Lab 1

Pre-Lab Questions

1. What command will show you which groups you are a member of?
    a. Using the command “groups” will show you which groups you are a member of.

2. What does the environmental variable “$?” hold? (Hint: the command ‘echo $?’ will show you this on your screen)
    a. The environment variable “$?” holds the exit code of the previous command.

3. What key combination will suspend a currently running process and place it as a background process?
    a. In order to suspend a currently running process and place it as a background process you can press “Ctrl-z”, which will suspend the background process then using the “bg” command to send it to the background process.

4. With what command (and arguments) can you find out your kernel version and the “nodename”? [The output should not include any other information]
    a. Using the command “uname -v -n” will show you the kernel version and the “nodename”. The “-v” will show you the operating system version and the “-n” will show you the “nodename”.

5. What is the difference between the paths “.”, “..”, and “~”? What does the path “/” refer to when not preceded by anything?
    a. Using the “.” refers to the current directory when used in a command or a file path. Using “..” refers to the parent directory in the directory hierarchy. Using “~” refers to the home directory of the current user. When using “/” and it is not preceded by anything it refers to the root directory of the file system.

6. What is a pid? Which command would you use to find the “pid” for a running process?
    a. A pid is a Process identifier, which is a unique number that identifies every process running in the kernel. To find the “pid” for a running process, you would use “pidof <name of process>”.

7. Write a single command that will return every user’s default shell. [You may chain commands using piping and redirects] (Hint: See ‘Chapter 19: filters’ of linux-training.be as well as the man page for the /etc/passwd file: https://linux.die.net/man/5/passwd)

   a. Using the single command “getent passwd $LOGNAME | cut -d: f1,7” will return the default shell of every user.

8. What is the difference between “sudo” and “su root”?
   a. The main difference between “sudo” and “su root” is that using “su root” lets you switch users to the root user and lets you perform administrative tasks. In “sudo”, it does the same thing except it just runs the commands you give it, and you are limited to execute specific commands with elevated privileges.

9. How would you tell your computer to run a program or script on a schedule or set interval on Linux? E.g. Run this program once every 30 minutes.
   a. To run a program or script on a schedule or set interval on Linux, you can use the “crontab -e”, and then adding the following line “*/30 * * * * <path/to/script>”, and then saving it and it will run the program every 30 mins, everyday.

10. Write a shell script that only prints the even numbered lines of each file in the current directory. The output should be filename: line for each even numbered line. You do not need to print line numbers.

    dev_chodavadiya-script.sh

Resources:

https://serverfault.com/questions/409698/how-to-send-running-process-to-background (3)

https://linux-training.be/linuxfun.pdf (1,5)

https://www.ibm.com/docs/en/aix/7.2?topic=u-uname-command#uname__row-d3e145402 (4)

https://www.geeksforgeeks.org/get-process-id-of-linux-foreground-and-background-processes/ (6)

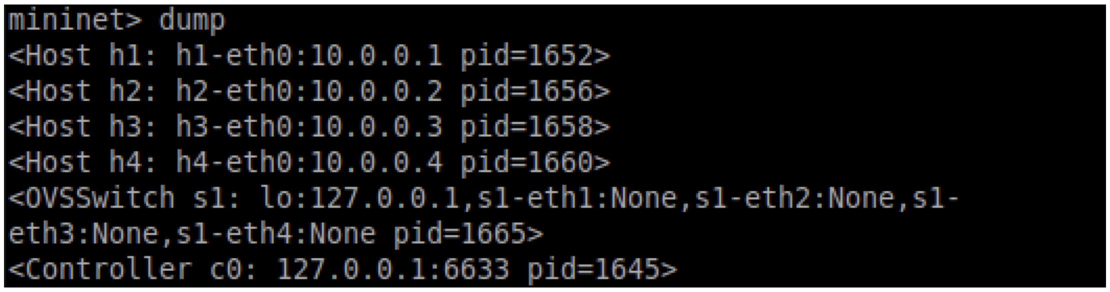
https://superuser.com/questions/294929/how-to-get-an-application-to-run-every-30-minutes#:~:text=Use%20cron%20to%20run%20it%20periodically.&text=Then%20save%20the%20crontab%2C%20which,and%20email%20you%20any%20output.(9)

https://man7.org/linux/man-pages/man1/getent.1.html (7)

