BAB 5
PERANCANGAN DAN IMPLEMENTASI

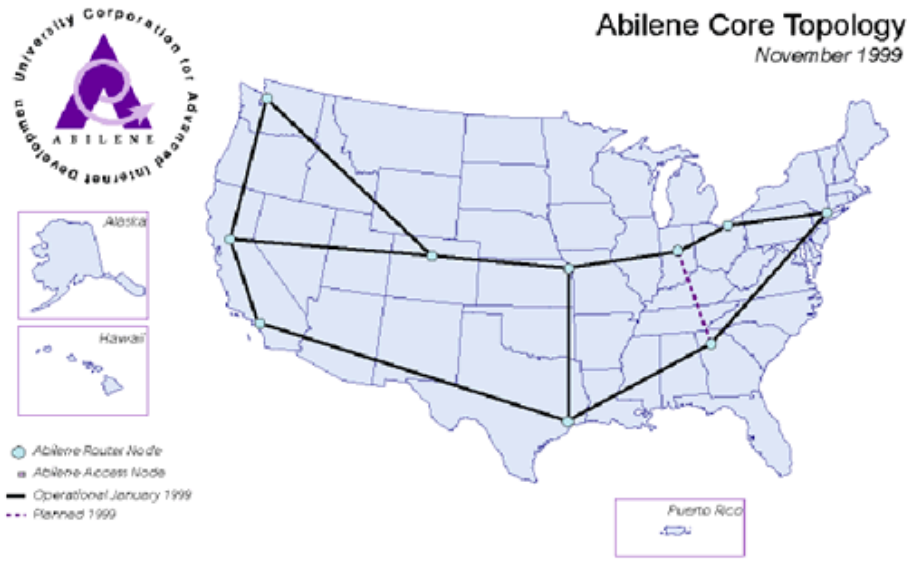
Pada bagian ini dijelaskan langkah-langkah untuk merancang dan mengimplementasikan sebuah sistem *multipath routing* dengan Y*en algorithm* pada *OpenFlow* SDN dengan berpedoman pada metodologi penelitian yang telah dibahas.

5.1 Perancangan

Perancangan akan dilakukan untuk kebutuhan yang telah di spesifikan di bab sebelumnya.

