

## BAB 3

### METODE PENELITIAN

Metode penelitian berisi mengenai uraian dari alur penelitian dan diagram simulasi yang berbentuk *flowchart* penelitian menjelaskan mengenai tahap penelitian. Selanjutnya alat dan bahan yang digunakan, rancangan sistem penelitian yang digunakan serta analisis hasil yang mencakup uraian tentang model dan cara menganalisis hasil.

#### 3.1 ALUR PENELITIAN

Penelitian dijalankan secara bertahap berdasarkan *flowchart* ditunjukkan pada gambar 3.1.



**Gambar 3.1 Flowchart Penelitian**

Pada Gambar 3.1 menunjukkan diagram alur penelitian dimulai dari penentuan topologi jaringan sebagai dasar dalam pengujian pada simulasi SDN. Kemudian penginstalan instalasi *controller* OpenDaylight dan Floodlight pada sistem operasi Ubuntu 16.04 LTS menggunakan *virtual box 6.1*. Setelah proses instalasi berhasil kemudian dilanjutkan dengan membuat topologi jaringan pada Mininet dengan melakukan konfigurasi *script* menggunakan bahasa pemrograman *Python*. Jika topologi berhasil dibuat, selanjutnya pengujian *controller* dengan cara menghubungkan antara *controller* dengan Mininet. Jika pengujian awal berhasil, kemudian melakukan pengujian sistem secara keseluruhan sesuai dengan parameter parameter yang telah ditentukan dengan cara membangkitkan beban trafik menggunakan *software* dan Iperf. Pengambilan hasil data pada pengujian pada *software* Wireshark dan Iperf. berupa parameter QoS meliputi *throughput*, *delay*, *packet loss* dan *jitter*. Setelah hasil data terkumpul, selanjutnya melakukan analisis terkait dari data-data yang diperoleh untuk mengetahui performansi masing-masing *controller*. Setelah Analisis terkumpul kemudian dilakukan penarikan kesimpulan penelitian dengan memperhatikan tujuan penelitian yang dicantumkan pada bab 1 agar diperoleh hasil yang diharapkan.

## **3.2 PERANGKAT YANG DIGUNAKAN**

Penelitian ini menggunakan dua perangkat dalam menjalankan simulasi jaringan yakni piranti perangkat keras (*hardware*) dan piranti perangkat lunak (*software*) yang nantinya akan digunakan pada bab 3.3-3.6.

### **3.2.1 PERANGKAT KERAS (*HARDWARE*)**

Perangkat keras yang digunakan untuk menjalankan simulasi SDN pada penelitian ini yakni satu buah laptop yang nantinya digunakan sebagai tempat instalasi *software*. Laptop yang digunakan memiliki spesifikasi *hardware* lengkap pada Tabel 3.1.

**Tabel 3.1 Spesifikasi Perangkat Keras Komputer**

Sistem Operasi	Ubuntu Desktop Version 16.04 LTS
<i>Processor</i>	AMD Ryzen™ 5 4500U
<i>Graphics Processing Unit</i>	AMD Radeon™ Graphics
RAM	8 GB DDR4 - 3200Mhz
<i>Storage</i>	512GB SSD M.2 2242 PCIe NVMe
Kecepatan internet	20 Mbps

### 3.2.2 PERANGKAT LUNAK (*SOFTWARE*)

*Software* yang digunakan untuk mensimulasikan penelitian ini yakni *virtual box* 6.1 yang berfungsi sebagai tempat instalasi *guest OS*. Pada *virtual box* dipasang OS *virtual* yakni Ubuntu 16.04 yang nantinya sebagai tempat untuk instalasi *controller* Opendaylight, Floodlight sebagai *control plane*, Mininet sebagai *data plane* yang berfungsi untuk membangun topologi jaringan. Perangkat lunak yang digunakan sebagai alat ukur menghitung hasil *QoS* adalah Iperf dan Wireshark seperti yang ditunjukkan pada Tabel 3.2.

