BAB 3
METODE PENELITIAN

3.1 ALAT DAN BAHAN

3.1.1 Perangkat keras (*Hardware*)

Perangkat keras yang digunakan untuk penelitian ini yaitu sebuah *Laptop* dengan spesifikasi sistem seperti berikut :

1. AMD Ryzen 5 2500U (8 CPUs) ~ 2.0GHz
2. Windows 10 Home Single Language 64-bit
3. RAM 8192 MB
4. Hardisk 1 TB

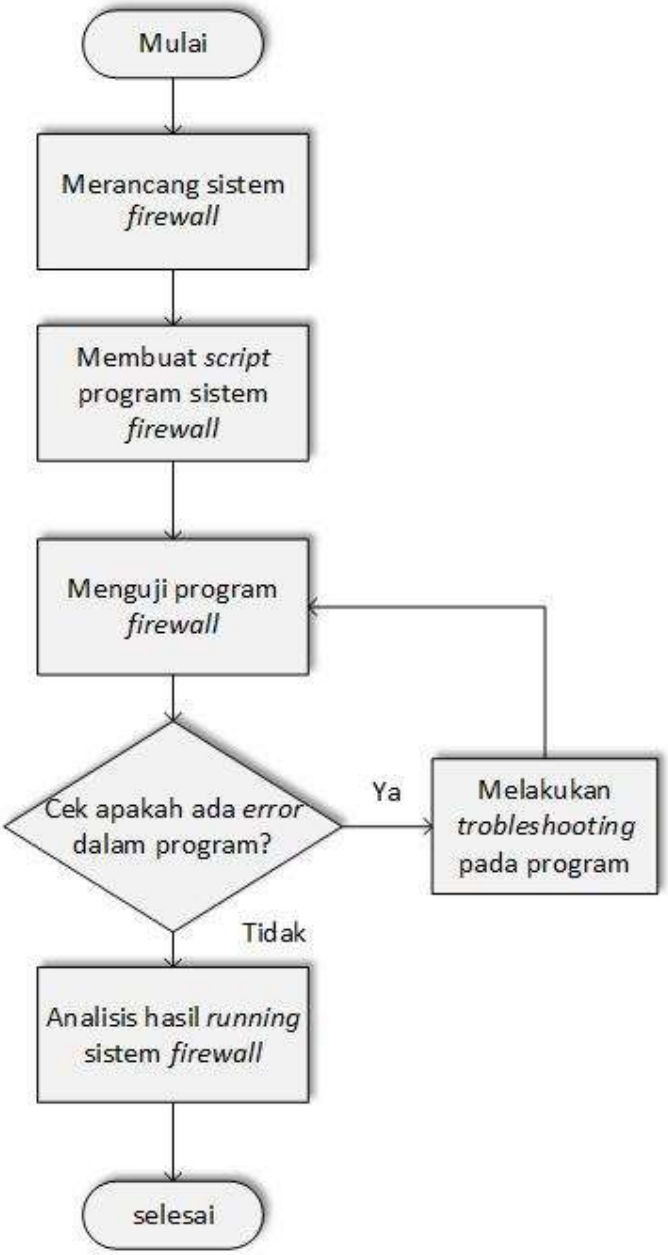
3.1.2 Perangkat lunak (*Software*)

Perangkat lunak yang akan digunakan adalah *VirtualBox* yang berisi sistem operasi *Kali Linux* yeng terpasang *Mininet, Ryu, Python,* dan *Redis database*. *VirtualBox* merupakan perangkat lunak yang nantinya akan digunakan untuk keperluan penelitian untuk pendukung sistem operasi virtual, *Mininet* untuk *platform* jaringan, *Ryu* sebagai kontroler jaringan SDN, *Redis database* sebagai sistem *database packet filtering* untuk *firewall* jaringan SDN*,* dan *Python* untuk antarmuka pemograman utamanya.