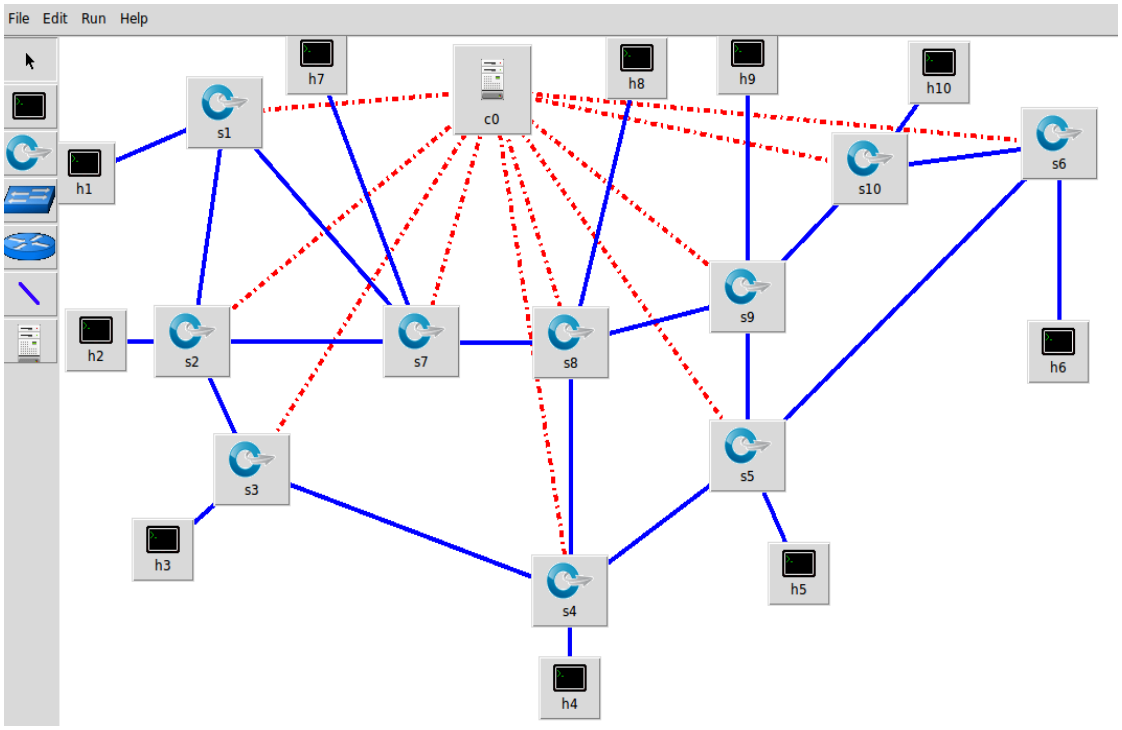
5.1.1 Perancangan Topologi

Untuk dapat memanfaatkan *multipath routing* dengan *Yes algorithm,* diperlukan topologi yang memiliki berbagai macam jalur yang dapat dilalui suatu *node* jaringan. Terdapat beberapa *path* yang dapat dilalui oleh set pasangan tersebut dapat dilihat pada gambar 5.1.



Gambar 5.1 Peta topologi *Abilene Network*

Sumber: stanford (2015)

