BAB 5 IMPLEMENTASI

Bab ini akan membahas tahapan implementasi kontroler Floodlight, kontroler Maestro, kontroler RYU, kontroler POX, dan kontroler ONOX, pada jaringan *Openflow* juga membentuk lingkungan pengujian dengan tujuan mendapatkan hasil yang dapat diambil sebuah kesimpulan pada bab kesimpulan. Langkah-langkah yang akan dilakukan mencangkup instalasi kontoler, instalasi simulator Cbench, selanjutnya melakukan pengujian kedua kontroler dengan skenario pengujian yang telah ditentukan.

5.1 Implementasi

Implementasi yang akan dilakukan menyesuaikan desain rancangan yang telah ditentukan. Dengan beberapa hal yang perlu disiapkan untuk menerapkan sistem, antara lain mempersiapkan perangkat keras yang akan digunakan dan juga instalasi perangkat lunak diperlukan.

Seperti pada **Gambar 5.1** dimana implementasi kontroler Floodlight, kontroler Maestro, kontroler RYU, kontroler POX, dan kontroler ONOX pada rancang jaringan *Openflow* yang akan dimulai dengan proses intalasi kontroler. Jika semua kontroler telah berhasil terinstal maka akan dilanjutkan pada tahap pengujian konektivitas kontroler dengan simulator mininet, dengan tujuan mengetahui berhasil atau tidaknya instalasi pada semua kontroler. Mininet adalah emulator jaringan yang menciptakan jaringan virtual berupa *host*, *switch*, kontroler, dan *links* untuk *custom routing* yang fleksibel dan *Software Defined Network* (SDN). Pada tahap instalasi kontroler dapat dikatakan berhasil jika semua kontroler yang terinstal dapat terhubung dengan komponen virtual seperti *switch* dan *host* yang diperlukan pada emulator mininet. Jika semua kontroler tidak dapat terhubung dengan komponen virtual pada emulator mininet maka tahap instalasi akan diulang hingga tahap ini berhasil. Kemudian pada tahap selanjutnya adalah proses instalasi simulator Cbench, dimana simulator Cbench akan difungsikan untuk menguji performansi dari semua kontroler yang akan diuji. Tahap terakhir adalah pengambilan data atau hasil dari pengujian *throughput* dan *latency*, dimana tahap ini dilakukan sesuai dengan skenario pengujian yang telah ditentukan pada bab sebelumnya.

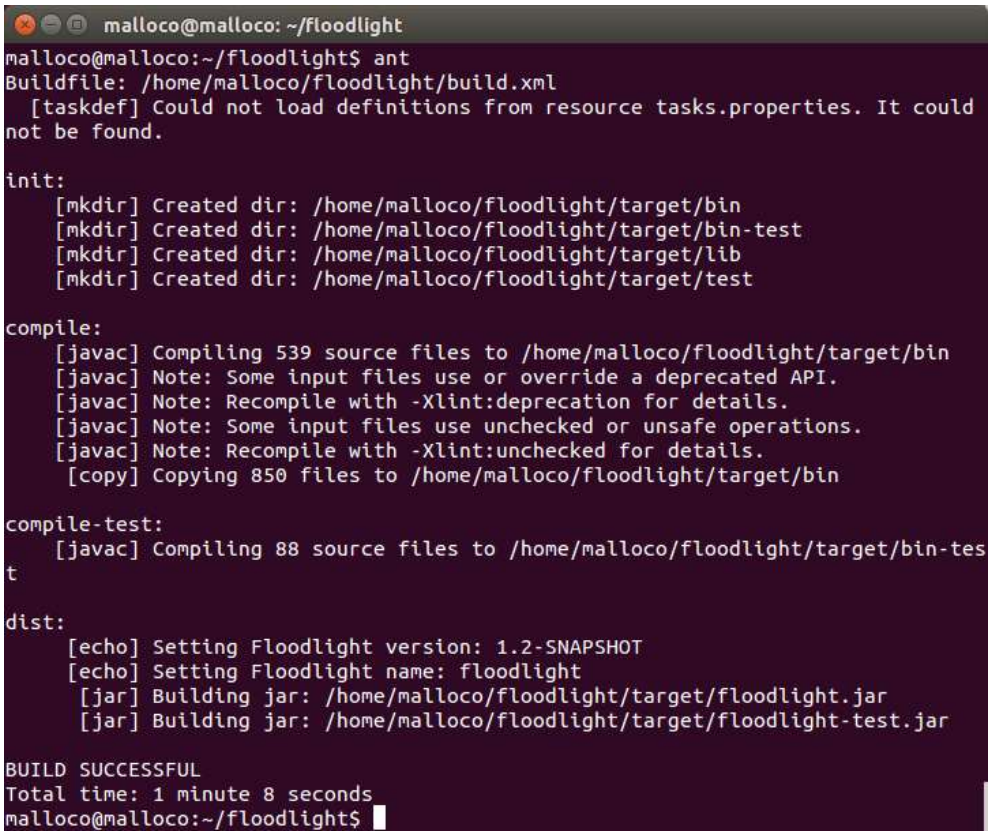
Gambar 5.1 Flowchart implementasi dan pengujian sistem

Sumber: Penulis

5.1.1 Instalasi Kontroler Floodlight

Berikut merupakan tahap-tahap dalam instalasi kontroler Floodlight:

