Part C:

1. **Pinging from h1 to h5**:

```
INFO:host_tracker:Learned 3 1 00:00:00:00:00:01
INFO:host_tracker:Learned 3 1 00:00:00:00:00:01 got IP 10.0.0.1
INFO:host_tracker:Learned 6 1 00:00:00:00:00:05
INFO:host_tracker:Learned 6 1 00:00:00:00:00:05 got IP 10.0.0.5
DEBUG:forwarding.l2_learning:installing flow for 00:00:00:00:00:05.1 -> 00:00:00:00:00:01.3
DEBUG:forwarding.l2_learning:installing flow for 00:00:00:00:00:05.1 -> 00:00:00:00:00:01.3
DEBUG:forwarding.l2_learning:installing flow for 00:00:00:00:00:05.2 -> 00:00:00:00:00:01.1
DEBUG:forwarding.l2_learning:installing flow for 00:00:00:00:00:05.3 -> 00:00:00:00:00:01.1
DEBUG:forwarding.l2_learning:installing flow for 00:00:00:00:00:05.3 -> 00:00:00:00:00:01.1
DEBUG:forwarding.l2_learning:installing flow for 00:00:00:00:00:01.1 -> 00:00:00:00:00:05.3
DEBUG:forwarding.l2_learning:installing flow for 00:00:00:00:00:01.1 -> 00:00:00:00:00:05.3
DEBUG:forwarding.l2_learning:installing flow for 00:00:00:00:00:01.1 -> 00:00:00:00:00:05.2
DEBUG:forwarding.l2_learning:installing flow for 00:00:00:00:00:01.3 -> 00:00:00:00:00:05.1
DEBUG:forwarding.l2_learning:installing flow for 00:00:00:00:00:01.3 -> 00:00:00:00:00:05.1
POX>
```

When we ping h5 from h1, packets from h1 travel through the switches s3, s2, s1, s5 and s6 in order.

The controller first gets the MAC and IP addresses of h1 and h5 in the following outputs:

host_tracker:Learned 3 1 00:00:00:00:00:01

host_tracker:Learned 3 1 00:00:00:00:00:01 got IP 10.0.0.1

host_tracker:Learned 6 1 00:00:00:00:00:05

host_tracker:Learned 6 1 00:00:00:00:00:05 got IP 10.0.0.5

Then the controller installs only flows necessary for the packet to travel from h1 to h5 via s3 -> s2 -> s1 -> s5 -> s6 and vice versa.

The forwarding.l2_learning logs only display source and destination matching MAC addresses and the input and output ports. There are 5 entries for flows required starting from h1 and 5 entries for flows required for travelling in the reverse direction for all the 5 switches in the path.

2. **RTT for first 5 pings**:

```
mininet@mininet-vm: ~/cs456-a3
File  Edit  View  Search  Terminal  Help
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet> h1 ping h5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=13.3 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=0.042 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=0.049 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=0.044 ms
64 bytes from 10.0.0.5: icmp_seq=5 ttl=64 time=0.052 ms
```

The first ping takes more time as it needs to send out an additional ARP request to figure out the MAC address of the destination host in the network, which is then stored in its ARP cache. Hence, the subsequent pings do not take a lot of time since there is no additional latency introduced due to ARP cache miss and the host already has the required MAC address.

3. **Flow table comparisons**:

```
mininet@mininet-vm:~/cs456-a3$ sudo ovs-ofctl dump-flows s1
 cookie=0x0, duration=28.254s, table=0, n_packets=14, n_bytes=574, priority=6500
0,dl_dst=01:23:20:00:00:01,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=28.214s, table=0, n_packets=0, n_bytes=0, priority=32769,a
rp,dl_dst=02:00:00:00:be:ef actions=CONTROLLER:65535
```

Figure 1: s1 before ping

```
mininet@mininet-vm:~/cs456-a3$ sudo ovs-ofctl dump-flows s3
 cookie=0x0, duration=90.618s, table=0, n_packets=19, n_bytes=779, priority=6500
0,dl_dst=01:23:20:00:00:01,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=90.578s, table=0, n_packets=0, n_bytes=0, priority=32769,a
rp,dl_dst=02:00:00:00:be:ef actions=CONTROLLER:65535
mininet@mininet-vm:~/cs456-a3$
```

Figure 2: s3 before ping

All the flow tables for all switches before pinging contained two entries and both pointed towards the controller. This is the initial state of flow tables and shows that whenever the first packet arrives, it would be sent to the controller whose destination MAC address is stated in the flow table.

```
mininet@mininet-vm:~/cs456-a3$ sudo ovs-ofctl dump-flows s1
 cookie=0x0, duration=13.152s, table=0, n_packets=8, n_bytes=672, hard_timeout=1
800, priority=65001,dl_src=00:00:00:00:00:05,dl_dst=00:00:00:00:00:01 actions=ou
tput:"s1-eth1"
 cookie=0x0, duration=13.141s, table=0, n_packets=7, n_bytes=630, hard_timeout=1
800, priority=65001,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:05 actions=ou
tput:"s1-eth2"
 cookie=0x0, duration=292.147s, table=0, n_packets=119, n_bytes=4879, priority=6
5000,dl_dst=01:23:20:00:00:01,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=292.107s, table=0, n_packets=0, n_bytes=0, priority=32769,
arp,dl_dst=02:00:00:00:be:ef actions=CONTROLLER:65535
```

Figure 3: s1 after ping

After the ping, flow tables for all switches that the packet had passed through (namely, s1, s2, s3, s5, s6) were installed with required new entries for communication between hosts h1 and h5.

