Name: Rajan Shantanu Chaturvedi NYU ID: rsc9044@nyu.edu Date: 05/16/2022
Section: 001

Assignment 8: Final Project

_____Part 3 Functionality [Max 70 points]
_____Part 4 Functionality [Max 25 points]
_____Style [Max 25 points]
_____Part 3 Extra credit questions [Max 20 points (10 each)]
_____Total [Max 100 points + Extra Credit points]
Total in points: _____
Total in Extra credit points: _____
Professor's Comments: _____

Affirmation of my Independent Effort: Rajan Shantanu Chaturvedi

Project Report

ABSTRACT

In this project, a layer-3 routing application has been created. The main function this application is to install the rules in SDN switches to forward traffic to hosts using a valid shortest path through the network. To identify the shortest path in the network between a host and switches, the application uses the Dijkstra's Algorithm. A distributed load balancer is also implemented in this application. The function of load balancer application is to redirect new TCP connections to hosts using round robin method.

The project was implemented in an emulated network inside of an Ubuntu Virtual Machine called as Mininet. The hosts in the emulated network run standard Linux network software and its switches uses OpenFlow for Software-Defined Networking and customer routing. Mininet creates hosts to communicate over the simulated network, the kernel or user-space OpenFlow Switches, Controllers of the Switches. Mininet uses virtual ethernet pairs to connect hosts and switches.

(Keywords: OpenFlow, Virtual Network, Mininet, Software Defined Networking, SDN Controller, Distributed Load Balancing Routing Algorithm, Networking Devices, Switches, Hosts, Shortest Path Algorithm, Dijkstra's algorithm)

INTRODUCTION

Software-Defined Networking (SDN) is an approach to networking that uses software-based controllers or application programming interfaces (APIs) to communicate with underlying hardware infrastructure and direct traffic on a network. This model differs from that of traditional networks, which use dedicated hardware devices, i.e., routers and switches to control network traffic. SDN can create and control a virtual network – or control a traditional hardware – via software.

It is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services.

In basic terms, SDN is an architecture that abstracts different, distinguishable layers of a network to make networks agile and flexible. The goal of SDN is to improve network control by enabling enterprises and service providers to respond quickly to changing business requirements.

Dijkstra's Algorithm

Dijkstra's Algorithm is an algorithm that is used for finding the shortest distance, or path, from starting node to target node in a weighted graph. In our case, the weighted graph is the network, and the nodes are the switches. For the algorithm, we use a variable "processed" and a Priority Queue "to_processed". The variable "processed" is used to

keep track of the total 'hops' from the starting switch to each destination switch. It stores the current total weight of the smallest weight path (cost) from the start to the next switch in question and keeps updating per iteration. For every switch in the network, the algorithm iterates only once.

The priority queue "to_processed" is used to control the order of the iteration over the switches. The algorithm uses the priority queue to greedily choose the nearest switch that has not been visited yet and executes the relaxation process on all its edges. When a switch is first created "processed" is set to a very large number (in the java implementation, it is set to Integer.MAX_VALUE - 1, i.e., 2147483647).

Dijkstra's algorithm makes use of weights of the edges for finding the path that minimizes the total distance (weight) among the source node and all other nodes. When the algorithm concludes the distances and the predecessor links for each switch in the network are set correctly.

SDN Load Balancing

The main function of a SDN Load Balancer is physically separate the network control plane from the forwarding plane. Using SDN load balancer multiple devices can be controlled simultaneously. This feature optimizes the load balancing. SDN load balancing promotes better network management and diagnosis by hardware level removing protocols. The SDN load balancing makes data path controls decisions independent of network equipment algorithms. SDN load balancer also reduces running time by control the entire network and web servers. It promotes faster delivery of the requests by finding the best pathway.

Software Requirement for the Project

- Microsoft Word
- Win Zip as necessary
- Oracle VirtualBox
- Virtual Box Mininet Image
- Java Programming language, eclipse, Intellij, etc.
- JDK and JRE
- Windows OS

METHODOLOGY

To completely implement this project, following steps were taken:

1. Environment Setup:
    - Installed Oracle Virtual Box on a Windows Machines.
    - Imported the Mininet VM image into the virtual box.
    - Updated network Setting of Virtual box to accept Bridged Adapter to enable network on the virtual machine.

- All the software requirements were downloaded and installed from the below links.
    - https://cs.nyu.edu/~jcf/classes/CSCI-GA.2262-001/
- Retrieved the IP of the VM Mininet using *ifconfig* command.
- Using Windows terminal accessed VM Mininet using the SSH command.

2. Coding: In shortest path java class and Load balancer java class, the missing pieces of code (TODO part) were written to complete the class. (Coding part is referenced from Github repository mentioned under references.)