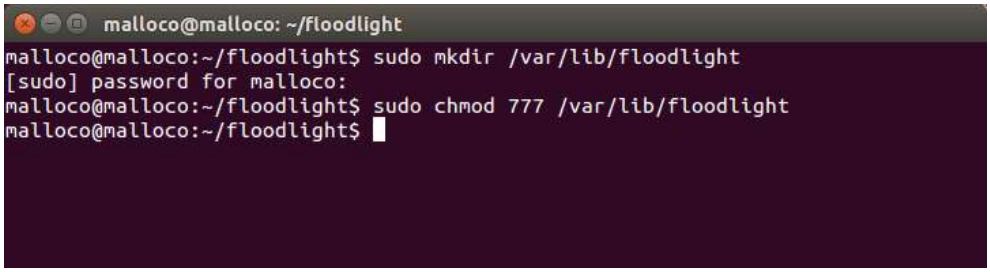
1. Instalasi dependensi Floodlight
   $sudo apt-get install build-essential ant maven python-dev
2. Mengkloning file Floodlight melalui website github
   $git clone git://github.com/floodlight/floodlight.git
3. Kemudian masuk ke direktori floodlight
   $cd floodlight
4. Melakukan update versi yang paling stabil
   $git checkout stable
5. Build Floodlight dengan perintah
   $ant



Gambar 5.2 Built Floodlight

Sumber: Penulis

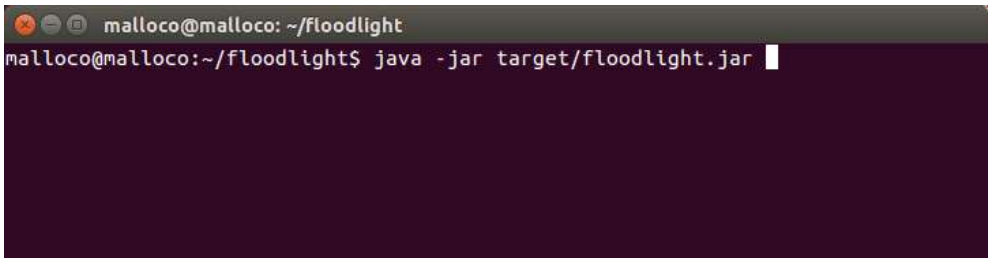
6. Membuat direktori baru pada */var/lib/* dengan nama floodlight yang dapat diakses dengan perintah
   $sudo mkdir /var/lib/floodlight
   $sudo chmod 777 /var/lib/floodlight



Gambar 5.3 Built Floodlight

Sumber: Penulis

Jika semua telah berhasil terinstal maka Floodlight dapat dijalankan dengan perintah *$ java -jar target/floodlight.jar*seperti yang terlihat pada **Gambar 5.4**.



Gambar 5.4 Perintah untuk menjalankan Floodlight

Sumber: Penulis

Jika pada instalasi kontroler Floodlight sudah benar, selanjutnya adalah tahap instalasi kontroler Maestro.

5.1.2 Instalasi Kontroler Maestro

Berikut merupakan tahap-tahap dalam instalasi kontroler Maestro:

1. Mengunduh file kontroler Maestro yang disediakan pada website maestro-platform.googlecode.com
2. Karena file yang didapat berformat .zip maka tahap selanjutnya adalah dengan meng-unzip file Maestro yang telah diunduh. Maka akan secara otomatis mendapatkan direktori *Maestro-0.1.0*
3. Tahap selanjutnya masuk pada direktori *Maestro-0.1.0*
4. Mengkompilasi semua file dengan perintah *$ ant*
5. Jika tahap kompilasi telah berhasil dilakukan maka tahap selanjutnya adalah memasuki direktori *Maestro-0.1.0*
6. Dan untuk menjalankan atau mengaktifkan kontroler Maestro dapat digunakan perintah
   $java -cp build/ sys.Main conf/Openflow.conf conf/learningSwitch.dag 1

5.1.3 Instalasi Kontroler RYU

Berikut merupakan tahap-tahap dalam instalasi kontroler RYU:

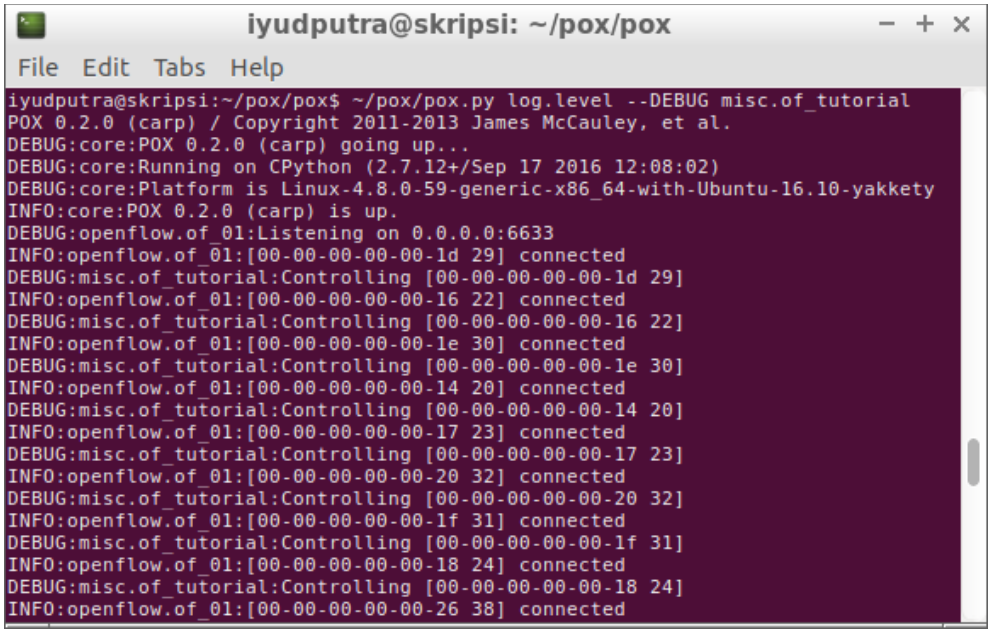
1. Mengkloning file RYU melalui website github
   $git clone git://github.com/osrg/ryu.git
2. Pada tahap install dengan perintah
   $cd ryu; python ./setup.py install

