COEN 241: Homework 3

Name: Mridul Amitkumar Gupta
SCU ID: 1607438

Task 1

1) The output of "nodes" is:
   available nodes are:
   h1 h2 h3 h4 h5 h6 h7 h8 s1 s2 s3 s4 s5 s6 s7

   The output of "net" is:
   h1 h1-eth0:s3-eth2
   h2 h2-eth0:s3-eth3
   h3 h3-eth0:s4-eth2
   h4 h4-eth0:s4-eth3
   h5 h5-eth0:s6-eth2
   h6 h6-eth0:s6-eth3
   h7 h7-eth0:s7-eth2
   h8 h8-eth0:s7-eth3
   s1 lo: s1-eth1:s2-eth1 s1-eth2:s5-eth1
   s2 lo: s2-eth1:s1-eth1 s2-eth2:s3-eth1 s2-eth3:s4-eth1
   s3 lo: s3-eth1:s2-eth2 s3-eth2:h1-eth0 s3-eth3:h2-eth0
   s4 lo: s4-eth1:s2-eth3 s4-eth2:h3-eth0 s4-eth3:h4-eth0
   s5 lo: s5-eth1:s1-eth2 s5-eth2:s6-eth1 s5-eth3:s7-eth1
   s6 lo: s6-eth1:s5-eth2 s6-eth2:h5-eth0 s6-eth3:h6-eth0
   s7 lo: s7-eth1:s5-eth3 s7-eth2:h7-eth0 s7-eth3:h8-eth0

2) The output of "h7 ifconfig" is:
   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::202d:4ff:fe61:ea78 prefixlen 64 scopeid 0x20<link>
   ether 22:2d:04:61:ea:78 txqueuelen 1000 (Ethernet)
   RX packets 237 bytes 36730 (36.7 KB)
   RX errors 0 dropped 0 overruns 0 frame 0
   TX packets 11 bytes 886 (886.0 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) Function call graph:
start switch : _handle_PacketIn() -> act_like_hub() -> resend_packet() -> send(msg)

2) h1 ping -c100 h2
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 98683ms
rtt min/avg/max/mdev = 1.003/1.489/3.982/0.499 ms

h1 ping -c100 h8
--- 10.0.0.8 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99155ms
rtt min/avg/max/mdev = 4.318/5.844/12.741/1.603 ms

a) h1 ping h2 - Average ping is 1.489 ms
h1 ping h8 - Average ping is 5.844 ms

b) h1 ping h2 - Minimum ping observed is 1.003 ms
h1 ping h8 - Minimum ping observed is 4.318 ms

h1 ping h2 - Maximum ping observed is 3.982 ms
h1 ping h8 - Maximum ping observed is 12.741 ms

c) As can be observed from the above values the time taken to ping from h1 to h8 is significantly higher than that of pinging from h1 to h2 this difference is due to the fact that there is only one switch between h1 and h2 i.e., switch s3 whereas there are five switches between h1 and h8 namely s3,s2,s1,s5 and s7 hence there are more hops required in order for data to be sent from h1 to h8.

3) a) Iperf is an open-source tool for helping the administrators measuring the bandwidth for the network performance and quality of a network line. The network link is restricted by two hosts running iperf. It is used to measure the throughput between any two nodes in a network line.

b) mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['10.63 Mbits/sec', '12.81 Mbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['3.27 Mbits/sec', '3.95 Mbits/sec']

c) As observed from the above values the throughput is higher between h1 and h2 than between h1 and h8 because of network congestion and latency. The number of hops between h1 and h2 are less and therefore more data can be transmitted in a shorter time. While the number of hops between h1 and h8 are more, and therefore less data can be transmitted in each time.

4) By adding log.info("Switch observing traffic: %s" % (self.connection) in the line number 107 "of_tutorial" controller we can view the information which helps us to observe the traffic. After seeing that, we can conclude that all the switches view the traffic, specifically when all are flooded with packets. The _handle_PacketIn function is the event listener so it's called every time a packet is received.

Task 3.

1) The act_like_switch function can map or "learn" which MAC addresses are located. So, if a MAC address is discovered to be a desired address that a sender sends to, the controller can map that MAC address to a port for simplicity. This also improves the performance of the controller when sending packets to already known addresses, as it just directs the packet to that known port. If the destination is not already known, the function simply floods the packet to all destinations. MAC Learning Controller also helps to improve the ping times and throughputs as flooding happens less often.

2) h1 ping -c100 h2
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 98789ms
rtt min/avg/max/mdev = 1.038/1.316/2.997/0.271 ms

h1 ping -c100 h8
--- 10.0.0.8 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 98992ms
rtt min/avg/max/mdev = 3.976/4.842/13.853/1.761 ms

a) h1 ping h2 - Average ping is 1.316 ms
h1 ping h8 - Average ping is 4.842 ms

b) h1 ping h2 - Minimum ping observed is 1.038 ms
h1 ping h8 - Minimum ping observed is 3.976 ms

h1 ping h2 - Maximum ping observed is 2.997 ms
h1 ping h8 - Maximum ping observed is 13.853 ms

c) Compared to task 2 the value for h1 ping h2 takes slightly less time in task 3, although the difference is not so significant. While in case of h1 and h8, the difference is significant for ping time values as it must go through a lot more switches. Clearly task 3 is much quicker, since only initial few packets are flooded in task 3. Once the destination MAC address is found in the map, the switches will resend the packet only to the corresponding port that is mapped to in the “mac_to_port" mapping. And hence the subsequent pings are much faster as there will not be much network congestions.

3) a) mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['30.24 Mbits/sec', '36.11 Mbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['3.58 Mbits/sec', '4.73 Mbits/sec']

b) The throughput for task 3 is larger than that for task 2 in both the cases. This is due to lesser network congestion as in task 3, as there will not be flooding of packets after mac_to_port map has learnt all the ports and the switches will not be burdened much.
We can see in h1 and h2, task 1 and 2 the throughput had approximately 3 times the avg improvement in throughputs, given the routes are more pre-computed and learnt with changes in controller. While in case of h1 and h8 there is not major improvement, but had slight improvement due to the number of hops and dropping packet.