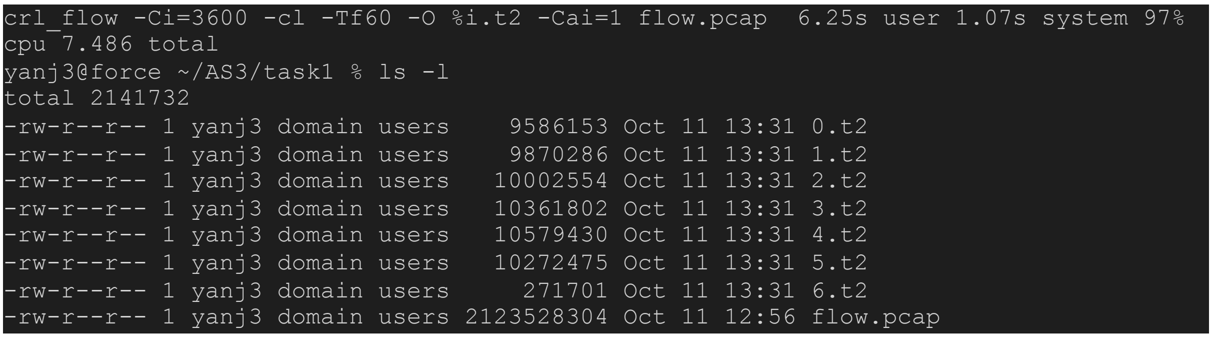
Report for the 3rd Assignment of ELEC-E7130

[redacted]

Report, task 1

First of all, convert the given sample pcap file into flows through CoralReef software. After executing the cmd, the flow information (*.t2 files) is generated from the flow.pcap file.

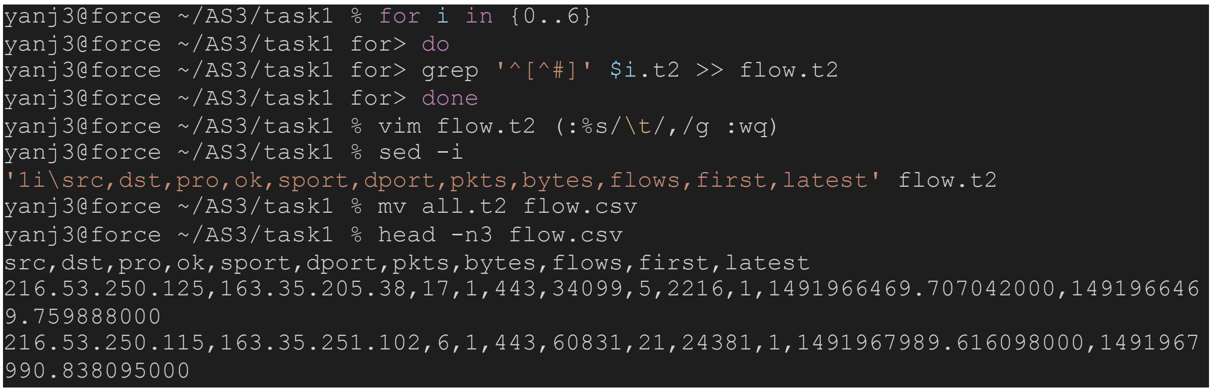
```
yanj3@force ~ % source /work/courses/unix/T/ELEC/E7130/general/use.sh
yanj3@force ~ % mkdir -p AS3/task1
yanj3@force ~ % cd AS3/task1
yanj3@force ~/AS3/task1 % cp
/work/courses/unix/T/ELEC/E7130/general/trace/capture/flow.pcap ./
yanj3@force ~/AS3/task1 % time crl_flow -Ci=3600 -cl -Tf60 -O %i.t2 -Cai=1
flow.pcap
```



```
crl_flow -Ci=3600 -cl -Tf60 -O %i.t2 -Cai=1 flow.pcap  6.25s user 1.07s system 97%
cpu 7.486 total
yanj3@force ~/AS3/task1 % ls -l
total 2141732
-rw-r--r-- 1 yanj3 domain users    9586153 Oct 11 13:31 0.t2
-rw-r--r-- 1 yanj3 domain users    9870286 Oct 11 13:31 1.t2
-rw-r--r-- 1 yanj3 domain users   10002554 Oct 11 13:31 2.t2
-rw-r--r-- 1 yanj3 domain users   10361802 Oct 11 13:31 3.t2
-rw-r--r-- 1 yanj3 domain users   10579430 Oct 11 13:31 4.t2
-rw-r--r-- 1 yanj3 domain users   10272475 Oct 11 13:31 5.t2
-rw-r--r-- 1 yanj3 domain users     271701 Oct 11 13:31 6.t2
-rw-r--r-- 1 yanj3 domain users 2123528304 Oct 11 12:56 flow.pcap
```

Merge all 7 files ended with ".t2" to just 1 ".t2" file while deleting all sentences with "#" started, and then transfer the ".t2" file to a csv file

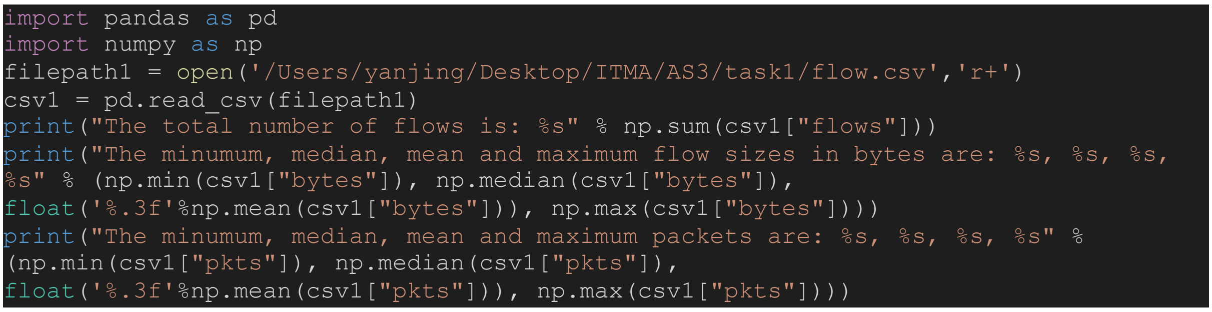
More detailed about "flow.csv" can be seen in AS3.zip file



