Brenda Wang
COEN 241 Cloud Computing W23
13 March 2023
HW3

Task 1

1) What is the output of “nodes” and “net”?

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

> “nodes” lists all the controllers, hosts, and switches for the topology
> “net” lists all the links between the nodes

2) What is the output of “h7 ifconfig”?

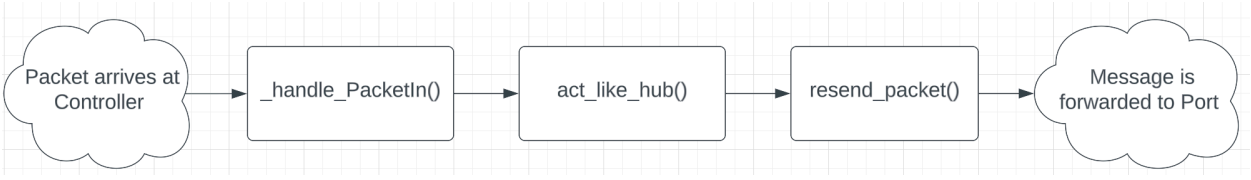
```
mininet> h7 ifconfig
h7-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.7  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::e0da:29ff:feb3:da29  prefixlen 64  scopeid 0x20<link>
        ether e2:da:29:b3:da:29  txqueuelen 1000  (Ethernet)
        RX packets 32  bytes 2432 (2.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9  bytes 726 (726.0 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
```

> ifconfig checks the IP of a virtual host

Task 2

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(event)                 // event = packet arrives at controller
act_like_hub (packet, packet_in)        // define out_port for switch's forwarding behavior
resend_packet (packet_in, out_port)     // message forwarded to out_port
```

2) Have h1 ping h2, and h1 ping h8 for 100 times (e.g., h1 ping -c100 p2).

```
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99344ms
rtt min/avg/max/mdev = 10.186/25.829/77.350/9.496 ms
```

```
--- 10.0.0.8 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99301ms
rtt min/avg/max/mdev = 51.860/95.974/148.119/14.766 ms
```

How long does it take (on average) to ping for each case?

h1 ping h2 -> 25.829 ms
h1 ping h8 -> 95.974 ms

What is the minimum and maximum ping you have observed?

h1 ping h2	min -> 10.186	max -> 77.350
h1 ping h8	min -> 51.860	max -> 148.119

What is the difference, and why?

The ping between h1 and h2 is faster on average than the ping between h1 and h8. The distance (number of links to travel) between them is shorter: h1-s3-h2 vs. h1-s3-s2-s1-s5-s7-h8, so the message travels faster.

3) Run “iperf h1 h2” and “iperf h1 h8”

What is “iperf” used for?

According the `mininet>help iperf`, it is a simple TCP test between 2 hosts. This can be used to evaluate network performance.

What is the throughput for each case?

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['653 Kbits/sec', '1.03 Mbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['472 Kbits/sec', '838 Kbits/sec']
```

What is the difference, and explain the reasons for the difference?

As stated previously, there are more switches between h1 and h8 versus between h1 and h2, so the throughput, or transfer rate between sender and receiver, is not as high since the packet must be received and sent out by each switch, limiting the total packets received in the same amount of time.

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).

