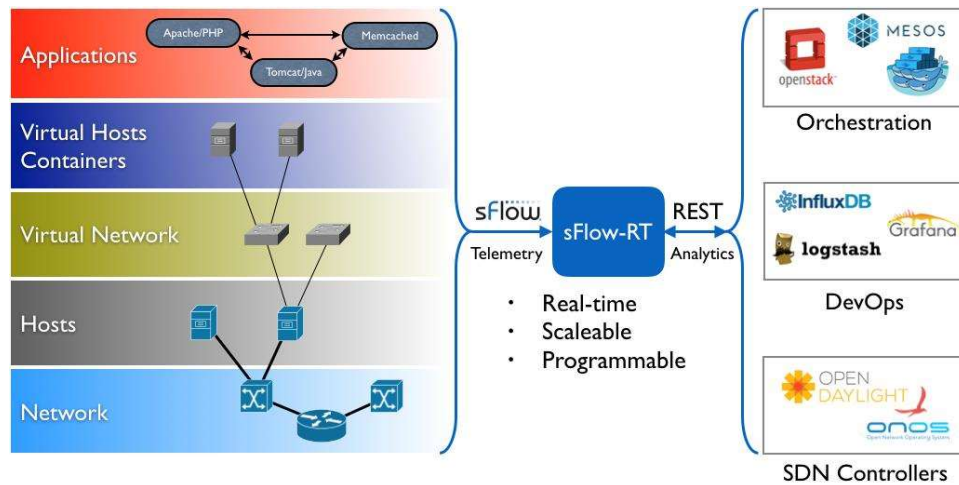
Gambar 2-4 Aplikasi sFlow

2.4.1 Agent sFlow

Agen sFlow adalah sebuah sistem kerja yang mengimplementasikan packet sampling technology untuk menangkap trafik data pada sebuah switch. Trafik data yang ditangkap oleh Agen sFlow dikenal dengan istilah sFlow Datagram, sFlow Datagram ini kemudian akan diteruskan ke Kolektor sFlow.

2.4.2 Kolektor sFlow

Kolektor sFlow adalah sebuah server pusat yang mengumpulkan sFlow datagram dari semua Agen sFlow untuk dianalisis atau diteruskan ke database.



Gambar 2-5 Fungsi sFlow

Gambar 2-5 merupakan representasi dari sFlow-RT yang mana memiliki beberapa manfaat sebagai berikut :

- *Collector* yang dapat dioperasikan berdasarkan script dengan menggunakan REST atau JavaScript
- Dapat menerima metric, mengatur thresholds dan menerima notifikasi
- Dapat digunakan oleh beberapa instance yang berbeda (*Application, Virtual Host, Virtual Networks, Hosts, Networks*)
- Di dukung oleh beberapa domain yang berbeda dan platform yang berbeda (*Orchestration, DevOps, SDN Controllers*).

2.5 IPERF

IPERF adalah salah satu tool yang dapat digunakan untuk mengukur throughput bandwidth dalam sebuah link network, agar bisa dilakukan pengukuran diperlukan Iperf yang terinstall point to point, baik disisi server maupun client. Iperf sendiri bisa digunakan untuk mengukur performance link dari sisi TCP maupun UDP.

2.6 DOS

Denial of Services suatu serangan pada jaringan dengan mekanismenya yaitu dengan cara membanjiri lalu lintas jaringan dengan banyak data. Konsep *Denial of Service* bisa dibagi menjadi 3 tipe penggunaan yaitu :

- *Request flooding* merupakan teknik yang digunakan dengan membanjiri jaringan menggunakan banyak request. Akibatnya, pengguna lain yang terdaftar tidak dapat dilayani.
- *Traffic flooding* merupakan teknik yang digunakan dengan membanjiri lalu lintas jaringan dengan banyak data. Akibatnya, pengguna lain tidak bisa dilayani.
- Mengubah sistem konfigurasi atau bahkan merusak komponen dan server juga termasuk tipe denial of service, tetapi cara ini tidak banyak digunakan karena cukup sulit untuk dilakukan.

2.7 Penelitian Sebelumnya

- Pada penelitian yang berjudul “*Large Flows Detection, Marking, and Mitigation based on sFlow Standard in SDN*” yang terbit pada *Journal of Korea Multimedia Society* oleh Afaq, M., Rehman, S., dan Song, W.-C. (2015). Dalam penelitiannya dapat melakukan deteksi dan mitigasi dalam system SDN dengan menggunakan teknologi pengambilan sample sFlow. Dalam sistem SDN yang digunakan dapat memanfaatkan OpenFlow dan sFlow dengan manajemen control terpusat berbasis perangkat lunak *OpenFlow Controller* yang memungkinkan operator jaringan dapat memeriksa, memprediksi dan mengatur perilaku data yang dikirimkan dalam jaringan. Teknologi pengambilan sampel berbasis sFlow memerlukan tiap OpenFlow switch untuk mengirimkan sampel paket data aliran untuk dikirimkan ke sebuah alat untuk menganalisa semua informasi arus lalu lintas data seperti contohnya sFlowTrend, sFlow-RT, Ganglia dan sebagainya. Yang kemudian dapat dimanfaatkan untuk menentukan keberadaan aliran paket besar berdasarkan sampel data yang digunakan.
- Pada penelitian yang berjudul “*Utilizing OpenFlow and sFlow to Detect and Mitigate SYN Flooding Attack*” yang terbit pada *Journal of Korea Multimedia Society* oleh Nugraha, M., Paramita, I., Musa, A., Choi, D., dan Cho, B. (2014). Dalam penelitiannya menjelaskan bagaimana OpenFlow dan sFlow dapat melakukan deteksi dan mitigasi terhadap *SYN Flood Attack*.

Dalam penelitian tersebut menggunakan 4 switch dan gabungan serangan dari setiap switch yang ada dan kemudian *Controller* akan menentukan arus data yang masuk akan diidentifikasi menjadi tipe serangan atau hanyalah arus data normal. Hasil dari penelitian tersebut memperlihatkan bahwa dengan memanfaatkan OpenFlow dan sFlow lebih baik daripada menggunakan jaringan tradisional dikarenakan OpenFlow dapat memanfaatkan banyak switch untuk melakukan tindakan pencegahan yang mana ini dilakukan dari *Controller*. Dengan menggunakan OpenFlow dan sFlow dapat mengurangi penggunaan sumber daya dan meningkatkan waktu deteksi terhadap serangan.

- Pada penelitian yang berjudul “Perancangan dan Analisis *Software Defined Network* Pada Jaringan LAN : Penerapan dan Analisis Metode Penjaluran Path Calculating Menggunakan Algoritma Dijkstra” yang terbit pada e-Proceeding of Engineering oleh Brayana Anggita Linuwih, Agus Virgono, Budhi Irawan. (2016). Dalam penelitian tersebut dilakukan pembuktian penerapan layanan routing menggunakan metode path calculating algoritma dijkstra sebagai rekayasa kontrol penjaluran pada jaringan LAN berbasis Software-Defined Network dan menganalisa perbandingan kinerja metode pemilihan jalur terbaik dalam lalu lintas jaringan pada jaringan LAN berbasis Software-Defined Network dengan jaringan konvensional yang juga diterapkan algoritma dijkstra dengan arsitektur yang sama namun tidak memiliki perangkat control plane. Pembuktian dilakukan dengan melakukan emulasi jaringan sebuah yang terdiri dari 11 buah switch yang saling terhubung dengan perangkat control plane sebagai pengendali sebuah jaringan.

Hasil pengujian performansi penerapan algoritma dijkstra berbasis jaringan SDN menunjukkan bahwa nilai dari keempat parameter QoS masih berada pada nilai yang menjadi standar ITU-T G.1010 serta memiliki nilai delay dan jitter yang lebih baik dibandingkan penerapan OSPF berbasis jaringan konvensional. Nilai QoS untuk UDP pada layanan data, VoIP masing-masing bervariasi dan berada di kisaran (5.17 – 145.81) ms untuk delay,

(0.05 - 0.93) ms untuk jitter, (13.22 – 73.41) kbps untuk throughput, 8,05 % - 33,81 % untuk packet loss di semua skenario yang telah dibuat. Adapun nilai QoS untuk TCP pada layanan data masing- masing bervariasi dan berada di kisaran (10.32 – 20.21) ms untuk delay, (0.5 - 3.02) ms untuk jitter, (38.12 – 38.27) kbps untuk throughput di semua skenario yang telah dibuat.