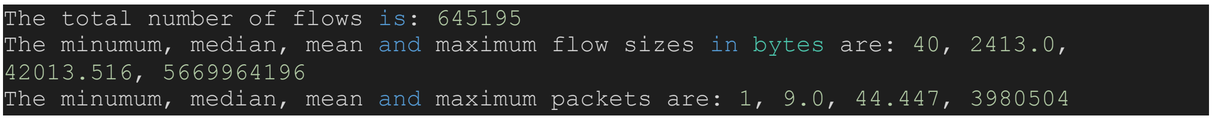
```
yanj3@force ~/AS3/task1 % for i in {0..6}
yanj3@force ~/AS3/task1 for> do
yanj3@force ~/AS3/task1 for> grep '^[^#]' $i.t2 >> flow.t2
yanj3@force ~/AS3/task1 for> done
yanj3@force ~/AS3/task1 % vim flow.t2 (:%s/\t/,/g :wq)
yanj3@force ~/AS3/task1 % sed -i
'1i\src,dst,pro,ok,sport,dport,pkts,bytes,flows,first,latest' flow.t2
yanj3@force ~/AS3/task1 % mv all.t2 flow.csv
yanj3@force ~/AS3/task1 % head -n3 flow.csv
src,dst,pro,ok,sport,dport,pkts,bytes,flows,first,latest
216.53.250.125,163.35.205.38,17,1,443,34099,5,2216,1,1491966469.707042000,149196646
9.759888000
216.53.250.115,163.35.251.102,6,1,443,60831,21,24381,1,1491967989.616098000,1491967
990.838095000
```

I Descriptive statistics

About total number of flows, - minimum, median, mean and maximum flow sizes in bytes and packets, the following are the code and result:



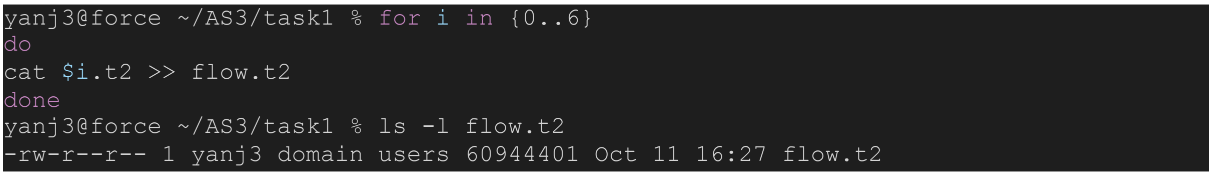
```
import pandas as pd
import numpy as np
filepath1 = open('/Users/yanjing/Desktop/ITMA/AS3/task1/flow.csv','r+')
csv1 = pd.read_csv(filepath1)
print("The total number of flows is: %s" % np.sum(csv1["flows"]))
print("The minumum, median, mean and maximum flow sizes in bytes are: %s, %s, %s,
%s" % (np.min(csv1["bytes"]), np.median(csv1["bytes"]),
float('%.3f'%np.mean(csv1["bytes"])), np.max(csv1["bytes"])))
print("The minumum, median, mean and maximum packets are: %s, %s, %s, %s" %
(np.min(csv1["pkts"]), np.median(csv1["pkts"]),
float('%.3f'%np.mean(csv1["pkts"])), np.max(csv1["pkts"])))
```



```
The total number of flows is: 645195
The minumum, median, mean and maximum flow sizes in bytes are: 40, 2413.0,
42013.516, 5669964196
The minumum, median, mean and maximum packets are: 1, 9.0, 44.447, 3980504
```

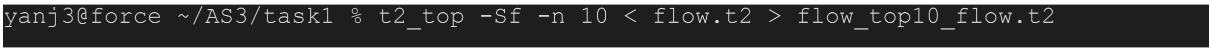
II Tables of top-ten host pairs

Merge all 7 files ended with ".t2" to just 1 ".t2" file and then use "t2_top" cmd to sort the data

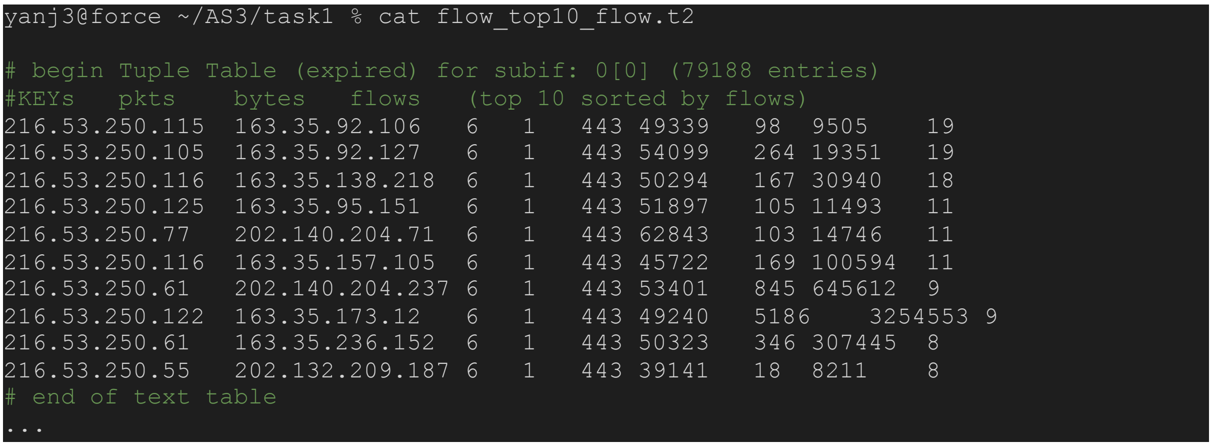


```
yanj3@force ~/AS3/task1 % for i in {0..6}
do
cat $i.t2 >> flow.t2
done
yanj3@force ~/AS3/task1 % ls -l flow.t2
-rw-r--r-- 1 yanj3 domain users 60944401 Oct 11 16:27 flow.t2
```

