Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.202183367
Vol. 8, No. 6, Desember 2021, hlm. 1117-1126 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

PENGEMBANGAN APLIKASI PEMANTAUAN JARINGAN BERBASIS WEB PADA SOFTWARE-DEFINED NETWORKING DENGAN PROTOKOL SFLOW

Muhammad Ilham[1], Nur Rohman Rosyid[*2]

[1,2]Universitas Gadjah Mada, Yogyakarta
Email: [1]muhammad.ilham.ugm@mail.ugm.ac.id, [2]nrohmanr@ugm.ac.id
[*]Penulis Korespondensi

(Naskah masuk: 03 Juli 2020, diterima untuk diterbitkan: 15 November 2021)

Abstrak

Software-defined networking (SDN) merupakan salah satu arsitektur jaringan yang dapat diprogram untuk memudahkan manajemen jaringan menggunakan aplikasi pengontrol.. SDN *controller* seperti ONOS hanya berfungsi untuk manajemen *flow table* dan tidak memiliki fitur pemantauan *traffic* jaringan yang cukup untuk mendukung proses manajemen jaringan. Oleh karena itu, untuk melakukan pemantauan *traffic* SDN maka diperlukan protokol lain seperti sFlow. Pada penelitian ini dikembangkan aplikasi sistem pemantauan *traffic* SDN berbasis web dengan menggunakan ONOS sebagai SDN *controller*, sFlow-RT sebagai sFlow *collector,* dan Node.js sebagai web *server*. Hasil pengembangan aplikasi menghasilkan tiga buah fitur utama yaitu topologi, grafik *traffic*, dan laporan. Visualisasi topologi dibuat berdasarkan data topologi dari API ONOS dan ditampilkan menggunakan pustaka vis.js. Kemudian untuk grafik *throughput* dibuat berdasarkan data *traffic* dari API sFlow-RT dan ditampilkan menggunakan pustaka dygraph. Data topologi dan *traffic* yang tertampil pada aplikasi diperbarui setiap 10 detik. Pengujian aplikasi dilakukan dengan *black-box testing* menunjukkan bahwa semua fungsi pada aplikasi berhasil dilakukan. Hasil survei menunjukkan bahwa aplikasi memiliki tampilan informatif dan ramah pengguna, serta dapat memudahkan pemantauan *traffic* SDN.

Kata kunci: *software-defined networking, sflow, pemantauan jaringan, topologi jaringan, throughput jaringan*

DEVELOPMENT OF WEB-BASED SOFTWARE-DEFINED NETWORK MONITORING APPLICATION WITH SFLOW PROTOCOL

Abstract

Software-defined networking (SDN) is one of the network architectures which programmable to ease network management using the controller application. SDN controllers such as ONOS only function for flow table management and do not have enough network traffic monitoring features to support network management processes. Therefore, to monitor SDN traffic other protocols such as sFlow are needed. In this research, the web-based SDN traffic monitoring system application was developed by using ONOS as SDN controller, sFlow-RT as sFlow collector, and Node.js as a web server. The results of application development produce three main features namely topology, traffic graphs, and reports. The topology visualization is based on topology data from the ONOS API and is displayed using the vis.js library. Then the throughput graph is made based on data traffic from the sFlow-RT API and displayed using the dygraph library. Topology and traffic data displayed on the application are updated every 10 seconds. Application testing is done with black-box testing showing that all functions and features of the application can function properly. The survey conducted shows that the application has an informative and user-friendly display, and can facilitate monitoring of SDN traffic.

Keywords: *software-defined networking, sflow, network monitoring, network topology, network throughput*

1. PENDAHULUAN

Teknologi *Software-Defined Network* (SDN) merupakan sebuah konsep pendekatan baru untuk mendesain, mengelola, dan mengimplementasikan arsitektur jaringan yang memisahkan antara sistem kontrol (*control plane*) dan sistem *forwarding* (*data plane*) pada perangkat jaringan (Mulyana, 2015). Pada SDN dibutuhkan sebuah *controller* yang bertindak sebagai pengendali perangkat-perangkat yang ada pada jaringan, termasuk diantaranya mendefinisikan jaringan serta mengatur proses *routing* dan *forwarding*.

Pada jaringan komputer diperlukan suatu sistem pemantauan yang baik untuk mendukung proses manajemen jaringan. Pemantauan jaringan merupakan kegiatan memantau jaringan secara terus menerus dan menyediakan notifikasi kepada pengelola jaringan ketika ada sebuah elemen dalam jaringan yang gagal (Adato dan Hale, 2016). Beberapa *controller* SDN seperti OpenDaylight, Floodlight, dan ONOS hanya memiliki fungsi dasar untuk pengaturan *flow* dan tidak mempunyai fitur pemantauan *traffic* untuk mendukung proses manajemen jaringan. Hal tersebut dikarenakan protokol OpenFlow yang merupakan protokol standar yang digunakan untuk komunikasi antar perangkat SDN seperti OpenvSwitch dan *controller*, tidak mempunyai fitur pemantauan *traffic* karena OpenFlow berfokus pada pengaturan *flow* pada setiap perangkat. Pembuatan aplikasi pemantauan *traffic* pada SDN sebelumnya pernah dilakukan menggunakan Floodlight *controller* dengan hanya memanfaatkan protokol OpenFlow dengan cara membuat beberapa modul Floodlight tambahan, kemudian dari modul-modul tersebut dapat dibuat sebuah aplikasi berbasis web dengan bahasa pemrograman python yang dapat digunakan untuk memantau *traffic* pada *control plane* (Isolani et al., 2015), bisa juga dibuat sebuah aplikasi pemantauan berbasis *desktop* menggunakan pemrograman java untuk memantau *traffic* hingga tingkat *flow* (Hartung dan Körner, 2017). Selain itu, pembuatan aplikasi berbasis web untuk menampilkan statistik jaringan pernah dibuat menggunakan Ryu *controller* pada jaringan mininet (Bertier et al., 2016), tetapi karena keterbatasan data dari *controller* maka pada aplikasi tersebut dibuat sebuah modul tingkat rendah secara terpisah untuk melengkapi informasi jaringan yang belum tersedia pada *controller* tersebut.

