MIKROTIK RB750 ROUTERBOARD SEBAGAI ALTERNATIF SWITCH *OPENFLOW* SOFTWARE-BASE

Rikie Kartadie
Pendidikan Teknologi Informasi
STKIP PGRI Tulungagung
Email: rikie.kartadie@stkippgritulungagung.ac.id

ABSTRAK

Software-Defined Networking (SDN) merupakan cara untuk mengurangi kompleksitas konfigurasi jaringan dan manajemen jaringan. Solusi yang ditawarkan oleh SDN memberikan nuansa baru pada jaringan komputer. Pengimplementasian arsitektur SDN/*OpenFlow* membutuhkan biaya yang tinggi, sedangkan penggunaan emulator mininet mampu memberikan simulasi yang baik pada skala penelitian, namun dalam kenyataannya implementasinya membutuhkan *hardware*. Pengimplementasian *OpenFlow* pada switch telah merambah kebeberapa vendor, diantaranya MikroTik yang telah menambahkan *OpenFlow agent* pada OS versi 6.17 pada *RouterOS* nya dan memungkinkannya untuk dapat diimplementasikan pada arsitektur SDN/*OpenFlow* dengan biaya yang lebih terjangkau. Implementasi *OpenFlow* agent pada *RouterOS* MikroTik layak untuk diuji performanya, sebagai alternatif switch *OpenFlow* software-base. Penelitian ini adalah penelitian awal dari rangkaian penelitian yang akan dilakukan. Langkah-langkah penelitian adalah: (1)Merancang topologi, (2)Simulasi menggunakan emulator mininet sebagai data pembanding (dianggap sebagai representatif dari *hardware-base switch*), (3)Memodifikasi MikroTik RB750 sebagai prototipe switch *OpenFlow software-base*, (4) Pengujian prototipe, (5) Analisis hasil dan menarik kesimpulan. Dari hasil uji *latency*, prototipe memberikan nilai lebih tinggi dibandingkan dengan nilai pembandingnya, nilai throughput TPC dan UDP 1937.5 kbps dan 8.64 kbps dan nilai *jitter* 0.0093 msec lebih rendah dibandingkan nilai pembandingnya. Prototipe dapat dijadikan alternatif pengganti switch *OpenFlow Software-based*, walaupun prototipe masih memberikan nilai performa yang rendah.

Kata kunci: *mininet, mikrotik, software-defined network, openflow.*

ABSTRACT

Software-Defined Networking (SDN) is a way to reduce network configuration complexity and network management. Solution offered by SDN is a fresh new look in a computer network. The implementation of SDN/OpenFlow architecture need a high cost, and the use of mininet emulator is able to give a good simulation at research scale. But, in reality, the implementation of SDN/OpenFlow architecture requires hardware. The implementation of OpenFlow in switch has been expanded to several vendors such as MikroTik. MikroTik added OpenFlow agent on OS version 6.17 on the RouterOS so that the MikroTik can be implemented in SDN/OpenFlow architecture in affordable cost. The performance of the implementation of OpenFlow agent on MikroTik RouterOS is feasible to be tested as an alternative to software-base OpenFlow switch. This study is the beginning of a series of studies that will be done. Research steps are: (1) Designing topology, (2) Simulation using mininet emulator as comparative data (considered as representative of the hardware-base switch), (3) Modify MikroTik RB750 as a prototype OpenFlow software-base switch, (4) Prototype testing, (5) Analysis of the results and draw conclusions. From latency testing results, prototype gives a higher value than the comparison value, the value of TPC and UDP throughput 1937.5kbps and 1.925kbps and 0.0093msec jitter value is lower than comparison value. The prototype can be used as an alternative to software-based OpenFlow switch, though the prototype still gives low performance.

Keywords: *mininet, mikrotik, software-defined network, openflow.*

1. PENDAHULUAN

Software Defined Networking (SDN) merupakan cara untuk mengurangi kompleksitas konfigurasi jaringan dan manajemen jaringan. Solusi yang ditawarkan oleh SDN memberikan nuansa baru pada jaringan komputer. *Software-Defined Networking* (SDN) atau *split* arsitektur adalah sebuah konsep yang memungkinkan/mengizinkan operator jaringan untuk mengelola router dan switch secara fleksibel menggunakan software yang berjalan di server eksternal [1],[2]. Open Network Foundation sendiri mendefinisikan SDN sebagai sebuah sebuah arsitertur jaringan baru dimana kontrol jaringan dipisahkan dari *forwarding* dan diprogram secara langsung [3],[4].