3.2 ALUR PENELITIAN

Pada alur penelitian ini, berisi langkah-langkah yang akan dilakukan selama penelitian dilakukan. Pertama peneliti melakukan perancangan sistem *firewall* serta installasi *package Kali Linux* yang dibutuhkan untuk sistem seperti installasi *Ryu* sebagai kontroler SDN*, Python* untuk antarmuka pemograman, *Redis database* untuk *database packet filtering,* dan *Mininet* sebagai *platform* pembuatan jaringan SDN. Selanjutnya dilakukan pembuatan *script program firewall* menggunakan *Python* yang sudah ada didalam *Kali Linux*, setelah program selesai maka dilakukan uji coba menggunakan *Mininet* yang sudah dibuat jaringan virtual didalamnya serta *Ryu* sebagai kontroler jaringan SDN yang akan menjalankan perintah atau fungsi yang ada didalam program. Bila terdapat error maka peneliti

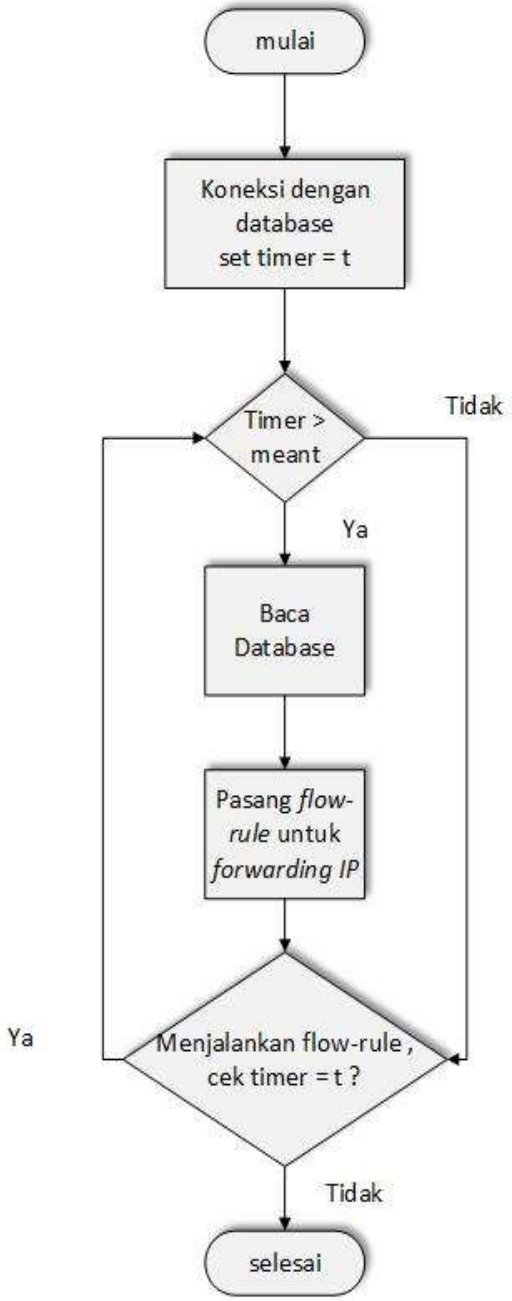
akan melakukan *troubleshooting* program dan jika program sudah berjalan dengan sesuai, maka peneliti akan menganalisa hasil uji coba *firewall* yang sudah sesuai. *Berikut* merupakan *flowchart* untuk alur penelitian ini:



Gambar 3.1 Flowchart Alur Penelitian.