The top-ten host-pairs based on number of flows: using “t2_top” cmd and showing *part of the result, more detailed can be seen through the “flow_top10_flow.t2” file in AS3.zip file*

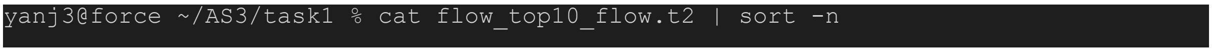


```
yanj3@force ~/AS3/task1 % t2_top -Sf -n 10 < flow.t2 > flow_top10_flow.t2
```

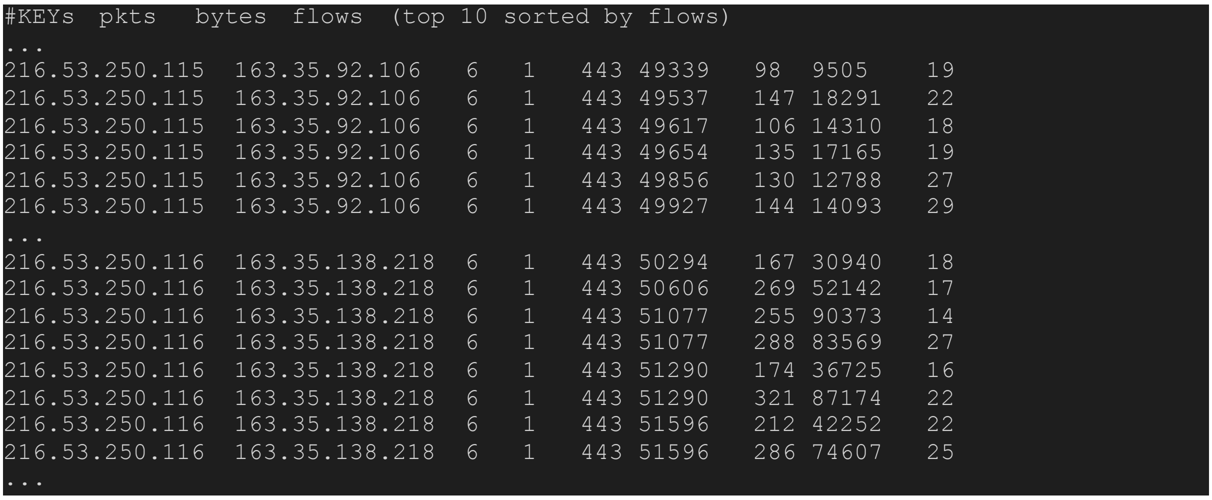


```
yanj3@force ~/AS3/task1 % cat flow_top10_flow.t2

# begin Tuple Table (expired) for subif: 0[0] (79188 entries)
#KEYs   pkts    bytes   flows   (top 10 sorted by flows)
216.53.250.115  163.35.92.106   6   1   443 49339   98  9505    19
216.53.250.105  163.35.92.127   6   1   443 54099   264 19351   19
216.53.250.116  163.35.138.218  6   1   443 50294   167 30940   18
216.53.250.125  163.35.95.151   6   1   443 51897   105 11493   11
216.53.250.77   202.140.204.71  6   1   443 62843   103 14746   11
216.53.250.116  163.35.157.105  6   1   443 45722   169 100594  11
216.53.250.61   202.140.204.237 6   1   443 53401   845 645612  9
216.53.250.122  163.35.173.12   6   1   443 49240   5186    3254553 9
216.53.250.61   163.35.236.152  6   1   443 50323   346 307445  8
216.53.250.55   202.132.209.187 6   1   443 39141   18  8211    8
# end of text table
...
```

By using the following cmd, I can see some records with the same host pairs in “flow_top10_flow.t2” file, and showing part of the results.

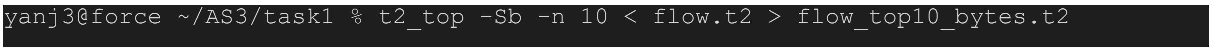


```
yanj3@force ~/AS3/task1 % cat flow_top10_flow.t2 | sort -n
```