Pengimplementasian arsitektur SDN/*OpenFlow* membutuhkan biaya yang tinggi, sedangkan penggunaan emulator mininet mampu memberikan simulasi yang baik pada skala penelitian, namun dalam kenyataannya implementasi membutuhkan *hardware[5]*.

Berdasarkan tesis yang penulis kerjakan dengan judul "Prototipe Infrastruktur Software-Defined Network Dengan Protokol *OpenFlow* Menggunakan Ubuntu Sebagai Kontroler", *software-based OpenFlow* switch memiliki kemampuan yang realtif sama dengan *hardware-based OpenFlow* switch, yang tentu saja dengan berbagai kekurangannya. Pada tesis tersebut digunakan switch berbasih OpenWRT yang di-implementasi-kan pada switch dari *vendor TP-LINK [5]*.

Hasil uji switch *OpenFlow* berbasis OpenWRT pada penelitian sebelumnya Performa switch OF *software-based* dapat dikatakan baik dengan hasil gap rata-rata pada setiap pengujian menunjukkan angka yang tidak tinggi. Bahkan pada pengujian *jitter* dapat dikatakan sama[6].

Pengimplementasian *OpenFlow* pada switch telah merambah kebeberapa vendor, diantaranya MikroTik. Tanutama dalam bukunya menyatakan bahwa MikroTik adalah sistem operasi independen berbasis Linux khusus untuk komputer yang difungsikan sebagai router. MikroTik didesain untuk mudah digunakan dan sangat baik digunakan untuk keperluan administrasi jaringan komputer seperti merancang dan membangun sebuah sistem jaringan komputer skala kecil hingga yang kompleks. MikroTik mulai didirikan tahun 1995 yang pada awalnya ditujukan untuk perusahaan jasa layanan Internet (*Internet Service Provider*, ISP) yang melayani pelanggannya menggunakan teknologi nirkabel. Saat ini MikroTik memberikan layanan kepada banyak ISP nirkabel untuk layanan akses Internet di banyak negara di dunia dan juga sangat populer di Indonesia. Mikrotik pada standar perangkat keras berbasiskan *Personal Computer* (PC) dikenal dengan kestabilan, kualitas kontrol dan fleksibilitas untuk berbagai jenis paket data dan penanganan proses rute (*routing*). MikroTik yang dibuat sebagai router berbasiskan komputer banyak bermanfaat untuk sebuah ISP yang ingin menjalankan beberapa aplikasi mulai dari hal yang paling ringan hingga tingkat lanjut. Selain routing, MikroTik dapat digunakan sebagai manajemen kapasitas akses (*bandwidth, firewall, wireless access point (WiFi), backhaul link*, sistem *hotspot*, *Virtual Private Network server)* dan masih banyak lainnya[7].

MikroTik telah menambahkan secara *optional*/pilihan *OpenFlow agent* pada OS versi 6.17 pada *RouterOS* nya dan memungkinkannya untuk dapat di-implementasi-kan pada arsitektur SDN/*OpenFlow* dengan biaya yang lebih terjangkau. Implementasi *OpenFlow* agent pada *RouterOS* MikroTik layak untuk diuji performanya, sebagai alternatif switch *OpenFlow software-based*.

Switch *OpenFlow* terdiri dari dua jenis, yang pertama adalah *hardware-based switch*, yang telah dijual secara komersial oleh beberapa *vendor* [8]. Switch jenis ini telah memodifikasi *hardware*-nya, menggunakan TCAM· Ternary CAM memungkinkan kondisi pencocokan ketiga "X" atau "tidak peduli" untuk satu atau lebih bit dalam dataword yang disimpan, sehingga menambah fleksibilitas untuk pencarian. Misalkan, ternary CAM mungkin memiliki kata yang tersimpan dari "10XX0" yang akan cocok dengan empat kata pencarian "10000", "10010", "10100", atau "10110". fleksibilitas pencarian menggunakan sumberdaya yang lebih besar dibandingkan dengan CAM biner sehinga dibutuhkan sel memori internal tambahan yang harus menyandikan tiga kemungkinan bukan dua dari CAM biner [9]. Kondisi tambahan ini biasanya diimplementasikan dengan menambahkan sedikit topeng (bit "peduli" atau "tidak peduli") untuk setiap sel memori, dan menggunakan OS khusus untuk mengimplementasikan *Flow-Table* dan *OpenFlow* protokol. Jenis yang kedua adalah *software-based switch* yang menggunakan sistem UNIX / Linux untuk mengimplementasikan seluruh fungsi *OpenFlow* switch [10]. Pengaturan *flow* pada switch *OpenFlow* dilakukan oleh sebuah kontroler secara terpusat.