3.3 RANCANGAN SISTEM

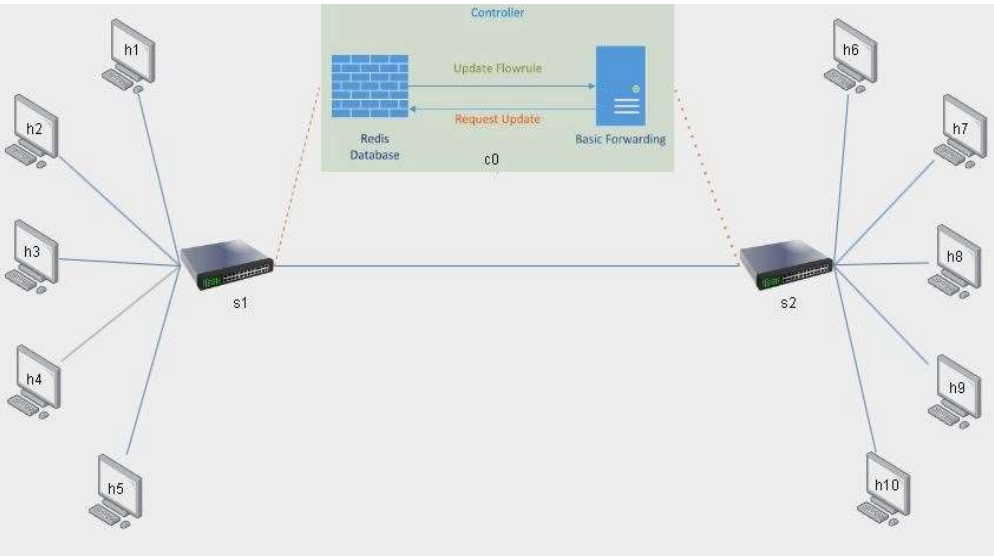
Pada rancangan sistem kali ini, peneliti akan menjelaskan cara kerja sistem *firewall* yang akan diterapkan di jaringan SDN. Berikut merupakan *flowchart* dari sistem *firewall* penelitian ini:



Gambar 3.2 *Flowchart* sistem *firewall* pada jaringan SDN menggunakan *Redis database.*

Untuk penjelasan alur flowchart pada Gambar 3.2, pada saat program dijalankan oleh kontroler, kontroler akan memulai pembentukan koneksi dengan *Redis database* yang kemudian akan menginstall fungsi-fungsi yang akan digunakan untuk *firewall* dari *library* program. Setelah inisiasi koneksi selesai, kontroler akan melakukan pencocokan waktu pengambilan data dari *database* yang ada didalam program, jika waktu yang sudah berjalan melebihi batas waktu yang ditentukan maka kontroler mulai menanyakan ke *database* untuk pengambilan data *packet filtering* dan memasang *flow-rule* yang ada di dalam program. Selanjutnya jika kontroler sudah mendapatkan data dan sudah tepasang *flow-rule* maka kontroler akan mulai menjalankannya untuk proses *firewall* dan *basic forwarding* terhadap paket yang keluar masuk *router/switch*. Jika waktu sudah mencapat batas, kontroler akan menghapus *flow-rule* dan data dari *database packet filtering* yang lama dan akan melakukan proses *update* kembali sehingga jika ada paket yang baru terdaftar didalam *Redis database* untuk *filtering* maka *contoller* tahu aksi yang akan dilakukan terhadap paket-paket yang terkena *filtering* tersebut.

3.4 TOPOLOGI JARINGAN



Gambar 3.3 Topologi jaringan *sdn whithelist packet-filtering* dengan menggunakan *redis database.*

Berdasarkan Gambar 3.3 diatas, topoligi jaringan pada penelitian kali ini menggunakan jenis topologi *tree* yang terdiri dari 10 perangkat *host*, 2 perangkat *switch*, dan 1 perangkat *controller*. Host 1 sampai host 5 terhubung langsung dengan switch 1, host 6 sampai host 10 terhubung langsung dengan switch 2, switch 1 terhubung langsungdengan switch 2, dan controller terhubung secara remote ke switch.

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)

    # Hosts connected to switch s1
    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')

    # Hosts connected to switch s2
```

```
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')

# Connect hosts to switches
for h, s in [(h1, s1), (h2, s1), (h3, s1), (h4, s1), (h5, s1)]:
    net.addLink(h, s, delay='1ms')

for h, s in [(h6, s2), (h7, s2), (h8, s2), (h9, s2), (h10, s2)]:
    net.addLink(h, s, delay='1ms')

# Connect switches
net.addLink(s1, s2)

net.build()
c0.start()
s1.start([c0])
s2.start([c0])