3. Jika yang digunakan adalah VM RYU, respositori sudah dikloning secara otomatis. Maka perlu update repositori dengan perintah
   $git pull
4. Dan untuk menjalankan atau mengaktifkan kontroler RYU menggunakan perintah
   $ ../bin/ryu-manager ryu/app/simple_switch.py

5.1.4 Instalasi Kontroler POX

POX secara langsung telah terpasang pada VM Mininet 2.2, namun juga dapat diinstal sendiri, dan berikut merupakan tahap-tahap dalam instalasi kontroler POX:

1. Mengkloning file POX melalui website github dengan perintah
   $git clone http://github.com/noxrepo/pox
2. Langkah selanjutnya adalah dengan memasuki direktori POX
   $cd pox
3. Untuk mengetahui informasi lebih banyak mengenai versi POX yang digunakan dengan menggunakan perintah
   $ git checkout dart
4. Dan untuk menjalankan atau mengaktifkan kontroler POX dapat menggunakan perintah
   $ ./pox/pox.py log.level –DEBUG misc.of_tutorial



Gambar 5.5 Perintah untuk menjalankan kontroler POX

Sumber: Penulis

5.1.5 Instalasi Kontroler ONOS

Berikut merupakan tahap-tahap dalam instalasi kontroler ONOS:

1. Mengunduh file ONOS yang disediakan pada website onosproject.org dengan perintah
   sudo wget -c http://downloads.onosproject.org/release/onos-$ONOS_VERSION.tar.gz
2. Membuka file dengan perintah
   sudo tar xzf onos-$ONOS_VERSION.tar.gz
3. Merubah nama direktori menjadi "onos"
   sudo mv onos-$ONOS_VERSION onos
4. Untuk menjalankan atau mengaktifkan kontroler ONOS dengan perintah
   $ /opt/onos/bin/onos-service start

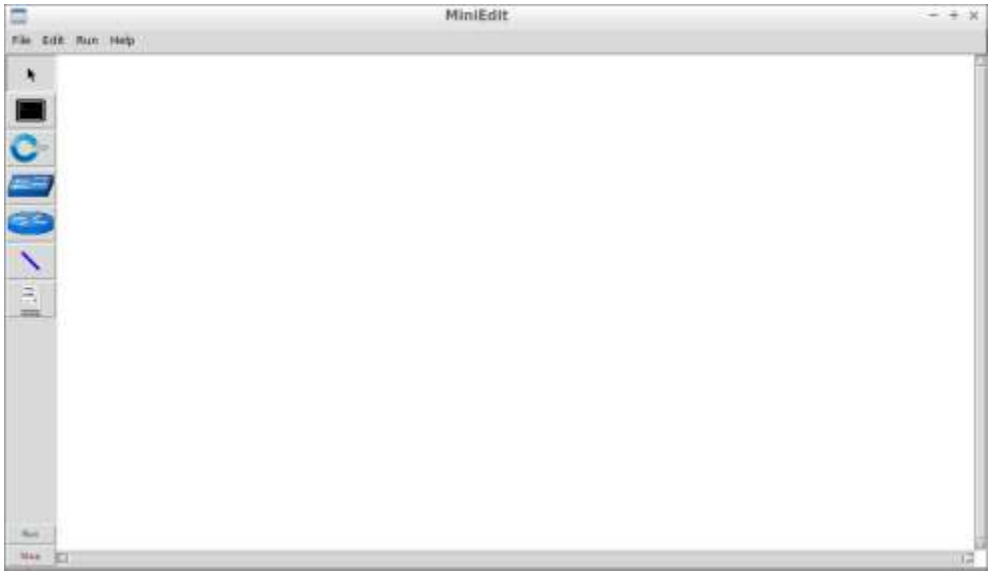
5.1.6 Instalasi Mininet dan Uji Konektivitas

Jika kedua kontroler baik kontroler Floodlight maupun kontroler Maestro telah berhasil diinstal, maka tahap selanjutnya adalah menguji konektivitas kontroler menggunakan emulator Mininet. Tahap ini dilakukan dengan tujuan untuk mengetahui konektivitas kontroler Floodlight dan kontroler Maestro terhadap infrastruktur jaringan seperti *Switch Openflow*.