- Pada penelitian yang berjudul “*Simulation of Minimum Path Estimation in Software Defined Networking Using Mininet Emulator*” yang terbit pada *British Journal of Mathematics dan Computer Science* oleh S. M. Shamim, M. Badrul, A. Miah, A. Sarker, A. N. Bahar, and A. Sarker. (2017). Dalam penelitian tersebut mengimplementasikan algoritma Bellman-Ford untuk melakukan komputasi terhadap jalur terpendek untuk sebuah jalur *vertex* untuk semua *vertices* lain pada *weighted digraph*. Algoritma ini dapat digunakan untuk banyak hal, dan memiliki fungsi untuk menangani *graph* dimana beberapa diantara *edge weights* memiliki nilai negative. Dalam simulasi penelitian ini menggunakan POX sebagai *OpenFlow Controller*, OpenvSwitch (OVS) sebagai fungsi pengiriman dan Mininet yang terinstall didalam Ubuntu *Virtual Machine* (VM). Hasil dari penelitian ini menunjukkan bahwa SDN dengan OpenvSwitch (OVS) dan POX *Controller* dapat menjalankan algoritma Bellman-Ford untuk menentukan jumlah jalur minimum diantara jalur topologi jaringan yang ada.
- Pada penelitian yang berjudul “*A Survey of Software-Defined Networking : Past , Present , and Future of Programmable Networks*” yang terbit pada *IEEE Communications Surveys & Tutorials* oleh B. A. A. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, dan T. Turletti. (2014). Dalam penelitian tersebut membahas bidang pengembangan terkait *Software-Defined Networking (SDN)*. Dimulai dari sejarah *programmable network*, dari pengenalan ide ini sampai dengan pengembangan terakhir. Dan juga membahas terkait arsitektur SDN secara detail termasuk juga didalamnya terdapat standarisasi OpenFlow. Dalam penelitian tersebut juga membahas implementasi SDN dan platform untuk testing dan pengujian terhadap

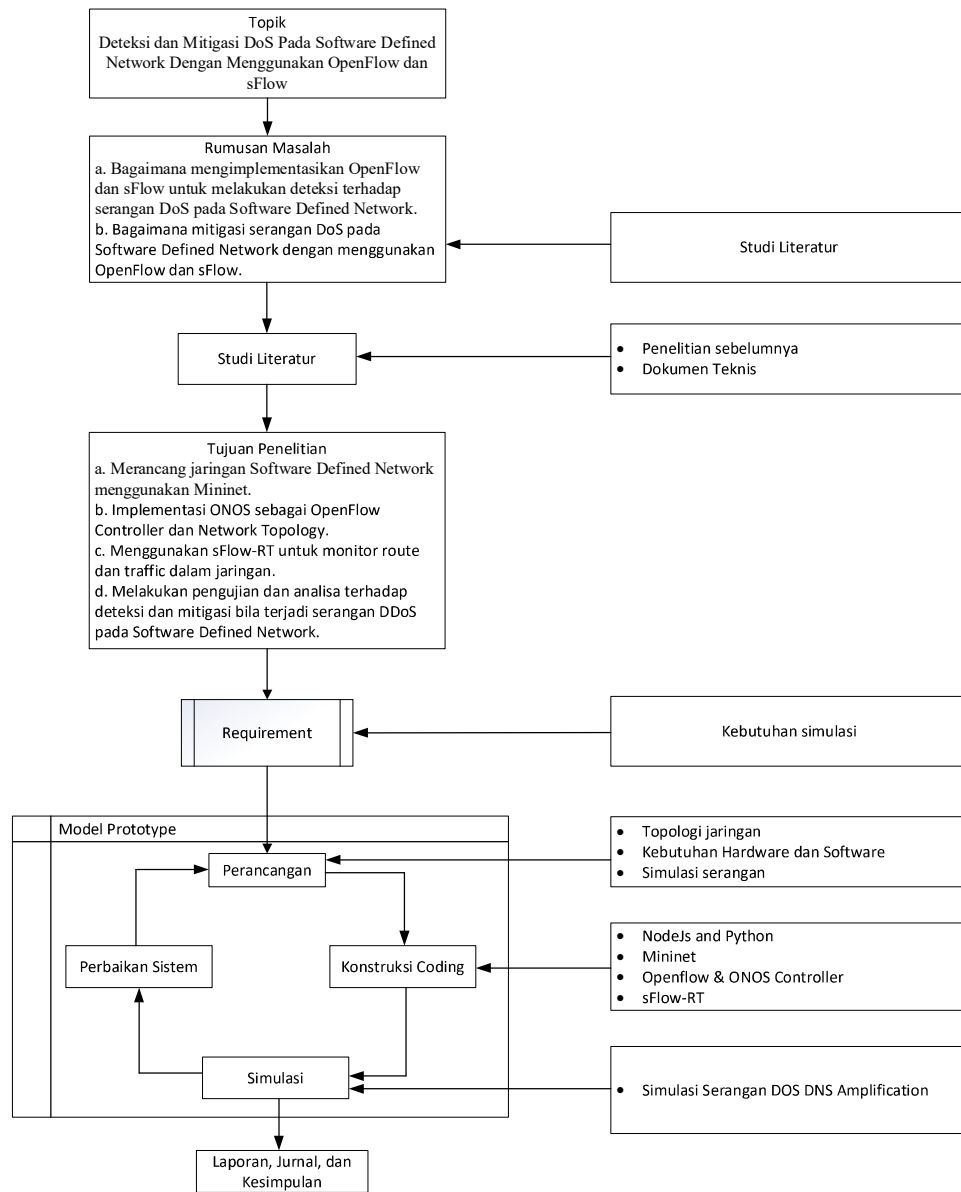
layanan jaringan dan aplikasi yang dibangun diatas paradigma SDN. Dalam penelitian tersebut juga membahas arah kedepan terkait SDN dari dukungan terhadap jaringan heterogen ke jaringan berbasis informasi.

BAB 3

METODE PENELITIAN

3.1 Pendahuluan

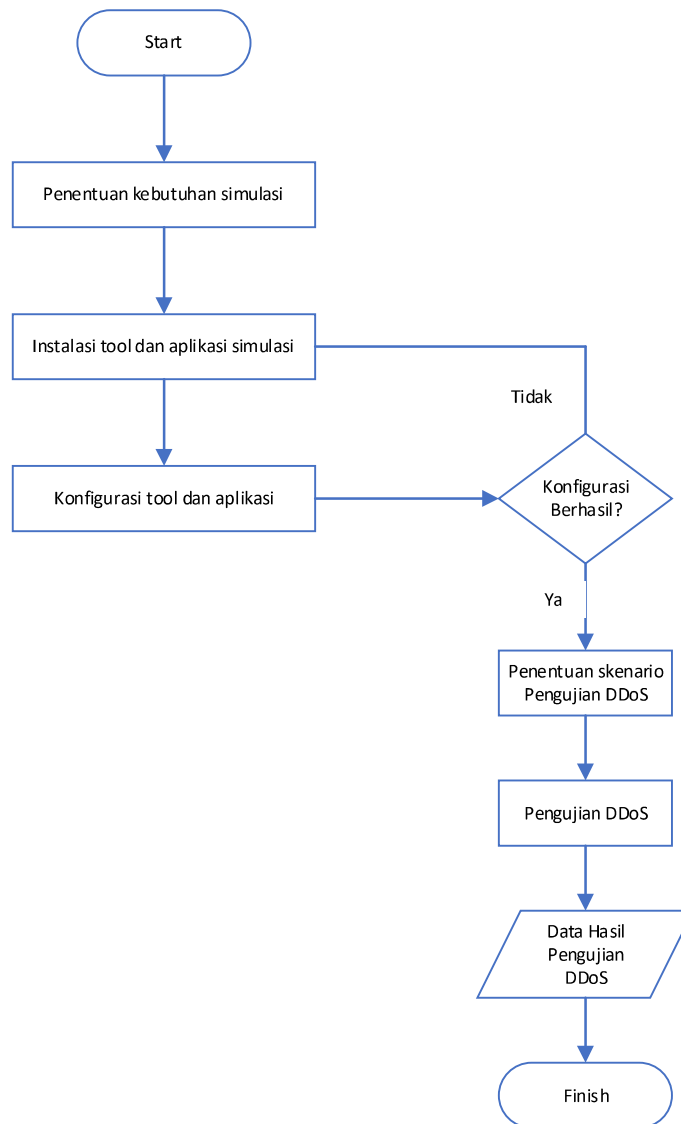
Secara garis besar penelitian ini ditujukan untuk menerapkan openflow dan sflow sebagai solusi serangan DOS pada *Software Defined Network*. Berikut adalah langkah-langkah yang akan dilakukan dalam penelitian ini dapat dilihat pada Gambar 3-1.



