Mininet & OpenFlow

Abhay Tamilselvan (W1606871)

TASK 1

1. Output of “nodes”

```
    available nodes are:
    c0 h1 h2 h3 h4 h5 h6 h7 h8 s1 s2 s3 s4 s5 s6 s7
```

Output of “net”

```
    h1 h1-eth0:s3-eth1
    h2 h2-eth0:s3-eth2
    h3 h3-eth0:s4-eth1
    h4 h4-eth0:s4-eth2
    h5 h5-eth0:s6-eth1
    h6 h6-eth0:s6-eth2
    h7 h7-eth0:s7-eth1
    h8 h8-eth0:s7-eth2
    s1 lo:  s1-eth1:s2-eth3 s1-eth2:s5-eth3
    s2 lo:  s2-eth1:s3-eth3 s2-eth2:s4-eth3 s2-eth3:s1-eth1
    s3 lo:  s3-eth1:h1-eth0 s3-eth2:h2-eth0 s3-eth3:s2-eth1
    s4 lo:  s4-eth1:h3-eth0 s4-eth2:h4-eth0 s4-eth3:s2-eth2
    s5 lo:  s5-eth1:s6-eth3 s5-eth2:s7-eth3 s5-eth3:s1-eth2
    s6 lo:  s6-eth1:h5-eth0 s6-eth2:h6-eth0 s6-eth3:s5-eth1
    s7 lo:  s7-eth1:h7-eth0 s7-eth2:h8-eth0 s7-eth3:s5-eth2
    c0
```

