Edbel Basaldua
Ebasaldu@ucsc.edu
ID: 1413712
CE150: Lab 1
January 29,2018

Pre lab questions:

1. What command will show you which groups you are a member of?

   Answer: Using the command 'groups' on the cli will show the user what groups you are a member of

   Source: https://www.cyberciti.biz/faq/linux-show-groups-for-user/

2. What does the environmental variable "$?" hold? (Hint: the command 'echo $?' will should you this on your screen)

   Answer: The environmental variable holds the value zero and displays '0'

   Source: Just used the cli and typed in the echo command

3. What key combination will suspend a currently running process and place it as a background process?

   Answer: In order to suspend a current process the Ctrl and z keys should be pressed. To place a process in the background the command 'bg' followed by the job id should be typed into the CLI.

   Source: https://unix.stackexchange.com/questions/45025/how-to-suspend-and-bring-a-background-process-to-foreground

4. With what command (and arguments) can you find out your kernel version and the "nodename"? [The output should not include any other information]

   Answer: In order to find out what kernel version is being used you can use the 'uname' command specifically for the kernel version the full command entered into the cli would be 'uname –v'. In order to get the node name or the host name the same command would be used except the instead of the v the command would be 'uname –n'.

   Source: https://www.liquidweb.com/kb/how-to-check-the-kernel-version-in-linux-ubuntu-centos/

5. What is the difference between the paths ".", "..", and "~"? What does the path "/" refer to when not preceded by anything?

   Answer: The path command "." Is representative of the current directory path. The path command ".." represent the parent directory of the current directory. The path command denoted by "~" is used to represent the home directory. Lastly the path command "/" is used to id the root directories to specify a certain path to take.

   Source: https://www.cs.jhu.edu/~joanne/unix.html

6. What is a pid? Which command would you use to find the "pid" for a running process?

   Answer: PID is an acronym for process identification number, which is given to all instances of processes. The command to find the pid of a job would be 'ps'. The combination of the command ps aux| grep –i application name will give the pid of the running processes.

   Source: https://www.cyberciti.biz/faq/howto-display-process-pid-under-linux-unix/

7. Write a single command that will return every user's default shell. [You may chain commands using piping and redirects]

   Answer: Using the command echo $SHELL will return the users default shell.

   Source: https://unix.stackexchange.com/questions/43499/difference-between-echo-shell-and-which-bash

8. What is the difference between "sudo" and "su root"?

   Answer: The difference between these two commands is the following. The sudo command will run a single command and give root privileges and will not switch to the root user or it wont prompt the user for another root password. On the other hand the su command will allow for the user to switch to the root user account and prompts for the respective root password.

   Source: https://www.howtogeek.com/111479/htg-explains-whats-the-difference-between-sudo-su/

Edbel Basaldua
Ebasaldu@ucsc.edu
ID: 1413712
CE150: Lab 1
January 29,2018

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.

Answer: In order to automate a program or script the crontab tool should be used it can be started by using the command 'crontab –e ' within an editor use the following format:

'minute(0-59) hour(0-23) day(1-31) month(1-12) weekday(0-6) <command>'

This format will allow for the time specifications of when a certain program or script should be executed.

For the example it may be something like:

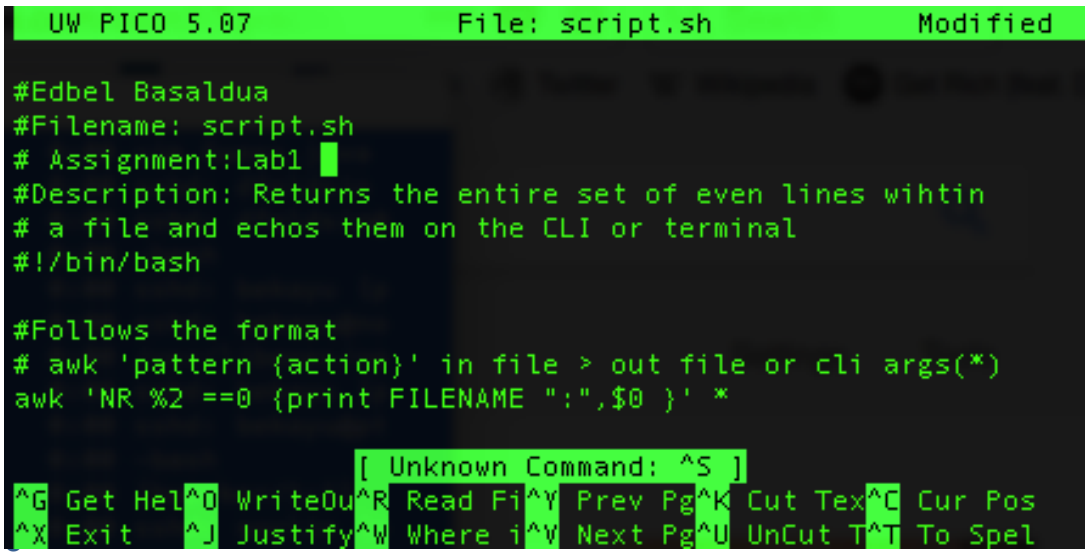
'29 * * * * <Desired program to be executed>'