Gambar 3-1 Tahapan Simulasi

3.2 Kerangka Simulasi

Jaringan virtual SDN yang sudah dibangun selanjutnya akan diuji dengan scenario serangan DOS yang berbeda-beda. Kerangka kerja yang dilakukan adalah sebagai berikut (lihat gambar 3-2)



Gambar 3-2 Kerangka Simulasi

- Penentuan kebutuhan simulasi.
Sesuai dengan konsep utama SDN yang memisahkan antara control plane dan data plane, maka host untuk menginstal *Controller* SDN (ONOS) dipisahkan dengan host untuk menginstall Mininet.
- Instalasi tool/aplikasi untuk simulasi.
Install Mininet dengan versi 2.2.2, install paket protokol openflow dengan versi 1.0.0 dan kemudian install *Controller* ONOS dengan versi 2.5.0.

Instalasi Mininet dan OpenFlow dilakukan pada host Mininet. Sedangkan *Controller* ONOS diinstall pada host *Controller* SDN.

- Konfigurasi tool/aplikasi.
Konfigurasi server Mininet agar dapat terhubung dengan *Controller* SDN. Termasuk konfigurasi sFlow pada server Mininet untuk dapat melakukan monitor traffic yang ada pada mininet. Parameter konfigurasi sudah berjalan dengan baik yaitu ketika node Mininet dapat menjalankan topologi jaringan yang kemudian dapat ditampilkan pada *Controller* ONOS.
- Pengecekan Konfigurasi.
Memastikan bahwa semua node Mininet yang dibuat dapat termonitor oleh ONOS termasuk dengan indikator informasi traffic dan informasi routing.
- Penentuan skenario pengujian jaringan.
- Pengujian jaringan dan pembahasan hasil uji.

Spesifikasi untuk *Controller* SDN dan Mininet akan di install diatas Virtual Machine. Adapun masing-masing virtual machine akan dialokasikan spesifikasi yang sama yaitu 1 processor (shared processor dengan host Intel Core i5), memory ram 2GB, harddisk 10GB dan OS Linux Ubuntu 12.04 untuk ONOS dan Ubuntu 14.04.4 untuk mininet.

3.3 Peralatan Yang Diperlukan

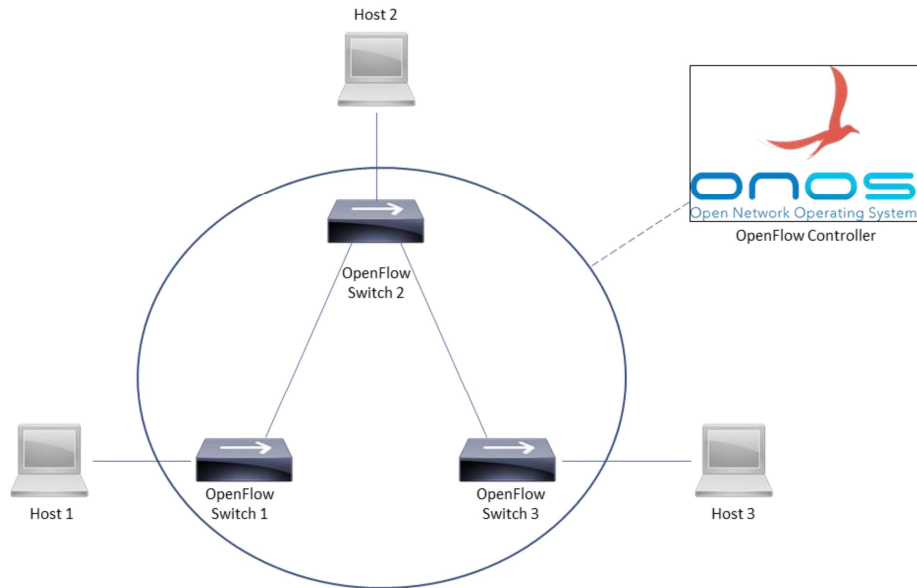
Dalam penelitian ini menggunakan beberapa peralatan yang diperlukan, diantaranya adalah sebagai berikut :

- Mininet Emulator yang bertindak sebagai dataplane (emulasi node)
- ONOS sebagai *OpenFlow Controller*
- sFlow digunakan untuk monitor route atau traffic dalam jaringan
- NodeJs dan Python sebagai bahasa pemrograman yang digunakan dalam pembuatan script untuk deteksi dan mitigasi serangan DOS.

3.4 Perancangan

3.4.1 Perancangan Topologi

Dalam simulasi serangan *DOS DNS Amplification* ini akan menggunakan desain topologi sebagai berikut.



Gambar 3-3 Topologi Jaringan Mininet

Topologi pada gambar 3-3 akan di proses melalui aplikasi Mininet dengan beberapa komponen sebagai berikut :

- 1 node *Controller* referensiional (c0)
- 3 buah node host (h1,h2,h3)
- 3 buah node switch open vswitch (s1,s2,s3)

3.4.2 Kebutuhan Hardware (Perangkat Keras)

Untuk melakukan simulasi tugas akhir ini akan menggunakan perangkat keras berupa sebuah laptop dengan spesifikasi sebagai berikut pada Tabel 1 :

Spesifikasi	Laptop
Processor	Intel Core i5 8250U Quad Core Eight Thread (up to 3.4GHz, 6MB L3 Cache)

RAM	8GB DDR4 2400MHz
Operating System	Windows 10
Fungsi	Host OS untuk Virtual Machine <i>OpenFlow Controller</i> dan Simulasi Jaringan

Tabel 1 Informasi Perangkat Keras

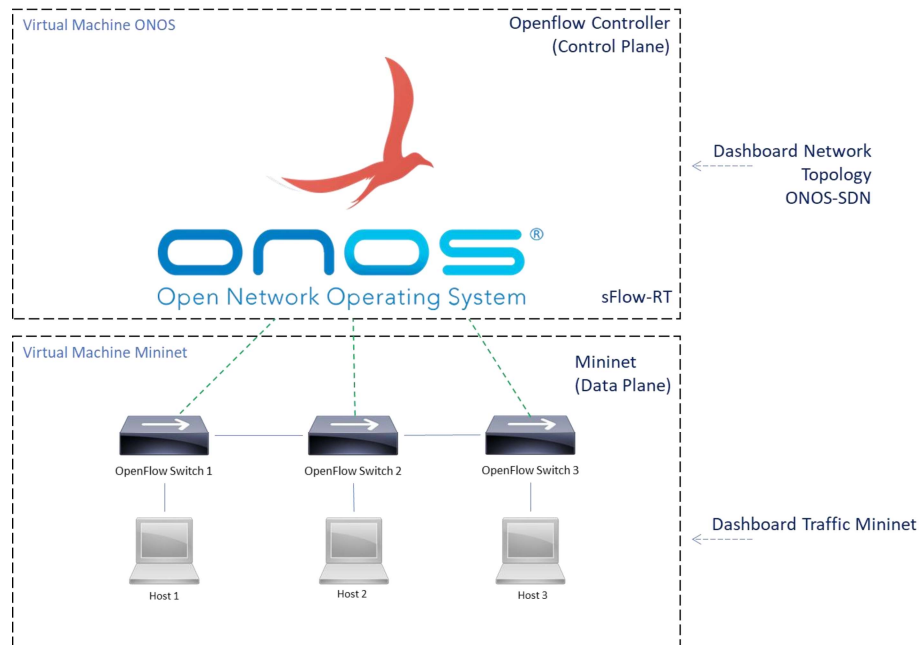
Laptop akan digunakan sebagai Host OS untuk menjalankan VirtualBox. Kemudian didalam VirtualBox akan dibuat 2 buah virtual machine yaitu ONOS sebagai *OpenFlow Controller* (control plane) dan Mininet sebagai simulator jaringan (data plane). Pemisahan *Controller* dan data plane pada virtual machine yang berbeda ini bertujuan untuk mendapatkan hasil yang maksimal dari setiap simulasi yang dilakukan.