    LoadBalancer.java:

- TODO: Perform other tasks, if necessary:

```
public void createswitch_rules(IOFSwitch sch, String mark){
    for(long ct:this.instances.keySet()){
        OFMatch om = new OFMatch();
        ArrayList<OFMatchField> field_list = new ArrayList<OFMatchField>();
        OFMatchField etype;
        OFMatchField match;
        if(mark.equals("arp")){
            log.info("ARP switch rules added");
            etype = new OFMatchField(OFOXMFieldType.ETH_TYPE, Ethernet.TYPE_ARP);
            match = new OFMatchField(OFOXMFieldType.ARP_TPA, ct);
            }
        else if(mark.equals("ip")){
            log.info("IP switch rules added");
            etype = new OFMatchField(OFOXMFieldType.ETH_TYPE, Ethernet.TYPE_IPv4);
            match = new OFMatchField(OFOXMFieldType.IPV4_DST, ct);
        }
        else{
            log.info("Wrong packet type");
            return;
        }

        field_list.add(etype);
        field_list.add(match);
        om.setMatchFields(field_list);

        OFActionOutput output_of = new OFActionOutput();
        output_of.setPort(OFPort.OFPP_CONTROLLER);
        ArrayList<OFAction> action_list = new ArrayList<OFAction>();
        action_list.add(output_of);

        OFInstructionApplyActions actions = new OFInstructionApplyActions(action_list);
        ArrayList<OFInstruction> instruction_lists = new ArrayList<OFInstruction>();
        instruction_lists.add(actions);

        SwitchCommands.installRule(sch, this.table, SwitchCommands.DEFAULT_PRIORITY,
        om, instruction_lists);
    }
```

```
public void createmore(IOFSwitch sch){

    log.info("Other switch rules added.");

    OFMatch om = new OFMatch();

    OFInstructionGotoTable instruct_tab = new OFInstructionGotoTable();

    instruct_tab.setTableId(ShortestPathSwitching.table);

    ArrayList<OFInstruction> instruction_list = new ArrayList<OFInstruction>();

    instruction_list.add(instruct_tab);

    SwitchCommands.installRule(sch, this.table, (short)(SwitchCommands.DEFAULT_PRIORITY-1), om, instruction_list);
}
```

- TODO: Install rules to send:

```
public void replyArp(Ethernet eth_packet, OFPacketIn packet_in, IOFSwitch sch){
    ARP arp_packt = (ARP) eth_packet.getPayload();

    long ip_virtual = IPv4.toIPv4Address(arp_packt.getTargetProtocolAddress());

    boolean mark = false;

    for(int validip:instances.keySet()){
        if(validip==ip_virtual){
            mark = true;
            break;
        }
    }

    if(mark&&(arp_packt.getOpCode() == ARP.OP_REQUEST)) {

        byte[] vh_mac = instances.get(ip_virtual).getVirtualMAC();

        log.info("Arp reply process!");

        arp_packt.setOpCode(ARP.OP_REPLY);
        arp_packt.setTargetHardwareAddress(arp_packt.getSenderHardwareAddress());
        arp_packt.setTargetProtocolAddress(arp_packt.getSenderProtocolAddress());
        arp_packt.setSenderHardwareAddress(vh_mac);
        arp_packt.setSenderProtocolAddress(ip_virtual);

        eth_packet.setDestinationMACAddress(eth_packet.getSourceMACAddress());
        eth_packet.setSourceMACAddress(vh_mac);
        log.info("Arp reply packet sending: "+arp_packt.toString());
        SwitchCommands.sendPacket(sch, (short) packet_in.getInPort(), eth_packet);
    }
}
```

```
public void reWrite(Ethernet ethrnt_packt, OFPacketIn packt_in, IOFSwitch sch){
    IPv4 ip_packt = (IPv4) ethrnt_packt.getPayload();
    if(ip_packt.getProtocol() != IPv4.PROTOCOL_TCP) return;
    TCP tcp_packt = (TCP) ip_packt.getPayload();
    long ip_virtual = ip_packt.getDestinationAddress();
    if(tcp_packt.getFlags() == TCP_FLAG_SYN){
        log.info("TCP SYNs rewriting.");
        long ip_src = ip_packt.getSourceAddress();

        boolean mark = false;

        for(int ip_valid:instances.keySet()){