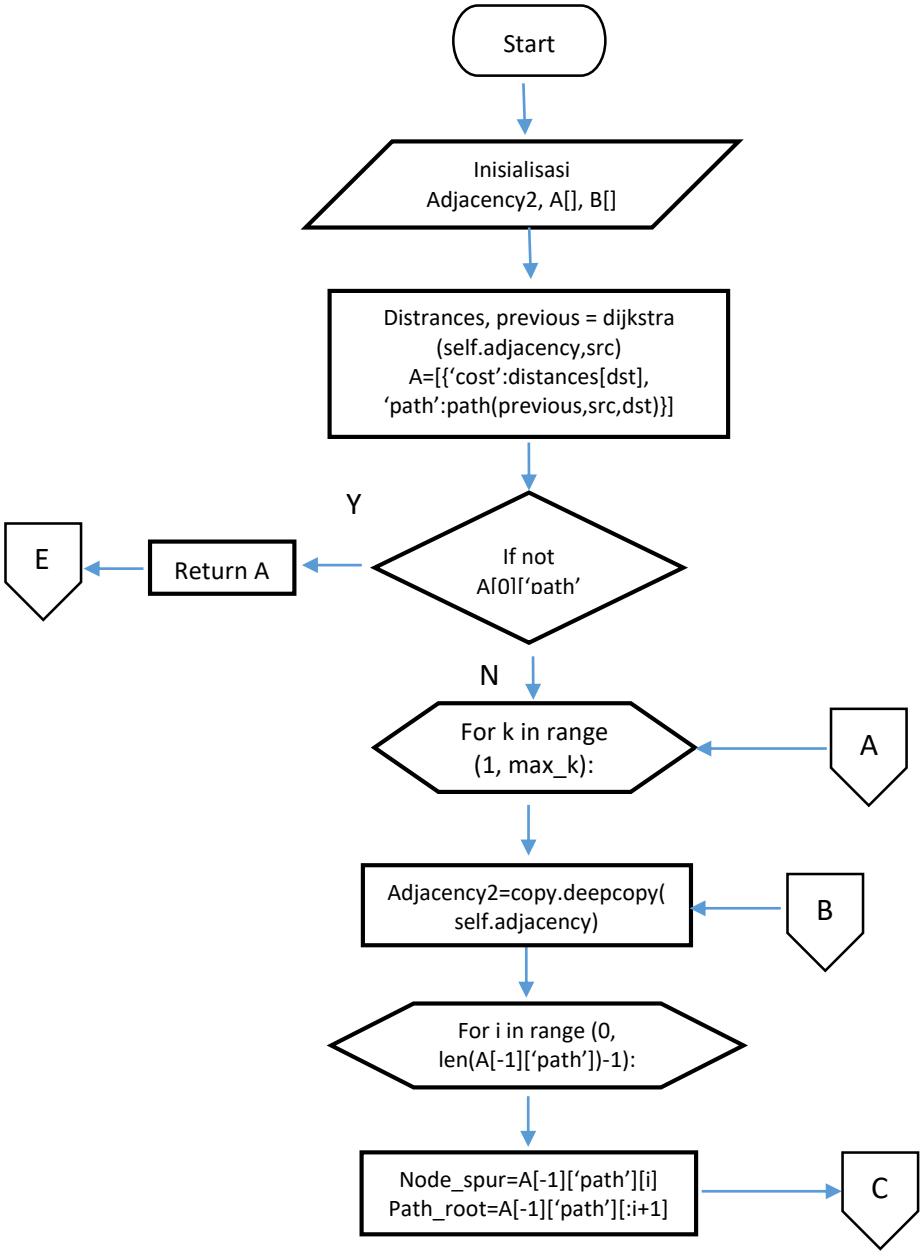


Gambar 5.2 Topologi *Abilene Network* pada *Mininet*

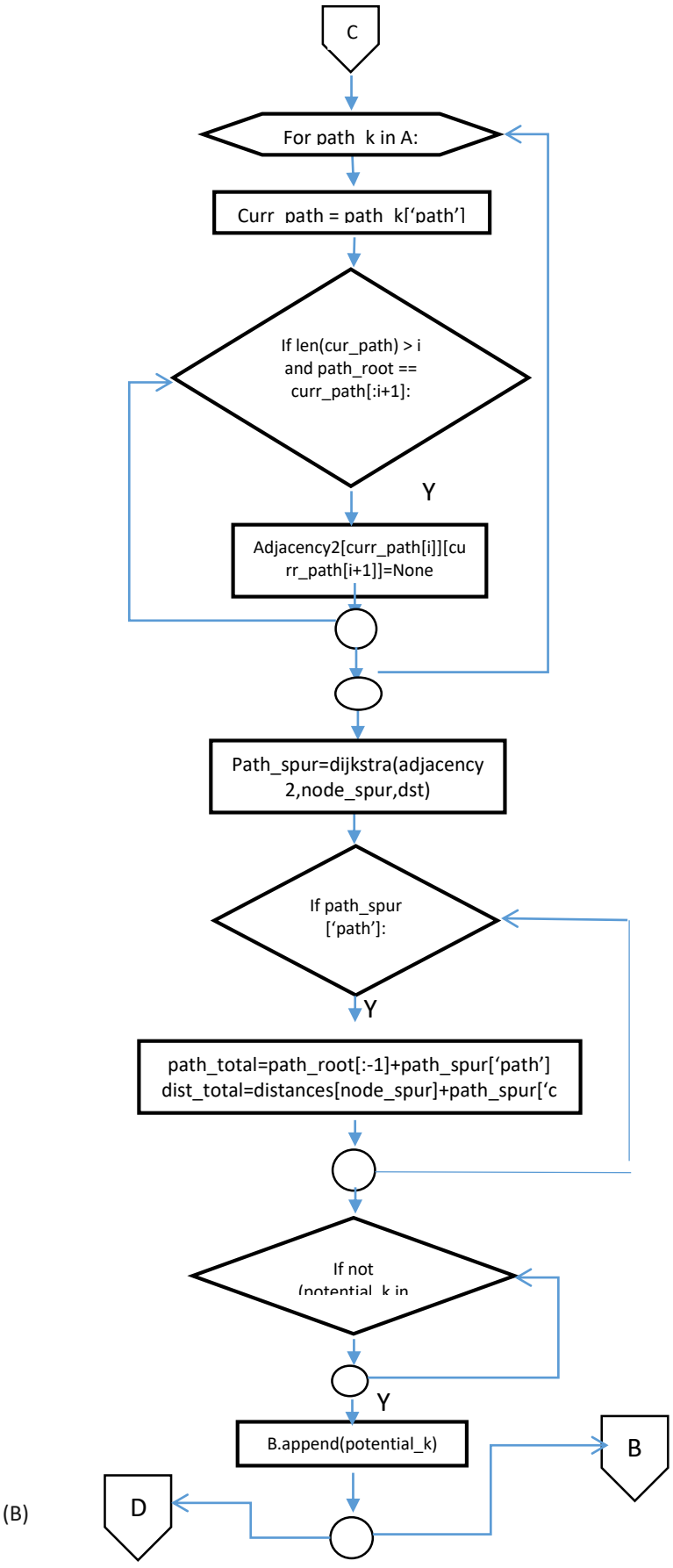
Pada topologi *Abilene* yang dipetakan pada *Mininet*, terdapat 10 *node* yang merepresentasikan suatu kota di Amerika Serikat yang bertindak sebagai *IP router node*. Di topologi ini jika *link* bertambah pada topologi akan membuat pemilihan jalur semakin banyak dan jika *switch* bertambah pada topologi akan menambah *cost* pada *link* tersebut. Dilakukan *setting* ukuran bandwidth pada tiap-tiap *link*. Bandwidth disetting pada ukuran 1000 Mbps.