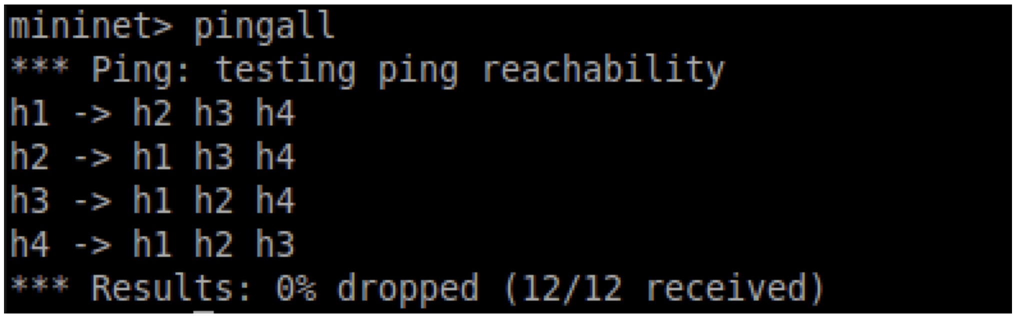
Lab 1 Questions

2. Save a screenshot of dump and pingall output. Explain what is being shown in the screenshot.

Dump: In this screenshot, it shows all four of the hosts, which are all connected to one switch. Host h1 has an IP address of 10.0.0.1 and has the process ID (pid) of 1652. Host h2 has an IP address of 10.0.0.2 and has the process ID (pid) of 1656. Host h3 has an IP address of 10.0.0.3 and has the process ID (pid) of 1658. Host h4 has an IP address of 10.0.0.4 and has the process ID (pid) of 1660. All the hosts are connected to switch s1. The switch has four ports, s1-eth1, s1-eth2, s1-eth3, and s1-eth4. The switch has an IP address of 127.0.0.1 and a process ID of 1665. It also has a controller c0, with an IP address and port number of 127.0.0.1:6633 and a process ID of 1645.

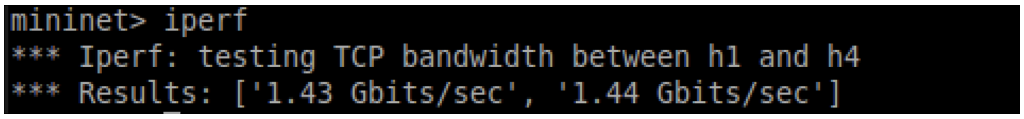

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=1652>
<Host h2: h2-eth0:10.0.0.2 pid=1656>
<Host h3: h3-eth0:10.0.0.3 pid=1658>
<Host h4: h4-eth0:10.0.0.4 pid=1660>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-
eth3:None,s1-eth4:None pid=1665>
<Controller c0: 127.0.0.1:6633 pid=1645>
```

Pingall: In this screenshot, it tests the connectivity between all the hosts within the network. The output shows that each host in the network successfully pinged the other hosts. For example host 1, h1, pinged h2, h3, and h4. Similarly host 2, h2, pinged h1, h3, h4, and so on for the rest of the hosts. In the results it shows, all 12/12 hosts were pinged successfully, no packets were lost.