Dengan begitu emulator Mininet harus diinstalasi untuk pengujian konektivitas kedua kontroler yang akan diuji. Adapun tahap-tahap instalasi emulator Mininet;

1. Mengunduh file Mininet pada website github menggunakan perintah
   $git clonegit://github.com/download/mininet/mininet
2. Selanjutnya masuk ke direktori mininet dengan perintah
   $cd mininet
3. Kemudian memilih versi Mininet yang akan digunakan dengan perintah
   $git tag
4. Untuk mengaktifkan versi Mininet yang akan diinstal atau yang akan digunakan dapat dilakukan dengan perintah
   $git checkout <versi Mininet>
5. Selanjutnya adalah mengeksekusi file *install.sh* dengan perintah
   $/mininet/util/install.sh -a

Setelah tahap instalasi mininet telah dilakukan, maka emulator mininet dapat dijalan dengan menjalankan file miniedit.py yang terdapat pada direktori /mininet/example.



Gambar 5.6 Tampilan GUI emulator Miniedit

Sumber: Penulis

5.1.6.1 Uji Konektivitas Kontroler Floodlight

Pada tahap ini akan dijelaskan mengenai langkah-langkah pada uji konektivitas kontroler Floodlight. Topologi yang akan digunakan dalam uji konektivitas menggunakan topologi yang ditetapkan sebagai topologi pengujian. Adapun langkah-langkah uji konektivitas tersebut adalah sebagai berikut:

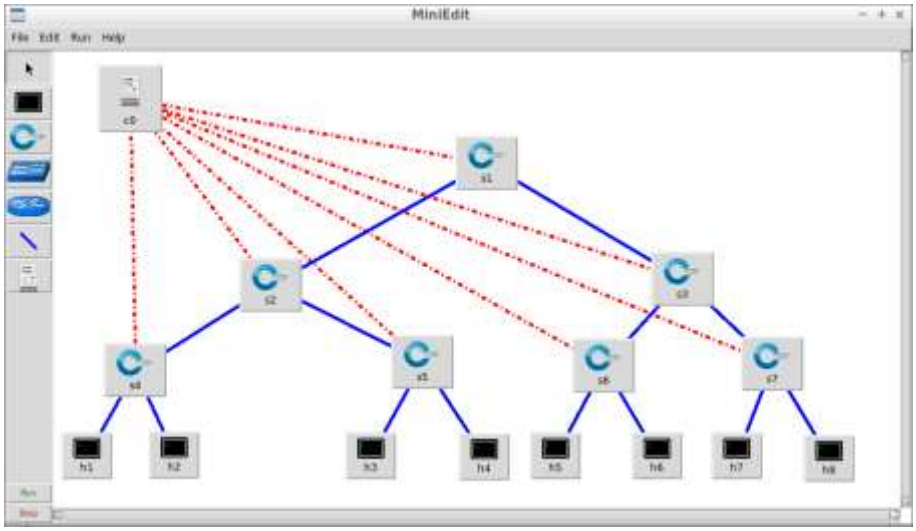
1. Menjalankan kontroler Floodlight dengan perintah

   $java –jar target / floodlight.jar yang berada pada direktori floodlight.

2. Menjalankan emulator Mininet dengan membuka file miniedit.py yang berada pada direktori /miniedit/example dengan perintah

   $ ./sudo miniedit.py

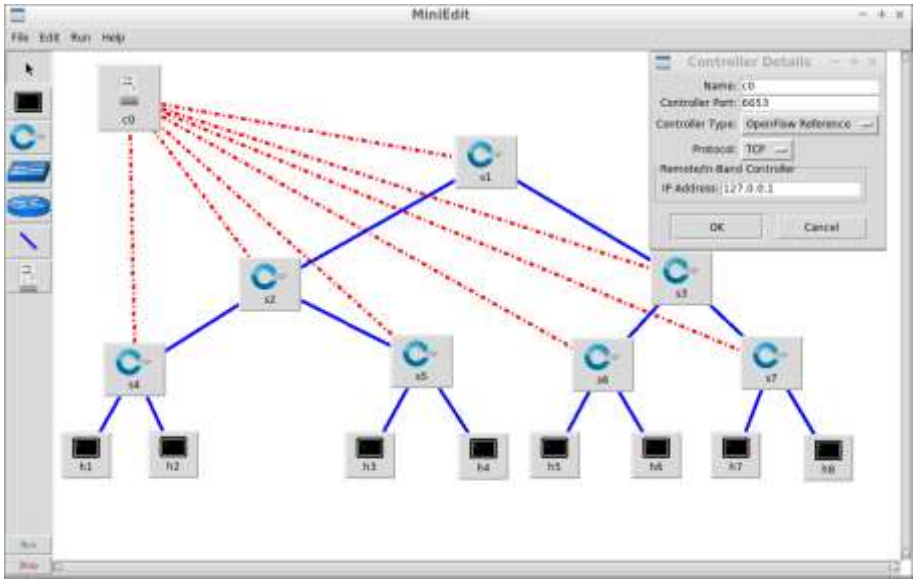
3. Pada emulator Mininet digunakan satu buah kontroler, diberikan tujuh buah *Switch Openflow*, dan delapan buah PC sebagaimana topologi yang akan digunakan dalam pengujian. Dengan cara menarik komponen yang akan digunakan dari panel komponen kemudian menghubungkan perangkat dengan menggunakan komponen kabel yang juga disediakan dalam emulator mininet. Seperti halnya pada **Gambar 5.7**



Gambar 5.7 Komponen topologi pada Mininet

Sumber: Penulis

4. Dikarenakan kontroler Floodlight berjalan pada *port* 6653, maka pada kolom *controller port* diberikan *port* 6653. Dan pada *controller type* menggunakan type *remote controller.*



Gambar 5.8 Setting remote controller untuk Floodlight

Sumber: Penulis

5. Jika pada *controller details* telah selesai dilakukan maka selanjutnya adalah menjalankan emulator Mininet dengan menekan tombol *run* yang berada di kiri bawah emulator.

