Task 1

Questions:

1. What is the output of “nodes” and “net”
   Nodes:

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

   Net:

```
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
c0
```

2. What is the 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
      ether 6a:dd:40:fe:36:8e  txqueuelen 1000  (Ethernet)
           RX packets 220  bytes 20440 (20.4 KB)
           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

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
            inet 127.0.0.1  netmask 255.0.0.0
            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

Questions:

1. Draw the function call graph of this controller. For example, once a packet comes to the controller, which function is the first to be called, which one is the second, and so forth?
   `_handle_PacketIn()` → `act_like_hub()` → `resend_packet()` → `self.connection.send()`

2. Have h1 ping h2, and h1 ping h8 for 100 times (e.g., h1 ping -c100 p2).
   a. How long does it take (on average) to ping for each case?
      h1 & h2: 7.521 ms
      h1 & h8: 20.477 ms

   b. What is the minimum and maximum ping you have observed?
      h1 & h2, Min : 2.243 ms, Max : 29.842 ms
      h1 & h8, Min : 10.594 ms, Max : 53.347 ms

   c. What is the difference, and why?
      h1 to h2 is faster than h1 to h8. There is a straightforward explanation for this. h1 and h2 have the same parent switch which is s3. For h1 to h8 we have to traverse through several switches s3, s2, s1, s5, s7. Hence this makes sense.

3. Run “iperf h1 h2” and “iperf h1 h8”
   a. What is “iperf” used for?
      Iperf is a commonly used network testing tool that measures the maximum achievable bandwidth between two devices over a network. It can be used to test

both TCP and UDP protocols and can generate a large amount of network traffic to test network performance.

b. What is the throughput for each case?
h1-h2: 10.6 Mbits/sec and 12.4 Mbits/sec
h1-h8: 3.03 Mbits/sec and 3.52 Mbits/sec

c. What is the difference, and explain the reasons for the difference.
The throughput of the first case is more than the second case, I think it's because of the same reason as the previous question, h1 pings to h8 has to go through more switches, so it will cause more packets to be dropped.

4. Which of the switches observe traffic? Please describe your way for observing such traffic on switches (e.g., adding some functions in the "of_tutorial" controller).
Traffic is observed in all the switches. We can add log statements in the _handle_packetIn() method to keep track of each packet.

Task 3

Questions:

1. Describe how the above code works, such as how the "MAC to Port" map is established. You could use a 'ping' example to describe the establishment process (e.g., h1 ping h2).
The above code consists of two primary components. Initially, it verifies whether the source MAC address is present in the "MAC to Port" map. In the event of a new source MAC, the code stores the source MAC and its corresponding port as a key-value pair in the "MAC to Port" map. This process involves learning the port for the source MAC. The code then proceeds to examine whether the destination is recognized. In the case of recognition, it transmits the packet only to the target port. Alternatively, it sends the packet to all the ports. For example, if we execute "h1 ping h2," and h1 is unrecognized, the code learns the port of h1 from the packet. It then checks whether h2 is recognized. If it is, the code transmits the packet directly to h2. If not, it forwards the packet to all ports.

2. (Comment out all prints before doing this experiment) Have h1 ping h2, and h1 ping h8 for 100 times (e.g., h1 ping -c100 p2).
    a. How long did it take (on average) to ping for each case?
    h1 to h2: 4.234 ms
    h1 to h8: 20.343 ms

    b. What is the minimum and maximum ping you have observed?
    h1 to h2, Min : 2.325, Max : 18.457ms
    h1 to h8, Min : 11.453 ms, Max : 56.347 ms

c. Any difference from Task 2 and why do you think there is a change if there is?
For h1 to h2 the performance improved noticeably but not that much for h1 to h8. Ping measures how fast data travels between two devices, and when devices are located far apart like h1 and h8, switching from a hub to a switch infrastructure may not make a big difference in terms of speed.

3. Q.3 Run “iperf h1 h2” and “iperf h1 h8”.
    a. What is the throughput for each case?
    h1-h2: 5.7 Mbits/sec and 6.9 Mbits/sec
    h1-h8: 10.5 Mbits/sec and 12.43 Mbits/sec

    b. What is the difference from Task 2 and why do you think there is a change if there is?
    It's logical to expect a significant change in throughput between Task 2 and the previous implementation with act_like_hub. In the previous implementation, the switch would send the packet to all ports, hoping it would reach its intended destination. However, this resulted in wasteful traffic and consumed bandwidth unnecessarily. In contrast, once the mappings are established, switches only need to forward to a specific port, reducing the competition for network resources. Consequently, there is a substantial improvement in throughput between both pairs of hosts.