**Tabel 3.2 Software Pada Komputer**

Mesin <i>Virtual</i>	Virtual Box 6.1
<i>Emulator</i>	<i>Mininet</i>
<i>Controller 1</i>	<i>Opendaylight</i>
<i>Controller 2</i>	Floodlight
Alat Ukur QoS	Iperf, Wireshark

### 3.3 KONFIGURASI SISTEM SIMULASI JARINGAN SDN

Pada penelitian ini untuk dapat membangun sebuah jaringan SDN perlu dilakukan konfigurasi sistem simulasi *software defined network* yang digunakan sebagai tempat membangun sebuah jaringan SDN. Terdapat dua konfigurasi yang

dibutuhkan yakni konfigurasi *control plane* dan konfigurasi *data plane* yang dilakukan pada Ubuntu 16.04.

### 3.3.1 KONFIGURASI CONTROL PLANE

Konfigurasi *control plane* pada penelitian ini terdiri dari 2 konfigurasi yakni instalasi *controller* Floodlight dan Opendaylight. Proses instalasi *controller* dapat dilakukan dengan cara mengetikkan perintah pada *terminal* Ubuntu.

#### 1. Konfigurasi Floodlight

```
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java8-installer
$ sudo apt-get install build-essential ant maven python-dev
$ git clone git://github.com/Floodlight/Floodlight.git
$ cd Floodlight
$ git submodule init
$ git submodule update
$ ant
$ sudo mkdir /var/lib/Floodlight
$ sudo chmod 777 /var/lib/Floodlight
$ cd Floodlight
$ java -jar target/Floodlight.jar
```

**Gambar 3.2 Perintah Instalasi Floodlight**

Gambar 3.2 merupakan perintah dasar untuk melakukan instalasi Floodlight pada Ubuntu. Selanjutnya perintah yang digunakan untuk melihat *user interface controller* dapat menggunakan *web UI* dengan cara mengetikkan *IP address* lokal laptop pada *web browser*, *192.0.0.1:8080*.

#### 2. Konfigurasi Opendaylight

Konfigurasi Opendaylight diawali dengan instalasi *Java Development Kit* yang berfungsi sebagai paket fungsi dari API untuk bahasa pemrograman Java yang isinya terdiri dari *Java Runtime Environment (JRE)* dan *Java Virtual Machine (JVM)*. Cara kerja JDK dengan melakukan proses penerjemahan dari kode Java ke *bytecode* agar dapat dipahami dan dapat dieksekusi oleh JRE. Setelah JDK terinstal langkah selanjutnya adalah instalasi karaf dengan melakukan perintah *sudo -E karaf* pada terminal Ubuntu. Opendaylight memiliki

beberapa versi yang tersedia diinternet. Penelitian ini menggunakan versi Opendaylight *Oxygen*.

```
$ apt-get -y install openjdk-8-jre
$ sudo update-alternatives --config java
$ sudo echo 'export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-
amd64/jre' >> ~/.bashrc
$ source ~/.bashrc
$ sudo mkdir /usr/local/karaf
$ sudo mv karaf-0.8.4.zip /usr/local/karaf
$ sudo unzip /usr/local/karaf/karaf-0.8.4.zip -d /usr/local/karaf/
$ sudo update-alternatives --install /usr/bin/karaf karaf
/usr/local/karaf/karaf-0.8.4/bin/karaf 1
$http://nexus.opendaylight.org/content/repositories/opendaylight.release/
org/opendaylight/integration/karaf/0.8.4/karaf-0.8.4.zip
$sudo update-alternatives --config karaf
$which karaf
#run.opendaylight
$sudo -E karaf
$feature:install odl-l2switch-switch-ui
$logout
$sudo nano /etc/systemd/system/opendaylight.service
WantedBy=multi-user.target
$sudo chmod 0644 /etc/systemd/system/opendaylight.service
$cd /usr/local/karaf/
$systemctl daemon-reload
$sudo systemctl enable opendaylight.service
$sudo systemctl start opendaylight
$sudo systemctl status opendaylight
```

**Gambar 3.3 Perintah Instalasi Opendaylight**

Gambar 3.3 merupakan perintah dasar untuk melakukan instalasi Floodlight pada Ubuntu. Selanjutnya perintah yang digunakan untuk melihat *user interface controller* dapat menggunakan *web UI* dengan cara mengetikan *IP address* lokal laptop pada *web browser*, *192.0.0.1:8181/index.html*.