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

```
    # Set default gateway for hosts
    h1.cmd('route add default gw 10.0.0.11')  # Set gateway to the IP of s1
    h2.cmd('route add default gw 10.0.0.11')  # Set gateway to the IP of s1
    h3.cmd('route add default gw 10.0.0.11')  # Set gateway to the IP of s1
    h4.cmd('route add default gw 10.0.0.11')  # Set gateway to the IP of s1
    h5.cmd('route add default gw 10.0.0.11')  # Set gateway to the IP of s1

    h6.cmd('route add default gw 10.0.0.12')  # Set gateway to the IP of s2
    h7.cmd('route add default gw 10.0.0.12')  # Set gateway to the IP of s2
    h8.cmd('route add default gw 10.0.0.12')  # Set gateway to the IP of s2
    h9.cmd('route add default gw 10.0.0.12')  # Set gateway to the IP of s2
    h10.cmd('route add default gw 10.0.0.12') # Set gateway to the IP of s2

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

    CLI(net)

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

net.stop()
```

Perintah ‘from’ digunakan untuk memanggil library dari aplikasi yang akan dijalankan lewat python agar kode fungsi program yang digunakan dapat teridentifikasi, perintah ‘import’ untuk memanggil kode fungsi dari library yang dipanggil. Dalam konfigurasi header program ini diatur agar topologi jaringan yang dibuat dapat dijalankan oleh mininet dalam perintah ‘import Mininet’, secara command line promp dalam perintah ‘import CLI’, menggunakan kontroler remote dan OpenvSwitch protokol dalam perintah ‘import RemoteController, OVSSwitch’, dan agar perangkat yang terhubung dapat menggunakan ip address dengan perintah ‘import TCLink’.

Pada main program mininet, kontroler diatur agar terhubung secara remote dengan jalur port 6633 dan switch yang digunakan untuk jaringan ini yaitu OpenVSwitch, konfigurasi mac address untuk kontroler diatur auto. 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 01:01:01:00:00:06, host 2 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 01:01:01:00:00:09, dan host 10 dengan ip 10.0.0.10 dan mac 01:01:01:00:00:10. Untuk switch 1 dan switch 2 diatur menggunakan protokol OpenFlow13 yang mendukung control remote dan pengalamatan ip.

Untuk perangkat host 1 sampai dengan host 5 diatur agar terhubung langsung dengan switch 1 dan host 6 sampai dengan host 10 diatur agar terhubung langsung dengan switch 2, switch 1 dan switch 2 terhubung langsung dan untuk kontroler dihubungkan secara remote ke switch 1 dan switch 2 menggunakan konfigurasi protokol OpenFlow13. Untuk pengalamatan ip antar switch yaitu 10.0.0.11 untuk switch 1 dan 10.0.0.12 untuk switch 2 dengan netmask 255.255.255.0 atau ip kelas C.

Untuk pengaturan rute gateway host 1 sampai host 5 menggunakan ip address switch 1 yaitu 10.0.0.11 dan untuk host 6 sampai host 10 menggunakan ip address switch 2 yaitu 10.0.0.12. gateway ip tiap host menggunakan ip tiap switch yang terhubung langsung dengan perangkat masing-masing host.

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

3.6 KONFIGURASI RYU CONTROLER

Pada penelitian ini menggunakan Ryu sebagai controller dan menggunakan bahasa pemrograman python sebagai dasarnya, disini controller bertugas sebagai pengatur jalannya paket yang masuk dan keluar baik itu icmp dan arp, jika ada ip address atau mac address baru yang masuk dalam jaringan sdn maka controller akan mendaftarkan ip dan mac address dari perangkat yang terkoneksi dan menentukan apakah perangkat tersebut dapat berkomunikasi dengan perangkat lainnya baik itu pada layer 2 atau mac address dalam bentuk paket arp maupun pada layer 3 atau ip address dalam bentuk paket icmp dan dapat mekaukan tes koneksi atau ping. Berikut kode program yang akan dijalankan dalam ryu controller untuk mengatur jalannya paket dalam jaringan ini :

```
from ryu.base import app_manager

from ryu.controller import ofp_event

from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER

from ryu.controller.handler import set_ev_cls

from ryu.ofproto import ofproto_v1_3

from ryu.lib.packet import packet

from ryu.lib.packet import ethernet

from ryu.lib.packet import arp

from ryu.lib.packet import ipv4

from ryu.lib.packet import icmp

from ryu.lib.packet import ether_types


class MyL2Switch(app_manager.RyuApp):

    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]
```