5.1.6.2 Uji Konektivitas Kontroler Maestro

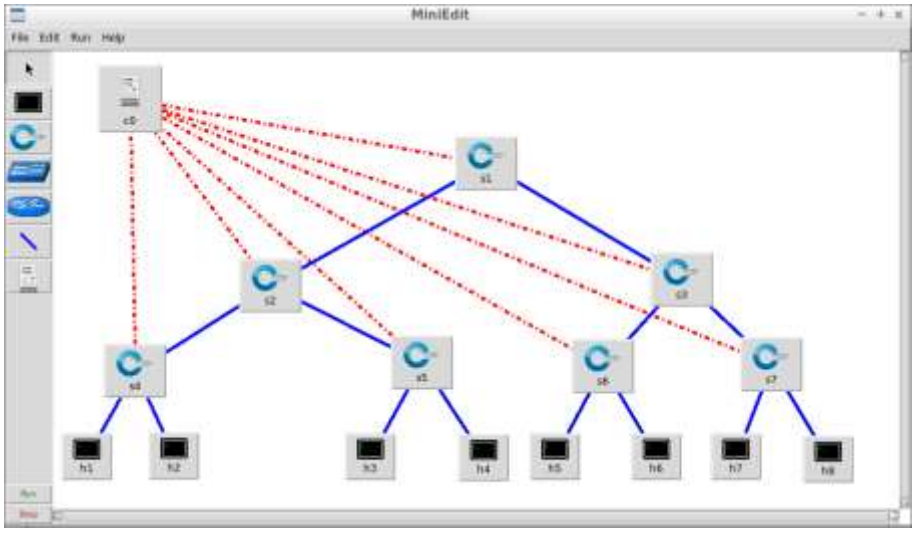
Seperti halnya pada tahap kontroler floodlight, uji konektivitas pada kontroler Maestro juga akan dilakukan dengan tujuan mengetahui konektivitas kontroler Maestro dengan topologi yang digunakan untuk pengujian. Berikut adalah langkah-langkah konektivitas kontroler Maestro:

1. Menjalankan kontroler Maestro dengan perintah

   $java -cp build/ sys.Main conf/Openflow.conf conf/learningSwitch.dag 1 yang berada pada direktori *Maestro-0-1-0*

2. Menjalankan emulator Mininet dengan membuka file miniedit.py yang berada pada direktori */miniedit/example* dengan perintah *./sudo miniedit.py*

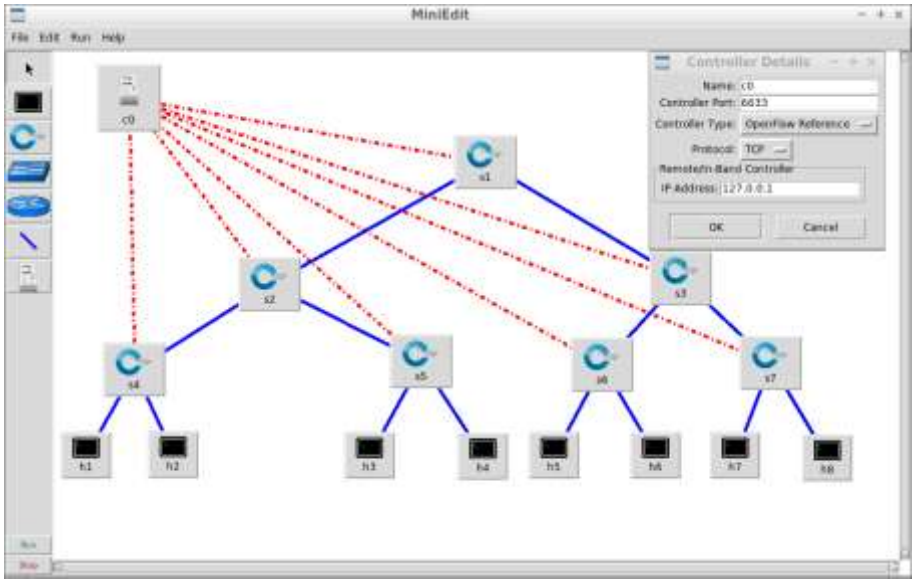
3. Pada emulator Mininet digunakan satu buah kontroler, diberikan tujuh buah *Switch Openflow*, dan delapan buah PC sebagaimana topologi yang akan digunakan dalam pengujian. Dengan cara menarik komponen yang akan digunakan dari panel komponen kemudian menghubungkan perangkat dengan menggunakan komponen kabel yang juga disediakan dalam emulator mininet. Seperti halnya pada **Gambar 5.9**



Gambar 5.9 Komponen topologi pada Mininet

Sumber: Penulis

4. Dikarenakan kontroler Floodlight berjalan pada *port* 6633, maka pada kolom *controller port* diberikan *port* 6633. Dan pada *controller type* menggunakan type *remote controller.*



Gambar 5.10 Setting remote controller untuk Maestro

Sumber: Penulis

5. Sama halnya pada Floodlight. Jika pada *controller details* telah selesai dilakukan maka selanjutnya adalah menjalankan emulator Mininet dengan menekan tombol *run* yang berada di kiri bawah emulator.