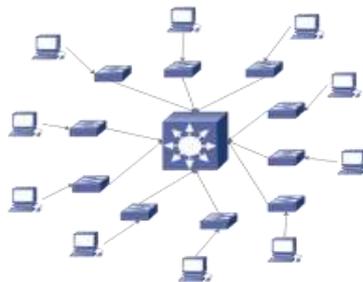
### 3.3.2 KONFIGURASI DATA PLANE

Konfigurasi yang dilakukan pada *data plane* diawali dengan instalasi Mininet untuk dapat membagan topologi jaringan dan sebagai penghubung antara *control plane* dengan *data plane*. Terdapat dua konfigurasi yang dilakukan pada

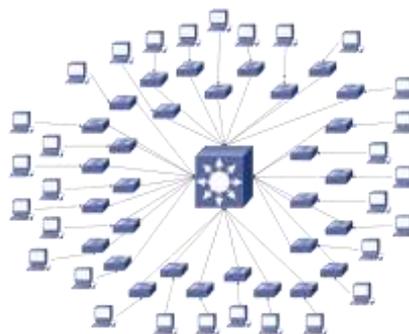
Mininet yakni membangun topologi jaringan dan menghubungkan Mininet dengan *controller*. Topologi jaringan pada Mininet dapat dibuat dengan cara konfigurasi *script* pemrograman berbahasa *python* yang berisikan informasi-informasi mengenai jumlah *switch*, jumlah *host*, IP address yang digunakan dari setiap *host* & *switch* dan pengaturan *link* antara *switch* dengan *switch* atau *host* dengan *switch*. Menghubungkan Mininet dengan *controller* dapat dilakukan dengan cara mengetikkan perintah pada *terminal* yang ada pada Ubuntu.

### 3.4 TOPOLOGI JARINGAN

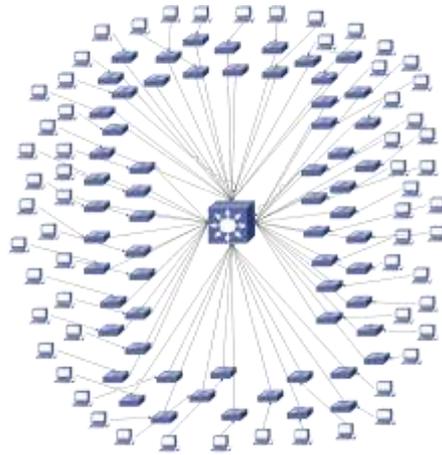
Topologi jaringan pada penelitian ini didasarkan pada Tabel 3.3 dan 3.4 membentuk topologi *ring* yang saling terhubung. Rancangan pembuatan topologi dirancang pada mininet dengan cara melakukan konfigurasi *script* topologi jaringan menggunakan bahasa pemograman *Phyton*. *Script* yang dibuat memuat informasi-informasi mengenai jumlah *host*, *switch*, pengalamatan *IP address* serta mengatur koneksi antara *switch* dengan *host* maupun *switch* dengan *switch*. Terdapat dua skenario pengujian yang terdiri dari 10 topologi jaringan yang ditunjukkan pada Gambar 3.4-3.5.



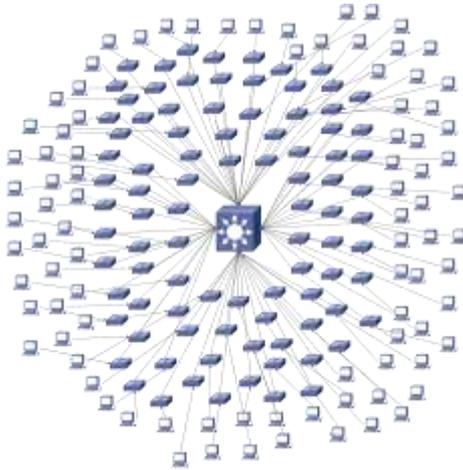
(a)



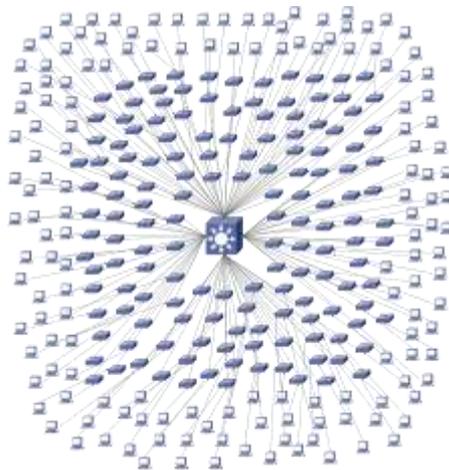
(b)



