BAB III
METODE PENELITIAN

3.1 Alat dan Bahan

Perangkat keras yang digunakan untuk melakukan penelitian ini dengan laptop dengan spesifikasi sistem seperti berikut :

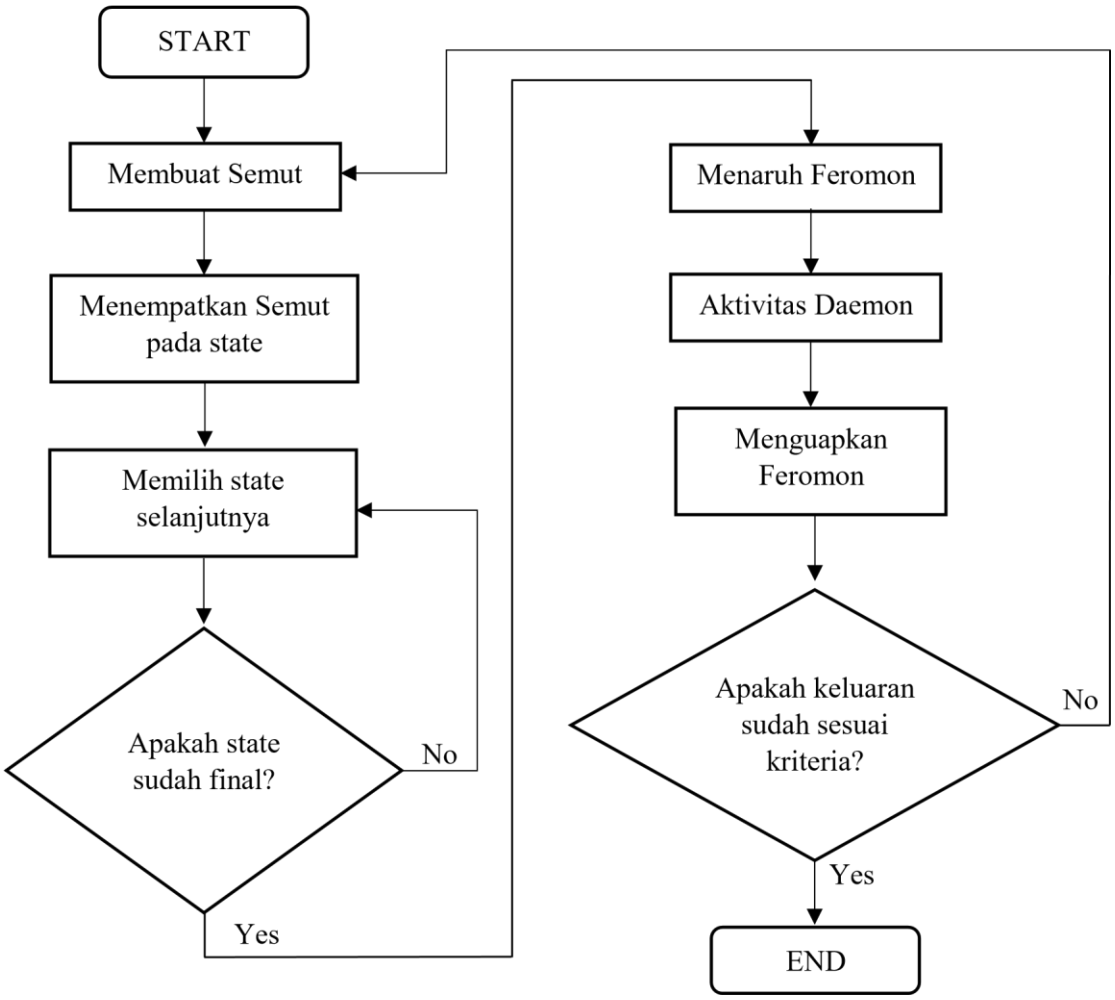
1. Intel(R) Celeron(R) CPU 847 @ 1.1-GHz (2 CPUs), 1.1GHz
2. Windows 10 Pro 64-bit (10.0, Build 19045)
3. RAM 4098 MB

1.2 Perangkat Lunak (*Software*)

Perangkat lunak yang digunakan adalah *VirtualBox* yang berisi sistem operasi *Kali Linux* yang terpasang *Mininet, Ryu,* dan *POX Controller*. *VirtualBox* adalah perangkat lunak yang digunakan untuk penelitian untuk mendukung sistem operasi virtual, *Mininet* untuk *platform* jaringan, POX digunakan sebagai *controller* jaringan SDN.