5.1.6.3 Uji Konektivitas Kontroler RYU

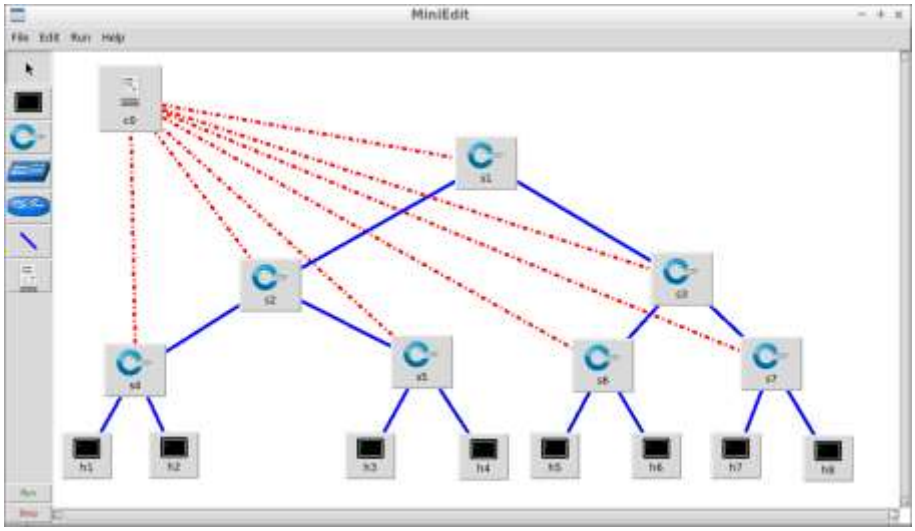
Pada tahap ini akan dijelaskan mengenai langkah-langkah pada uji konektivitas kontroler RYU. Topologi yang akan digunakan dalam uji konektivitas menggunakan topologi yang ditetapkan sebagai topologi pengujian. Adapun langkah-langkah uji konektivitas tersebut adalah sebagai berikut:

1. Menjalankan kontroler RYU dengan perintah

   $ PYTHONPATH ../bin/ryu-manager ryu/app/simple_switch.py

2. Menjalankan emulator Mininet dengan membuka file miniedit.py yang berada pada direktori */miniedit/example* dengan perintah *./sudo miniedit.py*

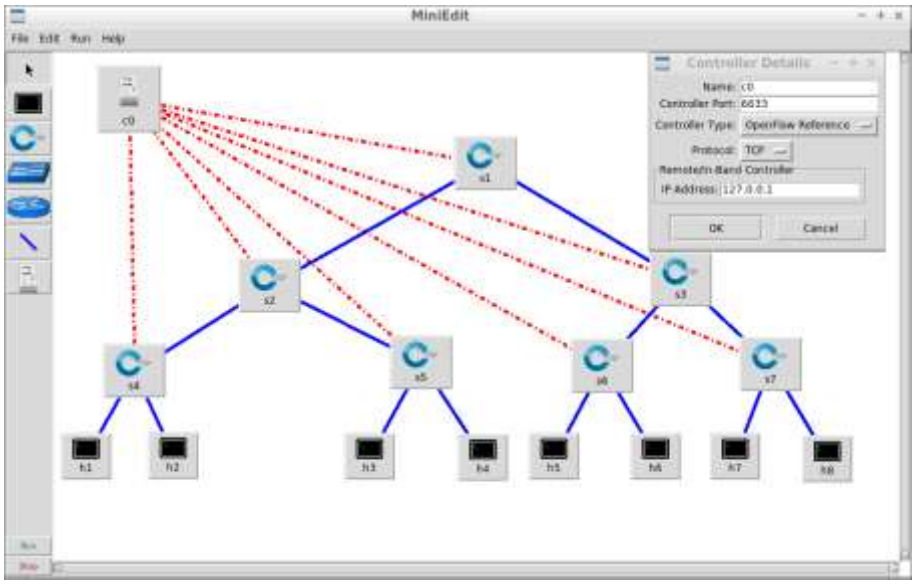
3. Pada emulator Mininet digunakan satu buah kontroler, diberikan tujuh buah *Switch Openflow*, dan delapan buah PC sebagaimana topologi yang akan digunakan dalam pengujian. Dengan cara menarik komponen yang akan digunakan dari panel komponen kemudian menghubungkan perangkat dengan menggunakan komponen kabel yang juga disediakan dalam emulator mininet. Seperti halnya pada **Gambar 5.11**



Gambar 5.11 Setting remote controller untuk RYU

Sumber: Penulis

4. Dikarenakan kontroler RYU secara default berjalan pada *port* 6633, maka pada kolom *controller port* diberikan *port* 6633. Dan pada *controller type* menggunakan type *remote controller.*



Gambar 5.12 Setting remote controller untuk RYU

Sumber: Penulis

5. Sama halnya pada kontroler-kontroler yang sebelumnya. Jika pada *controller details* telah selesai dilakukan maka selanjutnya adalah menjalankan emulator Mininet dengan menekan tombol *run* yang berada di kiri bawah emulator.