3.4.3 Kebutuhan Software (Perangkat Lunak)

Ada beberapa perangkat lunak yang akan digunakan dalam penelitian berikut diantaranya sebagai berikut :

- Iperf, digunakan untuk mengenerate traffic
- VirtualBox, digunakan untuk menjalankan virtual machine
- Ubuntu OS, digunakan sebagai guest OS untuk menjalankan ONOS *Controller* dan Mininet
- sFlow, sebagai aplikasi pendukung untuk monitoring route dan traffic dalam jaringan
- NodeJs dan Python, sebagai bahasa pemrograman untuk pembuatan aplikasi mitigasi DOS

3.4.4 Integrasi Control Plane dan Data Plane



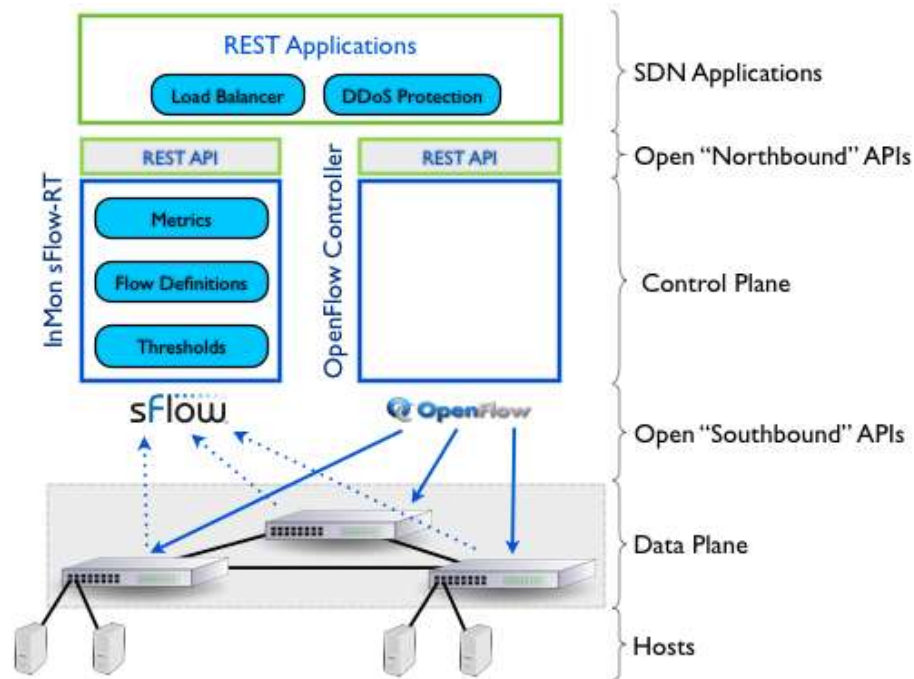
Gambar 3-4 Integrasi ONOS dan Mininet

Pada gambar 3-4 merupakan rencana topologi integrasi antara control plane dan data plane dimana pada tahap ini akan diintegrasikan antara data plane yang berada pada Mininet dan ONOS sebagai *OpenFlow Controller* sebagai control plane. Pada sistem model diatas disimulasikan dengan menggunakan perangkat lunak Mininet yang merupakan aplikasi untuk simulasi jaringan. Mininet menggunakan Linux container dan Open vSwitch untuk mensimulasikan jaringan virtual dari tiap host dan switch untuk dapat di jalankan dengan menggunakan virtual machine.

Pada diagram 3-4, permodelan sistem yang dirancang terdiri dari 3 virtual switch dan 3 hosts. Virtual switch tersebut nantinya akan terhubung dengan ONOS *OpenFlow Controller* dan *sFlow-RT Controller*. sFlow kemudian akan dikonfigurasi pada switch untuk dapat melakukan capture packet sesuai dengan sampling rate yang didefinisikan oleh sFlow. Sample tersebut kemudian akan dikirim dan akan diolah untuk menjadi *measurement datagram* dari setiap switch atau agent kepada sFlow-RT sebagai *real-time analytics engine*. Informasi *measurement datagram* yang diperoleh dari sFlow yang di proses secara *real-time*

oleh sFlow-RT yang mana data tersebut akan digunakan untuk informasi statistik ke aplikasi pengontrol melalui REST API. Flow yang sudah didefinisikan untuk dibuat kategori untuk digunakan keperluan deteksi dan mitigasi lebih lanjut.

3.4.5 Implementasi sFlow-RT untuk monitor route dan traffic

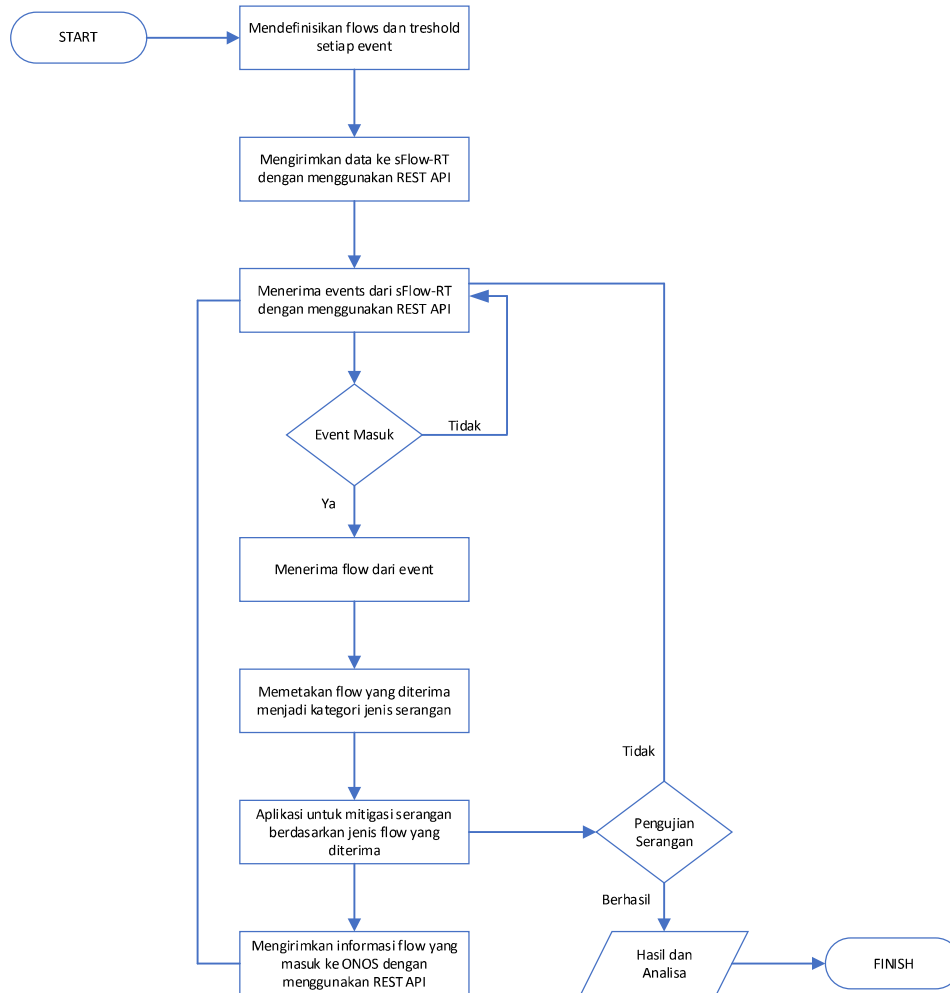


Gambar 3-5 Integrasi sFlow

Pada gambar 3-5 merupakan rencana integrasi sFlow-RT untuk dimanfaatkan sebagai masukan data yang kemudian akan di olah control plane sebagai parameter penentuan keputusan terhadap event yang diterima.

Untuk dapat berkomunikasi dengan sFlow analytics engine seperti sFlow-RT, switch perlu dikonfigurasi untuk menggunakan sFlow protocol pada control plane. Dengan tambahan switch juga perlu dikonfigurasi untuk menggunakan *OpenFlow Controller* seperti *ONOS Controller* pada penelitian ini sebagai control plane. Perangkat lunak control plane seperti *OpenFlow Controller* dan sFlow menggunakan Open Northbound API untuk mengakomodir fungsi kontrol dan informasi statistik terkini kepada aplikasi SDN yang kemudian dari data tersebut

bisa dimanfaatkan untuk kepentingan lebih lanjut seperti mitigasi *DOS*, *Load Balancer* dan *Large Flow Marking* dan sebagainya.