Hingga saat ini tidak ada protokol khusus untuk pemantauan SDN karena protokol standar yang digunakan pada jaringan konvensional masih dapat digunakan pada SDN dengan baik. Pemantauan jaringan saat ini biasanya menggunakan protokol SNMP. Protokol SNMP memiliki kekurangan yaitu tidak dapat memantau *traffic flow*. Selain itu, SNMP tidak berfokus pada pemantauan metrik perangkat saja, tetapi juga konfigurasi perangkat. SNMP juga tidak tersedia pada perangkat OpenvSwitch. Oleh karena itu, pembuatan sistem pemantauan SDN sebaiknya menggunakan protokol khusus pemantauan jaringan seperti Netflow atau sFlow. Dibanding Netflow, sFlow dikembangkan secara eksklusif sebagai teknologi pemantauan perangkat jaringan. sFlow menawarkan skalabilitas dan detail pelaporan yang lebih besar, memberikan informasi yang rinci, *real-time*, dari L2 hingga L7 tentang *traffic* di seluruh jaringan (InMon, 2019). Pada protokol sFlow, sebuah aplikasi sFlow *collector* pada komputer server dibutuhkan untuk menerima sFlow *datagram* dari perangkat sFlow *agent* seperti OpenvSwitch. Salah satu aplikasi sFlow *collector* yang dapat digunakan yaitu sFlow-RT. Aplikasi sFlow-RT bukanlah aplikasi yang siap digunakan untuk pemantauan jaringan melainkan menyediakan *Application Programming Interfaces* (API) yang dapat dimanfaatkan untuk mengembangkan aplikasi pemantauan sesuai kebutuhan.

Sebelumnya penerapan sFlow pada SDN sudah pernah dilakukan pada jaringan OF@TEIN menggunakan Floodlight *controller* (Rehman, Song dan Kang, 2014), tetapi data dari sFlow-RT tidak dikembangkan menjadi sebuah aplikasi independen melainkan data tersebut disimpan ke dalam *database* Whisper RRD kemudian data sFlow ditampilkan dalam bentuk grafik menggunakan aplikasi graphite.

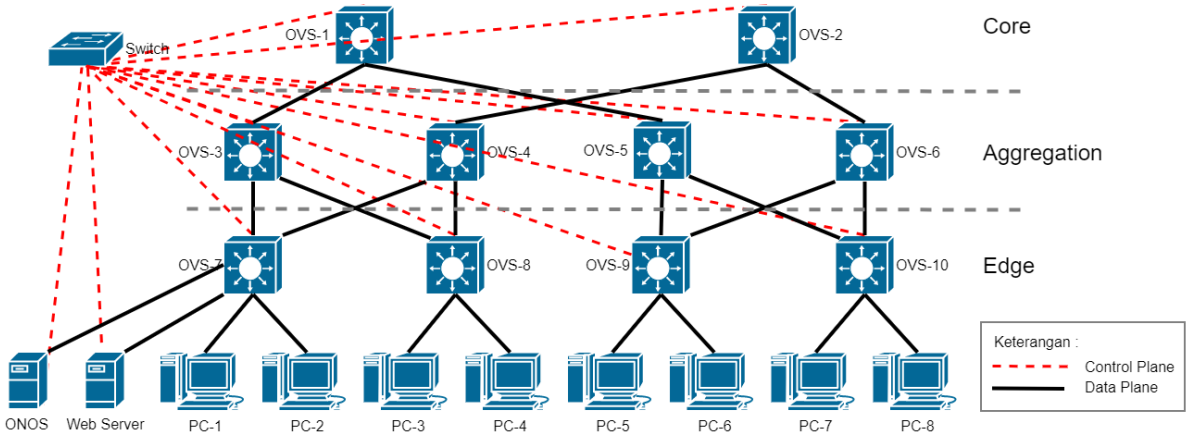
Setelah pembahasan singkat terkait pemantauan jaringan dan SDN, pada penelitian ini dibuatlah sebuah aplikasi pemantauan SDN berbasis web dengan tiga fitur utama yaitu menampilkan topologi, *throughput* perangkat, dan laporan jaringan. Pada penelitan terdahulu sudah pernah dilakukan pembuatan visualisasi topologi dengan membuat modul untuk POX *controller* pada jaringan mininet (Pantuza et al., 2014), tetapi visualisasi tersebut harus dilakukan secara manual dengan mengambil data dari modul POX yang dibuat, kemudian digunakan dimasukkan ke aplikasi graphviz. Oleh karena itu, pada aplikasi penelitian ini berinovasi dengan membuat fitur topologi yang dapat muncul secara otomatis dengan menggunakan data dari ONOS. Kemudian fitur *throughput* perangkat dan laporan jaringan dibuat berdasarkan data dari sFlow-RT yang diperbarui secara berkala.

2. PENGEMBANGAN SISTEM

Sistem aplikasi ini dikembangkan menggunakan metode *Rapid Application Development* (RAD). RAD melakukan pendekatan terhadap proses pengembangan perangkat lunak dengan mengurangi penekanan pada tahap perencanaan dan meningkatkan penekanan pada tahap pengembangan perangkat lunak. Pada tahap pengembangan sistem dilakukan perencanaan alat dan bahan yang digunakan, perancangan topologi jaringan, identifikasi persyaratan aplikasi, perancangan model aplikasi, kemudian dilakukan pembuatan aplikasi. Tetapi sebelumnya, terlebih dahulu dilakukan perencanaan alat dan bahan serta topologi jaringan yang digunakan.

2.1. Alat dan Bahan

Penelitian ini dilakukan di dalam simulasi sehingga diperlukan sebuah perangkat komputer dengan spesifikasi RAM 12 GB. Perangkat lunak yang digunakan dibagi menjadi dua jenis yaitu perangkat lunak untuk simulasi jaringan dan perangkat lunak untuk pengembangan aplikasi. Perangkat lunak yang digunakan untuk simulasi jaringan yaitu GNS3, OpenvSwitch, ipterm, vmware



Gambar 1. Topologi Jaringan Penelitian

Ubuntu Server, ONOS, sFlow-RT, dan Iperf3. Kemudian perangkat lunak yang digunakan untuk pengembangan aplikasi yaitu Node.js, InfluxDB, serta pustaka javascript express, axios, vis, dygraph, bootstrap, dan jquery.