2. Output of “h7 ifconfig”

```
h7-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.7  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::f8bc:d0ff:fe5e:17d6  prefixlen 64  scopeid
0x20<link>
        ether fa:bc:d0:5e:17:d6  txqueuelen 1000  (Ethernet)
        RX packets 73  bytes 5542 (5.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12  bytes 936 (936.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
```

```
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

TASK 2

1. The functional call graph of the “of_tutorial” controller is as follows:

    1. launch function
    2. start_switch function
    3. Class Tutorial constructor
    4. handle_PacketIn method
    5. act_like_hub method
    6. resend_packet method

2.

    a. H1 ping h2 -> average ping time 3.104 ms
       h1 ping h8 -> average ping time 10.163 ms

    b. H1 ping h2
        - min ping time 1.760
        - max ping time 7.151

       h1 ping h8
        - min ping time 6.507
        - max ping time 21.652

    c. It took a longer time to ping from h1->h8 compared to h1->h2 because the packet needed to go through more number of switches. It had to climb all the way to the root switch(s1) before it could reach its destination.

    h1 ping h2 – h1 -> s3 -> h2
    h1 ping h8 – h1 -> s3 -> s2 -> s1 -> s5 -> s7 -> h8

3.

    a. iperf is used to test the bandwidth between two hosts.

    b. H1 -> h2 throughput is 11.1 MBits/sec
       h1 -> h8 throughput is 4.52 MBits/sec

    c. The throughput between h1->h2 is much greater than the throughput between h1->h8 because of two main reasons – **network congestion** and **latency.** Network congestion occurs when there is too much traffic passing through the switches. Latency is the time it

takes a packet to travel from sender to receiver. This is especially true for TCP connections because the next packet can be sent only after receiving the acknowledgement(ACK) from the receiver.

4. All the switches observe traffic because the switches currently act like a hub sending all packets to all ports beside the input port. I was able to observe the traffic by adding the following line of code in the **handle_packetIn function,**

log.debug("Switch observing traffic: %s" % (self.connection))

The handle_packetIn function is the event listener for the switch so whenever a switch receives a new packet, the handle_packetIn function is called. So by using the above line of code, we can observe which all switches are observing traffic since *self.connection* will show the current switch. We can also check the details of the packet being received by using the following commands:

log.debug("Parsed packet date: %s" % (event.parsed))
log.debug("Actual ofp_packet_in message : %s" % (event.ofp))

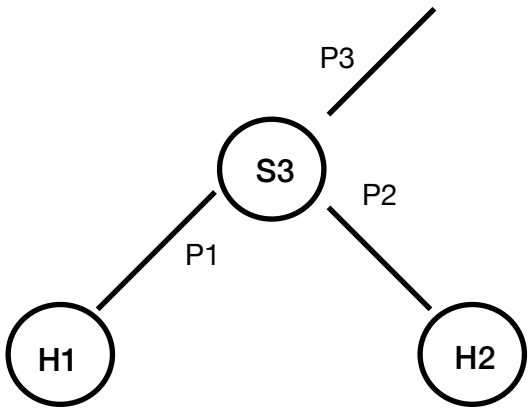
TASK 3

1. Each switch has a dictionary called *self.mac_to_port* which is used to map the MAC addresses of the host to the corresponding port. The dictionary is initially empty.

   When a switch receives a new packet, the act_like_switch function is called. It checks if the source MAC address is already in the dictionary. If it is not, then it maps the source MAC address to the corresponding port.

   In the next step, it checks if the destination MAC address is in the dictionary. If it is present, it resends the packet to that address. Else, it floods the packet to all the ports.

   The flow can be better explained through an example. Let's say we ping h2 from h1 for 100 times. The steps taking place would be as follows:



First Ping H1 -> H2

• H1 sends packet to s3 which it receives through port P1.
• S3 maps the source MAC address to that port -> {“H1 MAC address”:”P1”}.
• S3 doesn’t have the destination MAC address. So S3 forwards the packet to all other ports which is P2 and P3.
• The packet sent through P3 never reaches its destination.
• The packet sent through P2 reaches H2 which is connected to the destination MAC address.
• H2 sends an ACK packet back to S3. S3 will now add the new source address to its dictionary. {“H1 MAC address”:”P1”, “H2 MAC address”:”P2”}
• It checks for the destination MAC address which is H1 MAC address which is present in the dictionary. So it forwards the packet through P1.

Second Ping H1 -> H2

• H1 sends packet to s3 which it receives through port P1.
• S3 already has the source address. S3 also has the destination address in the dictionary. So it just forwards the packet to H2 through the P2 port.

Rest of the pings from both H1->H2 and H2->H1 would be forwarded by S3 to the correct destination.

2.

a. H1 ping h2 -> average ping time 3.009 ms
   H1 ping h8 -> average ping time 9.410 ms

b. H1 ping h2
   min ping time 1.705 ms
   max ping time 6.317 ms

   h1 ping h8
   min ping time 6.128 ms
   max ping time 18.666 ms

c. The values for h1 ping h2 in task 3 takes less time than task 2. The difference is not that huge in this case because there is only one switch connecting h1 and h2 so the s3 switch doesn’t face any major network congestion or latency due to flooding.

H1 ping h2	TASK 2	TASK 3
Min ping time (ms)	1.760	1.705
Avg ping time (ms)	3.104	3.009
Max ping time (ms)	7.151	6.317

In contrast, the values for h1 ping h8 show a more significant difference in time taken because it has to go through a lot more switches for a packet from h1 to reach h8. The time taken for task 3 is clearly less than time taken for task 2.

H1 ping h8	TASK 2	TASK 3
Min ping time (ms)	6.507	6.128
Avg ping time (ms)	10.163	9.410
Max ping time (ms)	21.652	18.666

The reason that task 3 takes less time to ping than task 2 is due to the fact that only the initial few packets are flooded in task 3. Once the destination MAC address is found, the switches will resend the packet only to the corresponding port that is mapped to the destination MAC address. So the subsequent pings will reach their destination much quicker and the switches will not have network congestion.

3.

a. H1 -> h2 throughput is 52.6 MBits/sec
   h1 -> h8 throughput is 6.85 MBits/sec

b. The throughput for task 3 is greater than that for task 2 in both cases. The increased throughput is due to lesser network congestion. The switches will not be burdened with huge number of incoming packets as there will not be flooding of packets due to the mapping that takes place in each switch.