3.3 Perancangan Sistem

Pada perancangan analisis *load balancing* pada jaringan SDN yang disimulasikan pada tugas akhir ini adalah untuk meningkatkan performasi pada saat *request* masuk ke *server* dengan Teknik *load balancing,* dan untuk mengoptimalkan *reponse time* yang ada, serta untuk mengambil data dari *untulitas* CPU. Ruang lingkup analisis tugas akhir adalah pada emulator *mininet* dan perhitungan pada algoritma optimasi koloni semut jaringan SDN. Lingkup kerja dari tugas besar ini meliputi pengerjaan data analisis jaringan SDN terlebih dahulu, kemudian di lakukan tenik *load balncing* dan meneraprakan analisis pada algoritma optimasi koloni semut.



Penjelasan dari tahapan sebagai berikut :

1. Start adalah awalan sistem perencanaan beban yang akan dieksekusi.
2. Membuat semut adalah memasukkan data beban.
3. Menempatkan semut pada state adalah menempatkan beban pada *state* pertama.
4. Memilih state selanjutnya adalah mengirimkan beban pada *state* selanjutnya.
5. Jika *state* sudah final state akan menaruh *feromon* jika tidak, kembali memilih jalur *state* selanjutnya.
6. Menaruh *feromon* ditujukan untuk mencari jalur tercepat.

7. Aktifitas *daemon* merupakan proses pengiriman beban ke server dengan jalur terpendek dari *server* yang utilitasnya kecil.

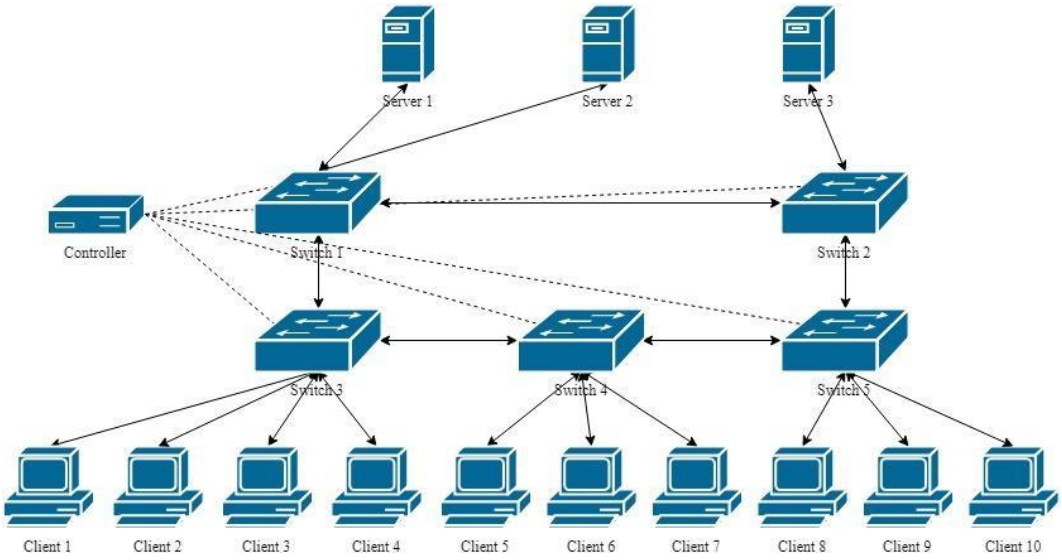
8. Menguapkan *feremon* merupakan penyimpanan jalur perjalanan semut.

9. Jika sudah sesuai kriteria maka data akan disimpan *server*, dan jika belom sesuai kriteria makan akan dikembalikan.

10. *End* adalah proses akhir setelah semua eksekusi dan analisis selesai.