            if(ip_valid==ip_virtual){

                mark = true;
                break;
            }
        }

        if(!mark) return;

        long port_src = tcp_packt.getSourcePort();
        long port_dst = tcp_packt.getDestinationPort();

        long ip_host = instances.get(ip_virtual).getNextip_host();
        byte[] mac_host = getmac_hostAddress(ip_host);

        log.info(String.format("Rewriting the IP address to %s, rewriting the MAC address to %s"
                , IPv4.fromIPv4Address(ip_host), MACAddress.valueOf(mac_host).toString()));

        OFMatch om;
        ArrayList<OFMatchField> field_list;
        ArrayList<OFAction> action_list;
        ArrayList<OFInstruction> instruction_list;
```

```
    for(long j=0; j<2; j++){

        field_list = new ArrayList<OFMatchField>();
        field_list.add(new OFMatchField(OFOXMFieldType.ETH_TYPE, Ethernet.TYPE_IPv4));
        field_list.add(new OFMatchField(OFOXMFieldType.IPV4_SRC, ip_src));
        field_list.add(new OFMatchField(OFOXMFieldType.IPV4_DST, ip_virtual));
        field_list.add(new OFMatchField(OFOXMFieldType.IP_PROTO, IPv4.PROTOCOL_TCP));

        if(j == 1){
            field_list.add(new OFMatchField(OFOXMFieldType.TCP_SRC, port_src));
            field_list.add(new OFMatchField(OFOXMFieldType.TCP_DST, port_dst));
            action_list = new ArrayList<OFAction>();
            action_list.add(new OFActionSetField(OFOXMFieldType.ETH_DST, mac_host));
            action_list.add(new OFActionSetField(OFOXMFieldType.IPV4_DST, ip_host));
        }

        else{
            field_list.add(new OFMatchField(OFOXMFieldType.TCP_SRC, port_dst));
            field_list.add(new OFMatchField(OFOXMFieldType.TCP_DST, port_src));
            action_list = new ArrayList<OFAction>();
            action_list.add(new OFActionSetField(OFOXMFieldType.ETH_SRC, instances.get(ip_virtual).getVirtualMAC()));
            action_list.add(new OFActionSetField(OFOXMFieldType.IPV4_SRC, ip_virtual));
        }


        om = new OFMatch();
        om.setMatchFields(field_list);
        OFInstructionApplyActions acts = new OFInstructionApplyActions(action_list);
        OFInstructionGotoTable instrct_got_tab = new OFInstructionGotoTable();
        instrct_got_tab.setTableId(ShortestPathSwitching.table);

        instruction_list = new ArrayList<OFInstruction>();
        instruction_list.add(acts);
        instruction_list.add(instrct_got_tab);
        SwitchCommands.installRule(sch, this.table, (short) (SwitchCommands.DEFAULT_PRIORITY + 1),
                om, instruction_list, SwitchCommands.NO_TIMEOUT, (short)IDLE_TIMEOUT);
    }
```

```
    else{

        log.info("TCP reset rewriting.");
        tcp_packt.setSourcePort(tcp_packt.getDestinationPort());
        tcp_packt.setDestinationPort(tcp_packt.getSourcePort());
        final byte tcprst_flag = 0x04;
        tcp_packt.setFlags((short) tcprst_flag);
        tcp_packt.setSequence(tcp_packt.getAcknowledge() );
        tcp_packt.setWindowSize((short) 0);
        tcp_packt.setChecksum((short) 0);
        tcp_packt.serialize();
        ip_packt.setPayload(tcp_packt);
        long destIp = ip_packt.getSourceAddress();
        long ip_src = ip_packt.getDestinationAddress();
        ip_packt.setDestinationAddress(destIp);
        ip_packt.setSourceAddress(ip_src);
        ip_packt.setChecksum((short) 0);
        ip_packt.serialize();
        ethrnt_packt.setPayload(ip_packt);
        byte[] destMac = ethrnt_packt.getSourceMACAddress();
        byte[] srcMac = ethrnt_packt.getDestinationMACAddress();
        ethrnt_packt.setDestinationMACAddress(destMac);
        ethrnt_packt.setSourceMACAddress(srcMac);
        SwitchCommands.sendPacket(sch, (short) packt_in.getInPort(), ethrnt_packt);
    }
}
```