Source: https://www.howtogeek.com/101288/how-to-schedule-tasks-on-linux-an-introduction-to-crontab-files/

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.

Note: The script for this program is also included in the online submission

Answer: The following is a screenshot of the file submitted



Source: I used the following documentation to write my code
https://www.gnu.org/software/gawk/manual/html_node/Auto_002dset.html#Auto_002dset

https://www.lifewire.com/write-awk-commands-and-scripts-2200573

Edbel Basaldua
Ebasaldu@ucsc.edu
ID: 1413712
CE150: Lab 1
January 29,2018

Lab 1:

1. **Change the default configuration to have 4 hosts connected to 1 switch**

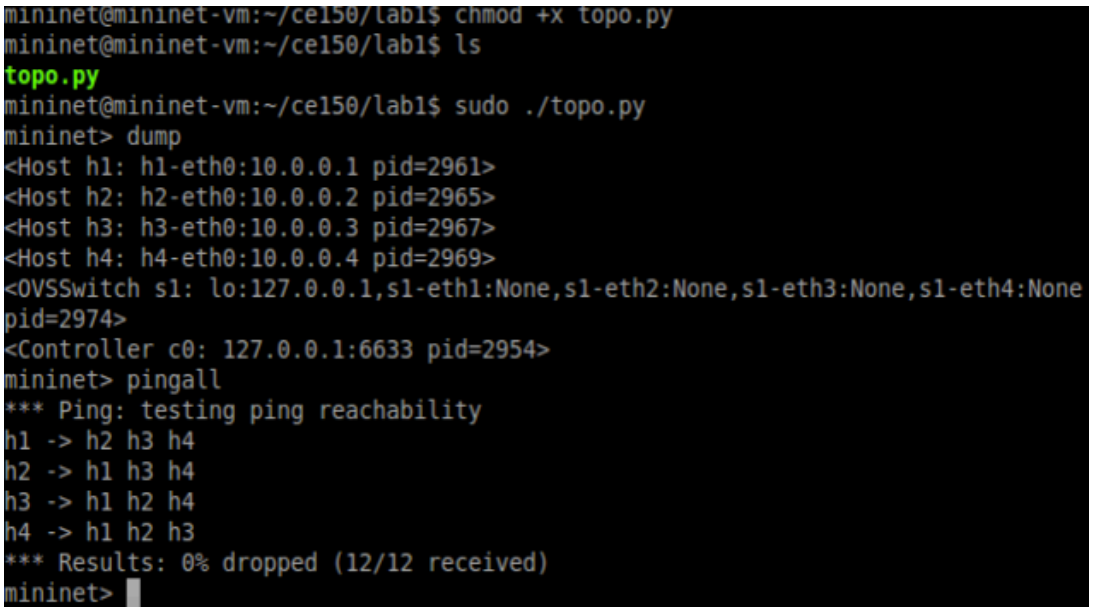
   Modifying the code in the starter code provided does this. The modification includes the addition of the following lines:

   Originally the topology is only two hosts and one switch. But with the addition of lines: host lines 16,17 and link lines 22,23. This network now becomes 4 hosts and 1 switch.

```
11      # Set Up Topology Here
12      switch = self.addSwitch('s1') ## Adds a Switch
13      host1 = self.addHost('h1') ## Adds a Host
14      host2 = self.addHost('h2') ## Adds a Host
15      ## The following lines are what was added to modify the base
16      host3 = self.addHost('h3') ## Adds new host h3
17      host4 = self.addHost('h4') ## Adds new host h4
18      ## add host modification ends here
19      self.addLink(host1, switch) ## Add a link
20      self.addLink(host2, switch) ## Add a link
21      ## Adds the respective links between the new hosts and switch
22      self.addLink(host3, switch) ## Adds link to the new host3
23      self.addLink(host4, switch) ## Add link to the new host4
24      ## Link addition ends here
```

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

After the running the executable for the topology we get the new modified network with 4 hosts and 1 switch.
After entering the dump and pingall commands the following output is given in the terminal window.