```
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)
```

3. Run the iperf command as well, and screenshot the output, how fast is the connect?

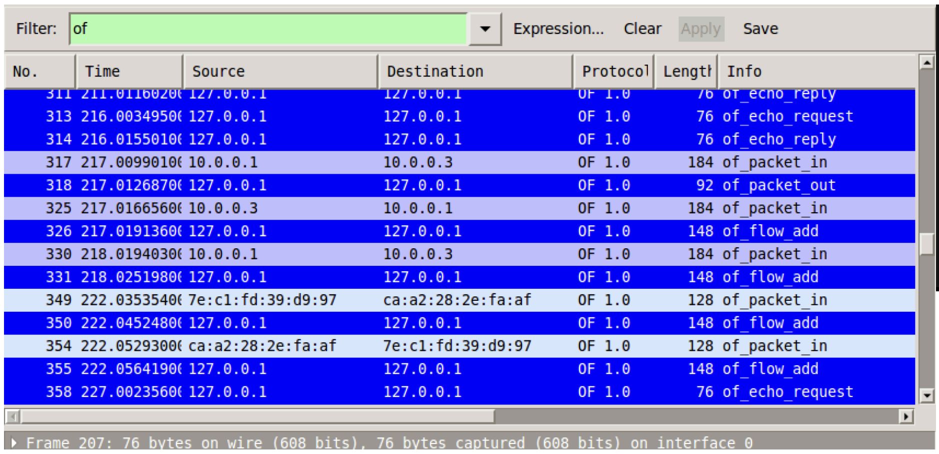
Running the "iperf" command, shows the connection between host 1 and host 4, as well as how strong the connection is between the two hosts. The connection speed is around 1.43 Gbits/sec for one direction and 1.44 Gbits/sec for the other direction.



```
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['1.43 Gbits/sec', '1.44 Gbits/sec']
```

4a. Run ping from a host to any other host using hX ping -c 5 hY. How many of_packet_in messages show up? Take a screenshot of your results.

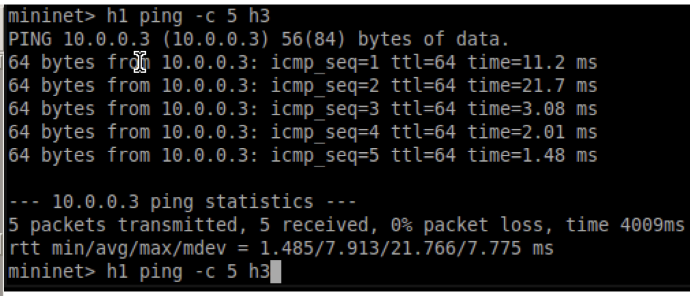
5 of_packet_in messages show up in wireshark when running "h1 ping -c 5 h3".



Filter: of

No.	Time	Source	Destination	Protocol	Length	Info
311	211.0116020(	127.0.0.1	127.0.0.1	OF 1.0	76	of_echo_reply
313	216.0034950(	127.0.0.1	127.0.0.1	OF 1.0	76	of_echo_request
314	216.0155010(	127.0.0.1	127.0.0.1	OF 1.0	76	of_echo_reply
317	217.0099010(	10.0.0.1	10.0.0.3	OF 1.0	184	of_packet_in
318	217.0126870(	127.0.0.1	127.0.0.1	OF 1.0	92	of_packet_out
325	217.0166560(	10.0.0.3	10.0.0.1	OF 1.0	184	of_packet_in
326	217.0191360(	127.0.0.1	127.0.0.1	OF 1.0	148	of_flow_add
330	218.0194030(	10.0.0.1	10.0.0.3	OF 1.0	184	of_packet_in
331	218.0251980(	127.0.0.1	127.0.0.1	OF 1.0	148	of_flow_add
349	222.0353540(	7e:c1:fd:39:d9:97	ca:a2:28:2e:fa:af	OF 1.0	128	of_packet_in
350	222.0452480(	127.0.0.1	127.0.0.1	OF 1.0	148	of_flow_add
354	222.0529300(	ca:a2:28:2e:fa:af	7e:c1:fd:39:d9:97	OF 1.0	128	of_packet_in
355	222.0564190(	127.0.0.1	127.0.0.1	OF 1.0	148	of_flow_add
358	227.0023560(	127.0.0.1	127.0.0.1	OF 1.0	76	of_echo_request

Frame 207: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface 0

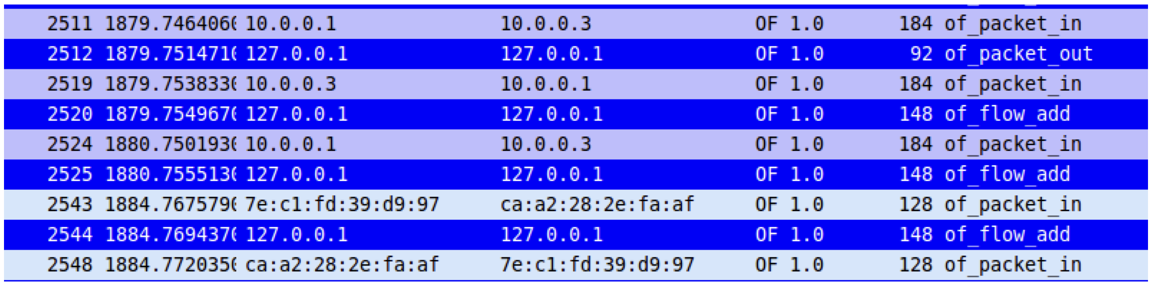