- TODO: Send an ARP reply for ARP requests for virtual IPs; for TCP

```
private byte[] getHostMACAddress(int hostIPAddress)
{
    Iterator<? extends IDevice> iterator = this.deviceProv.queryDevices(
            null, null, hostIPAddress, null, null);
    if (!iterator.hasNext())
    { return null; }
    IDevice device = iterator.next();
    return MACAddress.valueOf(device.getMACAddress()).toBytes();
}
```

ShortestPathSwitching.java

- TODO: Initialize other class variables, if necessary

```
public HashMap<IOFSwitch, HashMap<IOFSwitch, IOFSwitch>> bf_shortest_path() {
    Collection<IOFSwitch> swtchs = getSwitches().values();
    HashMap<IOFSwitch, HashMap<IOFSwitch, IOFSwitch>> shortest_path = new HashMap<IOFSwitch, HashMap<IOFSwitch, IOFSwitch>>();
    for(IOFSwitch x : swtchs) {
        HashMap<IOFSwitch, Integer> hm = new HashMap<IOFSwitch, Integer>();
        HashMap<IOFSwitch, IOFSwitch> prdcsr = new HashMap<IOFSwitch, IOFSwitch>();
        for(IOFSwitch z : swtchs) {
            hm.put(z, Integer.MAX_VALUE - 1);
            prdcsr.put(z, null);
        }
        hm.put(x, 0);
        for(int i=0;i<swtchs.size()-1;i++){
            for(Link link : getLinks()) {
                IOFSwitch src = getSwitches().get(link.getSrc());
                IOFSwitch dst = getSwitches().get(link.getDst());
                if(hm.get(src) +1 < hm.get(dst)) {
                    hm.put(dst, hm.get(src)+1);
                    prdcsr.put(dst, src);
                }else if(hm.get(dst) +1 < hm.get(src)){
                    hm.put(src, hm.get(dst)+1);
                    prdcsr.put(src, dst);
                }
            }
        }
        shortest_path.put(x, prdcsr);
    }
    return shortest_path;
}
/*********************************************************************/


/*********************************************************************/

 private void log_data() {

    StringBuilder msg = new StringBuilder();

    msg.append("\n#################### LOG DATA ####################################");
    msg.append(get_sp_string(this.paths));

    log.info(msg.toString());
}
```

```
private String get_sp_string(HashMap<IOFSwitch, HashMap<IOFSwitch, IOFSwitch>> shortest_path) {

    StringBuilder msg = new StringBuilder();

    msg.append("\n############ShortestPaths############\n");

        for (Map.Entry<IOFSwitch, HashMap<IOFSwitch, IOFSwitch>> inr : shortest_path.entrySet()) {
            Iterator<Map.Entry<IOFSwitch, IOFSwitch>> itr2 = inr.getValue().entrySet().iterator();
            msg.append(inr.getKey().getStringId()).append(": {");
            msg.append("\n");

            while (itr2.hasNext()) {
                Map.Entry<IOFSwitch, IOFSwitch> inr2 = itr2.next();
                msg.append("{ ");

                if (inr2.getKey() != null && inr2.getKey().getStringId() != null) {
                    msg.append(inr2.getKey().getStringId());
                } else {
                    msg.append("null");
                }
                msg.append(" : ");

                if (inr2.getValue() != null) {
                    msg.append(inr2.getValue().getStringId());
                } else {
                    msg.append("null");
                }
                msg.append(", ");
                msg.append("}\n");
            }
            msg.append("}\n");
        }

        return msg.toString();
    }
```

- TODO: Perform other tasks, if necessary. Initialize the shortest paths hash map here:

```
public byte getTable()
{ return table; }


private Collection<Host> getHosts()
{ return this.knownHosts.values(); }


private Map<Long, IOFSwitch> getSwitches()
{ return floodlightProv.getAllSwitchMap(); }


private Collection<Link> getLinks()
{ return linkDiscProv.getLinks().keySet(); }
```

```
public void set_flow(Host hst) {

    if(hst.isAttachedToSwitch()) {

        IOFSwitch s_hst = hst.getSwitch();
        OFMatch om = new OFMatch();
        ArrayList<OFMatchField> listofField = new ArrayList<OFMatchField>();
        OFMatchField etype = new OFMatchField(OFOXMFieldType.ETH_TYPE, Ethernet.TYPE_IPv4);
        OFMatchField mac = new OFMatchField(OFOXMFieldType.ETH_DST, Ethernet.toByteArray(hst.getMACAddress()));
        listofField.add(etype);
        listofField.add(mac);
        om.setMatchFields(listofField);

        for(IOFSwitch shc : getSwitches().values()) {
            OFActionOutput out_ofa = new OFActionOutput();
            if(shc.getId() == s_hst.getId()) {
                out_ofa.setPort(hst.getPort());
            } else {
                if(this.paths.containsKey(s_hst)&&this.paths.get(s_hst).containsKey(s)) {
                    IOFSwitch prd = this.paths.get(s_hst).get(s);
                    for (Link lnk : getLinks()) {
                        if ((prd.getId() == lnk.getDst()) && (shc.getId() == lnk.getSrc())) {
                            out_ofa.setPort(lnk.getSrcPort());
                        }
                    }
                }
            }

            ArrayList<OFAction> action_list = new ArrayList<OFAction>();
            ArrayList<OFInstruction> instruction_list = new ArrayList<OFInstruction>();
            action_list.add(out_ofa);
            instruction_list.add(new OFInstructionApplyActions(action_list));
            SwitchCommands.installRule(s, table, SwitchCommands.DEFAULT_PRIORITY, om, instruction_list,
                    SwitchCommands.NO_TIMEOUT, SwitchCommands.NO_TIMEOUT);
        }
    }
}
```

```
public void create_flow_tabs() {

    for(Host hst : getHosts()) {

        set_flow(hst);

    }
}


public void del_flow(Host hst) {

    OFMatch om = new OFMatch();
    ArrayList<OFMatchField> field_list = new ArrayList<OFMatchField>();
    OFMatchField etype = new OFMatchField(OFOXMFieldType.ETH_TYPE, Ethernet.TYPE_IPv4);
    OFMatchField dst_m = new OFMatchField(OFOXMFieldType.ETH_DST, Ethernet.toByteArray(hst.getMACAddress()));
    OFMatchField src_m = new OFMatchField(OFOXMFieldType.ETH_SRC, Ethernet.toByteArray(hst.getMACAddress()));

    field_list.add(etype);
    field_list.add(dst_m);
    field_list.add(src_m);
    om.setMatchFields(field_list);

    for(IOFSwitch shc : getSwitches().values()) {
        SwitchCommands.removeRules(shc, table, om);
    }
}


public void del_flow_tabs() {

    for(Host hst : getHosts()) {
        del_flow(hst);

    }
}
/*********************************************************************************/
```

- TODO: Update routing: change rules to route to host

```
del_flow(host);
create_flow_tabs();
```

3. Compilation: Compiled the Floodlight and SDN applications to produce the FloodlightWithApps. Jar file.
4. After Compilation, Floodlight and SDN applications were initiated.
5. Mininet was started. Networks with different topologies and different number of hosts were created.
6. To test the networks, we used following commands:
    - pingall
    - h1 ping h2
    - link s2 s3 down
    - link s2 s3 up
7. To check the load balancer, Floodlight and Load balancer was started.
8. To test the load balancer, incoming tcp messages on switch 1 were checked.