The **dump** command seems to spew out the IP and the process ID's of each host and switch. It also displays the respective ports of each host.

The **pingall** command test the connectivity between all hosts and does so through the use of connection requests. It also displays the percentage of packets dropped and gives the ratio of transferred packets to that of the total number of packets.

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

This command test the bandwidth between each host

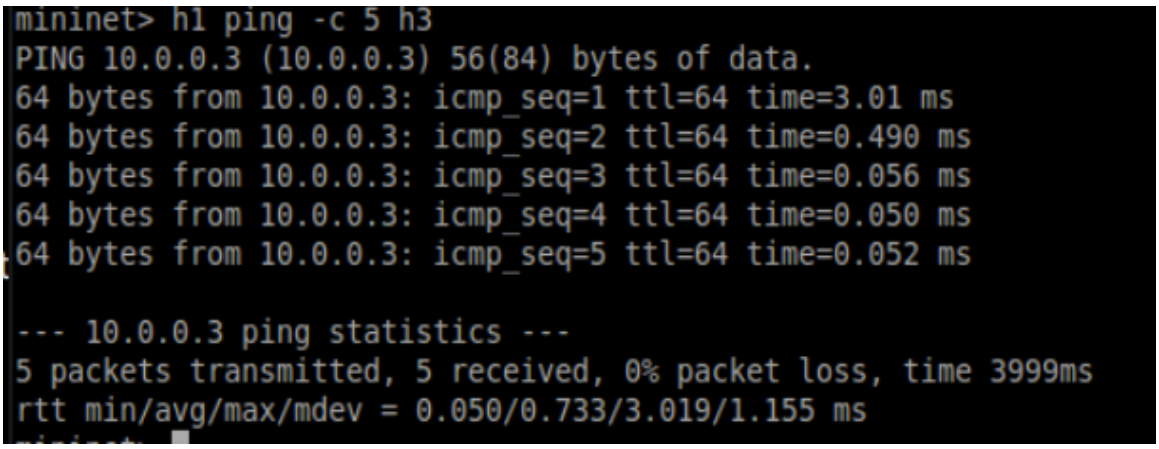
According to the output above the connect speed is **11.6Gbits/sec**

Edbel Basaldua
Ebasaldu@ucsc.edu
ID: 1413712
CE150: Lab 1
January 29,2018

4. Run wireshark, and using the display filter, filter for “of”. Note: When you run wireshark you should do so as “sudo wireshark”. When you choose an interface to capture on, you should select “any”.

    a. 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.

1. For this part of the assignment I chose to ping from host1 to host 3. The following is the output that I got from the terminal. It seems to show the 5 ping requests and the respective packets sent. It also provides some statistics of the round trip time. Also it shows the percentage of packets lost and the number of packets sent and received. From the wireshark out put it says that there were 184 or 28 *of_packet_in* and it showed 5 of them.



The wire shark output looks like:

```
821 1256.962228( 10.0.0.1      10.0.0.3      ICMP  100 Echo (ping) request  id=0x
822 1256.962261( 10.0.0.3      10.0.0.1      ICMP  100 Echo (ping) reply    id=0x
823 1256.962461( 10.0.0.3      10.0.0.1      ICMP  100 Echo (ping) reply    id=0x
824 1257.962647( 10.0.0.1      10.0.0.3      ICMP  100 Echo (ping) request  id=0x
825 1257.962823( 10.0.0.1      10.0.0.3      ICMP  100 Echo (ping) request  id=0x
826 1257.962885( 10.0.0.3      10.0.0.1      ICMP  100 Echo (ping) reply    id=0x
827 1257.962889( 10.0.0.3      10.0.0.1      ICMP  100 Echo (ping) reply    id=0x
828 1258.962327( 10.0.0.1      10.0.0.3      ICMP  100 Echo (ping) request  id=0x
829 1258.962339( 10.0.0.1      10.0.0.3      ICMP  100 Echo (ping) request  id=0x
830 1258.962351( 10.0.0.3      10.0.0.1      ICMP  100 Echo (ping) reply    id=0x
831 1258.962354( 10.0.0.3      10.0.0.1      ICMP  100 Echo (ping) reply    id=0x
832 1259.962306( 10.0.0.1      10.0.0.3      ICMP  100 Echo (ping) request  id=0x
833 1259.962318( 10.0.0.1      10.0.0.3      ICMP  100 Echo (ping) request  id=0x
834 1259.962330( 10.0.0.3      10.0.0.1      ICMP  100 Echo (ping) reply    id=0x
835 1259.962332( 10.0.0.3      10.0.0.1      ICMP  100 Echo (ping) reply    id=0x
```