Gambar 3-6 Flowchart Aplikasi

Berikut ini adalah penjelasan dari langkah-langkah yang dilakukan pada Gambar 3-6 :

- Mendefinisikan flows dan treshold dari setiap event
Fungsinya untuk memberikan batasan untuk membedakan mana event normal mana yang bisa disebut sebagai serangan.
- Mengirimkan data ke sFlow-RT dengan menggunakan REST API

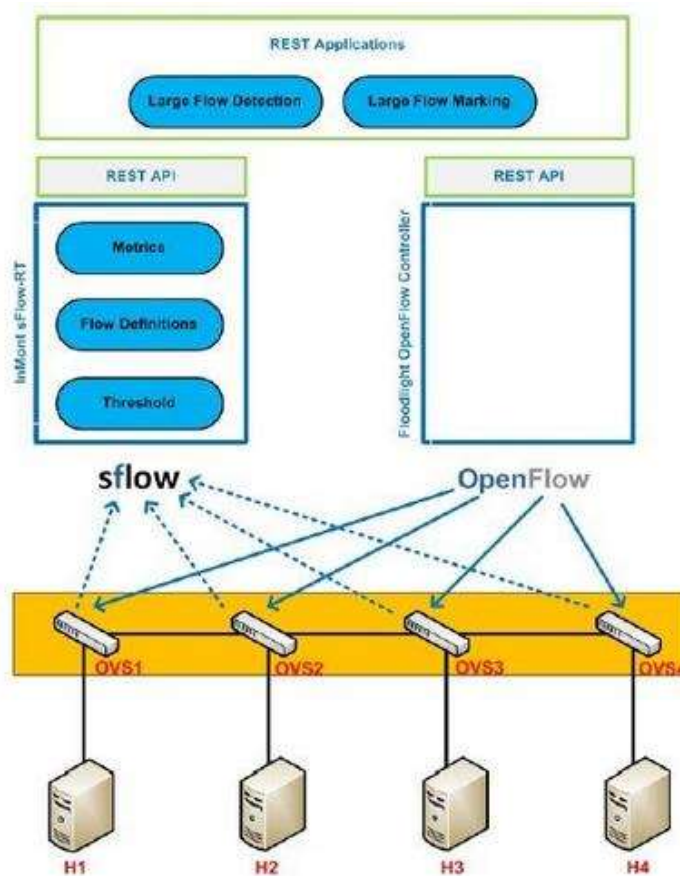
Pada bagian ini setiap node akan mengirimkan data ke sFlow-RT dengan menggunakan REST API terkait informasi routing maupun traffic yang ada pada tiap node.

- Menerima events dari sFlow-RT dengan menggunakan REST API
Pada bagian ini ONOS akan menerima informasi dari event tersebut dengan menggunakan REST API dan bila menerima event kemudian akan mengolah dan kemudian akan dipetakan kategori serangan tersebut sesuai flow yang diterima. Kemudian aplikasi yang akan dibuat akan digunakan untuk melakukan mitigasi atau melakukan drop packet berdasarkan flow yang sudah ditentukan.
- Mengirimkan informasi flow yang masuk ke ONOS dengan menggunakan REST API
Informasi yang sudah diterima kemudian di kirimkan kembali ke ONOS untuk dijadikan data library untuk deteksi serangan.
- Pengujian Serangan
Pada bagian ini setelah dibuat aplikasi untuk mitigasi. Dilakukan serangkaian sample serangan DOS dengan mentrigger serangan DNS Amplification untuk meverifikasi apakah aplikasi mitigasi yang sudah dibuat berhasil melakukan drop packet atau tidak.
- Hasil dan Analisa
Hasil akhir kemudian akan dijadikan bahan untuk analisa dalam penelitian ini.

3.5 Penentuan Skenario Pengujian

Dalam scenario pengujian ini menggunakan aplikasi Mininet sebagai emulator node jaringan. Mininet menggunakan linux container dan Open vSwitch untuk membangun realistic jaringan virtual host dan switch yang di jalankan dengan menggunakan virtual machine. Dalam simulasi ini juga menggunakan sFlow-RT untuk melakukan monitoring terhadap sFlow protocol dan melakukan pencatatan terhadap paket besar dalam jaringan.

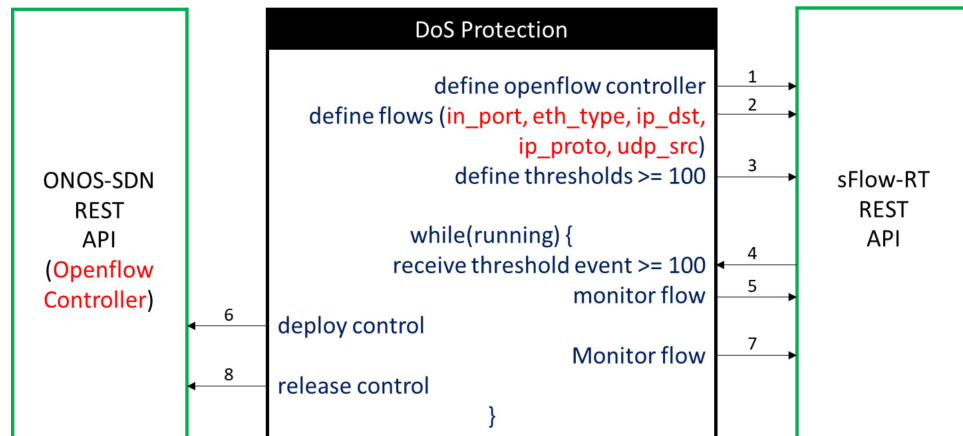
Dalam gambar 3-7 merupakan diagram jaringan yang terdiri dari 4 virtual switch. Setiap virtual switch terhubung kedalam 1 jaringan yang sama. Setiap virtual switch juga terhubung dengan ONOS *OpenFlow Controller* dan *sFlow-RT Controller*. Sampel paket data kemudian dikirimkan oleh setiap switch atau agent ke *sFlow-RT* sebagai mesin analisa secara real-time. Data stream kemudian diterima oleh *sFlow* yang secara langsung di proses oleh *sFlow-RT* yang dimana dapat memberikan *real-time summary statistics* untuk kemudian dapat di control oleh aplikasi melalui northbound REST APIs.



Gambar 3-7 Aplikasi sFlow dan OpenFlow

Aturan *Flows* kemudian ditentukan untuk melakukan deteksi dan pencatatan terhadap paket yang lewat dalam jaringan. Sebuah *Flow* ditentukan menggunakan

kolom *IN_PORT*, *ETH_TYPE*, *IPV_DST*, *IP_PROTO* dan *UDP_SRC* attribute. Pada gambar 3-8 merupakan *Flow* yang di definisikan java script yang dibangun untuk mengontrol menggunakan mitigasi dan pencatatan flow paket.



Gambar 3-8 Flow Script

ONOS Rest API kemudian digunakan untuk melakukan menambah atau menghapus filter terhadap *DOS traffic*. sFlow kemudian diatur untuk melakukan deteksi bila terdapat paket flow (UDP) besar yang telah ditentukan oleh *udp_reflection* flow yang berfungsi untuk mendeteksi serangan UDP amplification sesuai dengan gambar 3-8. Kontrol paket kemudian di aplikasikan terhadap port switch dimana traffic akan masuk ke dalam jaringan.

Pada gambar 3-8 terdapat fungsi *defineThreshold* yang berfungsi untuk mendefinisikan batasan *threshold* yang ditentukan. Dalam konfigurasi script diatas terdapat konfigurasi nilai 100 yang memiliki arti *threshold* 100 paket data per detik. Jadi ketika terdapat arus data dns dengan jumlah 100 paket dalam 1 detik dari sumber yang sama maka *controller* kemudian akan memberikan instruksi terhadap switch untuk melakukan blocking terhadap paket tersebut. Berdasarkan keterangan developer sFlow-RT dalam tulisan “*ONOS measurement based control*” Peter, Sflow.com . Nilai *threshold* ini dapat dirubah sesuai dengan kebutuhan, nilai 100 dalam percobaan simulasi ini digunakan hanya sebagai bentuk demonstrasi dengan

kapasitas link Mininet yang digunakan adalah 10 Mbit/s. Dalam lingkungan produksi perlu ada penyesuaian terhadap nilai *threshold* yang ditentukan.

BAB 4

HASIL DAN PEMBAHASAN

4.1 Kebutuhan Simulasi

4.1.1 Spesifikasi Kebutuhan Simulasi

Dalam analisis kebutuhan sistem, berikut merupakan spesifikasi kebutuhan yang diperlukan dalam membangun sistem ini.