```
mininet> h1 ping -c 5 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=11.2 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=21.7 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=3.08 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=2.01 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=1.48 ms

--- 10.0.0.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4009ms
rtt min/avg/max/mdev = 1.485/7.913/21.766/7.775 ms
mininet> h1 ping -c 5 h3
```

4b. What is the source and destination IP addresses for these entries? Find another packet that matches the "of" filter with the OpenFlow typefield set to OF_PACKET_OUT. What is the source and destination IP address for this entry? Take screenshots showing your results.

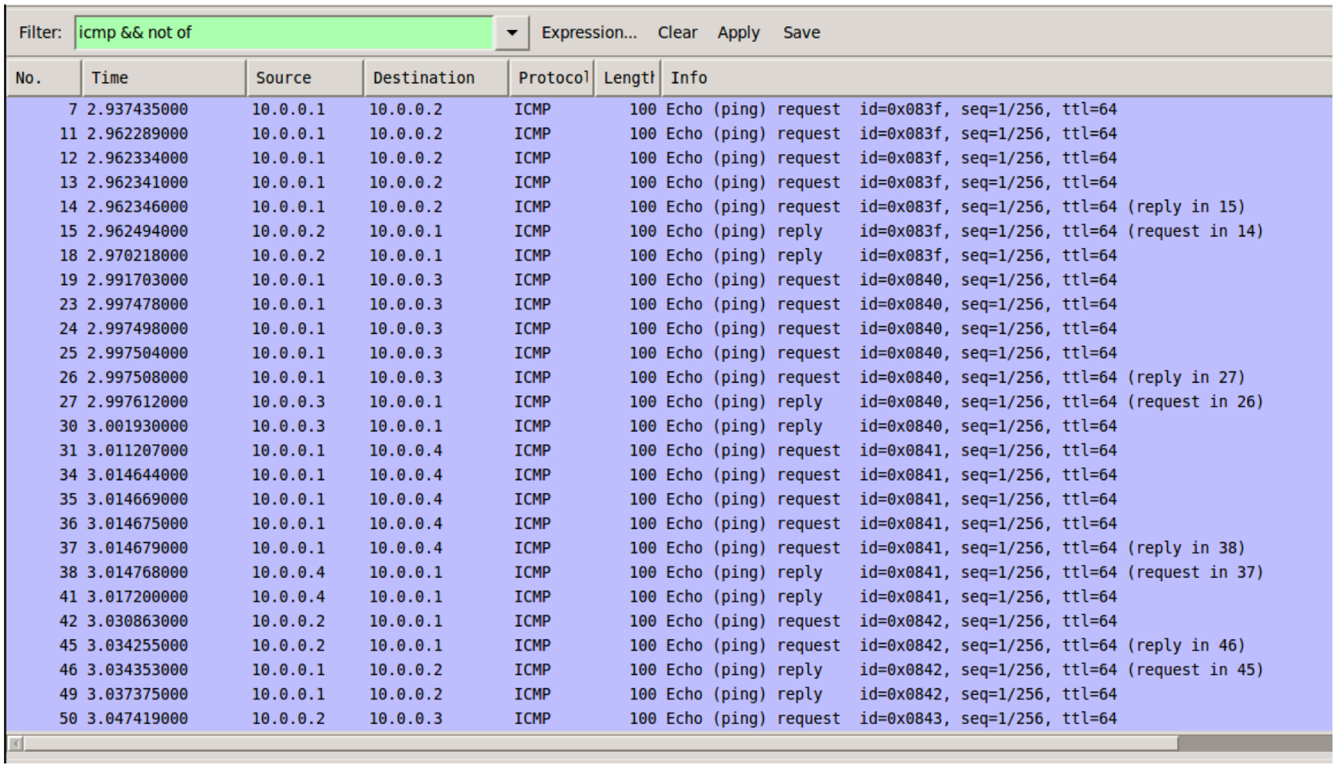
The source IP address when running "h1 ping -c 5 h3" was 10.0.0.1 and the destination was 10.0.0.3. Since I was pinging host 3 from host 1, it makes sense that the source was the IP address of host 1 and the destination address was the IP address of host 3. The source for the OF_PACKET_OUT is 12.0.0.1 and the destination is also 12.0.0.1. The source and destination are the same because icmp is pinging itself so it's going out from the same source and back to the same destination.