This one just has a line that was left out:

```
820 1256.962158( 127.0.0.1     127.0.0.1     TCP    68 44189 > 6633 [ACK] Seq=283
821 1256.962228( 10.0.0.1      10.0.0.3      ICMP  100 Echo (ping) request  id=0x
822 1256.962261( 10.0.0.3      10.0.0.1      ICMP  100 Echo (ping) reply    id=0x
823 1256.962461( 10.0.0.3      10.0.0.1      ICMP  100 Echo (ping) reply    id=0x
824 1257.962647( 10.0.0.1      10.0.0.3      ICMP  100 Echo (ping) request  id=0x
825 1257.962823( 10.0.0.1      10.0.0.3      ICMP  100 Echo (ping) request  id=0x
826 1257.962885( 10.0.0.3      10.0.0.1      ICMP  100 Echo (ping) reply    id=0x
827 1257.962889( 10.0.0.3      10.0.0.1      ICMP  100 Echo (ping) reply    id=0x
828 1258.962327( 10.0.0.1      10.0.0.3      ICMP  100 Echo (ping) request  id=0x
829 1258.962339( 10.0.0.1      10.0.0.3      ICMP  100 Echo (ping) request  id=0x
830 1258.962351( 10.0.0.3      10.0.0.1      ICMP  100 Echo (ping) reply    id=0x
831 1258.962354( 10.0.0.3      10.0.0.1      ICMP  100 Echo (ping) reply    id=0x
832 1259.962306( 10.0.0.1      10.0.0.3      ICMP  100 Echo (ping) request  id=0x
833 1259.962318( 10.0.0.1      10.0.0.3      ICMP  100 Echo (ping) request  id=0x
834 1259.962330( 10.0.0.3      10.0.0.1      ICMP  100 Echo (ping) reply    id=0x
```

With the Of filter:

Here there are 5 of_packet_in messages

```
991 1505.000134( 127.0.0.1     127.0.0.1     OF 1.0   76 of_echo_reply
994 1506.753489( 10.0.0.1      10.0.0.3      OF 1.0  184 of_packet_in
995 1506.754460( 127.0.0.1     127.0.0.1     OF 1.0   92 of_packet_out
1002 1506.754740( 10.0.0.3     10.0.0.1      OF 1.0  184 of_packet_in
1003 1506.755075( 127.0.0.1    127.0.0.1     OF 1.0  148 of_flow_add
1007 1507.754259( 10.0.0.1     10.0.0.3      OF 1.0  184 of_packet_in
1008 1507.754677( 127.0.0.1    127.0.0.1     OF 1.0  148 of_flow_add
1026 1511.758275( ca:5c:6e:e7:4a:36  de:1d:e3:b5:49:e9  OF 1.0  128 of_packet_in
1027 1511.758732( 127.0.0.1    127.0.0.1     OF 1.0  148 of_flow_add
1031 1511.759081( de:1d:e3:b5:49:e9  ca:5c:6e:e7:4a:36  OF 1.0  128 of_packet_in
```