- All the switches observe traffic.
- After adding `print(“%s – %s” % (self.connection, packets_in.total_len)` to the `act_like_hub()` function, the controller console prints the same the number of packets received for every switch (s1-s7) during both `h1 ping h2` and `h1 ping h8`.
- This behavior occurs because the switches are set to “act like a hub” and flood the packets received to any and every other switch, regardless of the destination.

```
def act_like_hub (self, packet, packet_in):
  """
  Implement hub-like behavior -- send all packets to all ports besides
  the input port.
  """

  # We want to output to all ports -- we do that using the special
  # OFPP_ALL port as the output port.  (We could have also used
  # OFPP_FLOOD.)
  self.resend_packet(packet_in, of.OFPP_ALL)

  # Note that if we didn't get a valid buffer_id, a slightly better
  # implementation would check that we got the full data before
  # sending it (len(packet_in.data) should be == packet_in.total_len)).
  print("%s – %s" % (self.connection, packet_in.total_len))
```

```
[00-00-00-00-00-03 5] – 98
[00-00-00-00-00-03 5] – 98
[00-00-00-00-00-02 6] – 98
[00-00-00-00-00-02 6] – 98
[00-00-00-00-00-04 2] – 98
[00-00-00-00-00-04 2] – 98
[00-00-00-00-00-01 3] – 98
[00-00-00-00-00-01 3] – 98
[00-00-00-00-00-05 7] – 98
[00-00-00-00-00-05 7] – 98
[00-00-00-00-00-07 1] – 98
[00-00-00-00-00-07 1] – 98
[00-00-00-00-00-06 4] – 98
[00-00-00-00-00-06 4] – 98
[00-00-00-00-00-03 5] – 98
[00-00-00-00-00-03 5] – 98
```

Task 3

```
def act_like_switch (self, packet, packet_in):
  # Learn the port for the source MAC
  # print("Src: ",str(packet.src),":", packet_in.in_port,"Dst:", str(packet.dst))
  if packet.src not in self.mac_to_port:
    print("Learning that " + str(packet.src) + " is attached at port " +
str(packet_in.in_port))
    self.mac_to_port[packet.src] = packet_in.in_port

  # if the port associated with the destination MAC of the packet is known:
  if packet.dst in self.mac_to_port:
    # Send packet out the associated port
    print(str(packet.dst) + " destination known. only send message to it")
    self.resend_packet(packet_in, self.mac_to_port[packet.dst])
  else:
    # Flood the packet out everything but the input port
    # This part looks familiar, right?
    print(str(packet.dst) + " not known, resend to everybody")
    self.resend_packet(packet_in, of.OFPP_ALL)
```

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 purpose of the function is to allow the switch to learn where to send packets by mapping a packets’ MAC addresses to their source or destination ports so that if the switch can know specifically where to route a packet to and decrease network congestion.

If the destination is unknown, the switch will flood the packet to every other switch, which gives them the opportunity to learn and map MAC to port.

For example, for h1 ping h2:
- h1 will send packets to s3
- s3 will map packet’s MAC to h1 and since the destination is unknown, flood it to s2 and h2
- h2 will respond to s3, which will then map the destination port
- when s3 receives another packet, it will check self.mac_to_port and find that the packet’s destination is h2 and send it only to h2 and not s2

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).

```
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99371ms
rtt min/avg/max/mdev = 22.820/38.836/75.131/7.844 ms
```

```
--- 10.0.0.8 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99352ms
rtt min/avg/max/mdev = 56.391/82.444/156.315/14.669 ms
```

How long did it take (on average) to ping for each case?

h1 ping h2 -> 38.836 ms
h1 ping h8 -> 82.444 ms

What is the minimum and maximum ping you have observed?

h1 ping h2 min -> 22.820 max -> 75.131
h1 ping h8 min -> 56.391 max -> 156.315

Any difference from Task 2 and why do you think there is a change if there is?

The ping between h1 and h8 is on average slightly faster here when the switches invoke act_like_switch(). They learn the mapping of packets to ports so that packets take the shortest route to their destination.

3) Run “iperf h1 h2” and “iperf h1 h8”.

What is the throughput for each case?

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['4.11 Mbits/sec', '4.75 Mbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['527 Kbits/sec', '944 Kbits/sec']
```

What is the difference from Task 2 and why do you think there is a change if there is?

The throughput between h1 and h2 is slower and the throughput between h1 and h8 is slightly faster when act_like_switch() is invoked compared to the two iperf tests when act_like_hub() is invoked. Again, the switches have learned exactly where to forward packets to so more can be sent and received in the same timeframe.