The match criteria in these flows is only dl_src (source MAC address) and dl_dst (destination MAC address) which shows that all the flows follow destination-based forwarding. In part-A, we had implemented a more general match plus action rule by matching over multiple fields and multiple actions were taken such as modifying source and destination MAC addresses along with changing the output port.

```
mininet@mininet-vm:~/cs456-a3$ sudo ovs-ofctl dump-flows s2
 cookie=0x0, duration=39.125s, table=0, n_packets=8, n_bytes=672, hard_timeout=1
800, priority=65001,dl_src=00:00:00:00:00:05,dl_dst=00:00:00:00:00:01 actions=ou
tput:"s2-eth1"
 cookie=0x0, duration=39.116s, table=0, n_packets=7, n_bytes=630, hard_timeout=1
800, priority=65001,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:05 actions=ou
tput:"s2-eth3"
 cookie=0x0, duration=318.121s, table=0, n_packets=193, n_bytes=7913, priority=6
5000,dl_dst=01:23:20:00:00:01,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=318.081s, table=0, n_packets=0, n_bytes=0, priority=32769,
arp,dl_dst=02:00:00:00:be:ef actions=CONTROLLER:65535
```

Figure 4: s2 after ping

```
mininet@mininet-vm:~/cs456-a3$ sudo ovs-ofctl dump-flows s3
 cookie=0x0, duration=56.926s, table=0, n_packets=8, n_bytes=672, hard_timeout=1
800, priority=65001,dl_src=00:00:00:00:00:05,dl_dst=00:00:00:00:00:01 actions=ou
tput:"s3-eth1"
 cookie=0x0, duration=56.918s, table=0, n_packets=7, n_bytes=630, hard_timeout=1
800, priority=65001,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:05 actions=ou
tput:"s3-eth3"
 cookie=0x0, duration=335.923s, table=0, n_packets=68, n_bytes=2788, priority=65
000,dl_dst=01:23:20:00:00:01,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=335.883s, table=0, n_packets=0, n_bytes=0, priority=32769,
arp,dl_dst=02:00:00:00:be:ef actions=CONTROLLER:65535
```

Figure 5: s3 after ping

```
mininet@mininet-vm:~/cs456-a3$ sudo ovs-ofctl dump-flows s5
 cookie=0x0, duration=105.249s, table=0, n_packets=8, n_bytes=672, hard_timeout=
1800, priority=65001,dl_src=00:00:00:00:00:05,dl_dst=00:00:00:00:00:01 actions=o
utput:"s5-eth3"
 cookie=0x0, duration=105.238s, table=0, n_packets=7, n_bytes=630, hard_timeout=
1800, priority=65001,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:05 actions=o
utput:"s5-eth1"
 cookie=0x0, duration=384.244s, table=0, n_packets=230, n_bytes=9430, priority=6
5000,dl_dst=01:23:20:00:00:01,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=384.204s, table=0, n_packets=0, n_bytes=0, priority=32769,
arp,dl_dst=02:00:00:00:be:ef actions=CONTROLLER:65535
```

Figure 6: s5 after ping

```
mininet@mininet-vm:~/cs456-a3$ sudo ovs-ofctl dump-flows s6
 cookie=0x0, duration=120.908s, table=0, n_packets=8, n_bytes=672, hard_timeout=
1800, priority=65001,dl_src=00:00:00:00:00:05,dl_dst=00:00:00:00:00:01 actions=o
utput:"s6-eth3"
 cookie=0x0, duration=120.895s, table=0, n_packets=7, n_bytes=630, hard_timeout=
1800, priority=65001,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:05 actions=o
utput:"s6-eth1"
 cookie=0x0, duration=399.902s, table=0, n_packets=81, n_bytes=3321, priority=65
000,dl_dst=01:23:20:00:00:01,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=399.862s, table=0, n_packets=0, n_bytes=0, priority=32769,
arp,dl_dst=02:00:00:00:be:ef actions=CONTROLLER:65535
```

Figure 7: s6 after ping

After the ping, switch 4 and 7 flow tables still do not have any new entries as the packet never had to pass through these switches when h5 was pinged from h1. So, no new entries have been installed by the controller.

```
mininet@mininet-vm:~/cs456-a3$ sudo ovs-ofctl dump-flows s4
 cookie=0x0, duration=359.748s, table=0, n_packets=73, n_bytes=2993, priority=65
000,dl_dst=01:23:20:00:00:01,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=359.708s, table=0, n_packets=0, n_bytes=0, priority=32769,
arp,dl_dst=02:00:00:00:be:ef actions=CONTROLLER:65535
```

Figure 8: s4 after ping

```
mininet@mininet-vm:~/cs456-a3$ sudo ovs-ofctl dump-flows s7
 cookie=0x0, duration=420.036s, table=0, n_packets=85, n_bytes=3485, priority=65
000,dl_dst=01:23:20:00:00:01,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=419.997s, table=0, n_packets=0, n_bytes=0, priority=32769,
arp,dl_dst=02:00:00:00:be:ef actions=CONTROLLER:65535
```

Figure 9: s7 after ping