Sebuah kontroler pada arsitek SDN bertanggung jawab untuk menambahkan atau menghilangkan flow dari *OpenFlow table* yang ada didalam perangkat *OpenFlow* itu sendiri. Ada 2 tipe dari kontroler : Statis, sebuah kontroler statis dapat berupa perangkat yang dapat menambahkan atau menghilangkan flow dari *flow table* secara statis. Dinamis, sebuah kontroler dinamis memanipulasi isi dari flow sehingga cocok untuk beberapa konfigurasi secara dinamis.

Floodlight yang digunakan pada penelitian ini merupakan kontroler yang digunakan dalam pengembangan proyek SDN (*Software Defined Network*). Floodlight berlisensi Apache dan berbasis JAVA. Floodlight dirancang untuk bekerja optimal sejalan dengan meningkatnya jumlah switch, router, switch virtual dan jalur akses yang mendukung standar *OpenFlow* [11].

Penelitian ini adalah penelitian awal dari rangkaian penelitian yang akan dilakukan. Pada penelitian ini dilakukan pengujian *latency* dan *througput* pada *switch OepnFlow software-based MikroTik*.

Latency didefinisikan sebagai penundaan waktu antara saat sesuatu dimulai dan saat ini atau menjadi terdeteksi atau dapat pula diartikan sebagai interval waktu antara stimulasi dan respon, dari sudut pandang yang lebih umum, sebagai waktu tunda antara penyebab dan efek dari beberapa perubahan fisik dalam sistem yang diamati. Dalam hal ini *latency* dari switch adalah jumlah respon yang dapat diberikan oleh switch dalam tiap detiknya [12].

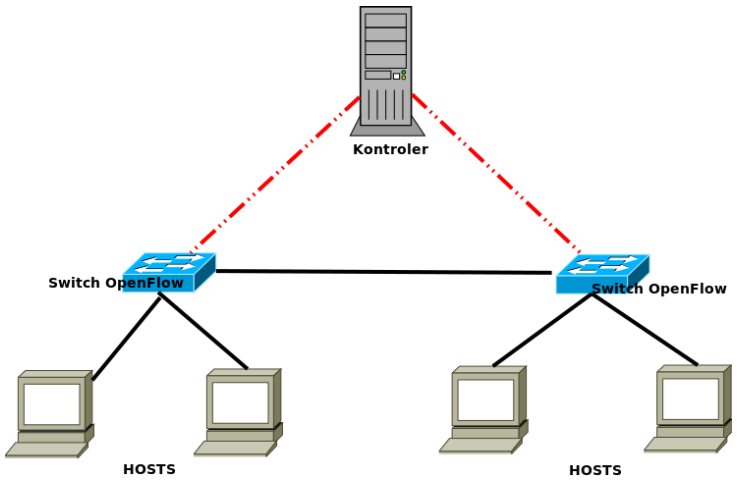
Throughput secara umum merupakan tingkat capaian atau tingkat di mana sesuatu dapat diproses, untuk jaringan adalah tingkat pengiriman pesan sukses melalui saluran komunikasi, atau dapat pula didefinisikan sebagai jumlah tingkat data yang tersampaikan disemua terminal dalam jaringan .

2. METODOLOGI PENELITIAN

Langkah-langkah pada penelitian ini adalah: (1)Merancang topologi, (2)Simulasi menggunakan emulator mininet sebagai data pembanding (dianggap sebagai representatif dari *hardware-base switch*), (3)Memodifikasi MikroTik RB750 sebagai prototipe switch *OpenFlow software-based*, (4) Pengujian prototipe, (5) Analisis hasil dan menarik kesimpulan.