2.2. Perancangan Topologi SDN

Bentuk topologi jaringan yang digunakan pada penelitian ini tidak berpengaruh terhadap pengembangan aplikasi sehingga bentuk topologi apapun dapat digunakan, tetapi pada penelitian ini menggunakan *fat-tree topology* yaitu topologi jaringan khusus yang digunakan pada *data center* (Al-Fares, Loukissas dan Vahdat, 2008). Topologi *data center* dipilih karena teknologi SDN banyak diterapkan pada *data center* (Kanagavelu dan Aung, 2015).

Jaringan yang digunakan untuk penelitian disimulasikan pada aplikasi GNS3. Perangkat SDN yang digunakan adalah OpenvSwitch dan ONOS *controller*. Setiap port manajemen OpenvSwitch terhubung langsung ke ONOS seperti pada Gambar 1 dengan garis putus-putus untuk membentuk jaringan dengan mode *out-of-band* sehingga jaringan *control plane* yang digunakan untuk mengontrol *switch* terpisah dengan jaringan *data plane* yang digunakan untuk jalur *traffic* data antar perangkat seperti yang ditunjukkan dengan garis tegas pada Gambar 1.

Pada ONOS *controller* terdapat dua aplikasi yang berjalan yaitu org.onosproject.openflow sebagai *driver* OpenFlow dan org.onosproject.fwd sebagai pengontrol tabel *flow* untuk membuat setiap perangkat OpenvSwitch berperilaku seperti L2 *learning switch.*

2.3. Identifikasi Kebutuhan Aplikasi

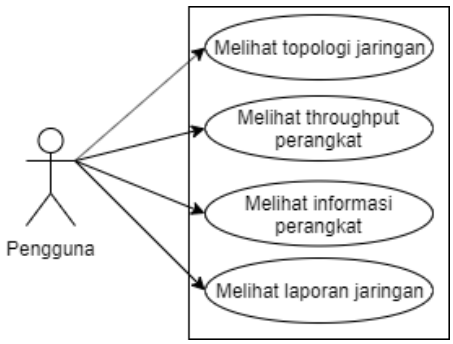
Identifikasi kebutuhan aplikasi dilakukan dengan cara mencari tahu kebutuhan aplikasi secara fungsional maupun non-fungsional. Kebutuhan fungsional berupa proses atau layanan yang harus disediakan oleh sistem. Berikut adalah kebutuhan fungsional pada sistem:

- Sistem dapat menampilkan topologi SDN
- Tata letak visualisasi topologi dapat dirubah pada sistem
- Sistem dapat menampilkan informasi perangkat switch seperti datapath id dan alamat ip
- Label nama perangkat pada sistem dapat dirubah secara visual
- Sistem dapat menggambarkan perubahan throughput antar perangkat pada topologi
- Sistem dapat menampilkan grafik throughput perangkat switch
- Sistem dapat mencetak laporan throughput perangkat berdasarkan periode yang ditentukan
- Sistem dapat menyimpan data throughput setiap perangkat pada database

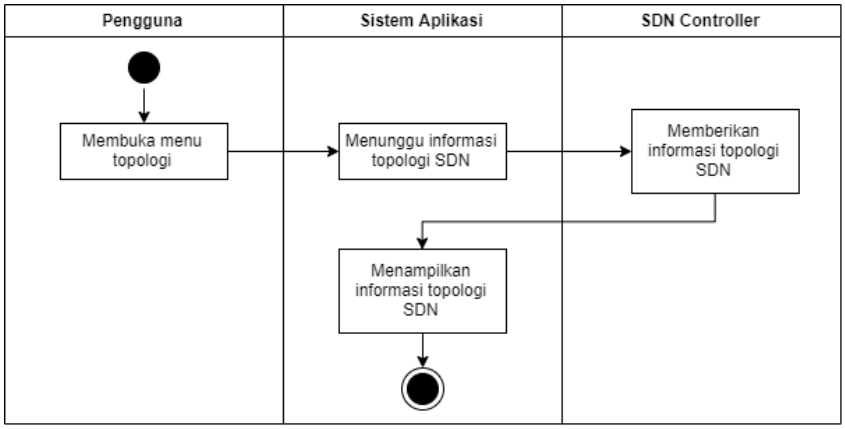
Selanjutnya, kebutuhan non-fungsional berupa properti perilaku yang disediakan oleh sistem. Berikut ini kebutuhan non-fungsional pada sistem:

- Sistem aplikasi memiliki antarmuka yang baik sehingga dapat memudahkan pengguna dalam menggunakan aplikasi
- Pengguna aplikasi dapat memahami tulisan dan grafik yang ada pada sistem dengan mudah.