5.1.2 Algoritme Pencarian dan Pemilihan Jalur

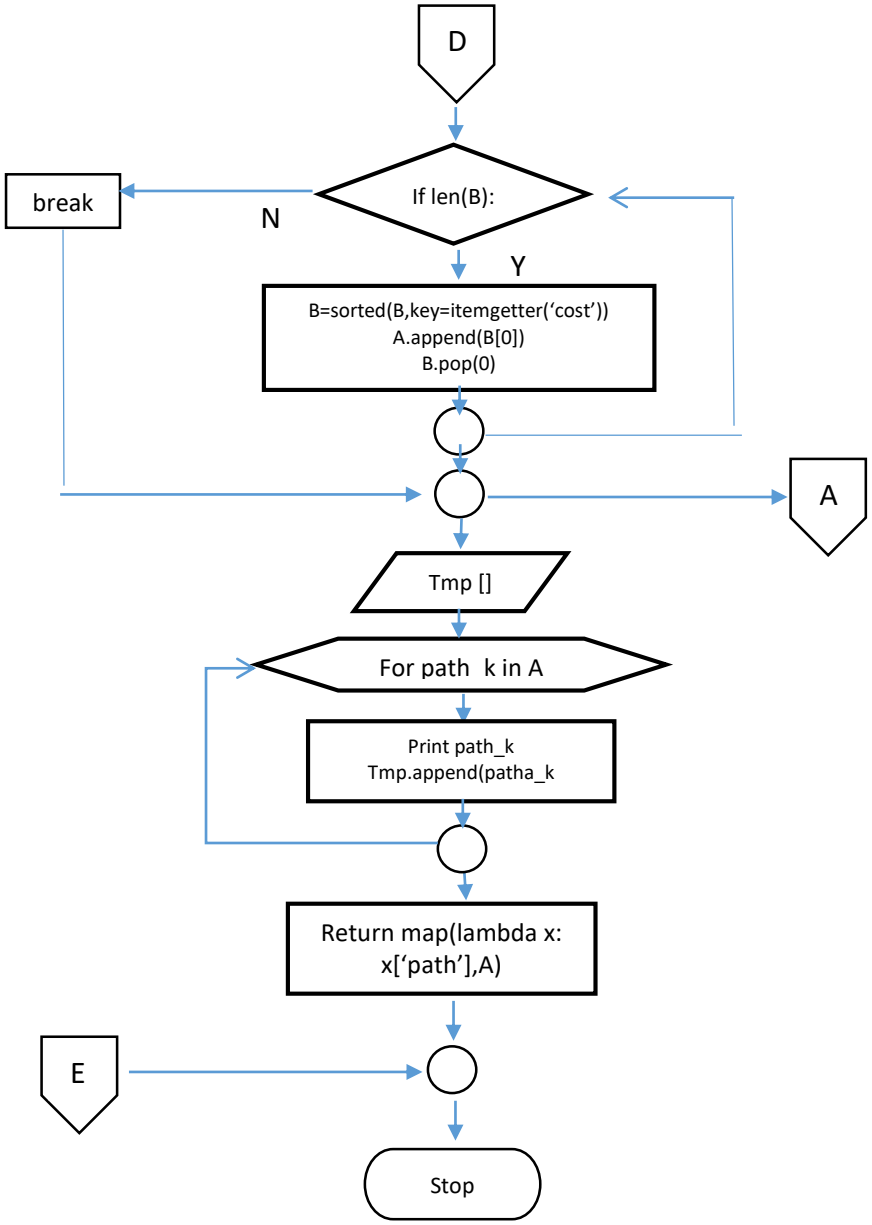
Berdasarkan perancangan topologi dan observasi terhadap sistem yang akan di bangun, algoritme pencarian jalur pada penelitian ini menggunakan *Yen algorithm*. *Yen algorithm* akan digunakan untuk mencari jalur terpendek sebanyak K yang dibutuhkan oleh sistem. *Source code* untuk *Yen algorithm* dapat dilihat pada gambar 2.3. Berdasarkan beberapa jalur yang telah didapatkan dengan algoritme pencarian jalur, maka dilakukan pemilihan *path* dengan mengurutkan *path* dengan *cost* terendah. Jalur yang memiliki *cost* paling kecil akan digunakan sebagai K-*Path* pengiriman data. Selengkapnya akan ditampilkan pada *flowchart* pencarian dan pemilihan jalur.