```
2511 1879.746406( 10.0.0.1            10.0.0.3            OF 1.0   184 of_packet_in
2512 1879.751471( 127.0.0.1           127.0.0.1           OF 1.0    92 of_packet_out
2519 1879.753833( 10.0.0.3            10.0.0.1            OF 1.0   184 of_packet_in
2520 1879.754967( 127.0.0.1           127.0.0.1           OF 1.0   148 of_flow_add
2524 1880.750193( 10.0.0.1            10.0.0.3            OF 1.0   184 of_packet_in
2525 1880.755513( 127.0.0.1           127.0.0.1           OF 1.0   148 of_flow_add
2543 1884.767579( 7e:c1:fd:39:d9:97   ca:a2:28:2e:fa:af   OF 1.0   128 of_packet_in
2544 1884.769437( 127.0.0.1           127.0.0.1           OF 1.0   148 of_flow_add
2548 1884.772035( ca:a2:28:2e:fa:af   7e:c1:fd:39:d9:97   OF 1.0   128 of_packet_in
```

4C. Replace the display filter for "of" to "icmp && not of". Run pingall again, how many entries are generated in wireshark? What types of icmp entries show up? Take a screenshot of your results.

After changing the filter to "icmp && not of", and running pingall again, a little over 100 entries are generated in wireshark. All the entries that show up in icmp are either "Echo (ping) request" or "Echo (ping) reply". Going more into detail, looking at the first line, it shows that our source's IP address is 10.0.0.1 which indicates the packet is coming from host 1. The destination of that packet is 10.0.0.2, which means that the packet is going to host 2. But notice that its also requesting for the packet, so there has to be a reply later. As we see a couple lines down, there is a reply from which the source is 10.0.0.2 to which the destination is 10.0.0.1. If you look closely it does that for every host till it pings all of them. It goes back and forth between each host till it's pinged all of the host.