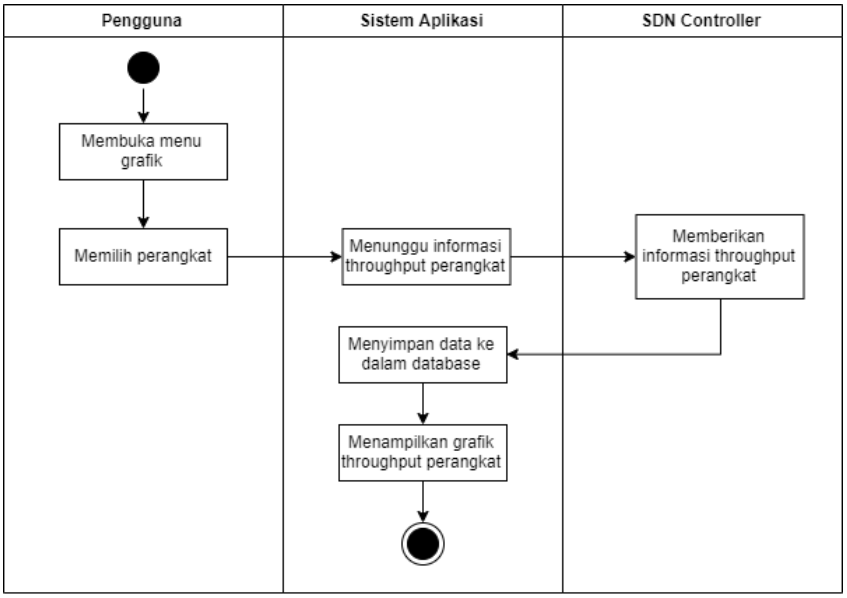
2.4. Perancangan Model



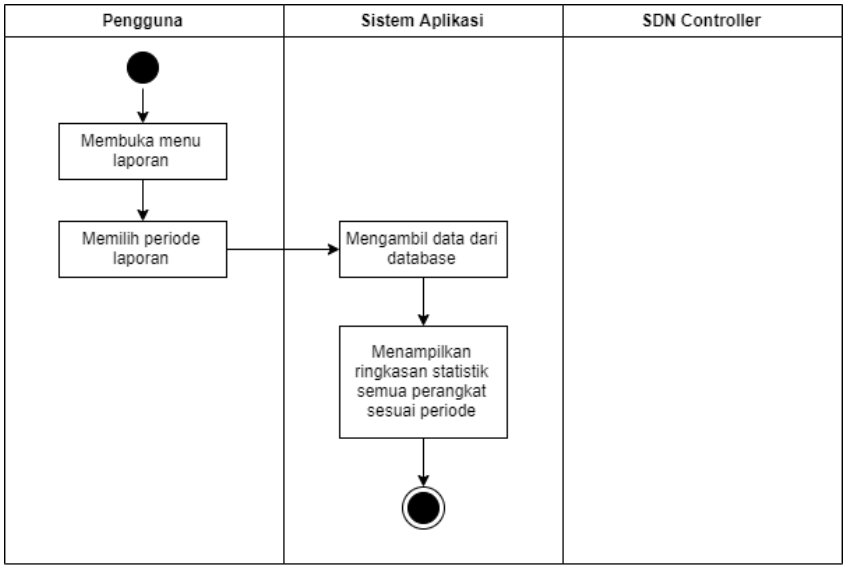
Gambar 2. *Use Case Diagram* Aplikasi Pemantauan



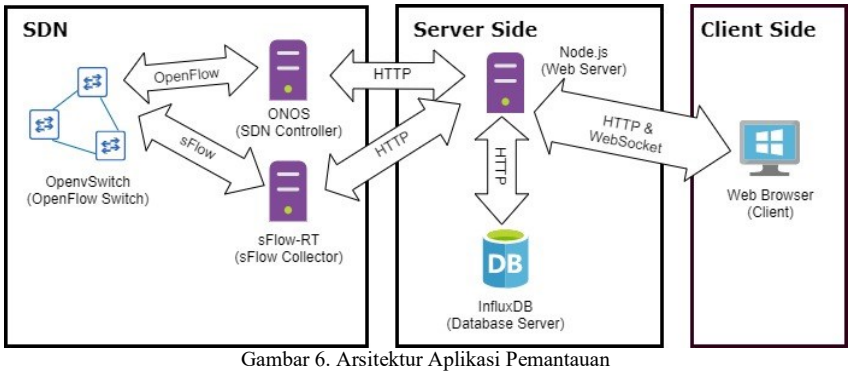
Gambar 3. *Activity Diagram* untuk Menu Topologi



Gambar 4. *Activity Diagram* untuk Menu Grafik



Gambar 5. *Activity Diagram* untuk Menu Laporan



Gambar 6. Arsitektur Aplikasi Pemantauan

Perancangan model dilakukan dengan menggunakan diagram UML untuk menggambarkan konsep aplikasi. Diagram UML yang digunakan yaitu *use case* diagram dan *activity* diagram. Pada penelitian ini terdapat empat *use case* dan satu aktor yang dapat dilihat pada Gambar 2. Kebutuhan aplikasi dapat dikelompokkan menjadi tiga menu aplikasi yaitu menu topologi, grafik, dan laporan. Kemudian setiap menu aplikasi dibuat *activity diagram* untuk menggambarkan aktivitas yang dilakukan pada menu tersebut. *Activity diagram* ini dibagi menjadi tiga sisi yaitu sisi pengguna, sistem aplikasi dan SDN *controller*. *Activity diagram* untuk menu topologi, grafik, dan laporan secara berturut-turut terlihat pada Gambar 3, Gambar 4, dan Gambar 5.

2.5. Pembuatan Aplikasi

Aplikasi dibuat berbasis web menggunakan bahasa javascript dengan *server* Node.js. Arsitektur aplikasi yang dibuat dapat dilihat pada Gambar 6. Pada Gambar 6 dibagi menjadi tiga komponen yaitu SDN, *server*, dan *client*. Pada komponen SDN merupakan bagian dimana jaringan SDN berjalan dan dipantau, disini terdapat perangkat OpenvSwitch, ONOS, dan sFlow-RT. Kemudian pada komponen *server* merupakan bagian dimana aplikasi dikembangkan, disini terdapat web *server* Node.js dan basis data influxDB. Lalu komponen terakhir adalah *client* yang mengakses aplikasi web, bagian ini hanya terdiri dari web *browser*. Pada setiap bagian terdapat panah dengan dua arah yang menunjukkan protokol yang digunakan untuk berkomunikasi antar perangkat, secara ringkas terdapat empat protokol yang digunakan yaitu OpenFlow, sFlow, HTTP, dan WebSocket. Sistem aplikasi yang dikembangkan tidak mengakses data dari protokol sFlow dan OpenFlow secara langsung, tetapi mengaksesnya menggunakan protokol HTTP melalui API sFlow-RT dan ONOS secara berturut-turut. Data dari kedua API tersebut diolah kemudian ditampilkan dalam bentuk visualisasi topologi SDN dan *link* secara dinamis. Selain itu, data dari sFlow-RT juga disimpan ke dalam *database* influxDB agar ada sejarah pencatatan statistik perangkat untuk pengolahan data kedepannya. Basis data influxDB dipilih karena influxDB merupakan *time-series database* yang cocok untuk menyimpan data metrik sFlow yang diambil setiap interval waktu tertentu. Setelah data diolah oleh aplikasi, data tersebut akan dikirim ke *client* yang sedang mengakses aplikasi web menggunakan websocket sehingga data pada aplikasi *client* selalu diperbarui secara berkala tanpa harus membuka ulang aplikasi.