- a. Perangkat lunak, perangkat lunak yang diperlukan untuk aplikasi yaitu sebagai berikut :
 - Sistem Operasi (Ubuntu Server 20.04.01)
 - *OpenFlow Controller* (Onos ONF)
 - sFlow
 - NodeJs dan Python
- b. Perangkat Keras, spesifikasi minimal perangkat keras yang dibutuhkan adalah sebagai berikut :
 - 70 GB Hardisk
 - 8 GB RAM (Memori)

4.1.2 Kebutuhan Fungsional

Berdasarkan Analisa terhadap Rancangan simulasi Deteksi dan Mitigasi DOS Pada *Software Defined Network* Dengan Menggunakan OpenFlow dan sFlow dapat dirumuskan kebutuhan fungsional sebagai berikut:

1. Menjalankan ONOS-SDN (*sFlow Controller*)
2. Menjalankan sFlow-RT
3. Menjalankan Mininet

4.2 Topologi Jaringan

Dalam perancangan simulasi serangan DOS ini menggunakan Mininet sebagai simulator host dan switch. Kemudian diintegrasikan dengan ONOS-SDN sebagai *OpenFlow Controller* dan Topologi Jaringan Gambar 4-1.



Tahapan ini merupakan process tahapan untuk menjalankan aplikasi, integrasi dan verifikasi.

Pada konfigurasi dibawah memperlihatkan bagaimana ONOS-SDN *OpenFlow Controller* dijalankan.

Come help out! Find out how at: contribute.onosproject.org

```
Hit '<tab>' for a list of available commands  
and '[cmd] --help' for help on a specific command.  
Hit '<ctrl-d>' or type 'logout' to exit ONOS session.
```

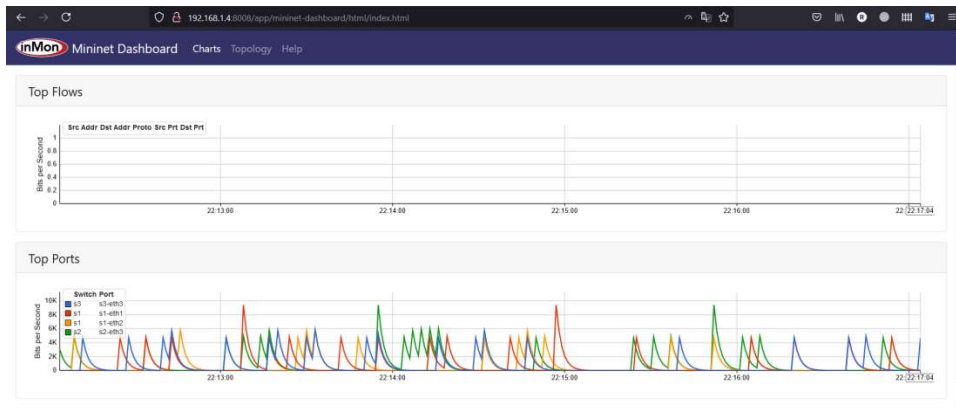
```
ubuntu@root > app activate org.onosproject.openflow          17:11:16  
Activated org.onosproject.openflow  
ubuntu@root > app activate org.onosproject.fwd              17:11:52  
Activated org.onosproject.fwd
```

4.3.2 Menjalankan sFlow-RT Controller

Pada konfigurasi dibawah memperlihatkan bagaimana script sFlow-RT dijalankan.

```
root@mininet-vm:/home/mininet/sflow-rt# env RTPROP=-Dscript.file=DOS.js  
./start.sh  
  
2021-06-03T04:46:12-07:00 INFO: Starting sFlow-RT 3.0-1596  
2021-06-03T04:46:14-07:00 INFO: Version check, 3.0-1599 available  
2021-06-03T04:46:14-07:00 INFO: Listening, sFlow port 6343  
2021-06-03T04:46:15-07:00 INFO: Listening, HTTP port 8008  
2021-06-03T04:46:15-07:00 INFO: DOS.js started  
2021-06-03T04:46:15-07:00 INFO: app/mininet-dashboard/scripts/metrics.js  
started  
root@mininet-vm:/home/mininet/sflow-rt#
```

Setelah sFlow-RT dijalankan kemudian dashboard sFlow dapat digunakan untuk melakukan monitoring terhadap jumlah arus lalu lintas data yang ada pada Mininet sesuai pada gambar 4-2.



Gambar 4-2 sFlow-RT Mininet Dashboard

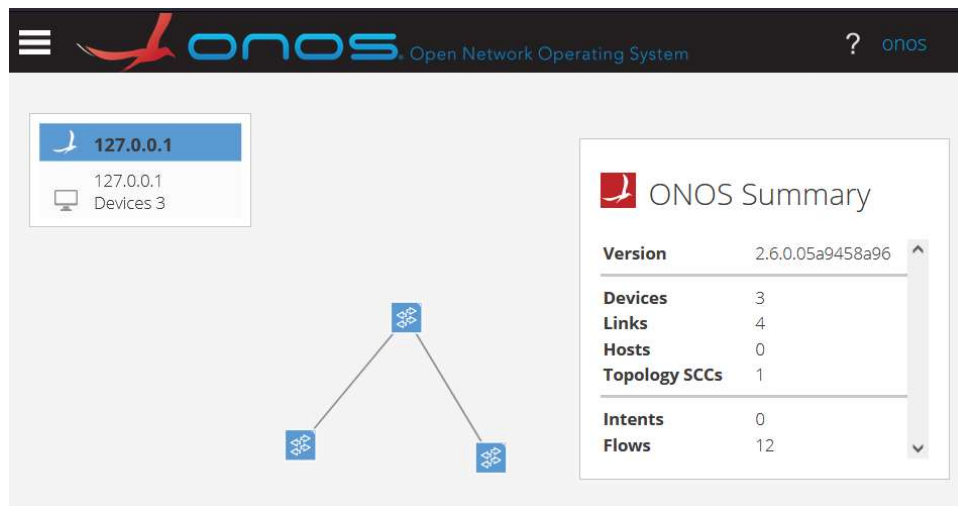
4.3.3 Menjalankan Mininet beserta integrasi ke ONOS-SDN

Kemudian pada kolom dibawah merupakan konfigurasi untuk menjalankan dan integrase antara ONOS-SDN dan Mininet.

```
root@mininet-vm:/home/mininet/sflow-rt# mn --custom extras/sflow.py --link
tc,bw=10 --Controller=remote,ip=192.168.1.3 --
switch=default,protocols=OpenFlow10 --topo tree,2,2
```

4.3.4 Visualisasi Jaringan pada ONOS-SDN

Setelah ONOS-SDN dijalankan, dapat kita lihat sesuai gambar 4-3 dan 4-4 bahwa terdapat 3 devices masing-masing switch terhubung satu dengan yang lain.



Gambar 4-3 Topologi Jaringan ONOS-SDN

The screenshot shows the ONOS Open Network Operating System interface with a list of devices. The table below represents the data shown in the interface:

	FRIENDLY NAME	DEVICE ID	MASTER	PORTS	VENDOR	H/W VERSION	S/W VERSION	PROTOCOL
✓	of0000000000000001	of0000000000000001	127.0.0.1	3	Nicira, Inc.	Open vSwitch	2.13.1	OF_10
✓	of0000000000000002	of0000000000000002	127.0.0.1	4	Nicira, Inc.	Open vSwitch	2.13.1	OF_10
✓	of0000000000000003	of0000000000000003	127.0.0.1	4	Nicira, Inc.	Open vSwitch	2.13.1	OF_10

Gambar 4-4 Daftar Perangkat pada ONOS-SDN

4.3.5 Verifikasi Monitoring

Pengujian dilakukan untuk memastikan bahwa sFlow dashboard dapat melihat traffic yang di generate oleh Mininet.