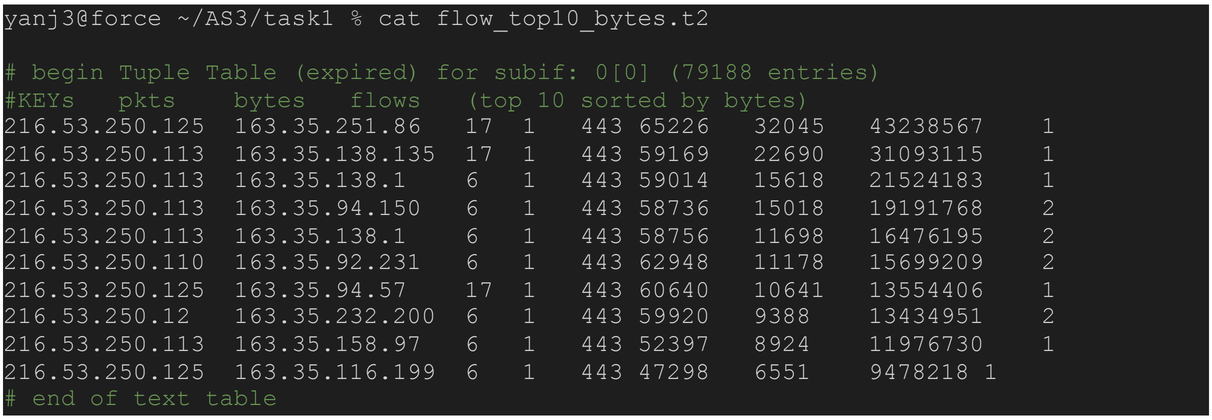


```
#KEYs  pkts   bytes  flows  (top 10 sorted by flows)
...
216.53.250.115  163.35.92.106   6   1   443 49339   98  9505    19
216.53.250.115  163.35.92.106   6   1   443 49537   147 18291   22
216.53.250.115  163.35.92.106   6   1   443 49617   106 14310   18
216.53.250.115  163.35.92.106   6   1   443 49654   135 17165   19
216.53.250.115  163.35.92.106   6   1   443 49856   130 12788   27
216.53.250.115  163.35.92.106   6   1   443 49927   144 14093   29
...
216.53.250.116  163.35.138.218  6   1   443 50294   167 30940   18
216.53.250.116  163.35.138.218  6   1   443 50606   269 52142   17
216.53.250.116  163.35.138.218  6   1   443 51077   255 90373   14
216.53.250.116  163.35.138.218  6   1   443 51077   288 83569   27
216.53.250.116  163.35.138.218  6   1   443 51290   174 36725   16
216.53.250.116  163.35.138.218  6   1   443 51290   321 87174   22
216.53.250.116  163.35.138.218  6   1   443 51596   212 42252   22
216.53.250.116  163.35.138.218  6   1   443 51596   286 74607   25
...
```

The top-ten host-pairs based on number of bytes: using “t2_top” cmd and showing *part of the result, more detailed can be seen through the “flow_top10_bytes.t2” file in AS3.zip file*



```
yanj3@force ~/AS3/task1 % t2_top -Sb -n 10 < flow.t2 > flow_top10_bytes.t2
```



```
yanj3@force ~/AS3/task1 % cat flow_top10_bytes.t2

# begin Tuple Table (expired) for subif: 0[0] (79188 entries)
#KEYs   pkts    bytes   flows   (top 10 sorted by bytes)
216.53.250.125  163.35.251.86   17  1   443 65226   32045   43238567    1
216.53.250.113  163.35.138.135  17  1   443 59169   22690   31093115    1
216.53.250.113  163.35.138.1    6   1   443 59014   15618   21524183    1
216.53.250.113  163.35.94.150   6   1   443 58736   15018   19191768    2
216.53.250.113  163.35.138.1    6   1   443 58756   11698   16476195    2
216.53.250.110  163.35.92.231   6   1   443 62948   11178   15699209    2
216.53.250.125  163.35.94.57    17  1   443 60640   10641   13554406    1
216.53.250.12   163.35.232.200  6   1   443 59920   9388    13434951    2
216.53.250.113  163.35.158.97   6   1   443 52397   8924    11976730    1
216.53.250.125  163.35.116.199  6   1   443 47298   6551    9478218 1
# end of text table
```

```
...
```

By using the following cmd, I can see some records with the same host pairs in "flow_top10_bytes.t2" file, and showing part of the results.

```
yanj3@force ~/AS3/task1 % cat flow_top10_bytes.t2 | sort -n

```

```
#KEYs  pkts   bytes  flows  (top 10 sorted by bytes)
...
216.53.250.125  163.35.137.89   17  1   443 51186   17122   22925812    1
216.53.250.125  163.35.137.89   17  1   443 52394   32171   43073612    1
216.53.250.125  163.35.137.89   17  1   443 54004   39508   52476639    1
216.53.250.125  163.35.137.89   17  1   443 56974   88957   118725206   1
216.53.250.125  163.35.137.89   17  1   443 56984   50399   67548325    1
216.53.250.125  163.35.137.89   17  1   443 57670   57254   76630623    1
216.53.250.125  163.35.137.89   17  1   443 59325   15071   20313210    1
216.53.250.125  163.35.137.89   17  1   443 61155   39377   53121691    1
...
```

III Plot the number of flows for the 100 most common pairs of hosts

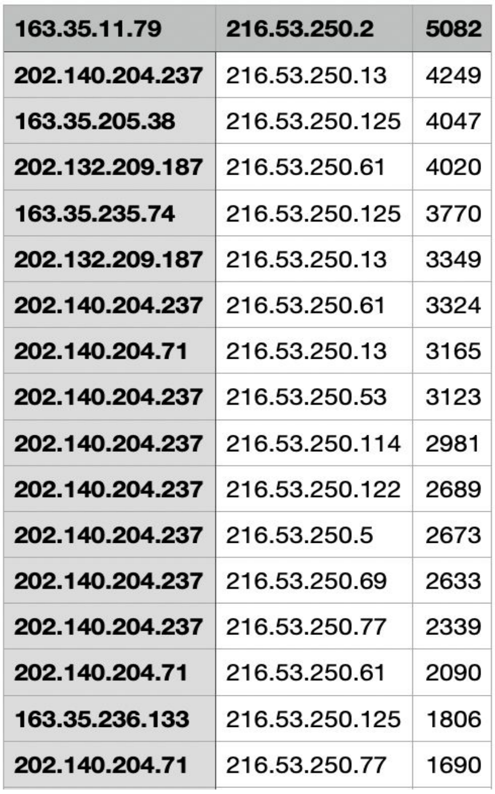
The source file for this question is the "flow.csv" generated at the beginning.
First of all, use the "flow.csv" file to find the most common pairs of hosts through the following python script and shell cmd.

```python
def getFileContext(path):
   with open(path,"r") as file:
       lines=file.readlines()
       return lines[1:]
def getKey(str1,str2):
   if str1 >str2:
       return str2+","+str1
   return str1+","+str2

lines=getFileContext("flow.csv")
maps={}
for item in lines:
   arr=item.split(',')
   key=getKey(arr[0],arr[1])
   if key in maps:
       maps[key]=maps[key]+1
   else:
       maps[key]=1
sorted(maps.items(),key=lambda item:item[1])
with open("result.csv","w") as file:
   for item in maps.keys():
       file.write(item+","+str(maps[item])+"\n")

print(len(lines))
print(len(maps))
```

```
awk -F ',' '{print $1" " $2 " "$3}' result.csv|sort -rnk 3 |awk '{print  $1"," $2
","$3}'  > top100_common.csv
```