2.6. Skenario Pengujian Aplikasi

Aplikasi diuji dengan metode *Black-Box Testing* untuk menguji kebutuhan fungsional aplikasi. Survei dilakukan untuk menguji kebutuhan non-fungsional aplikasi. Kuesioner untuk survei dibuat menggunakan skala likert dan diberikan kepada 30 orang responden.

3. PENGAMBILAN DATA

Pengambilan data dilakukan sebagai acuan pembuatan aplikasi. Data diambil dari dua sumber yaitu ONOS API dan sFlow-RT API. Data dari ONOS API digunakan untuk membuat visualisasi topologi pada menu topologi yang terdiri dari *nodes* dan *links*. *Nodes* terdiri dari perangkat OpenvSwitch dan *host*, kemudian *links* merupakan penghubung antar *nodes*. Data dari sFlow-RT digunakan untuk mendapatkan nilai *throughput* dari perangkat yang digunakan sebagai acuan pemberian warna garis *link* pada visualisasi topologi serta pembuatan grafik *throughput* setiap perangkat pada menu grafik. Selain itu, data dari sFlow-RT API disimpan ke basis data influxDB untuk pencatatan.

3.1. Pengambilan Data dari ONOS API

Pengambilan data dari ONOS API dilakukan untuk membuat visualisasi topologi pada menu topologi aplikasi penelitian. Pengambilan data dari ONOS API dilakukan setiap 5 detik sekali dikarenakan pengaturan bawaan ONOS hanya memberikan data yang diperbarui setiap 5 detik. Alamat untuk mengakses REST API milik ONOS memiliki awalan "http://IP_ADDRESS:8181/onos/v1/". Ada tiga ONOS API yang digunakan pada penelitian ini yaitu *devices*, *links*, dan *hosts*. Ketiga API tersebut diakses menggunakan HTTP *Request* GET.

Devices merupakan API yang digunakan untuk mengambil data daftar *switch* yang terhubung beserta informasinya. Data dari pemanggilan API ini digunakan untuk membuat daftar *nodes* pada aplikasi untuk mengetahui semua perangkat *switch* yang terhubung dengan ONOS beserta informasi terkait perangkat tersebut yang nantinya digunakan untuk visualisasi topologi pada menu topologi aplikasi. Pemanggilan *devices* API ini menghasilkan data JSON yang berisi himpunan objek perangkat. Objek perangkat memiliki banyak atribut, tetapi atribut yang penting hanya ada dua yaitu "id" yang menunjukkan *datapath* ID OpenFlow pada OpenvSwitch dan "annotations.managementAddress" yang menunjukkan alamat IP OpenvSwitch. Contoh salah satu isi data objek perangkat yaitu terlihat pada Gambar 7.


```
{
  "id": "of:00000ae9bc58464f",
  "type": "SWITCH",
  "available": true,
  "role": "MASTER",
  "mfr": "Nicira, Inc.",
  "hw": "Open vSwitch",
  "sw": "2.4.0",
  "serial": "None",
  "driver": "ovs",
  "chassisId": "ae9bc58464f",
  "lastUpdate": "1581405087347",
  "humanReadableLastUpdate": "connected 8m44s ago",
  "annotations": {
    "channelId": "192.168.111.18:33830",
    "managementAddress": "192.168.111.18",
    "protocol": "OF_13"
  }
}
```


Gambar 7. Struktur Data Objek Perangkat

Selanjutnya *links* API, *links* merupakan API yang digunakan untuk mengambil data daftar jalur koneksi antar *switch*. Data dari pemanggilan API ini digunakan untuk membuat daftar *links* antar perangkat pada aplikasi untuk visualisasi topologi. Pemanggilan API ini menghasilkan data JSON yang berisi himpunan objek *link*. Objek *link* memiliki beberapa atribut, atribut yang digunakan pada penelitian ini adalah src.port, src.device, dst.port, dan dst.device. Objek *link* ini bermanfaat untuk memetakan topologi perangkat OpenvSwitch secara otomatis. Contoh salah satu isi data objek *link* seperti pada Gambar 8.


```
{
  "src": {
    "port": "3",
    "device": "of:00005a9c31079543"
  },
  "dst": {
    "port": "1",
    "device": "of:0000d6d0f98d2a4b"
  },
  "type": "DIRECT",
  "state": "ACTIVE"
}
```


Gambar 8. Struktur Data Objek *Link*

ONOS API terakhir yang digunakan adalah *hosts* API. *Hosts* API merupakan API yang digunakan untuk mengambil data daftar *host* yang terdeteksi oleh ONOS. Data dari pemanggilan API ini digunakan untuk membuat daftar *hosts* dan *nodes* pada aplikasi untuk mengetahui semua perangkat *host* yang terdeteksi oleh ONOS beserta informasi terkait perangkat tersebut yang nantinya digunakan untuk visualisasi topologi pada menu topologi aplikasi. Pemanggilan API ini menghasilkan data JSON yang berisi himpunan objek *host*. Objek *host* memiliki beberapa atribut. Atribut yang penting disini yaitu mac dan ipAddresses untuk identifikasi *host* serta locations.elementID dan locations.port untuk menentukan letak *host* pada topologi. Contoh salah satu isi data objek *host* yaitu terlihat pada Gambar 9.


```
{
  "id": "C6:58:72:64:84:E8/None",
  "mac": "C6:58:72:64:84:E8",
  "vlan": "None",
  "innerVlan": "None",
  "outerTpid": "unknown",
  "configured": false,
  "ipAddresses": [
    "10.10.10.17"
  ],
  "locations": [{
    "elementId": "of:0000d6d0f98d2a4b",
    "port": "3"
  }]
}
```