Start
Inisialisasi
Adjacency2, A[], B[]
Distrances, previous = dijkstra
(self.adjacency,src)
A=[{'cost':distances[dst],
'path':path(previous,src,dst)}]
Y
If not
A[0]['path'
E
Return A
N
For k in range
(1, max_k):
A
Adjacency2=copy.deepcopy(
self.adjacency)
B
For i in range (0,
len(A[-1]['path'])-1):
Node_spur=A[-1]['path'][i]
Path_root=A[-1]['path'][:i+1]
C

(A)


C
For path k in A:
Curr path = path k['path']
If len(cur_path) > i
and path_root ==
curr_path[:i+1]:
Y
Adjacency2[curr_path[i]][cu
rr_path[i+1]]=None
Path_spur=dijkstra(adjacency
2,node_spur,dst)
If path_spur
['path']:
Y
path_total=path_root[:-1]+path_spur['path']
dist_total=distances[node_spur]+path_spur['c
If not
(potential k in
Y
B.append(potential_k)
B
D

(B)



(C)

Gambar 5.3 Flowchart algoritme pencarian dan pemilihan jalur

Flowchart algoritme pencarian jalur dan pemilihan jalur pengiriman data ditampilkan pada gambar 5.3 (a)(b)(c). Berikut penjelasan *flowchart:*

1. Sistem pertama kali melakukan inisialisasi variabel adjcency untuk menghubuhkan antara *switch.*

2. lalu sistem akan mengambil keluaran dari fungsi *Dijkstra* dan dimasukkan pada variabel yang berbeda. Data dari variabel *Dijkstra* ini akan digunakan sebagai *path* pertama dari beberapa jalur yang akan dicari.
3. Sistem akan melakukan perulangan untuk mencari path lain sebanyak MAX_K.
4. Urutan simpul dari sumber ke simpul jalur dari jalur k-terpendek sebelumnya.
5. Setelah sistem menemukan *path* sebanyak K yang dibutuhkan sistem akan melakukan *sorting* berdasarkan *cost* terendah.
6. Fungsi ini akan melakukan pengulangan sampai menemukan *path.*
7. Setelah semua *path* berhasil di dapatkan, sistem akan memasukan data ke tmp.

5.1.3 *PseudoCode Yen algorithm*

Untuk pencarian dan pemilihan jalur terpendek antara dua buah *host* pada jaringan digunakan *Yen algorithm.*

```
1  Function YenKSP (graph, source, dst, max_K) :
2  A[0] = Dijkstra (graph, source, dst)
3  B=[];
4  for K from 1 to max_K:
5  for i from 0 to [A − 1 ] − 1:
6  node_spur = A[-1]['path'][i]
7  path_root = A[-1]['path'][:i+1]
8  for each path in A:
9  if rootPath == p.nodes(0, i):
10 remove p.edge(i,i + 1) from Graph;
11 for each node rootPathNode in rootPath except spurNode:
12 remove rootPathNode from Graph;
13
14
```

```
15  spurPath = Dijkstra(graph, node_spur, dst);
16  totalPath = path_root + spurPath;
17  B.append(totalPath);
18  if B is empty:
19  break;
20  B.sort();
21  A[k] = B[0];
22  B.pop();
23  Return A
```

Gambar 5.4 *Pseudocode Yen algorithm*

Inputan *Yen Algorithm* adalah

Baris 1: Tentukan jalur terpendek dari sumber k.

Baris 2: Inisialisasi tumpukan untuk menyimpan jalur K terpendek.