2.1 Merancang Topologi

Topologi yang digunakan pada penelitian ini adalah topologi linear yang melibatkan 2 buah switch *OpenFlow software-based* berbasis MikroTik RB750 yang terhubung dengan sebuah kontroler. Topologi yang digunakan dapat dilihat pada gambar 1 dibawah ini.



Gambar 1. Topologi Yang Digunakan Dalam Penelitian

Topologi pada gambar 1 digunakan pada simulasi mininet dan digunakan pula pada topologi pada pengujian. Keterangan gambar:

Kontroler	Kontroler tersentralisasi yang melakukan kontrol terhadap kedua buah switch.
hosts	Host yang terkoneksi langsung ke switch *OpenFlow*
***Switch** OpenFlow*	Switch *OpenFlow* yang di uji, kedua buah switch tersebut terhubung satu sama lain, dan kedua switch terhubung langsung dengan kontroler.
———	Koneksi antar perangkat pada jaringan *OpenFlow*
———	Koneksi dari kontroler ke switch

2.2 Simulasi Menggunakan Emulator Mininet

Topologi yang telah dirancang sebelumnya diujikan pada emulator mininet. Pengujian ini dimaksudkan untuk mendapatkan data pembanding nilai *latency dan throughput* dengan data pada switch *OpenFlow software-based* MikroTik yang akan diujikan.

Topologi yang akan diujikan terlihat pada scrip dibawah ini dan disimpan dengan nama file ujibanding.py,

```
from mininet.topo import Topo

class MyTopo( Topo ):

   "Topologi prototipe."
    def __init__( self ):

"Membuat topologi untuk uji pembanding."
     # Initialize topology
      Topo.__init__( self )

     # Add hosts and switches
```

```
left0Host = self.addHost( 'h1', ip='192.168.10.1/24' )
right0Host = self.addHost( 'h2', ip='192.168.10.2/24' )
left1Host = self.addHost( 'h3', ip='192.168.10.3/24' )
right1Host = self.addHost( 'h4', ip='192.168.10.4/24' )
leftSwitch = self.addSwitch( 's1' )
rightSwitch = self.addSwitch( 's2' )

# Add links
self.addLink( left0Host, leftSwitch )
self.addLink( left1Host, leftSwitch )
self.addLink( leftSwitch, rightSwitch )
self.addLink( rightSwitch, right0Host )
self.addLink( rightSwitch, right1Host )

topos = { 'topoujibanding': ( lambda: MyTopo() ) }
```

Scrip tersebut dijalankan pada emulator mininet dengan kontroler diletakkan diluar virtualbox yang dijalankan oleh mininet. *Scrip* dijalankan dengan perintah dibawah ini.

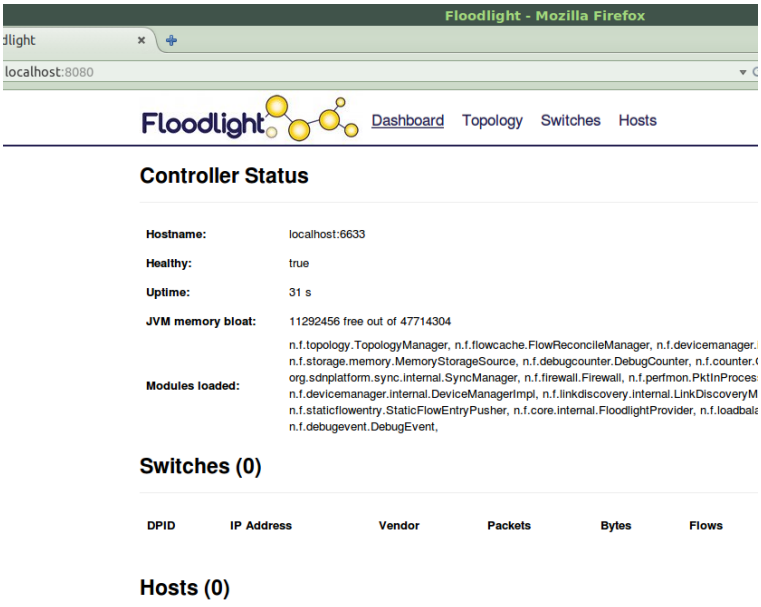
```
mininet@mininet#sudo mn --custom ujibanding.py --topo topoujibanding --controller=remote,
ip=192.168.1.76, port=6633 --mac --link tc,bw=100
```

Mininet dijalankan pada Laptop dengan spesifikasi sebagai berikut:

a) CPU: Intel® Core™ i3-4005U CPU @ 1.70GHz × 4
b) RAM: SODIMM DDR3 4Gb 1333MHz
c) Hardisk: 259, 1 Gbyte
d) Operation System: Linux UBUNTU 16.04 LTS.

