Networks Module B
Lab 2

Dillon Heald
HLDDIL001

30 April 2020

1 Part 1: Firewall

In this section, the questions for the firewall implementation will be answered.

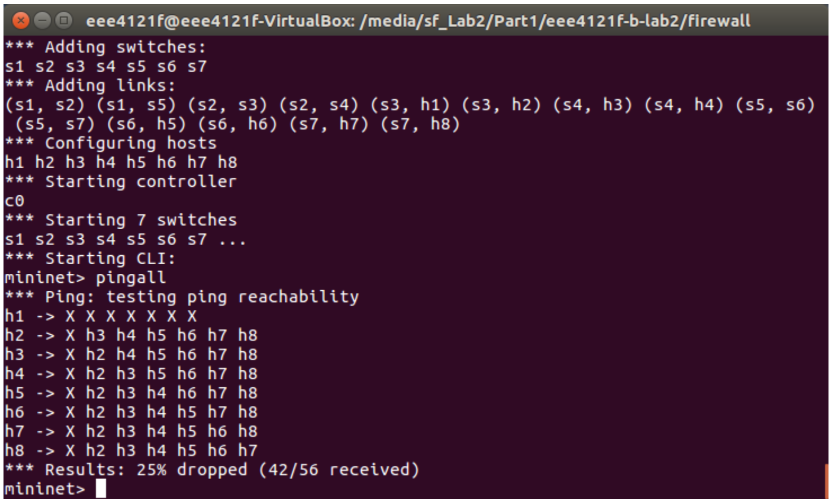
1.1 Question 1

A layer 2 learning switch functions by learning what source MAC addresses exist on the ports it has. Each time a host sends a frame through the switch to a destination, the switch learns what port is associated with that source MAC address and updates its routing table to reflect that. In the future when an external host wishes to send a frame to a host on connected to this switch, the L2 learning switch identifies the destination MAC address and forwards the packet to the port where it learnt that MAC address exists. L2 broadcast frames will still be sent to all of its ports. In the case of a unknown unicast, where the MAC address is not known yet, the switch broadcasts that frame out of all its ports with the hope that it will reach its intended destination.

1.2 Question 2

Initially, the firewall policies are loaded into a local variable within the L2_learning python script. Then within the `_handle_PacketIn()` function an if statement is added which checks the input packet's source MAC address against the blocked MAC addresses. If the input packet matches with a blocked address it is dropped.

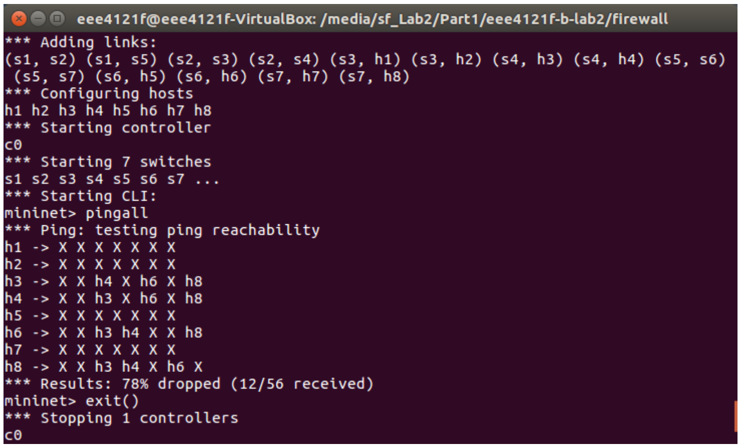
1.3 Question 3



eee4121f@eee4121f-VirtualBox: /media/sf_Lab2/Part1/eee4121f-b-lab2/firewall

```
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(s1, s2) (s1, s5) (s2, s3) (s2, s4) (s3, h1) (s3, h2) (s4, h3) (s4, h4) (s5, s6)
 (s5, s7) (s6, h5) (s6, h6) (s7, h7) (s7, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X X X X X
h2 -> X h3 h4 h5 h6 h7 h8
h3 -> X h2 h4 h5 h6 h7 h8
h4 -> X h2 h3 h5 h6 h7 h8
h5 -> X h2 h3 h4 h6 h7 h8
h6 -> X h2 h3 h4 h5 h7 h8
h7 -> X h2 h3 h4 h5 h6 h8
h8 -> X h2 h3 h4 h5 h6 h7
*** Results: 25% dropped (42/56 received)
mininet> 
```

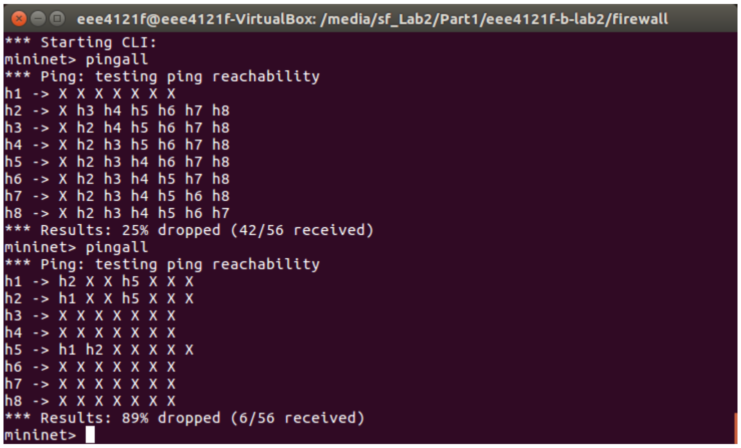
Figure 1: H1 blocked by firewall while all others are free to send/receive