Baris 4: simpul pertama ke simpul berikutnya dan kembali ke simpul sebelumnya dari jalur terpendek-k sebelumnya.

Baris 5: jalur yang di lewati dari jalur k-*shortest* sebelumnya.

Baris 6: Urutan simpul dari sumber ke simpul jalur dari jalur k-terpendek sebelumnya.

Baris 10: Hapus yang merupakan bagian dari jalur terpendek sebelumnya yang memiliki jalur yang sama.

Baris 15: Hitung jalur dari simpul sumber ke simpul tujuan.

Baris 16: Seluruh jalan terdiri dari jalur akar dan jalur pacu.

Baris 17: Menambahkan jalur K-terpendek.

Baris 18: Menangani kasus yang tidak ada jalur atau tidak ada jalur yang tersisa. Hal ini bisa terjadi jika jalur sudah habis atau tidak ada jalur sama sekali.

Baris 20: Urutkan jalur K-terpendek potensial dengan *cost*.

Baris 21: Tambahkan jalur *Cost* terendah menjadi jalur k-terpendek.

5.2 Implementasi

Berikut ini langkah-langkah yang dilakukan pada tahap implementasi sesuai dengan metodologi penelitian, yaitu sebagai berikut.

5.2.1 Instalasi

Instalasi memuat langkah-langkah yang dilakukan untuk memasang perangkat lunak pendukung untuk melakukan pengembangan sistem. Berikut ini beberapa perangkat lunak pendukung dengan cara instalasinya, yaitu sebagai berikut.

5.2.1.1 *Mininet*

Mininet merupakan lingkungan emulasi jaringan yang digunakan untuk pengembangan sistem ini. Berikut ini dijelaskan tentang tahap-tahap dalam melakukan instalasi Mininet.

1. Mengambil *source code Mininet* pada github resmi *Mininet* dengan perintah:

$ git clone git://github.com/mininet/mininet

2. Menginstall *Mininet* dengan perintah:

$ sudo mininet/util/install.sh –a

5.2.1.2 *Ryu Controller*

Berikut ini langkah-langkah melakukan instalasi Ryu sebagai SDN *Controller*, yaitu sebagai berikut.

1. Menginstall python-pip dengan perintah:

$ sudo apt-get install python-pip

2. Menginstall ryu dengan perintah:

$ sudo pip install ryu

5.2.1.3 *Mininam*

Berikut ini langkah-langkah melakukan instalasi Mininam sebagai pengecekan jalur animasi bergerak, yaitu sebagai berikut.

1. Menginstall python-imaging dengan perintah:

$ sudo apt-get install python-imaging

2. Menginstall MiniNAM dengan perintah:

$ git clone git://github.com/uccmisl/MiniNAM

3. Menginstall paping dengan perintah:

$ sudo chmod +x paping

5.2.1.4 Bwm-ng

1. Download source code dengan perintah:

$ wget http://www.gropp.org./bwm-ng/bwm-ng-0.6.tar.gz

2.Extrack file dengan perintah:

$ tar zxvf bwm-ng-0.6.tar.gz

3. Masuk ke directory dengan perintah:

$ cd bwm-ng-0.6

5.Configure dan install dengan perintah:

$ ./configure && make && make install

6. Jalankan bwm-ng dengan perintah:

$ bwm-ng

5.3 Membangun Topologi di *Mininet*

Setelah semua perangkat lunak pendukung telah terinstall, langkah selanjutnya adalah membangun topologi di Mininet sesuai dengan desain topologi yang telah dilakukan. Salah satu cara untuk melakukan pembangunan topologi di *Mininet* adalah dengan menggunakan GUI yang bernama *Miniedit*, yang akan dibahas di bagian ini. Berikut ini langkah-langkah membangun topologi di Miniedit, yaitu sebagai berikut.