Tentunya kontroler floodlight telah dijalankan terlebih dahulu, dengan perintah

```
$ java -jar floodlight.jar
```



Gambar 2. *Kontroler Floodlight*

GUI dari floodlight ini dapat diakses pada web browser dengan alamat http://localhost:8080/ui/index.html

2.3 Memodifikasi Mikrotik RB750 Sebagai Prototipe

Prototipe *OpenFlow* switch *software-based* dengan menggunakan MikroTik RB750 dapat dilakukan dengan 2 langkah, pertama dengan mengganti OS yang ada dengan OpenWRT dan selanjutnya menambahkan *OpenFlow* agen kedalamnya, kedua dengan menambahkan paket yang telah disediakan oleh MikroTik, walaupun MikroTik sendiri belum memberikan lisensi terhadap paket ini

Pada penelitian ini, digunakan cara yang kedua untuk menambahkan *agent OpenFlow* kedalam *RouterOS* MikroTik RB750. Karna peneliti beranggapan bila dilakukan cara yang pertama, maka tidak ada perbedaan signifikan dengan penelitian sebelumnya.

Langkah pertama yang dilakukan adalah mengupgrade OS MikroTik RB750 dengan versi 6.rc8 (januari 2013). Langkah kedua adalah dengan menambah paket *OpenFlow-6.32.2-mipsbe.npk* kedalam MikroTik dan mengkatifkan paket tersebut.

2.4 Pengujian Prototipe

Pengujian prototipe dilakukan dengan uji *latency* dan *throughput*. Pengujian *latency* yang dilakukan pada prototipe dilakukan pada besar paket ICMP dari 64 byte hingga 8192 byte. Pengujian dilakukan dengan pengulangan sebanyak 15 kali pada setiap paket data ICMP yang dikirimkan.

Pengujian *throughput* yang dilakukan pada prototipe, dilakukan pada protokol TCP dengan TCP *windows size* 128 kBps hingga 1024kBps dan UDP *Buffer size* dan *Bandwidth* sebesar 128 kBps hingga 1024kBps. Pengujian menggunakan jperf sebagai *tool* penguji. Pada sisi *server*, jperf diseting dengan *windows sizes* mulai dari 128kBps hingga 1024kBps.

iperf -s -P 0 -i 1 -p 5001 -w *1024K* -f k

Untuk UDP digunakan seting mulai dari packet size yang sama pula.

iperf -s -u -P 0 -i 1 -p 5001 -w *1024K* -f k

Sedang pada sisi *client*, jperf diseting dengan konfigurasi

iperf -c 10.0.0.1 -P 1 -i 1 -p 5001 -w *1024K* -f K -t 10

Untuk UDP pada sisi *client* digunakan seting sebagai berikut

iperf -c 10.0.0.1 -u -P 1 -i 1 -p 5001 -w 1024K -f k -b 1024M -t 10 -T 1

3. HASIL DAN PEMBAHASAN

Dari data yang diperoleh, baik data dari uji pembanding maupun data dari switch OF *software-based* MikroTik, dilakukan analisis pada nilai *latency* dan *throughput* yang dihasilkan. Data yang disajikan adalah nilai rata-rata pengujian.