5.1.6.4 Uji Konektivitas Kontroler POX

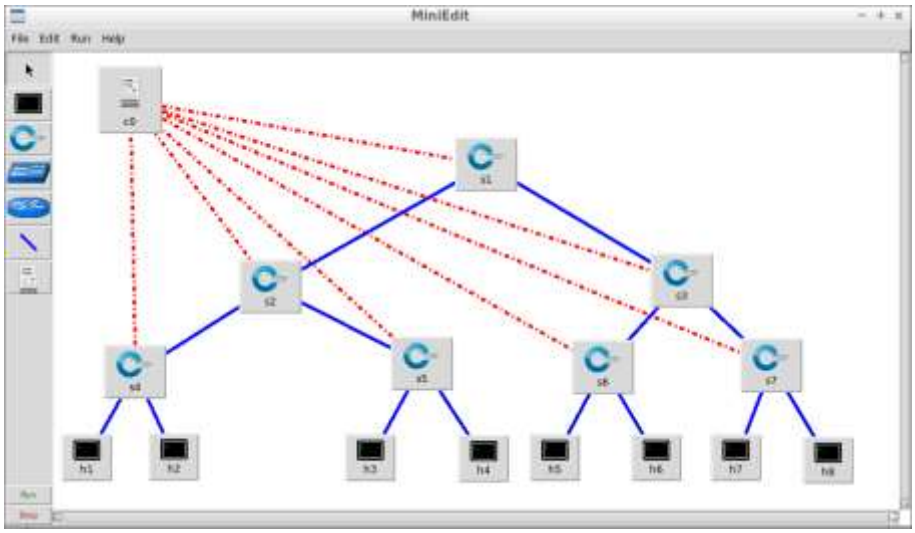
Pada tahap ini akan dijelaskan mengenai langkah-langkah pada uji konektivitas kontroler POX. Topologi yang akan digunakan dalam uji konektivitas menggunakan topologi yang ditetapkan sebagai topologi pengujian. Adapun langkah-langkah uji konektivitas tersebut adalah sebagai berikut:

1. Menjalankan kontroler POX dengan perintah

   $ ~pox/pox.py log.level –DEBUG misc.of_tutorial

2. Menjalankan emulator Mininet dengan membuka file miniedit.py yang berada pada direktori */miniedit/example* dengan perintah *./sudo miniedit.py*

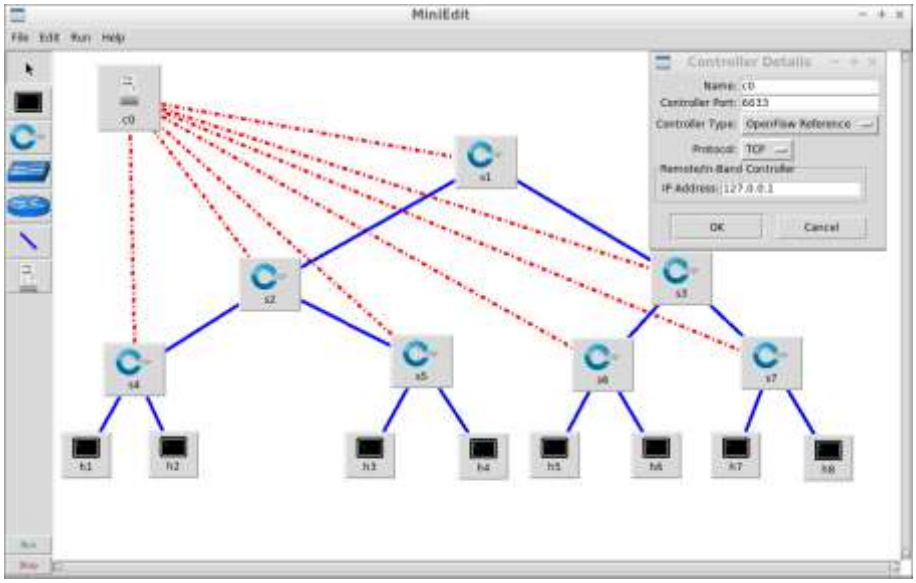
3. Pada emulator Mininet digunakan satu buah kontroler, diberikan tujuh buah *Switch Openflow*, dan delapan buah PC sebagaimana topologi yang akan digunakan dalam pengujian. Dengan cara menarik komponen yang akan digunakan dari panel komponen kemudian menghubungkan perangkat dengan menggunakan komponen kabel yang juga disediakan dalam emulator mininet. Seperti halnya pada **Gambar 5.13**



Gambar 5.13 Setting remote controller untuk POX

Sumber: Penulis

4. Dikarenakan kontroler POX secara default berjalan pada *port* 6633, maka pada kolom *controller port* diberikan *port* 6633. Dan pada *controller type* menggunakan type *remote controller.*



Gambar 5.14 Setting remote controller untuk POX

Sumber: Penulis

5. Sama halnya pada kontroler-kontroler yang sebelumnya. Jika pada *controller details* telah selesai dilakukan maka selanjutnya adalah menjalankan emulator Mininet dengan menekan tombol *run* yang berada di kiri bawah emulator.