OFP_VERSIONS: menentukan versi OpenFlow yang dipakai, disini memakai versi OF 1.3

```
def __init__(self, *args, **kwargs):
        super(MyL2Switch, self).__init__(*args, **kwargs)
        self.swList = []                    #daftar switch
        self.hostDB = {}        #berisi pairing antara host.ID - port number
        def add_flow(self, datapath, match, actions,priority=0):
        ofproto = datapath.ofproto
#instruksi dasar untuk mengeksekusi semua perintah di daftar actions
Inst=[datapath.ofproto_parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS, actions)]
        mod = datapath.ofproto_parser.OFPFlowMod(
            datapath=datapath,     #switch id
            cookie=0, cookie_mask=0,
            table_id=0,     #nomor Flow table dimana flow rule di install
            command=ofproto.OFPFC_ADD,
            idle_timeout=0,  hard_timeout=0,  #timeout  =  0  ->  tidak memiliki timeout
            priority=priority,                  #menentukan urutan matching
            buffer_id=ofproto.OFP_NO_BUFFER,
            out_port=ofproto.OFPP_ANY,
            out_group=ofproto.OFPG_ANY,
            flags=0,
```

```
                match=match,                    #perintah match
                instructions=inst)              #perintah actions
        datapath.send_msg(mod)
    def delflowintable(self,dp):
        ofp = dp.ofproto
        parser = dp.ofproto_parser
        del_flows = parser.OFPFlowMod(dp, table_id=ofp.OFPTT_ALL,
out_port=ofp.OFPP_ANY, out_group=ofp.OFPG_ANY,
command=ofp.OFPFC_DELETE)
        dp.send_msg(del_flows)
        print "deleting all flow entries in all table of sw-",dp.id
    @set_ev_cls(ofp_event.EventOFPSwitchFeatures,
CONFIG_DISPATCHER)
    def switch_features_handler(self, ev):
        msg = ev.msg
        dp = msg.datapath
        ofproto = dp.ofproto
        self.delflowintable(dp)
        self.swList.append(dp)          #daftar switch yang terkoneksi
        match = dp.ofproto_parser.OFPMatch(eth_type=0x806)  #0x806 ->
ARP ethertype
    actions=[dp.ofproto_parser.OFPActionOutput(ofproto.OFPP_FLOOD, 0),
    dp.ofproto_parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,
    ofproto.OFPCML_NO_BUFFER)]
        #FLOOD --> khusus topologi acyclic.
```

```
        #Pada topologi cyclic --> endless flooding --> overhead
        #install flow rule, pada switch dp, dengan skema match-actions
        self.add_flow(dp, match, actions)
@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def _packet_in_handler(self, ev):
        msg = ev.msg
        in_port = msg.match['in_port']
        dp = msg.datapath
        ofproto = dp.ofproto
        dpid = dp.id
        pkt = packet.Packet(msg.data)
        pkt_arp = pkt.get_protocols(arp.arp)[0]
        if pkt_arp:
                print ("receiving arp packet")
                #ambil IP pengirim dan asosiasikan dengan input port.
                pkt_arp = pkt.get_protocol(arp.arp)
                sipv4 =  pkt_arp.src_ip
                self.hostDB[sipv4] = in_port
#pasang flow rule, jika menerima paket ICMP dengan tujuan IP pengirim, kirim ke input_port
                match=dp.ofproto_parser.OFPMatch(eth_type=0x800,ip_proto=0x01,ipv4_dst=sipv4)
                actions=[dp.ofproto_parser.OFPActionOutput(self.hostDB[sipv4] , 0)]
```

```
                self.add_flow(dp, match, actions, 1)

                print ("install flow rule: match: ICMP to "+sipv4+"
            output:"+str(in_port))
```