3.1 Pengujian Latency

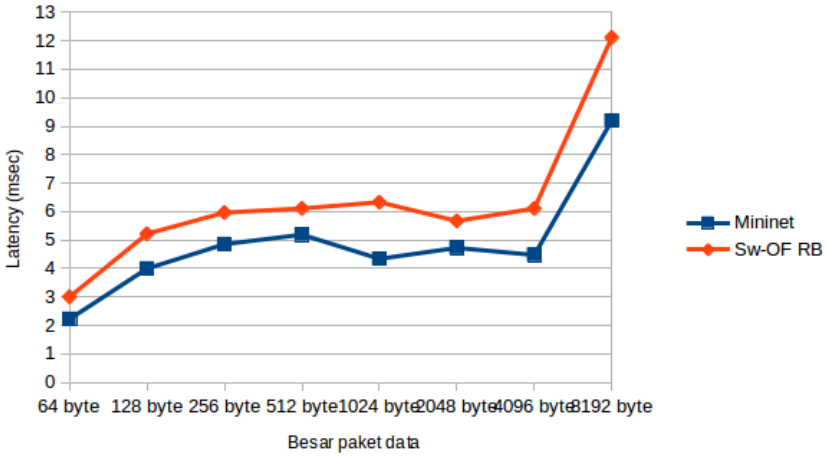
Pada tabel 1 dan gambar 3 dibawah ini, merupakan data nilai rata-rata *latency* yang dihasilkan pada setiap pengujian.

Tabel 1. Nilai rata-rata dari setiap pengujian

	64 byte	***128 byte***	***256 byte***	***512 byte***	***1024 byte***	***2048 byte***	***4096 byte***	***8192 byte***
Mininet	2.228	3.999	4.855	5.191	4.341	4.726	4.477	9.194
Sw-OF RB	3.007	5.221	5.966	6.11	6.331	5.672	6.108	12.111

Catatan*:*
Mininet : Nilai latency data pembanding dengan mininet
Sw-OF RB : Nilai latency prototipe
data disajikan dalam msec.

Diperoleh hasil bahwa prototipe mengalami *latency* yang tinggi diatas nilai *latency* dari data pembandingnya (mininet). Pada besar paket data ICMP tertentu prototipe mengalami keterlambatan yang lebih tinggi dibanding pembandingnya, seperti dapat dilihat pada gambar 3 dibawah ini.



Gambar 3. Grafik Pengujian *Latency*

3.2 Pengujian Throughput

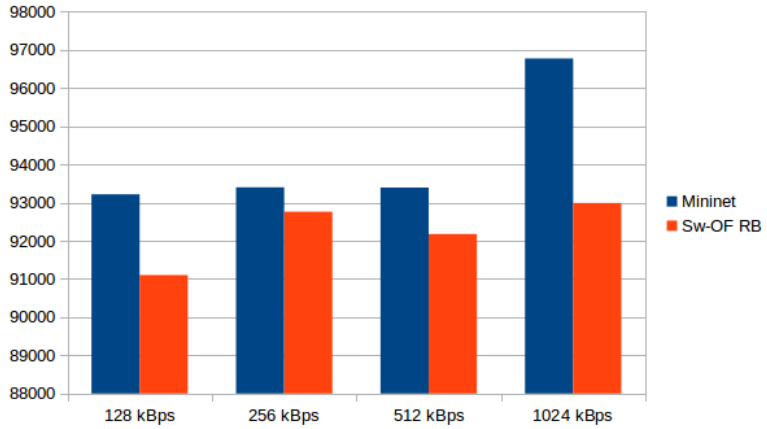
3.2.1 Protokol TCP

Pengujian *throughput* dengan protokol TCP yang diperoleh hasil *bandwidth* dari mininet dan prototipe seperti pada tabel 2 dan gambar 4 dibawah ini. Data yang digunakan adalah data Rx (*Receive*).

Tabel 2. Nilai TCP *Throughput* Rx

	128 kBps	***256 kBps***	***512 kBps***	***1024 kBps***
Mininet	93216	93400	93392	96771
Sw-OF RB	91101	92762	92176	92990
GAP rata^2		- 1937.5		

Catatan*:*
Mininet : Nilai throughput data pembanding dengan mininet
Sw-OF RB : Nilai throughput prototipe
gabrata^2 : Selisih rata-rata antar nilai throughput prototipe dengan pembanding.
data disajikan dalam kbps.



Gambar 4. TCP *Throughput* Rx

Dari grafik yang terlihat pada gambar 4, *throughput* protokol TCP yang diperoleh prototipe (switch OF *software-based* MikroTik) menunjukkan nilai yang kecil dibandingkan dengan pembanding mininet. Kemampuan dari prototipe belum dapat mendekati nilai pembanding pada besar paket 256kBps dan kemudian tidak mengalami kenaikan yang signifikan. Dapat dimaknai prototipe tidak memiliki kemampuan *throughput* yang baik, terlebih pada besar data diatas 256kBps. Selisih nilai yang diberikan masih besar yaitu 1937.5kbps lebih kecil dari pembandingnya.