5.1.6.5 Uji Konektivitas Kontroler ONOS

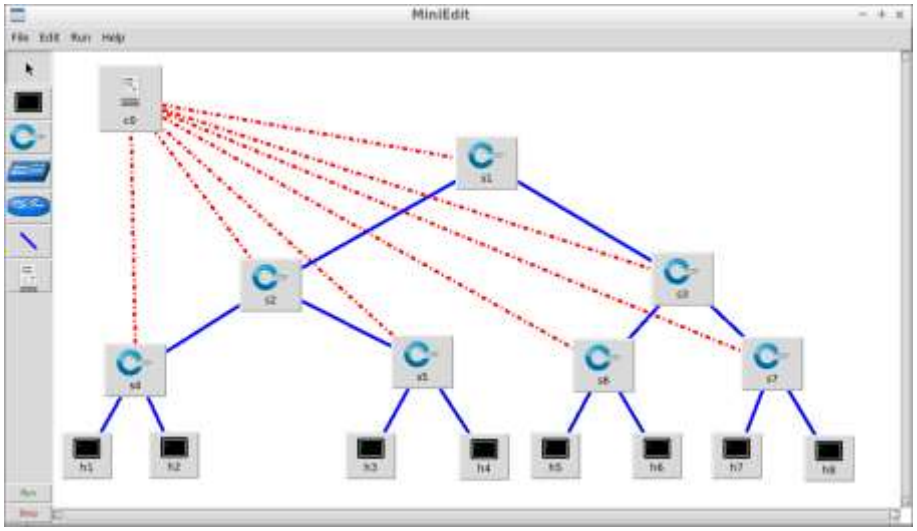
Pada tahap ini akan dijelaskan mengenai langkah-langkah pada uji konektivitas kontroler ONOS. Topologi yang akan digunakan dalam uji konektivitas menggunakan topologi yang ditetapkan sebagai topologi pengujian. Adapun langkah-langkah uji konektivitas tersebut adalah sebagai berikut:

1. Menjalankan kontroler ONOS dengan perintah

   $ sudo docker start onos1

2. Menjalankan emulator Mininet dengan membuka file miniedit.py yang berada pada direktori */miniedit/example* dengan perintah *./sudo miniedit.py*

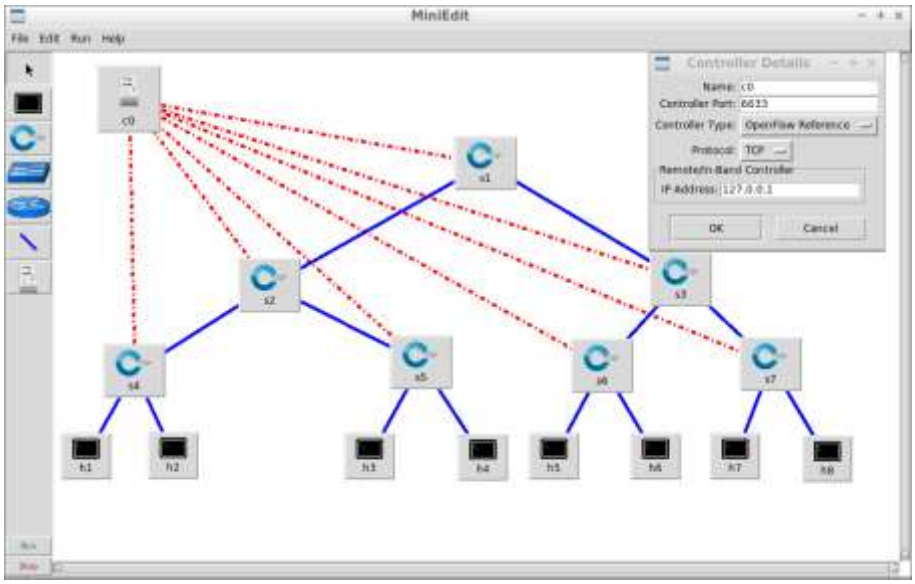
3. Pada emulator Mininet digunakan satu buah kontroler, diberikan tujuh buah *Switch Openflow*, dan delapan buah PC sebagaimana topologi yang akan digunakan dalam pengujian. Dengan cara menarik komponen yang akan digunakan dari panel komponen kemudian menghubungkan perangkat dengan menggunakan komponen kabel yang juga disediakan dalam emulator mininet. Seperti halnya pada **Gambar 5.15**



Gambar 5.15 Setting remote controller untuk ONOS

Sumber: Penulis

4. Dikarenakan kontroler ONOS secara default berjalan pada *port* 6633, maka pada kolom *controller port* diberikan *port* 6633. Dan pada *controller type* menggunakan type *remote controller.*



Gambar 5.16 Setting remote controller untuk ONOS

Sumber: Penulis

5. Sama halnya pada kontroler-kontroler yang sebelumnya. Jika pada *controller details* telah selesai dilakukan maka selanjutnya adalah menjalankan emulator Mininet dengan menekan tombol *run* yang berada di kiri bawah emulator.

5.1.7 Instalasi Simulator Cbench

Pada tahap ini akan dijelaskan mengenai langkah-langkah instalasi simulator Cbench. Berikut adalah langkah-langkah dalam menginstal Cbench:

1. Intalasi dependensi yang diperlukan dengan perintah

   $sudo apt-get install autoconf automake libtool libsnmp-dev libpcap-dev libpcap-dev libconfig8-dev

2. Mengunduh file *Openflow* melalu website *github* dengan perintah

   $git clone git://gitosis.stanford.edu/Openflow.git

3. Mengaktifkan versi source code *Openflow* dengan perintah

   $cd Openflow; git checkout -b mybranch origin/release/1.0.0

4. Mengunduh paket Oflops pada website github dengan perintah

   $git clone git://gitsis.stanford.edu/oflops.git

5. Mengaktifkan source core Oflops dengan perintah

   $cd oflops ; sh ./boot.sh ; ./configure –with-Openflow-src-dir=/home/iyudputra/Openflow; make; make install