Pada header library program dibuat agar kode yang akan dijalankan merupakan kode untuk ryu controller dalam pendefinisian ‘from ryu.base import app_manager’, selanjutnya untuk pengaturan definisi isi program untuk dijalankan pada ryu controller dimasukan library ‘from ryu.controller.handler’ dan dilengkapi dengan ‘import CONFIG_DISPATCHER’ untuk isi dari pengaturan jaringan sdn dan ‘import MAIN_DISPATCHER’ untuk eksekusi pengaturan yang ditetapkan pada jaringan sdn serta hal apa yang harus dilakukan oleh controller nantinya jika ada paket dan informasi perangkat dari jaringan sdn yang masuk ataupun jika ada perubahan topologi jaringan serta informasi perangkat didalamnya, untuk bisa menggunakan command line promp pada ryu di definisikan dengan ‘import set_ev_cls’, untuk protokol yang digunakan pada ryu controller harus disamakan dengan penggunaan protokol pada switch yaitu OpenFlow13 dan pengenalan library pemanggilan protokol didefinisikan dengan perintah ‘from ryu.ofproto import ofproto_v1_3’. Terakhir untuk kebutuhan fungsi yang akan dijalankan ryu controller akan didefinisikan dengan library ‘from ryu.lib.packet import’ dan fungsi jaringannya yaitu ‘packet’ untuk pengenalan paket yang digunakan, ‘ethernet’ untuk pengenalan ethernet, ‘arp’ untuk paket arp, ‘ipv4’ untuk pengenalan pengalanatan ipv4 dalam perangkat, ‘icmp’ untuk pengenalan paket icmp dan bisa melakukan tes ping, dan ‘ether_types’ untuk pengenalan tipe ethernet dari paket yang akan digunakan.

Untuk protokol yang digunakan oleh ryu controller diatur menjadi protokol OpenFlow13 agar disamakan dengan protokol yang digunakan perangkat switch agar controller dapat memberi perintah atau flowrule yang akan digunakan oleh jaringan sdn dan agar dapat mengenal pengalamatan ipv4, paket icmp, serta menjalankan perintah ping.

Perintah awal dari controller untuk pengenalan awal perangkat didalam jaringan yang dijalankan oleh mininet, pertama controller diminta untuk mencari apakah ada perangkat baru yang terdaftar lewat paket arp, setelah itu barulah semua

perangkat yang terbaca oleh controller didaftarkan kedalam flowrule dan mencatat berapa banyak switch dan host yang terhubung ke jaringan.

Jika sudah membuat daftar perangkat yang terhubung di jaringan maka controller mulai membuat flowrule baru, pertama controller mengecek apakah protokol yang digunakan oleh switch menggunakan protokol yang sama dengan controller, kedua setelah penggunaan protokol dinyatakan sama maka controller mulai melakukan cek apakah ada flowrule yang sudah ada didalam switch dan melakukan penghapusan flowrule lama dan akan digantikan flowrule baru oleh controller dan membuat urutan flowrule, ketiga jika sudah update flowrule yang baru maka controller akan memberitahu switch akan update flowrule dan mulai menginstall ke switch, terakhir switch akan memberitahu jika sudah memasang flowrule baru dan controller mulai mengecek ulang flowrule yang sudah dipasang pada switch dan siap dijalankan oleh jaringan sdn.

Controller diijinkan untuk menghapus semua flowrule lama yang sudah terpasang dalam switch dan melakukan update terhadap flowrule baru yang ada didalam controller.

Untuk konfigurasi flowrule awal untuk controller, controller akan melakukan pengiriman paket arp acak ke semua perangkat yang ada didalam jaringan sdn agar controller bisa mengenali semua perangkat yang terhubung dan mencatat semua mac address dalam daftar flowrule baik untuk host dan switch. Jika pengiriman gagal maka controller akan terus melakukan pengiriman paket arp secara acak sampai semua perangkat dapat dikenali dan siap dimasukan flowrule yang ada didalam controller. Hal ini juga dimaksudkan agar controller tidak salah memasukan flowrule ke perangkat host dan hanya memasukan flowrule kedalam perangkat switch saja, karena switch sebagai data plane pada jaringan sdn yang bertugas untuk meneruskan jalur dari paket yang dikirim oleh host dan menentukan apakah paket tersebut dapat diteruskan ke tujuan atau akan di drop, itu semua dilakukan sesuai flowrule yang diberikan oleh controller.