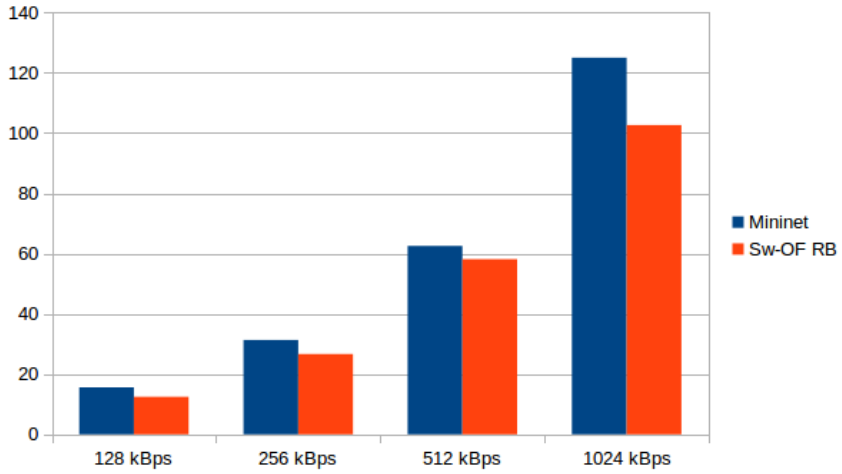
3.2.2 Protokol UDP

Diperoleh 2 nilai pada pengujian throughput protokol UDP yaitu besar bandwidth dan nilai jitter. Tabel 3 merupakan tabel yang menunjukkan perbedaan bandwidth yang dihasilkan switch pada protokol UDP. Pada gambar 5, dapat dilihat bahwa nilai throughput protokol UDP dari prototipe menunjukkan nilai yang lebih rendah dibandingkan nilai uji pembanding (mininet).

Tabel 3. Nilai UDP *Throughput* Rx

	128 kBps	***256 kBps***	***512 kBps***	***1024 kBps***
Mininet	15.6	31.3	62.5	125
Sw-OF RB	12.44	26.65	58.1	102.65
Gap Rata^2		-8.64		

Catatan*:*
Mininet : Nilai throughput data pembanding dengan mininet
Sw-OF RB : Nilai throughput switch OpenFlow Software-based MikroTik
gabrata^2 : Selisih rata-rata antar nilai throughput UDP prototipe dengan pembanding.
data disajikan dalam kbps.



Gambar 5. UDP *Throughput* Rx

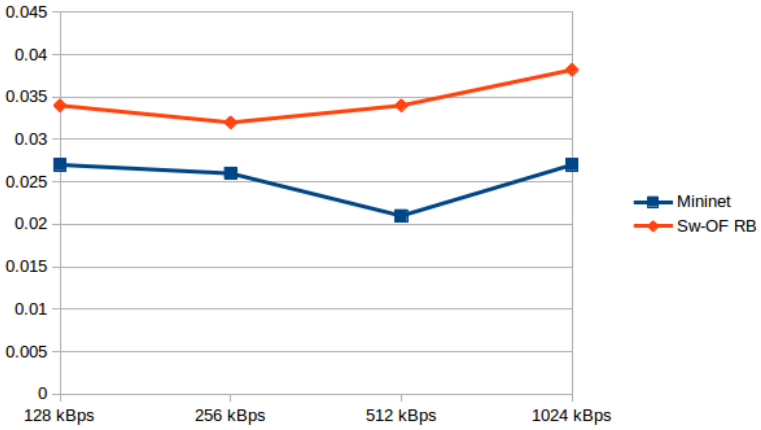
Nilai throughput UDP yang dihasilkan prototipe mempunyai nilai yang rendah dibandingkan dengan mininet sebagai pembanding, namun dengan nilai perbedaan/gap yang signifikan yaitu rata-rata mengalami perbedaan 8.64kBps.

Nilai *jitter* yang dihasilkan dapat di lihat pada tabel 4 dan grafik 6.