eee4121f@eee4121f-VirtualBox: /media/sf_Lab2/Part1/eee4121f-b-lab2/firewall

```
*** Adding links:
(s1, s2) (s1, s5) (s2, s3) (s2, s4) (s3, h1) (s3, h2) (s4, h3) (s4, h4) (s5, s6)
 (s5, s7) (s6, h5) (s6, h6) (s7, h7) (s7, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X X X X X
h2 -> X X X X X X X
h3 -> X X h4 X h6 X h8
h4 -> X X h3 X h6 X h8
h5 -> X X X X X X X
h6 -> X X h3 h4 X X h8
h7 -> X X X X X X X
h8 -> X X h3 h4 X h6 X
*** Results: 78% dropped (12/56 received)
mininet> exit()
*** Stopping 1 controllers
c0
```

Figure 2: H1, H2, H5, H7 blocked by firewall while all others are free to send/receive

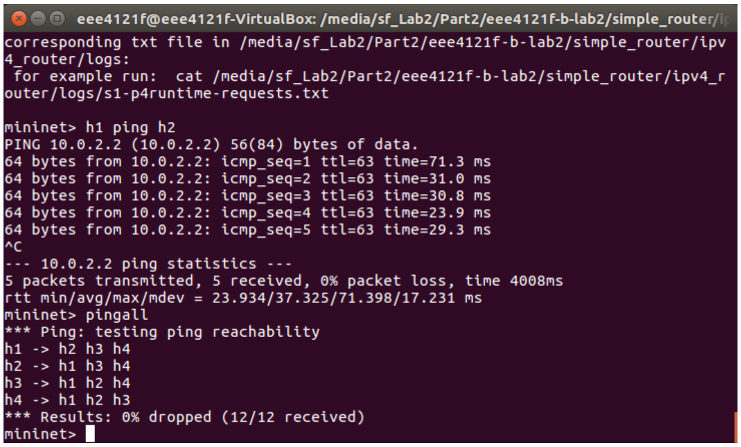


eee4121f@eee4121f-VirtualBox: /media/sf_Lab2/Part1/eee4121f-b-lab2/firewall

```
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X X X X X
h2 -> X h3 h4 h5 h6 h7 h8
h3 -> X h2 h4 h5 h6 h7 h8
h4 -> X h2 h3 h5 h6 h7 h8
h5 -> X h2 h3 h4 h6 h7 h8
h6 -> X h2 h3 h4 h5 h7 h8
h7 -> X h2 h3 h4 h5 h6 h8
h8 -> X h2 h3 h4 h5 h6 h7
*** Results: 25% dropped (42/56 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X X h5 X X X
h2 -> h1 X X h5 X X X
h3 -> X X X X X X X
h4 -> X X X X X X X
h5 -> h1 h2 X X X X X
h6 -> X X X X X X X
h7 -> X X X X X X X
h8 -> X X X X X X X
*** Results: 89% dropped (6/56 received)
mininet> 
```

Figure 3: H3, H4, H6, H7, H8 blocked by firewall while all others are free to send/receive

2 Part 2: Simple router in the data plane


eee4121f@eee4121f-VirtualBox: /media/sf_Lab2/Part2/eee4121f-b-lab2/simple_router/ip

```
corresponding txt file in /media/sf_Lab2/Part2/eee4121f-b-lab2/simple_router/ipv
4_router/logs:
 for example run:  cat /media/sf_Lab2/Part2/eee4121f-b-lab2/simple_router/ipv4_r
outer/logs/s1-p4runtime-requests.txt

mininet> h1 ping h2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
64 bytes from 10.0.2.2: icmp_seq=1 ttl=63 time=71.3 ms
64 bytes from 10.0.2.2: icmp_seq=2 ttl=63 time=31.0 ms
64 bytes from 10.0.2.2: icmp_seq=3 ttl=63 time=30.8 ms
64 bytes from 10.0.2.2: icmp_seq=4 ttl=63 time=23.9 ms
64 bytes from 10.0.2.2: icmp_seq=5 ttl=63 time=29.3 ms
^C
--- 10.0.2.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4008ms
rtt min/avg/max/mdev = 23.934/37.325/71.398/17.231 ms
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> 
```

Figure 4: Showing the result of a `pingall` command in mininet after implementation of the simple dataplane router.

The above figure 4 shows the results following the implementation of the simple packet forwarding expected of a router.

2.1 Question 4

To handle ARP requests, a custom header could be defined which responds to the ARP request with the router's MAC address.

2.2 Question 5

Traditionally a traceroute is done by incrementing the TTL of a packet by 1 and recording the ICMP packets returned by each router along its path as the TTL hits 0. While this is simple and works, it introduces extra traffic into a network and the results recorded can be inconsistent if the path changes frequently which results in a bad model of the topology.
Instead, I would add a probe header field which can be used to record the switch ID, the port egress port and the current and last time. This packet can then be used to describe the topology with a single pass through the network.

2.3 Question 6

The router works as expected since all it needs to do is forward packets according to where the SDN tells it to, thereby reducing the computation needed to be done by the router. This router does not collect measurement data which is something the SDN would need to compute best paths, thus; the router can be improved upon by addition of probing select packets for measurement purposes.