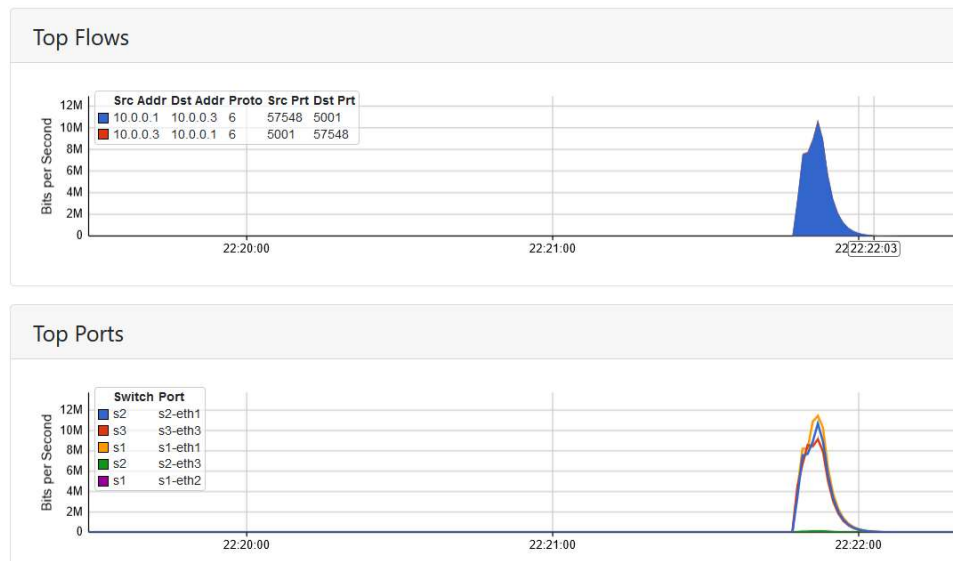
```
mininet> iperf h1 h3

*** Iperf: testing TCP bandwidth between h1 and h3

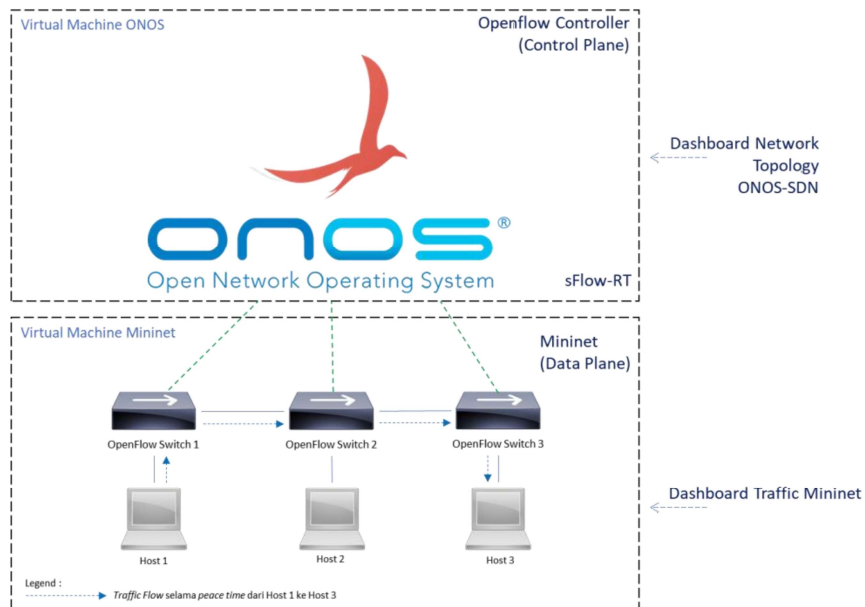
*** Results: ['9.54 Mbits/sec', '10.2 Mbits/sec']

mininet>
```

Pengujian traffic dalam kondisi normal pada gambar 4-5.



Gambar 4-5 Pengujian Traffic



Gambar 4-6 Traffic Flow Peace Time

Dalam tahap ini sesuai pada gambar 4-6 merupakan *peace time network condition* dimana arus komunikasi data masih normal dari Host 1 ke Host 3.

4.3.6 Simulasi Serangan DNS Amplification

Pengujian dilakukan dengan melakukan trigger serangan DNS Amplification dalam jumlah ukuran yang besar. Pengujian dilakukan untuk mengetahui apakah script aplikasi deteksi dan mitigasi DOS dapat berjalan seharusnya.

Berikut adalah command yang digunakan untuk trigger traffic DNS dalam jumlah besar.

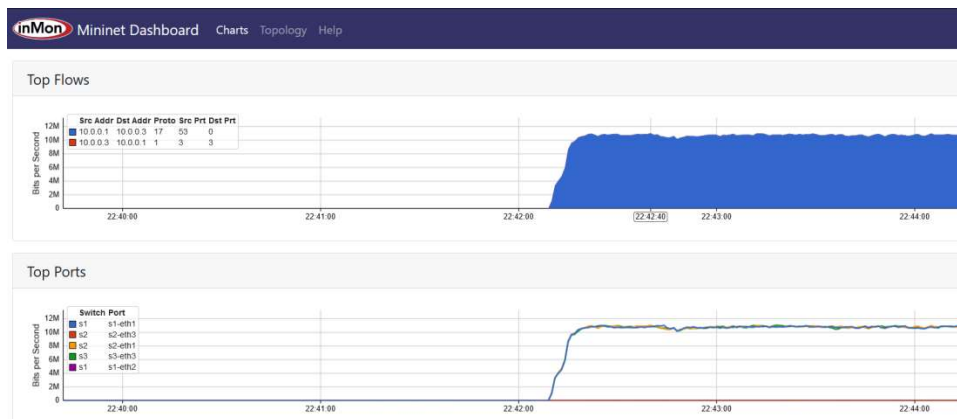
```
mininet> h1 hping3 --flood --udp -k -s 53 h3
```

HPING 10.0.0.3 (h1-eth0 10.0.0.3): udp mode set, 28 headers + 0 data bytes

hping in flood mode, no replies will be shown

4.4 Hasil simulasi serangan DNS

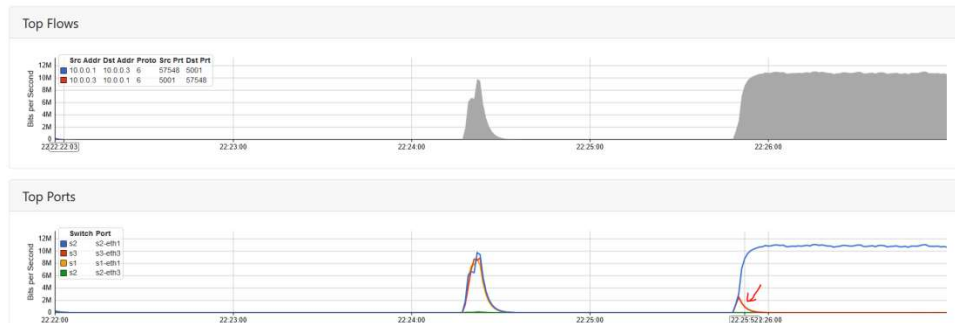
Berikut merupakan hasil pengujian yang dilakukan dalam pengujian serangan DNS Amplification.



Gambar 4-7 *Traffic Serangan DNS tanpa script*

Pertama dilakukan pengujian tanpa menggunakan script deteksi DOS. Dalam gambar 4-7 memperlihatkan bahwa arus lalu lintas data tetap berjalan meski terdapat paket besar atau DOS. Dalam kondisi ini bisa menyebabkan kapasitas jaringan akan penuh dan akan berdampak terhadap layanan yang berjalan dalam jaringan tersebut.

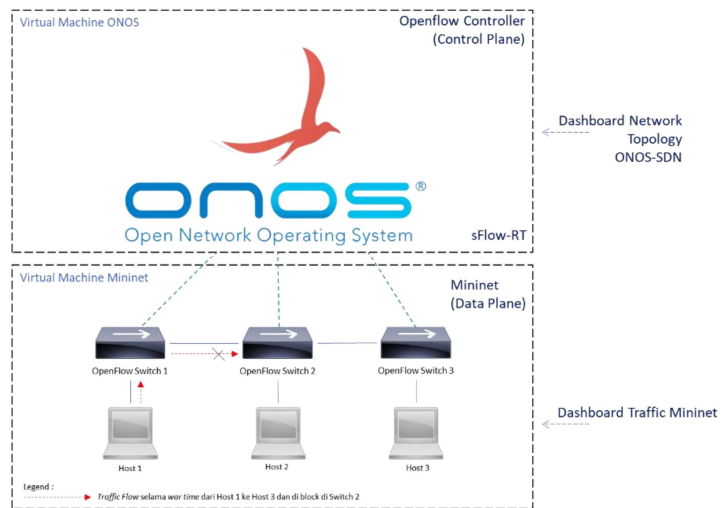
Dalam pengujian berikutnya, telah ditambahkan script deteksi dan mitigasi DOS sesuai pada gambar 4-8.



Gambar 4-8 *Traffic* Serangan DNS dengan script

Pada garis merah yang terdapat dalam gambar 4-8 diatas menunjukkan bahwa serangan DOS dapat berhasil di block.

Pada gambar di 4-6 menunjukkan test iperf diikuti oleh serangan simulasi. Bagan teratas menunjukkan arus teratas yang memasuki jaringan, menunjukkan lalu lintas serangan amplifikasi DNS dengan warna biru. Bagan tengah menunjukkan lalu lintas yang dipecah oleh port switch. Di sini, garis biru menunjukkan lalu lintas serangan yang tiba di swith s2 port s2-eth1 sedangkan garis oranye menunjukkan bahwa hanya sejumlah kecil lalu lintas yang diteruskan ke switch s3 port s3-eth3 sebelum serangan diblokir di switch s2 oleh *Controller*.