RESULTS

1) Layer-3 Routing and Shortest Path:

Topology: linear, Switches: 5

Initialization

```
root@mininet-VirtualBox:~/rsc9044_project# ./run_mininet.py linear,5
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1 s2 s3 s4 s5
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (s1, s2) (s2, s3) (s3, s4) (s4, s5)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
*** Starting 5 switches
s1 s2 s3 s4 s5
*** ARPing from host h1
*** Starting SimpleHTTPServer on host h1
*** ARPing from host h2
*** Starting SimpleHTTPServer on host h2
*** ARPing from host h3
*** Starting SimpleHTTPServer on host h3
*** ARPing from host h4
*** Starting SimpleHTTPServer on host h4
*** ARPing from host h5
*** Starting SimpleHTTPServer on host h5
*** Starting CLI:
mininet>
```

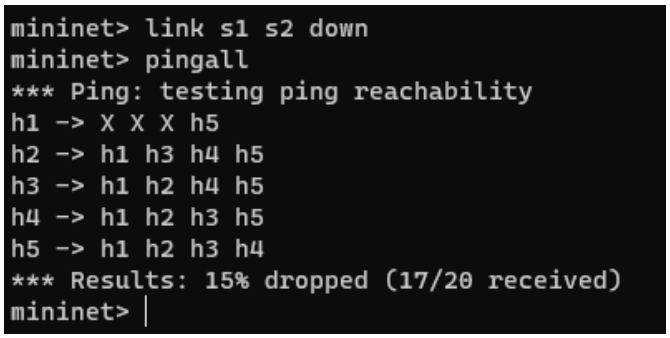
Pingall: No packets loss.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5
h2 -> h1 h3 h4 h5
h3 -> h1 h2 h4 h5
h4 -> h1 h2 h3 h5
h5 -> h1 h2 h3 h4
*** Results: 0% dropped (20/20 received)
mininet>
mininet>
mininet>
```

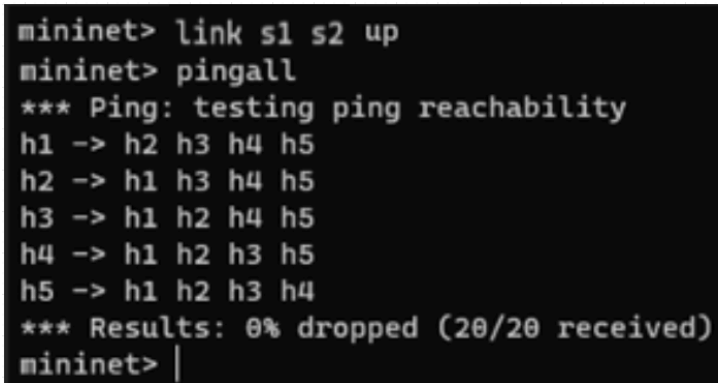
Ping from h1 to h3

```
mininet> h1 ping h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=18.6 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=1.01 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.128 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.085 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.150 ms
^C
--- 10.0.0.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4002ms
rtt min/avg/max/mdev = 0.085/4.010/18.675/7.340 ms
mininet>
```

Link between s1 and s2 is brought down. 15% packet dropped observed.

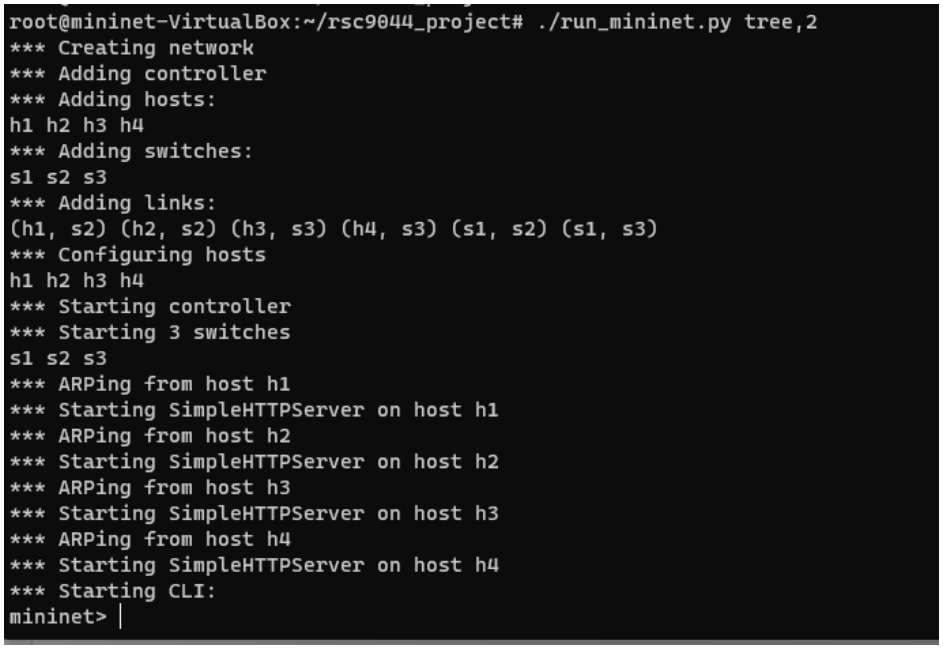


Link between s1 and s2 is brought up. No packet loss.



<u>Topology: linear, tree: 2</u>

Initialization



Pingall: No packet loss.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet>
```

