Lab1 Report

Johnson Le

03/31/19

CMPE 150/L

Pre-lab Questions

1) “Groups”
2) Holds the status of the most recent command. 0 indicates means success.
   Source:
   https://stackoverflow.com/questions/12741710/what-does-mean-in-a-shell-script
3) Ctrl+Z to pause. “fg” to continue.
4) “uname -n && uname -v”
   Or “uname -n -v” to see it all on one line.
   Source:
   https://www.cyberciti.biz/faq/howto-find-out-what-kernel-version-running/
5) “.” = current path
   “..” = parent path
   “~” = home path
   “/” = root path.
6) PID short for process identifier. Its some ID number that is used to identify a process that is running. To find the PID of a process, I would personally just view all processes like Window’s task manager. The ones I know of are “top” and good old “htop”.
7) Pretty sure there are plenty of ways to do it. I would do it:
   “cat /etc/passwd”
   This will read and print out what is in “passwd” file.
8) “su” is for switching user. Therefore, “su root” is for switching to root user. This would allow for commands that require root permissions to be available. “sudo <command>” will run the command with root permissions.
   Source:
   https://www.howtogeek.com/111479/htg-explains-whats-the-difference-between-sudo-su/
9) “watch -n x <command>” where x is number of seconds between executions. I tested this by pinging google every 10 seconds and values ping times changed.
   Source: http://www.linfo.org/watch.html
10) Refer to “johnsonle-script.sh” file.

The Lab

Part 1

I just modified the provided example script by adding 3 more hosts and linking them all to the same switch.

Part 2

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=1918>
<Host h2: h2-eth0:10.0.0.2 pid=1922>
<Host h3: h3-eth0:10.0.0.3 pid=1924>
<Host h4: h4-eth0:10.0.0.4 pid=1926>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None,s1-eth4:None
pid=1931>
<Controller c0: 127.0.0.1:6633 pid=1911>
```

Figure 1: Result for pingall and dump

“Pingall” will attempt to ping all available hosts. It then displays all hosts and the hosts they are connected to next to it. “dump” displays IP address and PID of each host and switch.

Part 3

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['4.77 Gbits/sec', '4.78 Gbits/sec']
```

Figure 2: Result of iperf

The bandwidth is 4.77 to 4.87 Gigabits per second between h1 and h4.

Part 4

a) A total of 5 “of_packet_in” messages show up

	Time	Source	Destination	Protocol	Length	Info
8	5.373571000	10.0.0.1	10.0.0.2	OF 1.0	184	of_packet_in
9	5.375160000	127.0.0.1	127.0.0.1	OF 1.0	92	of_packet_out
16	5.375969000	10.0.0.2	10.0.0.1	OF 1.0	184	of_packet_in
17	5.377304000	127.0.0.1	127.0.0.1	OF 1.0	148	of_flow_add
21	6.375166000	10.0.0.1	10.0.0.2	OF 1.0	184	of_packet_in
22	6.376455000	127.0.0.1	127.0.0.1	OF 1.0	148	of_flow_add
40	10.39125200(	06:dd:26:6c:a2:06	56:70:f8:b9:14:73	OF 1.0	128	of_packet_in
41	10.39310200(	127.0.0.1	127.0.0.1	OF 1.0	148	of_flow_add
45	10.39428500(	56:70:f8:b9:14:73	06:dd:26:6c:a2:06	OF 1.0	128	of_packet_in
46	10.39549200(	127.0.0.1	127.0.0.1	OF 1.0	148	of_flow_add

Figure 3: Result of h1 pinging h2 5 times

b) h1’s IP address is 10.0.0.1 and h2’s is 10.0.0.2. The switch or controller address is 127.0.0.1. The source is from h1 with destination h2.

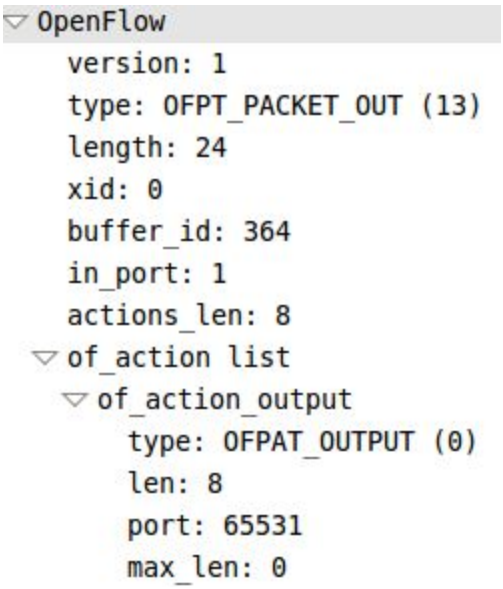

Figure 4: OpenFlow from of_packet_out

c) The result of pingall using the “icmp && not of” filter has 30 entries

generated. This displays the ip address of source and destination and what the source is doing such as requesting or replying. Results are shown in Figure 5.

	Time	Source	Destination	Protocol	Length	Info
7	5.373140000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x08c6, seq=1/256, ttl=64
11	5.375478000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x08c6, seq=1/256, ttl=64
12	5.375488000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x08c6, seq=1/256, ttl=64
13	5.375494000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x08c6, seq=1/256, ttl=64
14	5.375499000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x08c6, seq=1/256, ttl=64 (reply in 15)
15	5.375535000	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x08c6, seq=1/256, ttl=64 (request in 14)
18	5.377496000	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x08c6, seq=1/256, ttl=64
20	6.374737000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x08c6, seq=2/512, ttl=64
24	6.376705000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x08c6, seq=2/512, ttl=64 (reply in 25)
25	6.376732000	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x08c6, seq=2/512, ttl=64 (request in 24)
26	6.377230000	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x08c6, seq=2/512, ttl=64
27	7.376454000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x08c6, seq=3/768, ttl=64
28	7.376640000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x08c6, seq=3/768, ttl=64 (reply in 29)
29	7.376665000	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x08c6, seq=3/768, ttl=64 (request in 28)
30	7.376671000	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x08c6, seq=3/768, ttl=64
31	8.378078000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x08c6, seq=4/1024, ttl=64
32	8.378099000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x08c6, seq=4/1024, ttl=64 (reply in 33)
33	8.378120000	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x08c6, seq=4/1024, ttl=64 (request in 32)
34	8.378125000	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x08c6, seq=4/1024, ttl=64
35	9.382917000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x08c6, seq=5/1280, ttl=64
36	9.382939000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x08c6, seq=5/1280, ttl=64 (reply in 37)
37	9.382959000	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x08c6, seq=5/1280, ttl=64 (request in 36)
38	9.382964000	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x08c6, seq=5/1280, ttl=64

Figure 5: Result of “icmp && not of”