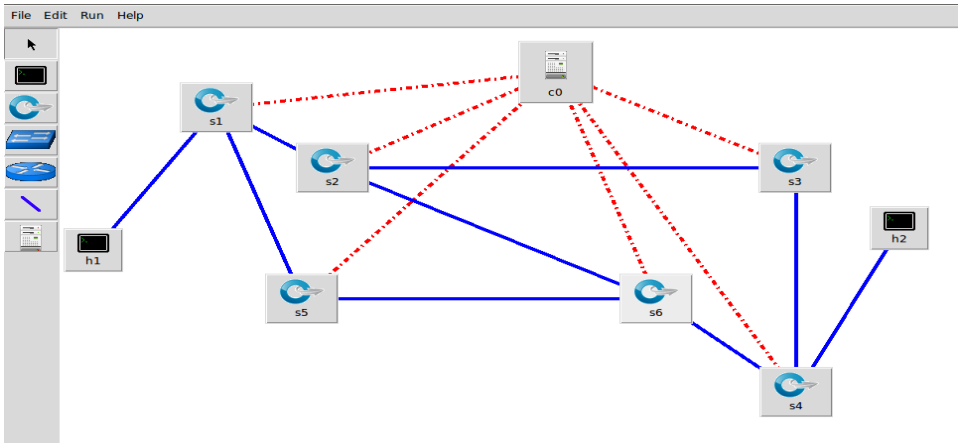
1. Menjalankan Miniedit pada terminal dengan perintah berikut:

$ sudo python mininet/examples/miniedit.py

2. Membangun topologi dengan mengacu daftar komponen yang ditunjukkan pada tabel 5.1. Salah satu contoh hasil pembangunan topologi dapat dilihat pada gambar 5.2.

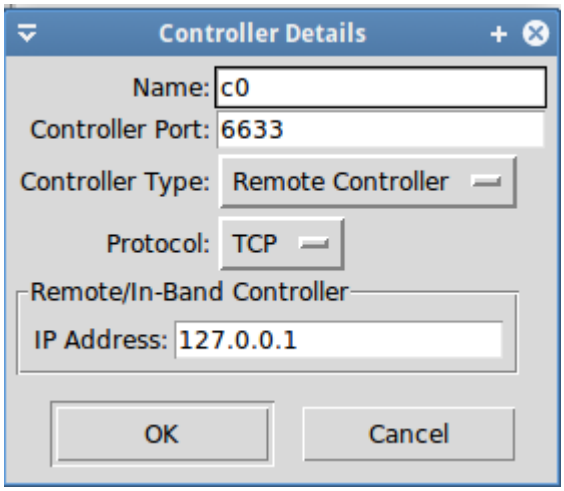
Tabel 5.1 Daftar komponen Mininet

No	Ikon	Nama	Fungsi
1		Kursor	Memindahkan dan memilih komponen-komponen
2		*Host/end Service*	Bertindak sebagai *host/end device* yang menggunakan layanan jaringan.
3		*OpenFlow switch*	Bertindak sebagai perangkat jaringan *OpenFlow* yang bertindak sebagai *data plane* SDN.
4		*Switch* tradisional	Bertindak sebagai *switch* tradisional yang meneruskan paket menggunakan pemetaan alamat MAC berdasarkan letak *port* bersambungnya.
5		*Router* tradisional	Bertindak sebagai *router* tradisional yang mengambil keputusan penerusan paket berdasarkan protokol *routing*.
6		*Link*	Menghubungkan antara satu komponen dengan yang lainnya.
7		*Controller*	Bertindak sebagai *Controller* SDN yang diakses oleh *Mininet* pada port tertentu.