Tabel 4. Nilai pengujian *jitter*

	128 kBps	***256 kBps***	***512 kBps***	***1024 kBps***
Mininet	0.027	0.026	0.021	0.027
Sw-OF RB	0.034	0.032	0.034	0.0382
Gab rata^2		0.0093		

Catatan*:*
Mininet : Nilai jitter data pembanding dengan mininet
Sw-OF RB : Nilai jitter switch OpenFlow Software-based MikroTik
gabrata^2 : Selisih rata-rata antar nilai jitter prototipe dengan pembanding.
data disajikan dalam msec.



Gambar 6. Nilai Jitter

Nilai jitter yang diperoleh menunjukkan bahwa prototipe memberikan nilai yang baik dengan nilai selisih rata-rata sebesar 0.0093 msec. Nilai jitter tersebut dapat dinyatakan sama dengan nilai jitter yang diberikan oleh pembandingnya.

4. KESIMPULAN

Dari data pengujian yang dilakukan dapat ditarik kesimpulan bahwa:

1) Pada nilai latency, prototipe mendapatkan nilai tinggi dibandingkan dengan pembandingnya, pada semua paket data.
2) Pada pengujian throughput protokol TCP, prototipe memberikan hasil yang rendah dibandingkan dengan pembandingnya, dengan Nilai gap rata-rata prototipe lebih rendah 1937.5 kbps.
3) Pada pengujian throughput protokol UDP, nilai gap throughput bandwidth protokol UDP rata-rata prototipe lebih rendah 8.64 kBps dibanding nilai pembanding mininet.
4) Pada pengujian jitter protokol UDP, nilai gap jitter protokol UDP rata-rata prototipe lebih tinggi 0.0093 msec dibanding nilai pembanding mininet. Nilai tersebut dapat dikatakan sama dengan nilai yang dihasilkan oleh pembandingnya.

Dengan melihat hasil yang diperoleh, dapat dikatakan bahwa prototipe dapat dijadikan alternatif pengganti switch OpenFlow Software-based, walaupun prototipe masih memberikan nilai performa yang rendah. Perlu dilakukan pengujian tentang pengolahan TCAM oleh prototipe yang diduga menyebabkan nilai yang diperoleh menjadi rendah dibandingkan nilai pembandingnya.

DAFTAR PUSTAKA

[1] Mckeown, N. et al,. "*OpenFlow*: Enabling Innovation in Campus Networks". *ACM SIGCOMM Comput. Commun. Rev*. vol.38. no.2. pp. 69–74. 2008.
[2] Shirazipour, M. et al,. "Realizing packet-optical integration with SDN and OpenFlow 1.1 extensions". *2012 IEEE Int. Conf. Commun*. pp.6633–6637. 2012.
[3] Paper, W. "Infrastructure SDN with Cariden Technologies". pp.1–14. 2012.
[4] Noname. "Software-Defined Networking: The New Norm for Networks [white paper]". *ONF White* Pap. pp.1–12. 2012.
[5] Kartadie, R. et al,. "Prototipe Infrastruktur Software-Defined Network Dengan Protokol *OpenFlow* Menggunakan UBUNTU Sebagai Kontroler". *DASI. 2014.*
[6] Kartadie, R. and Satya, B. "Uji Performa Kontroler Floodlight Dan Opendaylight Sebagai Komponen Utama Arsitektur Software-Defined Network". *SEMNASTEKNOMEDIA ONLINE*. 2015.
[7] Tanutama, L. (1996). *Jaringan Komputer*. 1st ed. Yogyakarta: Elex Media Komputindo.
[8] Chung Yik, E. "Implementation of an Open Flow Switch on Netfpga". Universiti Teknologi Malaysia. 2012.
[9] Hucaby, D. (2007). *CCNP BSCI Official Exam Certification Guide*. 1st ed. Vol.205. no.4594. Indiana: Cisco Press.
[10] Mateo, M, P. "*OpenFlow* Switching Performance", Politecnico Di Torino. 2009.
[11] Muntaner, G, R, D, T. "Evaluation of *OpenFlow* Controllers". p.90. 2012.
[12] Burge, J, E, et al. "What is Rationale and Why Does It Matter*?". Ration. Softw. Eng*. pp. 3–23. 2008