(c)



(d)

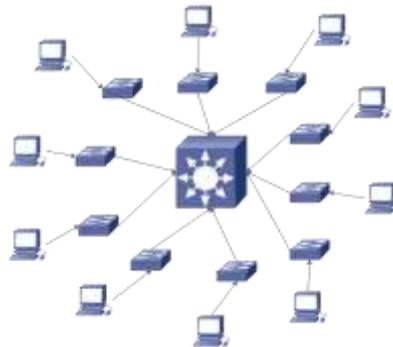


(e)

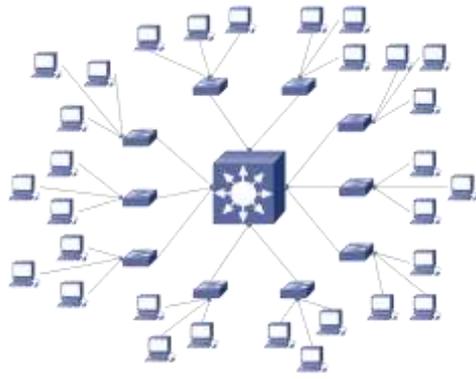
### Gambar 3.4 Topologi Skenario Penambahan *Switch & Host*

- (a) Topologi 10 *Switch* dan 10 *Host*
- (b) Topologi 30 *Switch* dan 30 *Host*
- (c) Topologi 60 *Switch* dan 60 *Host*
- (d) Topologi 90 *Switch* dan 90 *Host*
- (e) Topologi 150 *Switch* dan 150 *Host*

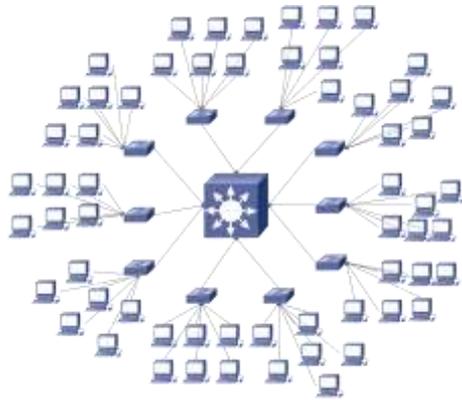
Gambar 3.4 merupakan topologi simulasi pengujian skenario pertama Topologi pengujian penelitian ini menggunakan topologi *ring*. Pada topologi skenario pertama penggunaan variasi penambahan *switch* dan *host* terdiri dari lima jenis yakni 10 *switch* dan 10 *host*, 30 *switch* dan 30 *host*, 60 *switch* dan 60 *host*, 90 *switch* dan 90 *host* dan 150 *switch* dan 150 *host*. Topologi simulasi pengujian dibuat dengan cara menuliskan program menggunakan bahasa pemrograman *python* dengan nama file *skripsi1-1.py*, *skripsi1-2.py*, *skripsi1-3.py*, *skripsi1-3.py*, *skripsi1-4.py*, *skripsi1-5.py*, yang nantinya akan dibangkitkan menggunakan *mininet* dan di hubungkan dengan *controller* *OpenDaylight* dan *Floodlight*.



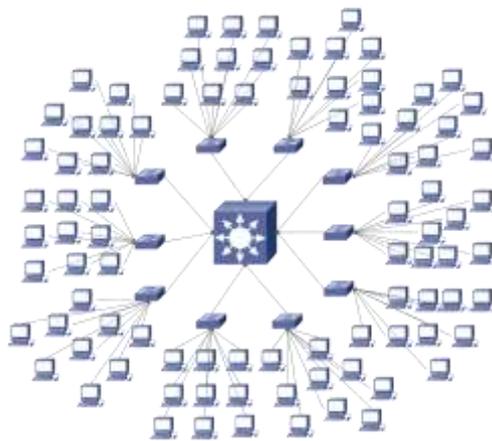
(a)



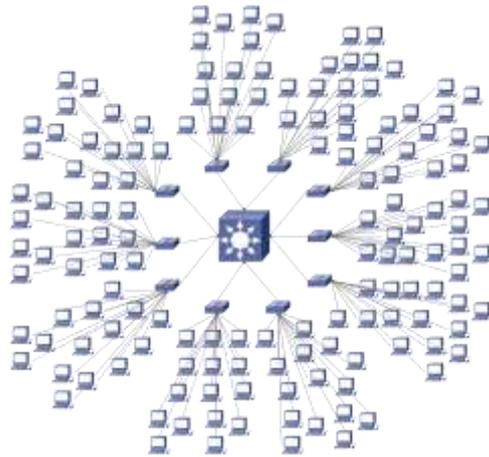
(b)