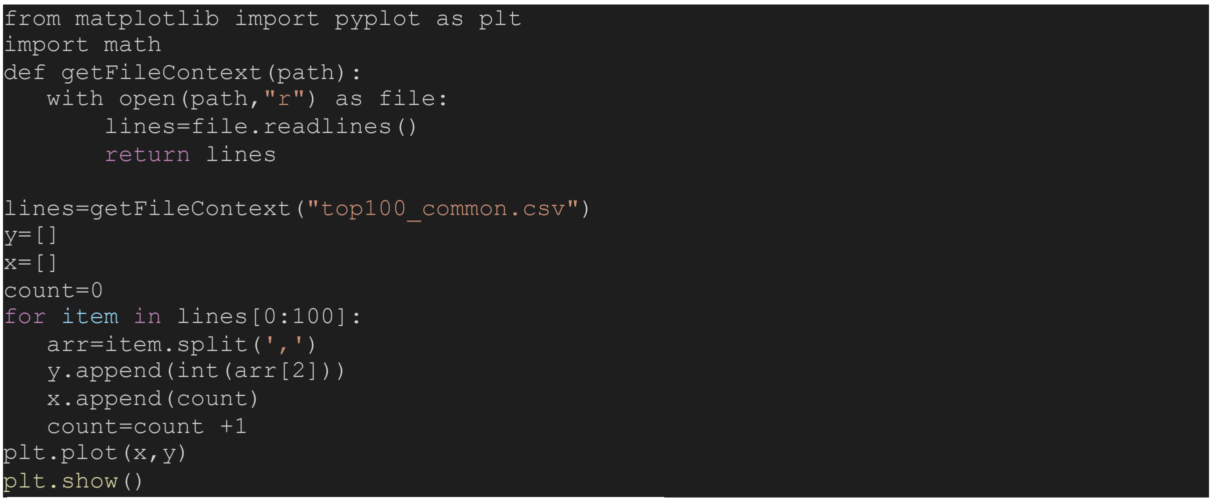
So in the "top100_common.csv" file, the first and the second columns are the host pairs, the third column is the frequency of the host pairs, which is also the number of the flows of the host pairs. The top 100 common host pairs can be seen from the line 1 to line 100.

Note: just showing part of the "top100_common.csv", more detailed can be seen through the "top100_common.csv" file in AS3.zip file

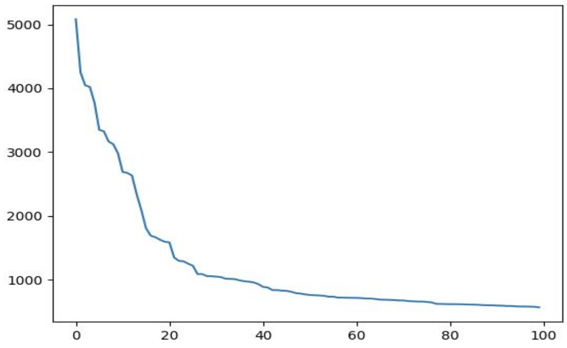


163.35.11.79	216.53.250.2	5082
202.140.204.237	216.53.250.13	4249
163.35.205.38	216.53.250.125	4047
202.132.209.187	216.53.250.61	4020
163.35.235.74	216.53.250.125	3770
202.132.209.187	216.53.250.13	3349
202.140.204.237	216.53.250.61	3324
202.140.204.71	216.53.250.13	3165
202.140.204.237	216.53.250.53	3123
202.140.204.237	216.53.250.114	2981
202.140.204.237	216.53.250.122	2689
202.140.204.237	216.53.250.5	2673
202.140.204.237	216.53.250.69	2633
202.140.204.237	216.53.250.77	2339
202.140.204.71	216.53.250.61	2090
163.35.236.133	216.53.250.125	1806
202.140.204.71	216.53.250.77	1690

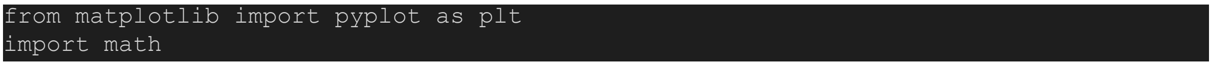
- Using linear scale



```
from matplotlib import pyplot as plt
import math
def getFileContext(path):
   with open(path,"r") as file:
       lines=file.readlines()
       return lines

lines=getFileContext("top100_common.csv")
y=[]
x=[]
count=0
for item in lines[0:100]:
   arr=item.split(',')
   y.append(int(arr[2]))
   x.append(count)
   count=count +1
plt.plot(x,y)
plt.show()
```

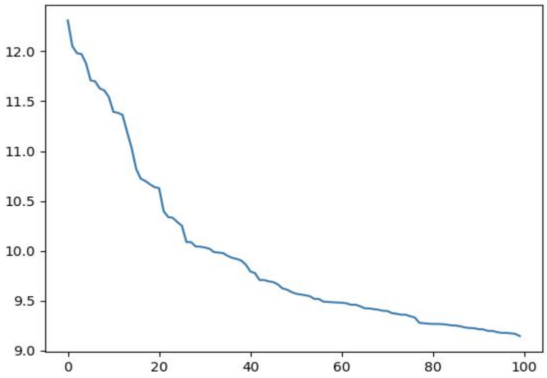


- Using logarithmic scale



```
from matplotlib import pyplot as plt
import math
```

```
def getFileContext(path):
    with open(path,"r") as file:
        lines=file.readlines()
        return lines

lines=getFileContext("top100_common.csv")
logy=[]
x=[]
count=0
for item in lines[0:100]:
    arr=item.split(',')
    logy.append(math.log2(int(arr[2])))
    x.append(count)
    count=count +1
plt.plot(x,logy)
plt.show()
```