3.4 Gambaran Umum Sistem

Dalam pengujian akan dirancang *load balancing* pada jaringan *Software Definend Network* (SDN) menggunakan algoritma optimasi koloni semut, dan penggunaan topologi *tree* pada jaringan SDN dan di jalankan dengan emulator *mininet*, konfigurasi menggunakan *controller* POX



Gambar 3. 1 merupakan gambaran umum dari *load balance* pada jaringan SDN yang akan diterapkan pada algoritma optimasi koloni semut.

Berikut adalah penjelasan dari topologi pada gambar 3.1:

1. *Client*, mengirim permintaan.
2. *Switch*, perangkat ini bertujuan meneruskan permintaan dari *controller*.

3. POX *Controller*, bertugas untuk mengatur switch yang sedang mengirim permintaan.
4. *Server*, bertugas sebagai penyedia layanan dari permintaan *client.*

3.5 Konfigurasi Jaringan

Pengaturan konfigurasi jaringan dilakukan dengan pemrograman menggunakan python untuk dijalankan lewat mininet baik dalam membuat topologi, pengaturan ip address dan mac address, hubungan antar perangkat, serta penggunaan protokol. Berikut kode program untuk konfigurasi jaringan pada penelitian ini :

```
from mininet.cli import CLI
from mininet.net import Mininet
from mininet.node import RemoteController, OVSSwitch
from mininet.link import TCLink

if __name__ == '__main__':
    net = Mininet(controller=RemoteController, switch=OVSSwitch, link=TCLink, autoSetMacs=True)

    c0 = net.addController('c0', controller=RemoteController, port=6633)

    h1 = net.addHost('h1', ip='10.0.0.1', MAC='01:01:01:00:00:01')
    h2 = net.addHost('h2', ip='10.0.0.2', MAC='01:01:01:00:00:02')
    h3 = net.addHost('h3', ip='10.0.0.3', MAC='01:01:01:00:00:03')
    h4 = net.addHost('h4', ip='10.0.0.4', MAC='01:01:01:00:00:04')
    h5 = net.addHost('h5', ip='10.0.0.5', MAC='01:01:01:00:00:05')
    h6 = net.addHost('h6', ip='10.0.0.6', MAC='01:01:01:00:00:06')
    h7 = net.addHost('h7', ip='10.0.0.7', MAC='01:01:01:00:00:07')
    h8 = net.addHost('h8', ip='10.0.0.8', MAC='01:01:01:00:00:08')
    h9 = net.addHost('h9', ip='10.0.0.9', MAC='01:01:01:00:00:09')
    h10 = net.addHost('h10', ip='10.0.0.10', MAC='01:01:01:00:00:10')

    # Switches
```

```
s1 = net.addSwitch('s1', protocols='OpenFlow13')
s2 = net.addSwitch('s2', protocols='OpenFlow13')
s3 = net.addSwitch('s3', protocols='OpenFlow13')
s4 = net.addSwitch('s4', protocols='OpenFlow13')
s5 = net.addSwitch('s5', protocols='OpenFlow13')

# Connect hosts to switches
for h, s in [(h1, s1), (h2, s1), (h3, s1), (h4, s1)]:
    net.addLink(h, s, delay='1ms')

for h, s in [(h5, s2), (h6, s2), (h7, s2)]:
    net.addLink(h, s, delay='1ms')

for h, s in [(h8, s3), (h9, s3), (h10, s3)]:
    net.addLink(h, s, delay='1ms')

# Connect switches
net.addLink(s1, s2)
net.addLink(s2, s3)
net.addLink(s1, s4)
net.addLink(s3, s5)
net.addLink(s4, s5)

net.build()
c0.start()
s1.start([c0])
s2.start([c0])
s3.start([c0])
s4.start([c0])
s5.start([c0])
```

```
s1.cmd('ovs-vsctl set Bridge s1 protocols=OpenFlow13')
s2.cmd('ovs-vsctl set Bridge s2 protocols=OpenFlow13')
s3.cmd('ovs-vsctl set Bridge s3 protocols=OpenFlow13')
s4.cmd('ovs-vsctl set Bridge s4 protocols=OpenFlow13')
s5.cmd('ovs-vsctl set Bridge s5 protocols=OpenFlow13')

# Set IP addresses for the switches
s1.cmd('ifconfig s1 10.0.0.11 netmask 255.255.255.0')
s2.cmd('ifconfig s2 10.0.0.12 netmask 255.255.255.0')
s3.cmd('ifconfig s3 10.0.0.13 netmask 255.255.255.0')
s4.cmd('ifconfig s4 10.0.0.14 netmask 255.255.255.0')
s5.cmd('ifconfig s5 10.0.0.15 netmask 255.255.255.0')

# Set default gateway for hosts
h1.cmd('route add default gw 10.0.0.11')
h2.cmd('route add default gw 10.0.0.11')
h3.cmd('route add default gw 10.0.0.11')
h4.cmd('route add default gw 10.0.0.11')
h5.cmd('route add default gw 10.0.0.12')

h6.cmd('route add default gw 10.0.0.12')
h7.cmd('route add default gw 10.0.0.12')
h8.cmd('route add default gw 10.0.0.13')
h9.cmd('route add default gw 10.0.0.13')
h10.cmd('route add default gw 10.0.0.13')

# Enable IP forwarding on the switches
s1.cmd('sysctl -w net.ipv4.ip_forward=1')
s2.cmd('sysctl -w net.ipv4.ip_forward=1')
s3.cmd('sysctl -w net.ipv4.ip_forward=1')
s4.cmd('sysctl -w net.ipv4.ip_forward=1')
```

```
s5.cmd('sysctl -w net.ipv4.ip_forward=1')

CLI(net)

s1.cmd('ovs-ofctl del-flows s1')
s2.cmd('ovs-ofctl del-flows s2')
s3.cmd('ovs-ofctl del-flows s3')
s4.cmd('ovs-ofctl del-flows s4')
s5.cmd('ovs-ofctl del-flows s5')


net.stop()
```