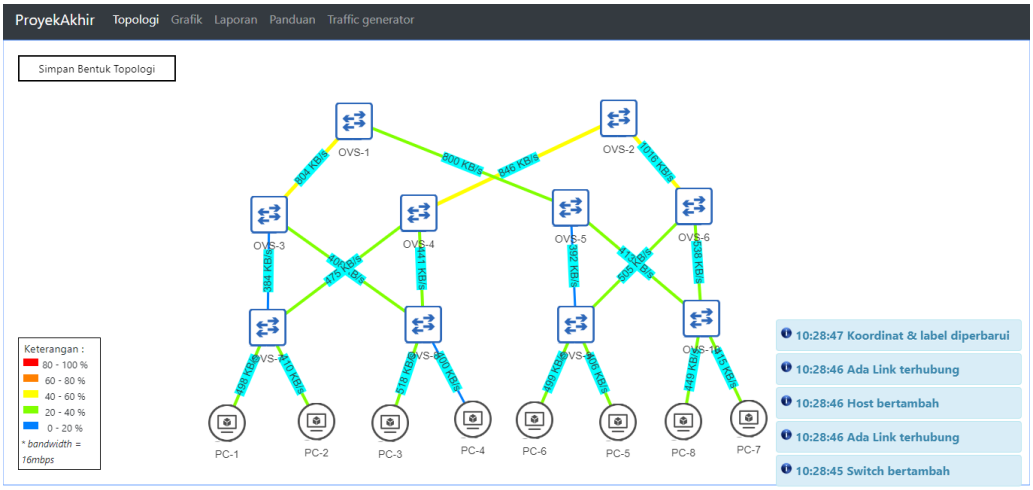

Gambar 9. Struktur Data Objek *Host*

3.2. Pengambilan Data dari sFlow-RT API

Data dari protokol sFlow yang sudah diproses secara otomatis oleh sFlow-RT dapat diakses menggunakan REST API milik sFlow-RT. Data dari pemanggilan sFlow-RT API digunakan untuk membuat grafik *throughput* pada menu grafik dan visualisasi topologi pada menu topologi. Sistem aplikasi pemantauan pada penelitian ini menggunakan pustaka javascript axios untuk menggunakan REST API tersebut. Pengaksesan data sFlow dilakukan setiap 10 detik sekali karena pengaturan bawaan sFlow-RT hanya memberikan data baru setiap 10 detik. Alamat untuk mengakses REST API sFlow-RT memiliki awalan "*http://IP_ADDRESS:8008/*". API yang digunakan pada penelitian ini yaitu *metrics* API yang berfungsi untuk mengambil data metrik semua *interfaces* pada setiap perangkat. API diakses dengan HTTP *Request* GET ke alamat "*/table/{agent}/{metric}/json*". Parameter *agent* diisi dengan nilai *ALL* yang berarti semua agen dipilih, lalu untuk parameter *metric* diisi dengan "*of_dpid, of_port, ifinutilization, ifoututilization, ifinoctets, ifoutoctets, ifspeed*" secara berturut turut merupakan OpenFlow *datapath* ID, OpenFlow *port*, utilisasi masuk *interface,* utilisasi keluar *interface*, jumlah byte masuk ke *interface* setiap detik, jumlah byte keluar setiap detik, dan kecepatan *interface*.


```
{
  "agent": "192.168.111.11",
  "metricName": "of_dpid",
  "lastUpdate": 8129,
  "metricValue": "0000baf78681d142",
  "dataSource": "44"
}
```


Gambar 10. Struktur Data Objek Metrik



Gambar 11. Tampilan Antarmuka untuk Menu Topologi

Pemanggilan API ini menghasilkan data JSON yang berisi himpunan dari himpunan objek metrik yang dikelompokkan berdasarkan *data source* yang sama. Contoh salah satu isi data objek metrik dapat dilihat pada Gambar 10 yang merupakan contoh objek metrik untuk parameter of_dpid.

3.3. Penerapan Basis Data InfluxDB

Basis data influxDB digunakan untuk menyimpan semua data dari sFlow *metrics* yang diambil. Pada aplikasi penelitian, basis data ini digunakan untuk membuat grafik pada menu grafik dan menu laporan aplikasi. InfluxDB dipilih karena merupakan *time-series database* yang cocok untuk menyimpan data dengan interval tertentu. Pada influxDB, *measurements* (seperti istilah tabel pada basis data relasional) tidak perlu dibuat terlebih dahulu, hanya *database* yang perlu dibuat terlebih dahulu, sehingga data langsung dimasukkan ke dalam *database*. Data dimasukkan secara otomatis oleh sistem aplikasi dengan pustaka javascript bernama influx setiap pengambilan data dari sFlow-RT API dilakukan. Data metrik of_dpid, of_port, ip, dan ifspeed dimasukkan dengan tipe *tags* pada influxDB untuk memberikan identitas pada data dan agar terindeks sehingga dapat dilakukan *query* dengan cepat. Kemudian data metrik ifinutilization, ifoututilization, ifinoctets, dan ifoutoctets dimasukkan menjadi tipe *fields* pada *database*.

4. HASIL DAN PEMBAHASAN

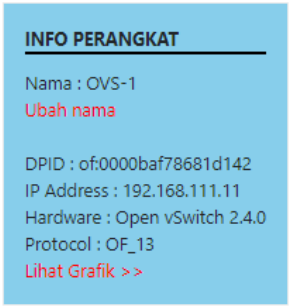
Aplikasi yang dikembangkan menghasilkan tiga menu utama yaitu topologi, grafik, dan laporan. Selain itu juga dibuat menu *traffic generator* untuk membantu pengujian aplikasi.

4.1. Menu Topologi

Menu topologi terletak pada halaman utama aplikasi. Pada halaman ini terdapat visualisasi topologi dan informasi *throughput* setiap perangkat dengan tampilan yang diperbarui secara berkala setiap 10 detik sesuai interval yang telah ditentukan pada ONOS, sFlow, dan aplikasi. Data topologi diambil dari kombinasi API *hosts*, *devices*, dan *links* milik ONOS. Sedangkan data *throughput* diambil dari API *metrics* milik sFlow-RT. Menu topologi digunakan untuk menampilkan visualisasi topologi, visualisasi *throughput* perangkat dan menampilkan informasi perangkat. Tampilan visualisasi topologi dibuat menggunakan pustaka javascript vis.js. Tampilan keseluruhan menu topologi dapat dilihat pada Gambar 11.

Saat aplikasi pertama kali dijalankan, tata letak topologi akan muncul secara otomatis dan tata letaknya diatur secara acak, untuk merapikannya pengguna dapat mengklik dan menggeser gambar perangkat secara manual kemudian klik tombol "Simpan Bentuk Topologi" di atas kiri halaman untuk menyimpannya agar setiap aplikasi dibuka tata letak visualisasi topologi tetap sama. Selain itu, nama label perangkat juga masih menggunakan DPID, untuk merubahnya dilakukan dengan cara mengklik gambar perangkat kemudian klik ubah nama pada panel info yang muncul di kanan halaman.