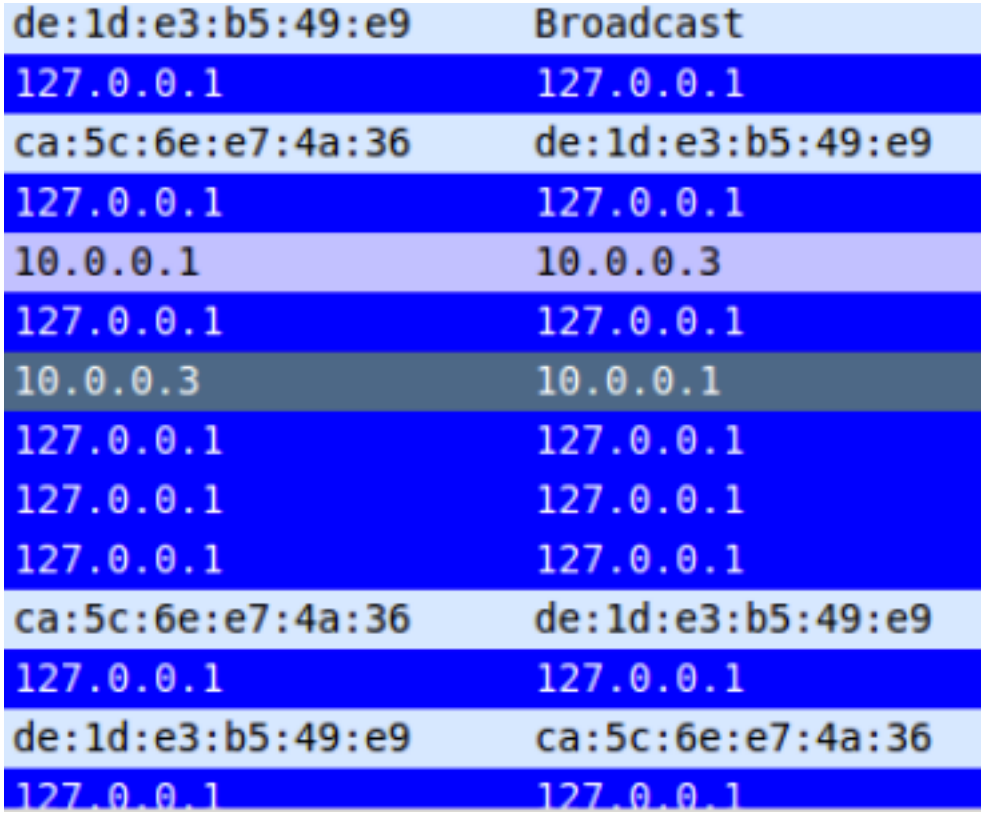
Edbel Basaldua
Ebasaldu@ucsc.edu
ID: 1413712
CE150: Lab 1
January 29,2018

311	440.6394660(	de:1d:e3:b5:49:e9	ca:5c:6e:e7:4a:36	OF 1.0	128 of_packet_in
312	440.6397970(	127.0.0.1	127.0.0.1	OF 1.0	148 of_flow_add
315	444.9997480(	127.0.0.1	127.0.0.1	OF 1.0	76 of_echo_request
316	445.0004710(	127.0.0.1	127.0.0.1	OF 1.0	76 of_echo_reply
318	449.9996690(	127.0.0.1	127.0.0.1	OF 1.0	76 of_echo_request
319	450.0007340(	127.0.0.1	127.0.0.1	OF 1.0	76 of_echo_reply
321	454.9999640(	127.0.0.1	127.0.0.1	OF 1.0	76 of_echo_request
322	455.0008110(	127.0.0.1	127.0.0.1	OF 1.0	76 of_echo_reply
324	459.9995890(	127.0.0.1	127.0.0.1	OF 1.0	76 of_echo_request
325	460.0002790(	127.0.0.1	127.0.0.1	OF 1.0	76 of_echo_reply
327	464.9989510(	127.0.0.1	127.0.0.1	OF 1.0	76 of_echo_request
328	464.9998000(	127.0.0.1	127.0.0.1	OF 1.0	76 of_echo_reply
330	469.9999400(	127.0.0.1	127.0.0.1	OF 1.0	76 of_echo_request
331	470.0018050(	127.0.0.1	127.0.0.1	OF 1.0	76 of_echo_reply
333	474.9997110(	127.0.0.1	127.0.0.1	OF 1.0	76 of_echo_request

Here there are 6 of_packet_in messages which include the broadcast one. (I didn't know if we were supposed to omit this or not)