Link between s2 and s1 is brought down. 8% packet dropped.

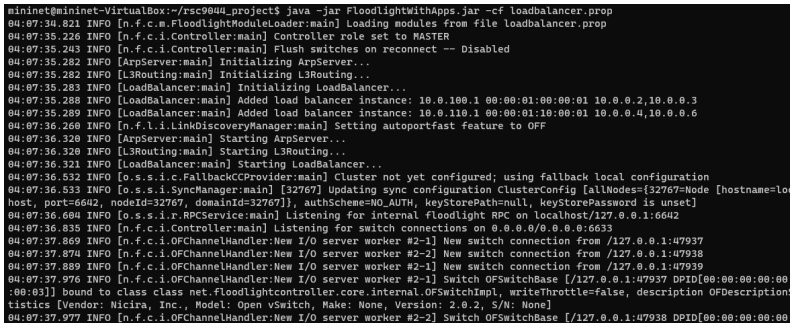
```
mininet> link s2 s1 down
mininet>
mininet>
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 8% dropped (11/12 received)
```

Link between s2 and s1 is brought up. 0% packet dropped.

```
mininet> link s2 s1 up
mininet>
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet>
```

Load Balancer:

Initialization

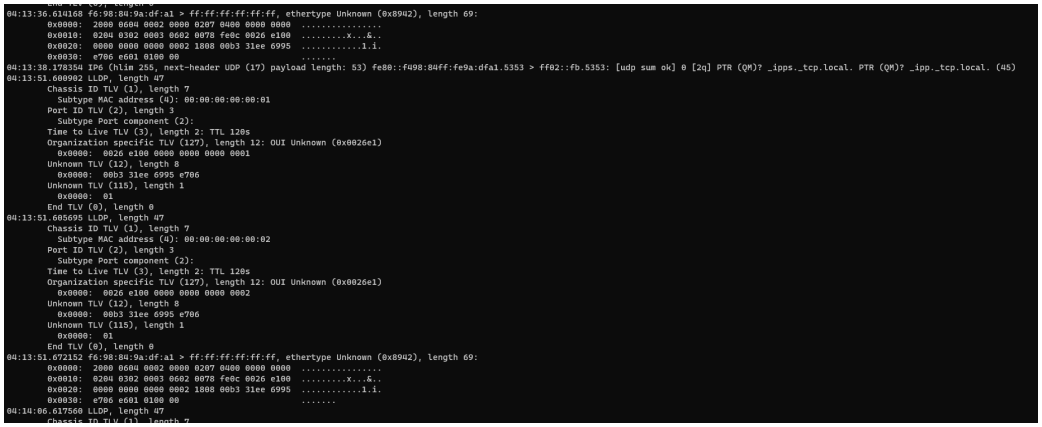


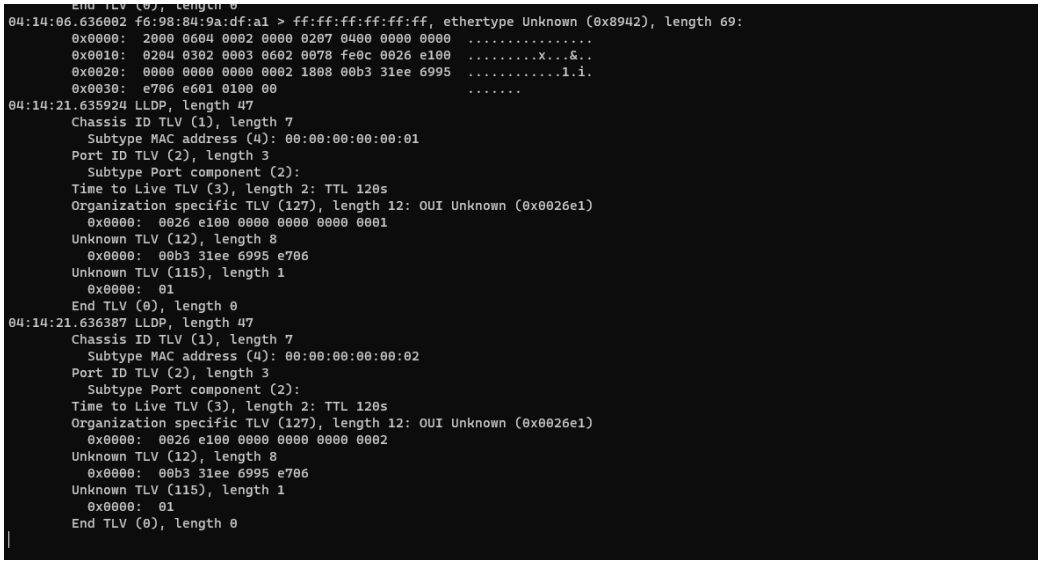
Checking TCP incoming messages on switch s1.

```
root@mininet-VirtualBox:~/rsc9044_project# tcpdump -v -n -i s1-eth1
tcpdump: WARNING: s1-eth1: no IPv4 address assigned
tcpdump: listening on s1-eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
04:12:31.657329 IP (tos 0x10, ttl 128, id 0, offset 0, flags [none], proto UDP (17), length 328)
    0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from f6:98:84:9a:df:a1, length 300, xid 0x1e18f50a, secs 42, Flags [none]
          Client-Ethernet-Address f6:98:84:9a:df:a1
          Vendor-rfc1048 Extensions
            Magic Cookie 0x63825363
            DHCP-Message Option 53, length 1: Discover
            Hostname Option 12, length 18: "mininet-VirtualBox"
            Parameter-Request Option 55, length 18:
              Subnet-Mask, BR, Time-Zone, Default-Gateway
              Domain-Name, Domain-Name-Server, Option 119, Hostname
              Netbios-Name-Server, Netbios-Scope, MTU, Classless-Static-Route
              NTP, Classless-Static-Route, Classless-Static-Route-Microsoft, Static-Route
              Option 252, NTP
04:12:33.869143 IP6 (hlim 1, next-header Options (0) payload length: 56) :: > ff02::16: HBH (rtalert: 0x0000) (padn) [icmp6 sum ok]
ICMP6, multicast listener report v2, 2 group record(s) [gaddr ff02::fb to_in, 0 source(s)] [gaddr ff02::1:ff9a:dfa1 to_ex, 0 source(
s)]
04:12:34.069125 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 24) :: > ff02::1:ff9a:dfa1: [icmp6 sum ok] ICMP6, neighbor so
licitation, length 24, who has fe80::f498:84ff:fe9a:dfa1
04:12:34.453203 IP6 (hlim 1, next-header Options (0) payload length: 56) :: > ff02::16: HBH (rtalert: 0x0000) (padn) [icmp6 sum ok]
ICMP6, multicast listener report v2, 2 group record(s) [gaddr ff02::fb to_in, 0 source(s)] [gaddr ff02::1:ff9a:dfa1 to_ex, 0 source(
s)]
04:12:35.069231 IP6 (hlim 1, next-header Options (0) payload length: 36) fe80::f498:84ff:fe9a:dfa1 > ff02::16: HBH (rtalert: 0x0000)
 (padn) [icmp6 sum ok] ICMP6, multicast listener report v2, 1 group record(s) [gaddr ff02::1:ff9a:dfa1 is_ex, 0 source(s)]
04:12:35.077155 IP6 (hlim 1, next-header Options (0) payload length: 36) fe80::f498:84ff:fe9a:dfa1 > ff02::16: HBH (rtalert: 0x0000)
 (padn) [icmp6 sum ok] ICMP6, multicast listener report v2, 1 group record(s) [gaddr ff02::fb to_ex, 0 source(s)]
04:12:35.174495 IP6 (hlim 255, next-header UDP (17) payload length: 53) fe80::f498:84ff:fe9a:dfa1.5353 > ff02::fb.5353: [udp sum ok]
 0 [2q] PTR (QM)? _ipps._tcp.local. PTR (QM)? _ipp._tcp.local. (45)
04:12:35.184923 IP6 (hlim 255, next-header UDP (17) payload length: 133) fe80::f498:84ff:fe9a:dfa1.5353 > ff02::fb.5353: [udp sum ok
```

```
] 0*- [0q] 2/0/0 _services._dns-sd._udp.local. PTR _workstation._tcp.local., _workstation._tcp.local. PTR mininet-VirtualBox [f6:98:
84:9a:df:a1]._workstation._tcp.local. (125)
04:12:36.521586 LLDP, length 47
        Chassis ID TLV (1), length 7