(c)



(d)



(e)

**Gambar 3.5 Topologi Skenario Penambahan *Host***

**(a) Topologi 10 *Switch* dan 10 *Host***

**(b) Topologi 10 *Switch* dan 30 *Host***

**(c) Topologi 10 *Switch* dan 60 *Host***

**(d) Topologi 10 *Switch* dan 90 *Host***

**(e) Topologi 10 *Switch* dan 150 *Host***

Gambar 3.4 merupakan topologi simulasi pengujian skenario kedua. Pada topologi skenario kedua, penggunaan variasi penambahan hanya berupa *host* yang terdiri dari lima jenis yakni 10 *switch* dan 10 *host*, 10 *switch* dan 30 *host*, 10 *switch* dan 60 *host*, 10 *switch* dan 90 *host* dan 10 *switch* dan 150 *host*. Topologi simulasi pengujian dibuat dengan cara menuliskan program menggunakan bahasa pemrograman *Python* dengan nama file *skripsi2-1.py*, *skripsi2-2.py*, *skripsi2-3.py*, *skripsi2-3.py*, *skripsi2-4.py*, *skripsi2-5.py*, yang nantinya akan dibangkitkan menggunakan *Mininet*.

### **3.5 SKENARIO PENGUJIAN**

Skenario pengujian dari penelitian ini terdiri dari 2 skenario. Pada skenario pertama dilakukan dengan menambah jumlah *switch* & *host* dimulai dari 10 *switch* & 10 *host*, 30 *switch* & 30 *host*, 60 *switch* & 60 *host*, 90 *switch* & 90 *host* dan 150 *switch* & 150 *host* dengan beban *traffic* sebesar 200 MB. Pengujian ini dilakukan untuk mengetahui manakah kinerja *controller* yang lebih unggul dalam

membangun jaringan SDN. Berikut adalah tabel skenario pertama yang ditampilkan pada Tabel 3.3.

**Tabel 3.3 Skenario Penambahan Jumlah *Switch* & *Host***

No		Jumlah <i>Switch</i>	Jumlah <i>Host</i>	<i>IP Address</i>
1	Percobaan ke 1	10	10	192.168.1.1 – 192.168.1.10
2	Percobaan ke 2	30	30	192.168.1.1 – 192.168.1.30
3	Percobaan ke 3	60	60	192.168.1.1 – 192.168.1.60
4	Percobaan ke 4	90	90	192.168.1.1 – 192.168.1.90
5	Percobaan ke 5	150	150	192.168.1.1 – 192.168.1.150

Pada skenario pengujian kedua dilakukan dengan menambahkan jumlah *host* tanpa menambah jumlah *switch* dimulai dari 10 *switch* & 10 *host*, 10 *switch* & 30 *host*, 10 *switch* & 60 *host*, 10 *switch* & 90 *host*, dan 10 *switch* & 150 *host* dengan beban *traffic* sama sebesar 200 MB. Pengujian ini dilakukan untuk mengetahui pengaruh penambahan *host* berdasarkan parameter QoS yakni *throughput*, *jitter*, *packet loss*, dan *delay*. Berikut tabel skenario pengujian kedua yang ditampilkan pada Tabel 3.4

**Tabel 3.4 Skenario Penambahan Jumlah *Host***

No		Jumlah <i>Switch</i>	Jumlah <i>Host</i>	<i>IP Address</i>
1	Percobaan ke 1	10	10	192.168.1.1 – 192.168.1.10
2	Percobaan ke 2	10	30	192.168.1.1 – 192.168.1.30
3	Percobaan ke 3	10	60	192.168.1.1 – 192.168.1.60
4	Percobaan ke 4	10	90	192.168.1.1 – 192.168.1.90
5	Percobaan ke 5	10	150	192.168.1.1 – 192.168.1.150