Untuk tiap perangkat host diatur *ip address* dan *mac address* secara manual dengan ketentuan pada host 1 menggunakan ip 10.0.0.1 dan mac 01:01:01:00:00:01, *host* 2 dengan ip 10.0.0.2 dan mac 01:01:01:00:00:02, host 3 dengan ip 10.0.0.3 dan mac 01:01:01:00:00:03, host 4 dengan ip 10.0.0.4 dan mac 01:01:01:00:00:04, host 5 dengan ip 10.0.0.5 dan mac 01:01:01:00:00:05, host 6 dengan ip 10.0.0.6 dan mac 01:01:01:00:00:07, host 7 dengan ip 10.0.0.7 dan mac 01:01:01:00:00:07, host 8 dengan ip 10.0.0.8 dan mac 01:01:01:00:00:08, host 9 dengan ip 10.0.0.9 dan mac 01:01:01:00:00:10. Untuk switch 1 sampai 5 diatur menggunakan protokol OpenFlow yang mendukung control remote dan pengalamatan ip.

Untuk perangkat host 1 sampai dengan host 4 diatur agar terhubung langsung dengan switch 3, host 5 sampai host 7 terhubung langsung ke switch 4, host 8 sampai dengan host 10 terhubung dengan switch 5. Untuk pengalamatan ip antar switch yaitu 10.0.0.11 sampai 10.0.0.15 untuk switch 1 sampai switch 5 dengan netmask 255.255.255.0.

Untuk pengaturan rute gateway host 1 sampai host 4 menggunakan alamat ip address switch 3 yaitu 10.0.0.13 dan untuk host 5 sampai dengan host 7 menggunakan ip address switch 4 yaitu 10.0.0.14. Gateway ip tiap host menggunakan ip tiap switch terhubung langsung dengan perangkat masing-masing host.

Tiap switch dikonfigurasikan untuk mengenali paket yang datang dari host lewat pengalamtan IPv4 agar dapat melakukan ping, saat ada flow-rule baru yang diberitahukan oleh kontroler maka semua switch akan melakukan penghapusan flow-rule lama dan melakukan update dlow-rule yang diberikan dari kontroler remote