Pada menu topologi juga terdapat informasi perangkat yang dapat diakses dengan mengklik gambar perangkat yang ada kemudian nanti akan muncul panel disebelah kanan layar. Jika perangkat switch diklik maka akan menampilkan nama label *switch*, DPID, IP *address*, tipe *hardware*, dan versi protokol seperti pada Gambar 12.



Gambar 12. Tampilan Antarmuka untuk Panel Informasi

Pada panel informasi *switch* terdapat link berwarna merah untuk mengubah label nama serta link "Lihat Grafik" untuk melihat grafik *throughput* perangkat. Perubahan nama hanya sekedar tampilan untuk memudahkan pengguna dalam mengidentifikasi perangkat pada aplikasi, perubahan nama tidak merubah hostname pada perangkat aslinya.

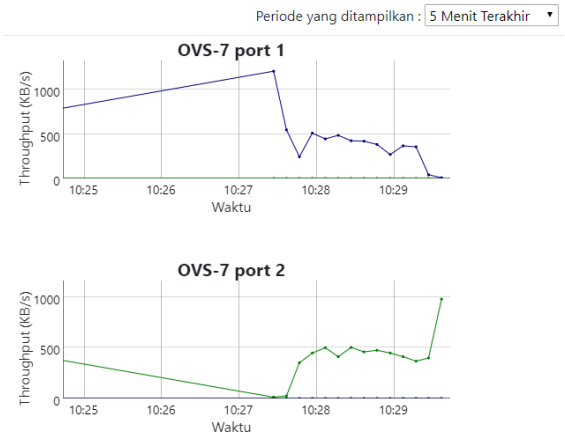
Warna *link* antar perangkat pada visualisasi topologi menyesuaikan utilisasi *port interface* salah satu dari dua perangkat *switch* yang saling terhubung. Nilai utilisasi ini dihitung dengan rumus pada persamaan (1).

$$Utilisasi = \frac{ifinoctets + ifoutoctets}{(bandwidth/8)} \times 100\% \qquad (1)$$

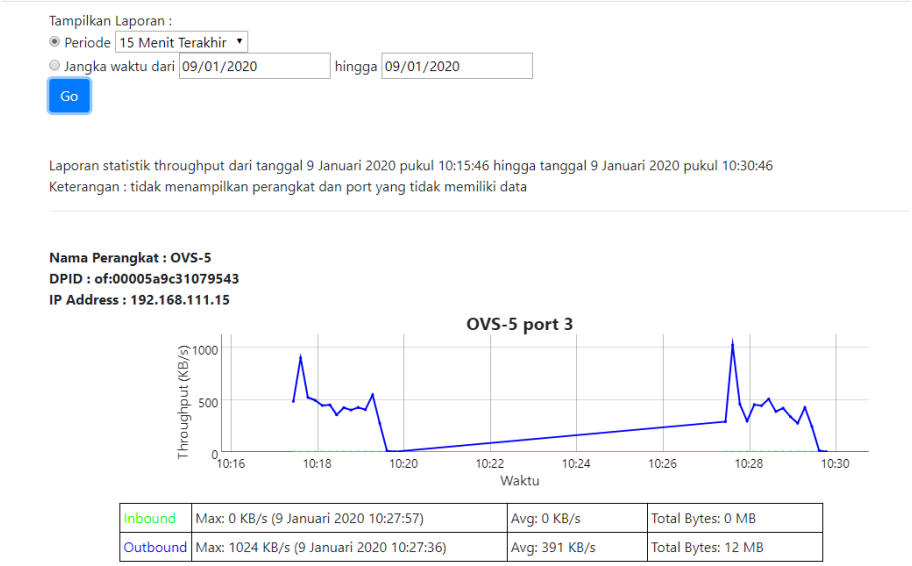
Pada persamaan (1), nilai ifinoctets dan ifoutoctets merupakan nilai *throughput* yang didapatkan dari *metrics* API milik sFlow-RT, sedangkan *bandwidth* merupakan nilai yang diatur pada kode aplikasi. sFlow-RT sebenarnya sudah menyediakan data utilisasi dalam persen, tetapi pada simulasi GNS3 nilai utilisasi tersebut tidak akurat dan nilai *bandwidth* yang diketahui sFlow adalah 10 mbps yang diketahui dengan memeriksa API *metrics* pada parameter ifspeed berbeda dengan *bandwidth interface* pada simulasi GNS3 yaitu 1 gbps.

4.2. Menu Grafik

Menu grafik digunakan untuk menampilkan grafik *throughput* dalam satuan kilobyte per detik dari semua *port* yang aktif pada perangkat yang dipilih. Grafik diperbarui setiap 10 detik berdasarkan interval sFlow yang ditentukan pada perangkat sFlow *agent* dan pada kode aplikasi web pemantauan. Tampilan grafik dibuat menggunakan pustaka javascript dygraph. Garis pada grafik ini ada dua warna yaitu hijau untuk *traffic inbound* dan biru untuk *traffic outbound*. Tampilan menu grafik terlihat seperti pada Gambar 13. Grafik dapat diatur agar menampilkan grafik dari waktu tertentu hingga waktu sekarang sesuai waktu yang dipilih pada bagian *dropdown* "Periode yang ditampilkan" yang memiliki rentang nilai dari yang terkecil 5 menit terakhir hingga nilai terbesar yaitu 1 tahun terakhir.