Gambar 4-9 *Traffic Flow War Time*

Dalam tahap ini sesuai pada gambar 4-9 merupakan *war time network condition* dimana terdapat arus data anomaly dalam jumlah besar dari Host dibawah Switch 1 dan kemudian *controller* memberikan instruksi untuk melakukan *block* terhadap paket yang sesuai dengan rule script detect DoS.

BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari pembahasan yang telah dijelaskan pada bab-bab sebelumnya, dapat disimpulkan bahwa :

- ONOS, Mininet dan sFlow-RT dapat dikombinasikan sebagai simulator untuk pengujian sistem SDN secara virtual. Dalam hal ini kemudian dapat digunakan untuk pengembangan dan pengujian aplikasi SDN lain dengan kebutuhan sumber daya yang cukup ringan dan biaya yang rendah dibandingkan jaringan fisik.
- Pengujian simulasi *DOS DNS Amplification* dalam jaringan SDN berjalan dengan baik. Dalam penelitian ini menunjukkan bahwa sistem yang dibangun dapat melakukan deteksi dan mitigasi terhadap paket DOS yang masuk kedalam jaringan melalui switch.

5.2 Saran

Simulasi yang dibangun masih memiliki beberapa kekurangan, oleh sebab itu beberapa hal yang dapat dikembangkan dari peneliti selanjutnya maupun industry yaitu sebagai berikut.

- Dalam penelitian ini hanya menguji serangan *UDP Amplification*, namun tidak membatasi bahwa dengan menggunakan metode serupa dapat juga digunakan untuk menguji tipe serangan yang lain.
- Penelitian ini hanya bersifat simulasi, dalam pengembangan ke arah produksi perlu adanya pengembangan lebih lanjut.
- Mininet tidak mendukung multiple session dalam melakukan pengujian, sehingga pengujian potensi kesalahan (*false positive*) terhadap *traffic DNS legitimate* tidak dapat dilakukan.

DAFTAR REFERENSI

- [1] Afaq, M., Rehman, S., & Song, W.-C. (2015). Large Flows Detection, Marking, and Mitigation based on sFlow Standard in SDN. *Journal of Korea Multimedia Society*. <https://doi.org/10.9717/kmms.2015.18.2.189>
- [2] Nugraha, M., Paramita, I., Musa, A., Choi, D., & Cho, B. (2014). Utilizing OpenFlow and sFlow to Detect and Mitigate SYN Flooding Attack. *Journal of Korea Multimedia Society*. <https://doi.org/10.9717/kmms.2014.17.8.988>
- [3] Linuwih Anggita, B., Virgono, A., & Irawan, B. (2016). Perancangan Dan Analisis *Software Defined Network* Pada Jaringan Lan : Penerapan Dan Analisis Metode Penjaluran Path Calculating Menggunakan Algoritma Dijkstra. *E-Proceeding Of Engineering*.
- [4] S. M. Shamim, M. Badrul, A. Miah, A. Sarker, A. N. Bahar, and A. Sarker, "Simulation of Minimum Path Estimation in *Software Defined Networking* Using Mininet Emulator," *Br. J. Math. Comput. Sci.*, vol. 21, no. 3, pp. 1–8, 2017.
- [5] B. A. A. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, and T. Turetli, "A Survey of Software-Defined Networking : Past , Present , and Future of Programmable Networks," *IEEE Commun. Surv. TUTORIALS*, vol. 16, no. 3, pp. 1–18, 2014.
- [6] W. Braun and M. Menth, "Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices," *Futur. Internet*, vol. 6, no. 2, pp. 302–336, 2014
- [7] Fepiliana, "Penentuan Jalur Terpendek Menggunakan Algoritma Bellman-Ford Pada *Software Defined Network*" Universitas Sriwijaya, 2018.
- [8] Rekha P and Dakshayini M, "A Study of Software Defined Networking with OpenFlow," *Int. J. Comput. Appl.*, vol. 122, no. 5, pp. 5–12, 2015.
- [9] C. Pal, "Implementation of Simplified Custom Topology Framework in Mininet," *Asia-Pacific Conf. Comput. Aided Syst. Eng. Implement.*, pp. 48–53, 2014.
- [10] I. Z. Bholebawa and U. D. Dalal, "Performance Analysis of SDN / *OpenFlow Controllers* : POX Versus Floodlight," *Wirel. Pers. Commun.*, no. August, 2017.

- [11] F. S. Rechia, “An Evaluation of SDN Based Network Virtualization Techniques,” no. April, 2016.
- [12] Peter Phaal, (2018). “ONOS Measurement Based Control”. sFlow.
<https://sflow.com/>

Lampiran 1. Kode Program – Deteksi dan Mitigasi DoS

```
var user = 'username';
var password = 'password';
var onos = 'ip_controller';
var controls = {};

setFlow('udp_reflection',
  {keys:'ipdestination,udpsourceport',value:'frames'});
setThreshold('udp_reflection_attack',
  {metric:'udp_reflection',value:100,byFlow:true,timeout:2});

setEventHandler(function(evt) {
  // don't consider inter-switch links
  var link = topologyInterfaceToLink(evt.agent,evt.dataSource);
  if(link) return;

  // get port information
  var port = topologyInterfaceToPort(evt.agent,evt.dataSource);
  if(!port) return;

  // need OpenFlow info to create ONOS filtering rule
  if(!port.dpid || !port.ofport) return;

  // we already have a control for this flow
  if(controls[evt.flowKey]) return;

  var [ipdestination,udpsourceport] = evt.flowKey.split(',');
  var msg = {
    flows: [
      {
        priority:4000,
        timeout:0,
        isPermanent:true,
        deviceId:'of:'+port.dpid,
        treatment:[],
        selector: {
          criteria: [
            {type:'IN_PORT',port:port.ofport},
            {type:'ETH_TYPE',ethType:'0x800'},
            {type:'IPV4_DST',ip:ipdestination+'/32'},
            {type:'IP_PROTO',protocol:'17'},
            {type:'UDP_SRC',udpPort:udpsourceport}
          ]
        }
      }
    ]
  }
```

```

    ]
    };

    var resp = http2({
    url:'http://'+onos+'8181/onos/v1/flows?appId=ddos',
    headers: {'Content-Type':'application/json','Accept':'application/json'},
    operation:'post',
    user:user,
    password:password,
    body: JSON.stringify(msg)
    });

    var {deviceId,flowId} = JSON.parse(resp.body).flows[0];
    controls[evt.flowKey] = {
    time:Date.now(),
    threshold:evt.thresholdID,
    agent:evt.agent,
    metric:evt.dataSource+'.'+evt.metric,
    deviceId:deviceId,
    flowId:flowId
    };

    logInfo("blocking " + evt.flowKey);
    },['udp_reflection_attack']);

    setIntervalHandler(function() {
    var now = Date.now();
    for(var key in controls) {
    let rec = controls[key];

    // keep control for at least 10 seconds
    if(now - rec.time < 10000) continue;
    // keep control if threshold still triggered
    if(thresholdTriggered(rec.threshold,rec.agent,rec.metric,key)) continue;

    var resp = http2({
    url:'http://'+onos+'8181/onos/v1/flows/'
    +encodeURIComponent(rec.deviceId)+'/'+encodeURIComponent(rec.flowId),
    headers: {'Accept':'application/json'},
    operation:'delete',
    user:user,
    password:password
    });

    delete controls[key];

```

```
    logInfo("unblocking " + key);  
  }  
});
```

Lampiran 2. Daftar Riwayat Hidup

DATA PRIBADI

Nama : Randy Mukti

Jenis Kelamin : Laki-Laki

Tempat, Tanggal Lahir : Grobogan, 3 Januari 1996

Alamat Lengkap : Ds. Depok Dsn. Pengkol RT 4 RW 5, Kec. Toroh
Kab. Grobogan, Jawa Tengah 58171

Nomor Telepon : 085713750130

Email : randymukti@gmail.com

PENDIDIKAN FORMAL

2018 – 2021 : Universitas Esa Unggul (Transfer)

2013 – 2018 : STMIK Widuri

2013 – 2010 : SMK N 1 Purwodadi (Teknik Komputer dan Jaringan)

2007 – 2010 : SMP N 2 Toroh

2001 – 2007 : SD N 5 Tambirejo

RIWAYAT PEKERJAAN

2015 – 2021 (saat ini) : PT. Sisindokom Lintasbuana (Network Engineer)

2014 – 2015 : PT. Noosc Global (Information Security Analyst)