          Subtype MAC address (4): 00:00:00:00:00:02
        Port ID TLV (2), length 3
          Subtype Port component (2):
        Time to Live TLV (3), length 2: TTL 120s
        Organization specific TLV (127), length 12: OUI Unknown (0x0026e1)
          0x0000:  0026 e100 0000 0000 0000 0002
        Unknown TLV (12), length 8
          0x0000:  00b3 31ee 6995 e706
        Unknown TLV (115), length 1
          0x0000:  01
        End TLV (0), length 0
04:12:36.526026 LLDP, length 47
        Chassis ID TLV (1), length 7
          Subtype MAC address (4): 00:00:00:00:00:01
        Port ID TLV (2), length 3
          Subtype Port component (2):
        Time to Live TLV (3), length 2: TTL 120s
        Organization specific TLV (127), length 12: OUI Unknown (0x0026e1)
          0x0000:  0026 e100 0000 0000 0000 0001
        Unknown TLV (12), length 8
          0x0000:  00b3 31ee 6995 e706
        Unknown TLV (115), length 1
          0x0000:  01
        End TLV (0), length 0
04:12:36.958156 IP6 (hlim 255, next-header UDP (17) payload length: 273) fe80::f498:84ff:fe9a:dfa1.5353 > ff02::fb.5353: [udp sum ok
] 0*- [0q] 5/0/0 mininet-VirtualBox [f6:98:84:9a:df:a1]._workstation._tcp.local. (Cache flush) TXT "", mininet-VirtualBox.local. (Ca
```

```
04:12:51.945660 f6:98:84:9a:df:a1 > ff:ff:ff:ff:ff:ff, ethertype Unknown (0x8942), length 69:
        0x0000:  2000 0604 0002 0000 0207 0400 0000 0000  ................
        0x0010:  020d 0302 0003 0602 0078 fe0c 0026 e100  .........x...&..
        0x0020:  0000 0000 0000 0002 1808 00b3 31ee 6995  ............1.i.
        0x0030:  e706 e601 0100 00                        .......
04:13:06.177212 IP6 (hlim 255, next-header UDP (17) payload length: 53) fe80::f498:84ff:fe9a:dfa1.5353 > ff02::fb.5353: [udp sum ok] 0 [2q] PTR (QM)? _ipps._tcp.local. PTR (QM)? _ipp._tcp.local. (45)
04:13:06.554749 LLDP, length 47
        Chassis ID TLV (1), length 7
          Subtype MAC address (4): 00:00:00:00:00:02
        Port ID TLV (2), length 3
          Subtype Port component (2):
        Time to Live TLV (3), length 2: TTL 120s
        Organization specific TLV (127), length 12: OUI Unknown (0x0026e1)
          0x0000:  0026 e100 0000 0000 0000 0002
        Unknown TLV (12), length 8
          0x0000:  00b3 31ee 6995 e706
        Unknown TLV (115), length 1
          0x0000:  01
        End TLV (0), length 0
04:13:06.555485 LLDP, length 47
        Chassis ID TLV (1), length 7
          Subtype MAC address (4): 00:00:00:00:00:01
        Port ID TLV (2), length 3
          Subtype Port component (2):
        Time to Live TLV (3), length 2: TTL 120s
        Organization specific TLV (127), length 12: OUI Unknown (0x0026e1)
          0x0000:  0026 e100 0000 0000 0000 0001
        Unknown TLV (12), length 8
          0x0000:  00b3 31ee 6995 e706
        Unknown TLV (115), length 1
          0x0000:  01
        End TLV (0), length 0
04:13:21.560756 LLDP, length 47
        Chassis ID TLV (1), length 7
          Subtype MAC address (4): 00:00:00:00:00:01
        Port ID TLV (2), length 3
          Subtype Port component (2):
        Time to Live TLV (3), length 2: TTL 120s
        Organization specific TLV (127), length 12: OUI Unknown (0x0026e1)
          0x0000:  0026 e100 0000 0000 0000 0001
        Unknown TLV (12), length 8
          0x0000:  00b3 31ee 6995 e706
        Unknown TLV (115), length 1
          0x0000:  01
        End TLV (0), length 0
```