```
Filter: icmp && not of                Expression...  Clear  Apply  Save

No.   Time          Source     Destination  Protocol Length Info
    7 2.937435000   10.0.0.1   10.0.0.2     ICMP      100 Echo (ping) request  id=0x083f, seq=1/256, ttl=64
   11 2.962289000   10.0.0.1   10.0.0.2     ICMP      100 Echo (ping) request  id=0x083f, seq=1/256, ttl=64
   12 2.962334000   10.0.0.1   10.0.0.2     ICMP      100 Echo (ping) request  id=0x083f, seq=1/256, ttl=64
   13 2.962341000   10.0.0.1   10.0.0.2     ICMP      100 Echo (ping) request  id=0x083f, seq=1/256, ttl=64
   14 2.962346000   10.0.0.1   10.0.0.2     ICMP      100 Echo (ping) request  id=0x083f, seq=1/256, ttl=64 (reply in 15)
   15 2.962494000   10.0.0.2   10.0.0.1     ICMP      100 Echo (ping) reply    id=0x083f, seq=1/256, ttl=64 (request in 14)
   18 2.970218000   10.0.0.2   10.0.0.1     ICMP      100 Echo (ping) reply    id=0x083f, seq=1/256, ttl=64
   19 2.991703000   10.0.0.1   10.0.0.3     ICMP      100 Echo (ping) request  id=0x0840, seq=1/256, ttl=64
   23 2.997478000   10.0.0.1   10.0.0.3     ICMP      100 Echo (ping) request  id=0x0840, seq=1/256, ttl=64
   24 2.997498000   10.0.0.1   10.0.0.3     ICMP      100 Echo (ping) request  id=0x0840, seq=1/256, ttl=64
   25 2.997504000   10.0.0.1   10.0.0.3     ICMP      100 Echo (ping) request  id=0x0840, seq=1/256, ttl=64
   26 2.997508000   10.0.0.1   10.0.0.3     ICMP      100 Echo (ping) request  id=0x0840, seq=1/256, ttl=64 (reply in 27)
   27 2.997612000   10.0.0.3   10.0.0.1     ICMP      100 Echo (ping) reply    id=0x0840, seq=1/256, ttl=64 (request in 26)
   30 3.001930000   10.0.0.3   10.0.0.1     ICMP      100 Echo (ping) reply    id=0x0840, seq=1/256, ttl=64
   31 3.011207000   10.0.0.1   10.0.0.4     ICMP      100 Echo (ping) request  id=0x0841, seq=1/256, ttl=64
   34 3.014644000   10.0.0.1   10.0.0.4     ICMP      100 Echo (ping) request  id=0x0841, seq=1/256, ttl=64
   35 3.014669000   10.0.0.1   10.0.0.4     ICMP      100 Echo (ping) request  id=0x0841, seq=1/256, ttl=64
   36 3.014675000   10.0.0.1   10.0.0.4     ICMP      100 Echo (ping) request  id=0x0841, seq=1/256, ttl=64
   37 3.014679000   10.0.0.1   10.0.0.4     ICMP      100 Echo (ping) request  id=0x0841, seq=1/256, ttl=64 (reply in 38)
   38 3.014768000   10.0.0.4   10.0.0.1     ICMP      100 Echo (ping) reply    id=0x0841, seq=1/256, ttl=64 (request in 37)
   41 3.017200000   10.0.0.4   10.0.0.1     ICMP      100 Echo (ping) reply    id=0x0841, seq=1/256, ttl=64
   42 3.030863000   10.0.0.2   10.0.0.1     ICMP      100 Echo (ping) request  id=0x0842, seq=1/256, ttl=64
   45 3.034255000   10.0.0.2   10.0.0.1     ICMP      100 Echo (ping) request  id=0x0842, seq=1/256, ttl=64 (reply in 46)
   46 3.034353000   10.0.0.1   10.0.0.2     ICMP      100 Echo (ping) reply    id=0x0842, seq=1/256, ttl=64 (request in 45)
   49 3.037375000   10.0.0.1   10.0.0.2     ICMP      100 Echo (ping) reply    id=0x0842, seq=1/256, ttl=64
   50 3.047419000   10.0.0.2   10.0.0.3     ICMP      100 Echo (ping) request  id=0x0843, seq=1/256, ttl=64
```