### 3.6 PENGAMBILAN DATA

Proses pengambilan hasil data dilakukan sebanyak 30 kali setiap percobaan, diawali dengan menghubungkan *controller* OpenDaylight dan Floodlight dengan mininet melalui terminal ubuntu dengan cara mengetikkan perintah dasar seperti pada Gambar 3.6.

```
admin1@admin: ~/mininet/custom
admin1@admin:~$ cd mininet/custom/
admin1@admin:~/mininet/custom$ sudo mn --custom skripsi1-2.py --topo skenario1-2 --controller=remote,ip=192.168.100.144
```

**Gambar 3.6** Perintah Menghubungkan *Controller* dengan Mininet

Setelah Mininet berhasil membangkitkan sejumlah *switch* dan *host* membentuk topologi *ring* sesuai dengan *script* program yang dibuat, selanjutnya perlu dilakukannya pengetesan *ping* terlebih dahulu keseluruhan *node* yang terhubung untuk memastikan ketersediaan jaringan berjalan secara normal seperti yang ditunjukkan pada Gambar 3.7.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Results: 0% dropped (90/90 received)
mininet>
```

**Gambar 3.7** Pengetesan Ketersediaan Jaringan

Proses pengambilan data pengukuran QoS menggunakan *software* Iperf dan Wireshark. Pengambilan data dilakukan dari komunikasi node h2 ke h7 menggunakan *xterm*. Pada *node* h2 akan menjalankan Iperf sebagai penerima data, sedangkan h7 menjalankan Iperf sebagai pengirim paket data. Sebagai contoh untuk h7 akan bertindak sebagai *client* kemudian mengirimkan data berbentuk *protocol* UDP dengan ukuran paket sebesar 200 *Megabytes* yang dikirimkan selama 30 detik menggunakan port 5566 ke alamat IP *address* 192.168.1.2 seperti yang ditunjukkan pada Gambar 3.8.

```
"Node: h7"
root@ananta-VirtualBox:~/mininet/custom# iperf -c 192.168.1.2 -u -b 200m -t 30 -p 5566
```

**Gambar 3.8** Perintah yang Dijalankan h7

Selanjutnya, perintah yang dijalankan pada h2 akan bertindak sebagai *server* dengan menerima *protocol* UDP yang dikirimkan h7 menggunakan *port* 5566 dengan *interval* 2 seperti yang ditunjukkan pada Gambar 3.9.

```

root@ananta-VirtualBox:~/mininet/custom# iperf -s -u -p 5566 -i 2
  
```

**Gambar 3.9 Perintah yang Dijalankan h2**

Pada saat proses pembangkitan trafik jaringan, *software* Iperf akan menghasilkan sebuah *live report* dengan *interval* 2 detik, dimana *report* tersebut yang nantinya akan digunakan sebagai hasil data penelitian. Hasil pengukuran yang diperoleh *software* Iperf terdiri dari nilai-nilai parameter QoS seperti *throughput*, *jitter*, dan *packet loss* seperti pada Gambar 3.10.

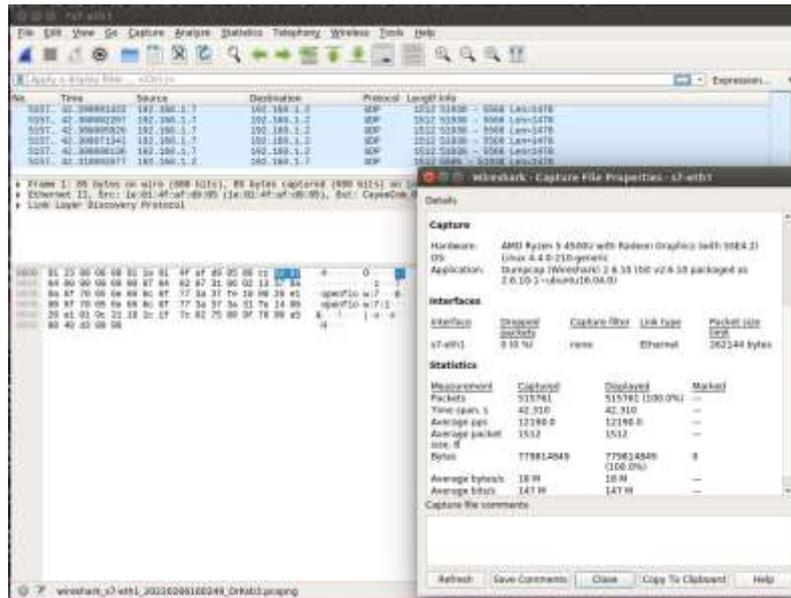
```