Terakhir untuk perintah untuk flowrule utama dalam ryu controller yaitu agar paket icmp dan pengalamatan ipv4 dapat dikenali dalam jaringan sdn dan dapat dilakukan pengetesan jaringan dengan perintah ping. Pertama, saat melakukan tes

koneksi menggunakan perintah ping akan terjadi paket loss karena controller tidak bisa mengenal pengalamatan ipv4 ataupun paket icmp, selanjutnya controller akan kembali melakukan pengiriman paket arp ke semua perangkat yang terhubung ke jaringan sdn dan controller akan mencatat daftar mac address semua perangkat yang terhubung dan menanyakan ip address tiap perangkat agar nanti dicatat didalam daftar flowrule baru didalam controller. Kedua, setelah controller mendapatkan semua ip address tiap perangkat, controller akan membuat flowrule untuk mengenali paket icmp dan menentukan perangkat apa saja yang mempunyai pengalamatan ipv4 agar dapat menerima paket icmp serta mencatat port mana saja yang akan menjadi jalur untuk dilewati paket icmp tersebut, dalam hal ini maka perintah ping akan dikenali dan tes koneksi jaringan lewat perintah ping akan berhasil dengan tanda muncul notifikasi reply dari perangkat tujuan. Terakhir jika ada flowrule untuk memblokir penerusan paket icmp ke tujuan yang tidak terdaftar dalam catatan flowrule untuk ip address maka paket tersebut akan di stop oleh switch dan akan otomatis di drop agar tidak sampai perangkat tujuan karena ip address perangkat tersebut tidak dikenali dan paket icmp tidak bisa dikirim hanya dengan mac address karena mac address hanya berada di layer 2 yang mengenali paket arp dan paket icmp hanya pada layer 3 yaitu ip address.

3.7 KONFIGURASI REDIS DATABASE

Pada penelitian ini, redis database akan digunakan untuk mencatat ip address yang akan diijinkan untuk melakukan komunikasi dalam jaringan sdn yaitu hanya pada sisi layer 3 atau untuk paket icmp, berikut kode program untuk redis database :

```
import redis


rd = redis.Redis(host='localhost', port=6379, db=0)

isExit = False

while not(isExit):
```

```
inputdata = raw_input("IP address:")
listip=inputdata.split(".")
if inputdata == "exit":
        isExit = True
elif inputdata == "read":
        for i in range(rd.llen("allowedIP")):
                print rd.lindex("allowedIP",i)
elif inputdata == "delete":
        inputdata = raw_input("del-record:")
        rd.lrem("allowedIP",2,inputdata)
elif len (listip)==4:
        isValid = True
        for i in range (len(listip)):
                if listip[i].isdigit():
                        if int(listip[i])<256 and int(listip[i])>=0:
                                isValid=True
                        else:
                                isValid=False
                                break
                else:
                        isValid=False
                        break
        if isValid:
```

```
ipnumber = inputdata

rd.rpush("allowedIP",ipnumber)
```

Pertama, redis database harus dapat dijalankan didalam controller secara bersamaan dengan kode program inti, karena itu diatur menggunakan port yang berbeda agar dari controller dapat mengenali mana perintah untuk mencatat flowrule untuk pengenalan paket arp, paket icmp, ipv4, dan mac address, dan flowrule untuk mencatat perizinan apakah paket dapat diteruskan ke tujuan berdasarkan ip address karena sudah tercatat dalam whitelist redis database. Disini redis database menggunakan port 6379 yaitu port default dan koneksi local didalam ryu controller.

Kedua, redis akan menanyakan ip address yang boleh diizinkan untuk berkomunikasi didalam jaringan sdn dan mulai mencatat whitelist, jika semua ip address sudah dimasukan kedalam whitelist maka redis database mulai mengirim flowrule baru kepada controller dan meneruskannya ke switch agar switch meneruskan paket icmp yang telah disetujui dalam daftar whithelist redis database ke tujuan. Karena program redis database dan program controller utama menggunakan port yang berbeda, controller dapat menjalankan baik program untuk basic data forwarding maupun firewall whithelist paket filtering sehingga admin bisa memasukan dan menghapus ip address untuk perangkat yang diizinkan melakukan komunikasi pengiriman paket icmp serta melakukan update flowrule tanpa memberhentikan layanan basic forwarding komunikasi antar perangkat yang sedang berjalan dalam jaringan sdn.