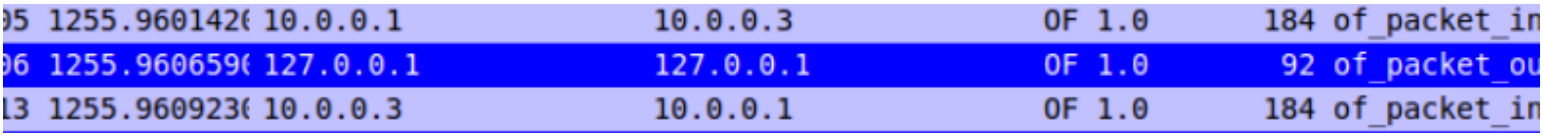
991	1505.000134(	127.0.0.1	127.0.0.1	OF 1.0	76 of_echo_reply
994	1506.753489(	10.0.0.1	10.0.0.3	OF 1.0	184 of_packet_in
995	1506.754460(	127.0.0.1	127.0.0.1	OF 1.0	92 of_packet_out
1002	1506.754740(	10.0.0.3	10.0.0.1	OF 1.0	184 of_packet_in
1003	1506.755075(	127.0.0.1	127.0.0.1	OF 1.0	148 of_flow_add
1007	1507.754259(	10.0.0.1	10.0.0.3	OF 1.0	184 of_packet_in
1008	1507.754677(	127.0.0.1	127.0.0.1	OF 1.0	148 of_flow_add
1026	1511.758275(	ca:5c:6e:e7:4a:36	de:1d:e3:b5:49:e9	OF 1.0	128 of_packet_in
1027	1511.758732(	127.0.0.1	127.0.0.1	OF 1.0	148 of_flow_add
1031	1511.759081(	de:1d:e3:b5:49:e9	ca:5c:6e:e7:4a:36	OF 1.0	128 of_packet_in

b. 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 *OFPT_PACKET_OUT*. What is the source and destination IP address for this entry? Take screenshots showing your results.



de:1d:e3:b5:49:e9	Broadcast
127.0.0.1	127.0.0.1
ca:5c:6e:e7:4a:36	de:1d:e3:b5:49:e9
127.0.0.1	127.0.0.1
10.0.0.1	10.0.0.3
127.0.0.1	127.0.0.1
10.0.0.3	10.0.0.1
127.0.0.1	127.0.0.1
127.0.0.1	127.0.0.1
127.0.0.1	127.0.0.1
ca:5c:6e:e7:4a:36	de:1d:e3:b5:49:e9
127.0.0.1	127.0.0.1
de:1d:e3:b5:49:e9	ca:5c:6e:e7:4a:36
127.0.0.1	127.0.0.1

The above image shows the respective source and destination IP addresses for these entries. This includes the lines: 1,3,5,7,11,12. Counting from the top down on this image.



95 1255.960142(	10.0.0.1	10.0.0.3	OF 1.0	184	of_packet_in
96 1255.960659(	127.0.0.1	127.0.0.1	OF 1.0	92	of_packet_ou
13 1255.960923(	10.0.0.3	10.0.0.1	OF 1.0	184	of_packet_in

The of_packet_out is highlighted in blue above it says that its 92. Its source and destination are both the same, which is: **127.0.0.1**

c. 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.

When using the filter and running **pingall** again there are 142 entries produced. These entries are ping reply and ping request . The following screenshot only shows a few of them but they go up to line 305.

Edbel Basaldua
Ebasaldu@ucsc.edu
ID: 1413712
CE150: Lab 1
January 29,2018

163	186.2984870(	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x1459, seq=1/25
166	186.2989950(	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x1459, seq=1/25
167	186.2990060(	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x1459, seq=1/25
170	186.2996540(	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x1459, seq=1/25
171	186.3020850(	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x145b, seq=1/25
174	186.3026030(	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x145b, seq=1/25
175	186.3026060(	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x145b, seq=1/25
176	186.3026070(	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x145b, seq=1/25
177	186.3026090(	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x145b, seq=1/25
178	186.3026190(	10.0.0.3	10.0.0.1	ICMP	100	Echo (ping) reply id=0x145b, seq=1/25
181	186.3032820(	10.0.0.3	10.0.0.1	ICMP	100	Echo (ping) reply id=0x145b, seq=1/25
193	186.3094500(	10.0.0.1	10.0.0.4	ICMP	100	Echo (ping) request id=0x145c, seq=1/25
196	186.3100820(	10.0.0.1	10.0.0.4	ICMP	100	Echo (ping) request id=0x145c, seq=1/25
197	186.3100970(	10.0.0.4	10.0.0.1	ICMP	100	Echo (ping) reply id=0x145c, seq=1/25
200	186.3107140(	10.0.0.4	10.0.0.1	ICMP	100	Echo (ping) reply id=0x145c, seq=1/25