[127] local 192.168.1.7 port 43068 connected with 192.168.1.2 port 5566
[11] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[127] 0.0-2.0 sec 725 Kbytes 203 Mbits/sec 0,045 ms 0/54456 (0,0056%)
[127] Sent 538940 datagrams
[127] Server Report:
[127] 0.0-30.0 sec 725 Kbytes 203 Mbits/sec 0,050 ms 29/538839 (0,0056%)
[127] 10.0-30.0 sec 10 datagrams received out-of-order
root@ananta-VirtualBox:~/mininet/custom# iperf -s -u -p 5566 -i 2
Server listening on UDP port 5566
Receiving 1470 byte datagrams
UDP buffer size: 206 Kbyte (default)

[127] local 192.168.1.2 port 5566 connected with 192.168.1.7 port 43068
[11] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[127] 0.0-2.0 sec 48.3 Kbytes 203 Mbits/sec 0,030 ms 0/54456 (0,0056%)
[127] 0 datagrams received out-of-order
[127] 2.0-4.0 sec 48.3 Kbytes 203 Mbits/sec 0,045 ms 0/54470 (0,0056%)
[127] 4.0-6.0 sec 48.3 Kbytes 203 Mbits/sec 0,046 ms 0/54423 (0,0056%)
[127] 6.0-8.0 sec 48.3 Kbytes 203 Mbits/sec 0,035 ms 0/54402 (0,0056%)
[127] 8.0-10.0 sec 48.3 Kbytes 203 Mbits/sec 0,030 ms 0/54457 (0,0056%)
[127] 10.0-12.0 sec 48.3 Kbytes 203 Mbits/sec 0,030 ms 0/54425 (0,0056%)
[127] 12.0-14.0 sec 48.3 Kbytes 203 Mbits/sec 0,027 ms 25/54477 (0,0056%)
[127] 14.0-16.0 sec 48.3 Kbytes 203 Mbits/sec 0,040 ms 0/54462 (0,0056%)
[127] 16.0-18.0 sec 48.3 Kbytes 203 Mbits/sec 0,024 ms 0/54455 (0,0056%)
[127] 18.0-20.0 sec 48.3 Kbytes 203 Mbits/sec 0,030 ms 0/54485 (0,0056%)
[127] 20.0-22.0 sec 48.3 Kbytes 203 Mbits/sec 0,035 ms 0/54470 (0,0056%)
[127] 22.0-24.0 sec 48.3 Kbytes 203 Mbits/sec 0,045 ms 0/54441 (0,0056%)
[127] 24.0-26.0 sec 48.3 Kbytes 203 Mbits/sec 0,030 ms 0/54479 (0,0056%)
[127] 26.0-28.0 sec 48.3 Kbytes 203 Mbits/sec 0,050 ms 29/538839 (0,0056%)
[127] 0.0-30.0 sec 725 Kbytes 203 Mbits/sec
  
```

**Gambar 3.10 Hasil Live Report QoS**

Sebagai contoh pada gambar 3.10 pada hasil *live report* dapat diperoleh nilai parameter QoS *throughput* sebesar 203 *Mbits/sec*, *jitter* sebesar 0,050 *millisecond (ms)* dan *packet loss* sebesar 0,0056%.



**Gambar 3.11 Pengambilan Hasil Data Delay**

Pengambilan data selanjutnya adalah nilai *delay* yang menggunakan bantuan *software* Wireshark, dilakukan dengan cara yang sama menggunakan Iperf sebagai aplikasi untuk membangkitkan beban trafik yang kemudian dilakukan proses *capture* menggunakan Wireshark. Nilai *delay* diperoleh dengan cara menghitung waktu pengiriman paket data ke-2 dikurangi waktu pengiriman paket data ke-1 atau rata-rata *delay* didapatkan dengan cara menghitung total waktu tunggu pengiriman dibagi dengan total paket yang diterima. Contoh pada Gambar 3.19 dimana total waktu pengiriman sebesar 42,310 detik dibagi dengan 515.761 paket yang diterima sehingga dihasilkan  $8,20341 \times 10^{-5}$  detik atau diubah kedalam *milliseconds* menjadi 0,082034 ms.