IV Discussion on resource memory requirements

Test the time and memory resources for “crl_flow” cmd, using “time” cmd to check the time resource and using “free -m” to check memory resources.

```
# In the 1st terminal
yanj3@force ...AS3/task1/test % time crl_flow -Ci=3600 -cl -Tf60 -O %i.t2 -Cai=1
flow.pcap
crl_flow -Ci=3600 -cl -Tf60 -O %i.t2 -Cai=1 flow.pcap  6.31s user 0.96s system 97%
cpu 7.490 total
# In the 2rd terminal
yanj3@force ...AS3/task1/test % while true
do
free -m >> log
Done
yanj3@force ...AS3/task1/test % cat log| more
             total        used        free      shared  buff/cache   available
Mem:        774033        7564      710974         114       55494      761474
Swap:         7725        2677        5048
...
             total        used        free      shared  buff/cache   available
Mem:        774033        7565      710973         114       55494      761473
Swap:         7725        2677        5048
...
```

Test the time and memory resources for “netmate” cmd, using “time” cmd to check the time resource and using “free -m” to check memory resources.

```
# In the first terminal
yanj3@force ...AS3/task1/test % time netmate -r netAI-e7130.xml -f flow.pcap -l
flow.log
netmate -r netAI-e7130.xml -f flow.pcap -l flow.log  61.34s user 9.21s system 99%
cpu 1:10.64 total
```

```
# In the second terminal
yanj3@force ...AS3/task1/test % while true
do
free -m >> log
done
yanj3@force ...AS3/task1/test % cat log
...
             total        used        free      shared  buff/cache   available
Mem:         774033        7564      710905         114       55562      761474
Swap:          7725        2677        5048
             total        used        free      shared  buff/cache   available
Mem:         774033        7565      710905         114       55562      761473
Swap:          7725        2677        5048
...
```

From the results shown above, we can see that: 1> The “crl_flow” lasts 6.31s and nearly 1M memory. 2> The “netmate” lasts 61.34s and nearly 1M memory.
Basically, “netmate” cmd uses much more time than “crl_flow” cmd, while they consume nearly the same memory.

Report, task 2

I Capture network traffic

I use “tcpdump” cmd to capture packets, while listening to the en0 interface in my own computer, and then saving the captured packets into files.
More detailed about the following two packets (capture_one_hour.pcap and capture_fif_min.pcap) can be seen in AS3.zip file
- For a duration of one hour (during which, I browsed some websites, used some instant msg app, checked emails for a short time.)

```
yanjing@yanjingdeMacBook-Pro task2 % tcpdump -D
1.en0 [Up, Running]
...
yanjing@yanjingdeMacBook-Pro task2 % date; tcpdump -i en0 -n -w
capture_one_hour.pcap
Mon Oct 12 12:28:33 EEST 2020
tcpdump: listening on en0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C56921 packets captured
56923 packets received by filter
0 packets dropped by kernel
yanjing@yanjingdeMacBook-Pro task2 % date
Mon Oct 12 13:33:09 EEST 2020
yanjing@yanjingdeMacBook-Pro task2 % ls -lh capture_one_hour.pcap
-rw-r--r--  1 yanjing  staff    33M Oct 12 13:33 capture_one_hour.pcap
```

- For a duration of fifteen minutes (during which I do ping operation and iperf3 operation, however, the iperf servers in the previous assignment do not work, so I use some public iperf servers.)

```
# In the 1st terminal
yanjing@yanjingdeMacBook-Pro task2 % date; tcpdump -i en0 -n -w
capture_fif_min.pcap
Mon Oct 12 14:15:57 EEST 2020
tcpdump: listening on en0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C192973 packets captured
192973 packets received by filter
0 packets dropped by kernel
yanjing@yanjingdeMacBook-Pro task2 % date
Mon Oct 12 14:32:48 EEST 2020
yanjing@yanjingdeMacBook-Pro task2 % ls -lh capture_fif_min.pcap
```

```
-rw-r--r--  1 yanjing  staff   187M Oct 12 14:32 capture_fif_min.pcap

# In the 2nd terminal
yanjing@yanjingdeMacBook-Pro task2 % ping hlz-nz.ark.caida.org -c 5
yanjing@yanjingdeMacBook-Pro task2 % ping ok1.iperf.comnet-student.eu -c 5
yanjing@yanjingdeMacBook-Pro task2 % ping blr1.iperf.comnet-student.eu -c 5
yanjing@yanjingdeMacBook-Pro task2 % ping pna-es.ark.caida.org -c 5
yanjing@yanjingdeMacBook-Pro task2 % iperf3 -c bouygues.iperf.fr -p 9200
```

II Summary of capture data for both sessions

From the test results shown above, we can see that: 1> The size of the “capture_one_hour.pcap” file is 33M, and there are 56921 packets in the “capture_one_hour.pcap” file. 2> The size of the “capture_fif_min.pcap” file is 187M, and there are 192973 packets in he “capture_fif_min.pcap”

III Differences between capture file statistics and counters

The values in interface counters (I mean RX + TX) equals the number of packets in these two capture files. There should be one extra condition, which is before each capture, the RX and TX should be set as 0. As we know, tcpdump captures the packets both received by the interface and transferred by the interface, but for the interface counters, it distinguishes the receiving packets as RX and transferring packets as TX.
Note: As for the “interface counters” part, I have used “ifconfig interface_name” or “ethtool -g interface_name” cmds before and I do confirm that they can show the RX and TX, something likes as follows. But in my MacBook, I can not install “ethtool” successfully, and my “ifconfig” cmd does not display complete info as follows.