CONCLUSION

In the Project, we have successfully implemented the Layer-3 Routing application and Load Balancer application in the given software defined network.

SDNs create the centralized view of a vast network. This centralization eases the network management and service provisioning. An SDN provides a set of APIs which create a central console to manage Physical and Virtual Devices. SDN increases the real time visibility into the network performance. Using the increase visibility, performance of the network can be optimized. SDN provides capability of automatic real time re-routing or to withstand the new functions and routes. Uptime can also be increased without adding new hardware and that too in low cost. In SDN, automated change in scaling is possible. Along with extra visibility and automatic scaling empowers the management and engineers with operational capacity to normalize traffic across the wide network in less amount of time.

SDN Load balancer is easy to implement on demand in less time and cost. In turn, hardware load balancers are costly and difficult to be installed. Software based load balancer can also be used to provide the extra layer of security since it is placed between Client and Servers, and it can be used to reject suspicious data packets. Software load balancers can also be used to control multiple devices simultaneously hence increasing agility into the network.

REFERENCES

- https://en.wikipedia.org/wiki/OpenFlow
- [Coding] https://github.com/zqf0722/Software-Defined-Networking-Application
- https://avinetworks.com/glossary/sdn-load-balancing/
- https://docs.microsoft.com/en-us/windows-server/networking/technologies/network-load-balancing
- https://imaginenext.ingrammicro.com/data-center/7-advantages-of-software-defined-networking
- https://www.ieee.org/conferences/publishing/templates.html
- http://www.networkcomputing.com/cloud-infrastructure/7-essentials-software-definednetworking/1672
- Data Communication and Networks, Prof. Jean-Claude Franchitti, Slides and Class notes, NYU SPRING 2022