Gambar 13. Tampilan Antarmuka untuk Menu Grafik



Gambar 14. Tampilan Antarmuka untuk Menu Laporan

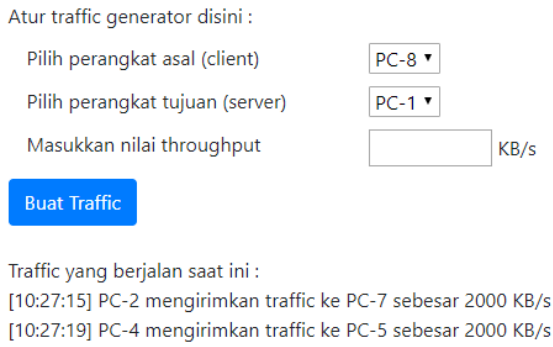
4.3. Menu Laporan

Menu laporan digunakan untuk menampilkan informasi ringkasan jaringan seperti jumlah perangkat terhubung, *link* antar perangkat, dan grafik *interface* perangkat. Grafik akan muncul di bawah halaman setelah pengguna memilih periode waktu grafik yang akan ditampilkan lalu menekan tombol "Go". Pada bagian bawah grafik terdapat info *throughput* maksimum, rata-rata, dan total *throughput* pada rentang waktu yang dipilih.

Tampilan menu laporan terlihat seperti pada Gambar 14.

4.4. Menu *Traffic Generator*

Menu *traffic generator* merupakan menu tambahan untuk memudahkan mensimulasikan *traffic* pada simulasi jaringan GNS3. *Traffic generator* ini dihasilkan dengan dengan menjalankan perintah "iperf3 -c IP_SERVER -b THROUGHPUT -t 120" pada PC *client*. Parameter IP_SERVER merupakan IP *address* iperf3 *server*. Parameter THROUGHPUT merupakan nilai *throughput* yang dimasukkan pada web. Kemudian parameter -t 120 berarti iperf3 akan berjalan selama 120 detik atau 2 menit. Perintah iperf3 akan dikirimkan oleh web *server* ke iperf3 *client* dengan telnet. Tampilan menu *traffic generator* ada pada Gambar 15.



Gambar 15. Tampilan Antarmuka untuk Menu *Traffic Generator*

4.5. Hasil Pengujian

Metode *Black-Box Testing* yang digunakan pada penelitian ini hanya berfokus pada pengujian fungsionalitas aplikasi, selain itu untuk menguji non-fungsionalitas aplikasi digunakan survei.

Black-Box Testing dilakukan dengan membuat tabel daftar pengujian aplikasi yang berisi daftar kasus pengujian, kemudian diisi apakah kasus tersebut berhasil atau tidak. Hasil pengujian sistem aplikasi ini yaitu seluruh fungsionalitas aplikasi dapat berfungis dengan baik seperti yang terlihat pada Tabel 1.

Survei dilakukan dengan memberikan kuesioner kepada 30 orang responden. Dari survei ini mendapatkan hasil setuju bahwa aplikasi memiliki tampilan informatif dan ramah pengguna, dan sangat setuju aplikasi dapat memudahkan pemantauan SDN.

Tabel 1. Hasil *Black-Box Testing*

No.	Menu Aplikasi	Kasus Pengujian	Hasil
1	Topologi	Menampilkan topologi jaringan SDN	Berhasil
2	Topologi	Merubah tata letak visualisasi topologi	Berhasil
3	Topologi	Menampilkan informasi perangkat	Berhasil
4	Topologi	Merubah nama perangkat	Berhasil
5	Topologi	Menampilkan perubahan warna link antar perangkat sesuai *throughput*	Berhasil
6	Grafik	Menampilkan grafik *throughput* perangkat	Berhasil
7	Laporan	Menampilkan laporan jaringan pada periode tertentu	Berhasil

5. KESIMPULAN

Pengembangan aplikasi menghasilkan tiga buah menu utama yaitu topologi, grafik, dan laporan. Pada menu tersebut dapat dilakukan pemantauan jaringan dalam bentuk visualisasi topologi dan grafik *throughput* perangkat. Hasil *Black-Box Testing* menunjukkan bahwa pengujian fungsionalitas pada aplikasi berhasil dilakukan. Hasil survei menunjukkan bahwa aplikasi memiliki tampilan informatif dan ramah pengguna, serta dapat memudahkan pemantauan SDN.

DAFTAR PUSTAKA

ADATO, L. DAN HALE, B., 2016. *Network Monitoring For Dummies*. SolarWinds ed. New Jersey: John Wiley & Sons, Inc.

AL-FARES, M., LOUKISSAS, A. DAN VAHDAT, A., 2008. A Scalable , Commodity Data Center Network Architecture. *SIGCOMM Comput. Commun. Rev.*, hal.63–74.

BERTIER, C., PORTELLI, R., STAGKOPOULOU, A., NIKAM, V., NEROUTSOS, E. DAN ZHANG, D., 2016. *Final Report -- Distributed Monitoring in SDN*. Stockholm.

HARTUNG, M. DAN KÖRNER, M., 2017. SOFTmon - Traffic Monitoring for SDN. In: *Procedia Computer Science*. [daring] Elsevier.hal.516–523. Tersedia pada: <https://www.sciencedirect.com/science/article/pii/S1877050917313236> [Diakses 7 Jul 2019].

INMON, 2019. *InMon: sFlow*. [daring] Tersedia pada: <https://inmon.com/technology/> [Diakses 12 Agu 2019].

ISOLANI, P.H., WICKBOLDT, J.A., BOTH, C.B., ROCHOL, J. DAN GRANVILLE, L.Z., 2015. Interactive monitoring, visualization, and configuration of OpenFlow-based SDN. In: *Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM 2015*. [daring] IEEE.hal.207–215. Tersedia pada: <http://ieeexplore.ieee.org/document/7140294/> [Diakses 7 Jul 2019].

KANAGAVELU, R. DAN AUNG, K.M.M., 2015. SDN controlled Local re-routing to reduce congestion in cloud Data Centers. *International Conference on Cloud Computing Research and Innovation*, hal.80–88.

MULYANA, E., 2015. *Buku Komunitas SDN-RG*. [daring] Bandung: GitBook. Tersedia pada: <https://eueung.gitbooks.io/buku-komunitas-sdn-rg/>.

PANTUZA, G., SAMPAIO, F., VIEIRA, L.F.M., GUEDES, D. DAN VIEIRA, M.A.M., 2014. Network management through graphs in Software Defined Networks. In: *10th International Conference on Network and Service Management (CNSM) and Workshop*. hal.400–405.

REHMAN, S.U., SONG, W.C. DAN KANG, M., 2014. Network-wide traffic visibility in OF@TEIN SDN testbed using sFlow. *APNOMS 2014 - 16th Asia-Pacific Network Operations and Management Symposium*, hal.1–6.