- *RX packets:96430 errors:0 dropped:0 overruns:0 frame:0*
- *TX packets:10274 errors:0 dropped:0 overruns:0 carrier:0*

Report, task 3

Deal with the “capture_one_hour.pcap” file first with the following cmds:
(I captured these packets in my own computer, while dealing with them in aalto shell server)
More detailed about the following two files (capture_one_hour.out and capture_one_hour.t2) can be seen in AS3.zip file

```
yanj3@force ~/AS3/task3 % time netmate -r netAI-e7130.xml -f capture_one_hour.pcap
-l capture_one_hour.log
netmate -r netAI-e7130.xml -f capture_one_hour.pcap -l capture_one_hour.log  0.11s
user 0.09s system 70% cpu 0.292 total
yanj3@force ~/AS3/task3 % cp /tmp/netmatee.out ./; mv netmatee.out
capture_one_hour.out
```

```
yanj3@force ~/AS3/task3 % time crl_flow -Ci=3600 -cl -Tf60 -O %i.t2 -Cai=1
capture_one_hour.pcap
crl_flow -Ci=3600 -cl -Tf60 -O %i.t2 -Cai=1 capture_one_hour.pcap  0.04s user 0.01s
system 68% cpu 0.075 total
yanj3@force ~/AS3/task3 % cat 0.t2 1.t2 2.t2 >> capture_one_hour.t2
```

I How many IP (and IPv6 if any) hosts are communicating?

After the “netmate”’s processing, basically, the host pairs will be column 1(which is my own computer) and column 3 (which is the remote website). So after checking the duplicated session, there are 993 host pairs communicating without IPv6 address here.

```
yanj3@force ~/AS3/task3 % cat capture_one_hour.out | awk -F',' '{print $1}' > tmp1
yanj3@force ~/AS3/task3 % cat capture_one_hour.out | awk -F',' '{print $3}' > tmp2
yanj3@force ~/AS3/task3 % paste -d ',' tmp1 tmp2 > hostpairs
yanj3@force ~/AS3/task3 % cat hostpairs | uniq | wc -l
993
```

II How many hosts were tried to contact, but communication failed for a reason or another? Can you identify different subclasses of failed communications?

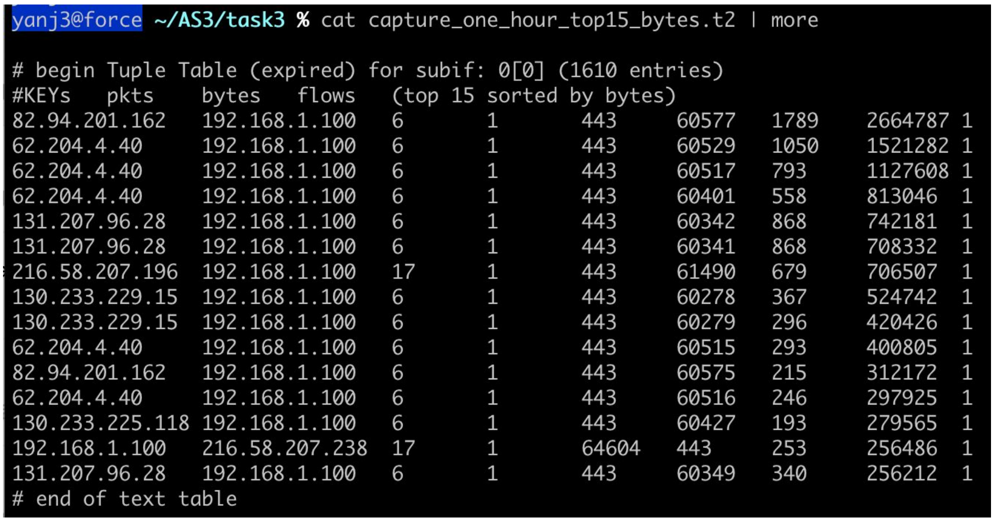
There is no failed session in the “capture_one_hour.pcap” file.
As for how to distinguish the failed communications, I just found some examples which were also generated by the “netmate” tool from a capture file. Two records as follows, the 1st one is a failed session cause the dst port is 0 and no any packets/bytes during this session.

```
216.53.250.66,0,163.35.157.90,31032,1,1,84,0,0,84,84,84,0,-1,-1,-1,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,1,84,0,0,0,0,0,0,28,0
216.53.250.77,443,202.132.241.114,58203,17,2,151,0,0,59,75,92,23,-1,-1,-1,0,11417,1
1417,11417,0,0,0,0,0,11417,11417,11417,11417,0,0,0,0,0,2,151,0,0,0,0,0,0,48,0
```

III Top 15 hosts by byte counts.

```
yanj3@force ~/AS3/task3 % t2_top -Sb -n 15 < capture_one_hour.t2 >
capture_one_hour_top15_bytes.t2
```

Note: Just display some of the results, more info can be seen in the “capture_one_hour_top15_bytes.t2” file in the AS3.zip.