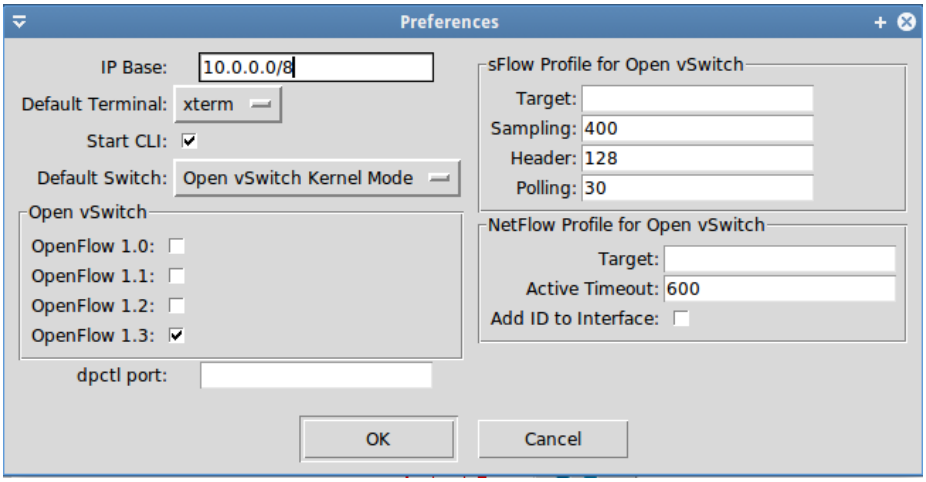
Gambar 5.5 Contoh topologi di *Mininet*

3. Melakukan pengaturan controller sesuai dengan gambar 5.6 dengan melakukan klik kanan pada logo *controller* (c0) dan menekan "*properties*". Hal ini dilakukan agar Mininet dapat terhubung dengan program *controller* yang telah dikembangkan. "*Controller Type*" diubah menjadi "*Remote Controller*".



Gambar 5.6 Pengaturan controller di *Mininet*

4. Melakukan pengaturan terhadap *preferences* dari *Mininet* untuk mengaktifkan CLI *Mininet* dan mengubah *versi OpenFlow* menjadi *versi* 1.3 sesuai dengan gambar 5.7. Hal tersebut dilakukan dengan memilih menu "*Edit*" kemudian memilih "*Preferences*".



Gambar 5.7 Pengaturan *preferences* di *Mininet*

5. Menjalankan simulasi topologi di *Mininet* dengan menekan tombol "*Run*".

6. Membuka terminal baru dan menjalankan program controller dengan ryu--manager menggunakan perintah berikut ini.

$ sudo ryu-manager --observe-links nama_program.py

```
carlo@carlo-X455LD: ~/ryu/ryu/app
carlo@carlo-X455LD:~$ cd ryu
carlo@carlo-X455LD:~/ryu$ cd ryu
carlo@carlo-X455LD:~/ryu/ryu$ cd app
carlo@carlo-X455LD:~/ryu/ryu/app$ ryu-manager --observe-links ryu_multipath_yens
.py
loading app ryu_multipath_yens.py
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
instantiating app ryu.topology.switches of Switches
instantiating app ryu_multipath_yens.py of ProjectController
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Gambar 5.8 Running *Controller* Ryu

Berdasarkan Gambar 5.8 di atas, apabila di terminal tercetak tampilan maka program telah berhasil di *upload* pada *controller Ryu*. Ketika program sudah terupload maka sistem telah siap dijalankan untuk mencari rute terpendek.

5.4 Pengembangan Program *Controller*

Pengembangan program *controller* terdiri atas langkah-langkah pemrograman yang dilakukan untuk *Ryu Controller* dengan menggunakan bahasa

Python versi 2.7.11 dalam mengimplementasi logika atau kecerdasan dari berdasarkan perancangan yang telah dilakukan.

5.4.1 Variabel dan Struktur Data

Berikut ini beberapa variabel dan struktur data yang digunakan dalam program, yaitu seperti yang ditunjukkan tabel 5.2.

Tabel 5.1 Variabel dan Struktur Data

No	Nama	Tipe Data	Deskripsi	Contoh
1	*switches*	*Dictionary*	Menyimpan informasi tentang suatu *switch*.	{1:{"ports":{1:{"ifindex":12,"ifname":"s1-eth1","bandwidth":100000000}},"capacity":10000000000}} Penjelasan: Switch 1 berkapasitas 10 Gbps atau 10^{10} bps memiliki port 1 (*interface index* 12, *interface name* s1-eth1, *bandwidth* 100 Mbps atau 10^8 bps).
2	*adjacency*	*Dictionary*	Menyimpan informasi ketetanggan (*adjacency*) antara dua *switch* dan *port* yang menghubungkannya	{1:{2:1,3:2}} Penjelasan: *Switch* 1 terhubung dengan *switch* 2 pada *port* 1 dan *switch* 3 pada *port* 2.
3	*Controller IP*	*Dictionary*	Menyimpan informasi pemetaan suatu alamat IP pada *switch* dan *port*-nya	"10.0.0.100" Penjelasan: Alamat IP 10.0.0.1 terhubung pada *switch* 1
4	*REFERENCE_BW*	*Integer*	*Reference Bandwidth* untuk perhitungan *cost* atau *weight*	1000000000 Penjelasan: 1 Gbps atau 10^9 bps.