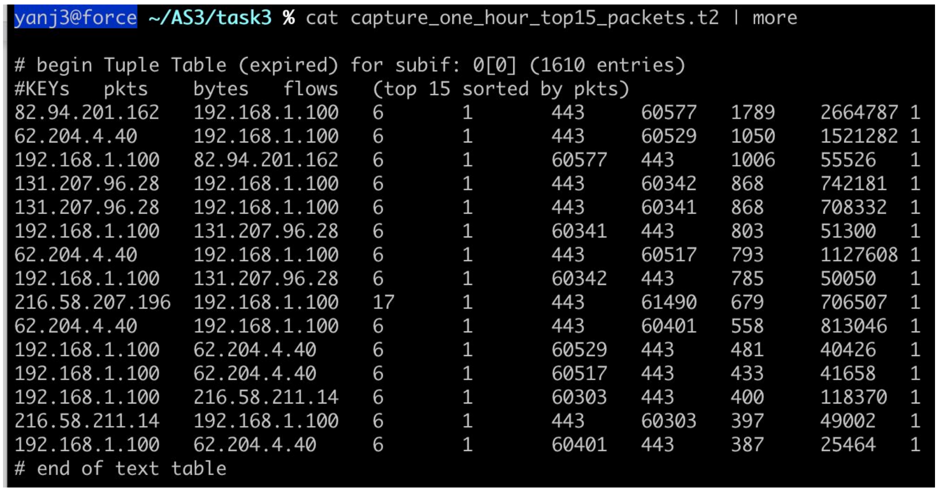


```
yanj3@force ~/AS3/task3 % cat capture_one_hour_top15_bytes.t2 | more

# begin Tuple Table (expired) for subif: 0[0] (1610 entries)
#KEYs    pkts     bytes    flows    (top 15 sorted by bytes)
82.94.201.162    192.168.1.100    6        1        443      60577    1789     2664787 1
62.204.4.40      192.168.1.100    6        1        443      60529    1050     1521282 1
62.204.4.40      192.168.1.100    6        1        443      60517    793      1127608 1
62.204.4.40      192.168.1.100    6        1        443      60401    558      813046  1
131.207.96.28    192.168.1.100    6        1        443      60342    868      742181  1
131.207.96.28    192.168.1.100    6        1        443      60341    868      708332  1
216.58.207.196   192.168.1.100    17       1        443      61490    679      706507  1
130.233.229.15   192.168.1.100    6        1        443      60278    367      524742  1
130.233.229.15   192.168.1.100    6        1        443      60279    296      420426  1
62.204.4.40      192.168.1.100    6        1        443      60515    293      400805  1
82.94.201.162    192.168.1.100    6        1        443      60575    215      312172  1
62.204.4.40      192.168.1.100    6        1        443      60516    246      297925  1
130.233.225.118  192.168.1.100    6        1        443      60427    193      279565  1
192.168.1.100    216.58.207.238   17       1        64604    443      253      256486  1
131.207.96.28    192.168.1.100    6        1        443      60349    340      256212  1
# end of text table
```

IV Top 15 hosts by packet counts.

```
yanj3@force ~/AS3/task3 % t2_top -Sp -n 15 < capture_one_hour.t2 >
capture_one_hour_top15_packets.t2
```

Note: Just display some of the results, more info can be seen in the “capture_one_hour_top15_packets.t2” file in the AS3.zip.



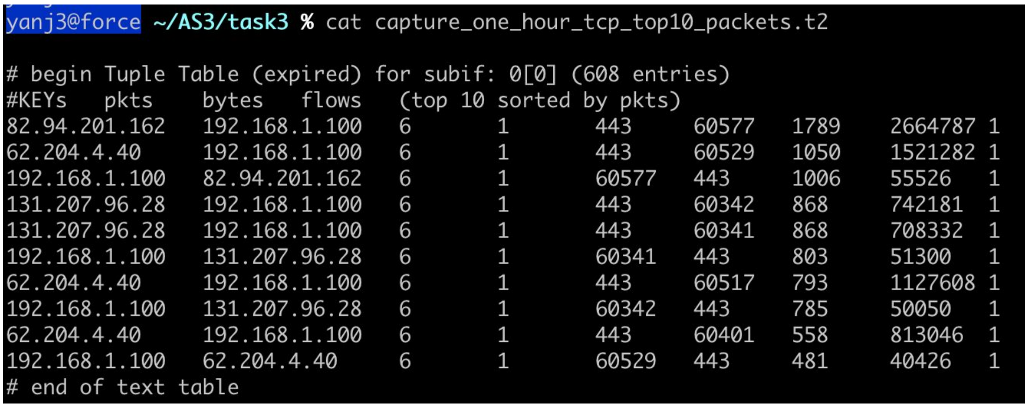
```
yanj3@force ~/AS3/task3 % cat capture_one_hour_top15_packets.t2 | more

# begin Tuple Table (expired) for subif: 0[0] (1610 entries)
#KEYs    pkts     bytes    flows    (top 15 sorted by pkts)
82.94.201.162   192.168.1.100   6       1       443     60577   1789    2664787 1
62.204.4.40     192.168.1.100   6       1       443     60529   1050    1521282 1
192.168.1.100   82.94.201.162   6       1       60577   443     1006    55526   1
131.207.96.28   192.168.1.100   6       1       443     60342   868     742181  1
131.207.96.28   192.168.1.100   6       1       443     60341   868     708332  1
192.168.1.100   131.207.96.28   6       1       60341   443     803     51300   1
62.204.4.40     192.168.1.100   6       1       443     60517   793     1127608 1
192.168.1.100   131.207.96.28   6       1       60342   443     785     50050   1
216.58.207.196  192.168.1.100   17      1       443     61490   679     706507  1
62.204.4.40     192.168.1.100   6       1       443     60401   558     813046  1
192.168.1.100   62.204.4.40     6       1       60529   443     481     40426   1
192.168.1.100   62.204.4.40     6       1       60517   443     433     41658   1
192.168.1.100   216.58.211.14   6       1       60303   443     400     118370  1
216.58.211.14   192.168.1.100   6       1       443     60303   397     49002   1
192.168.1.100   62.204.4.40     6       1       60401   443     387     25464   1
# end of text table
```

V Top 10 TCP and top 5 UDP port numbers (by packet count).

Extract the "TCP" packets from the "capture_one_hour.pcap" file and write them to a new file "cap_one_hour_tcp.pcap", then use "crl_flow" and "t2_top" cmds.

```
yanj3@force ~/AS3/task3 % /usr/sbin/tcpdump -ntt -r capture_one_hour.pcap tcp -w
capture_one_hour_tcp.pcap
yanj3@force ~/AS3/task3 % time crl_flow -Ci=3600 -cl -Tf60 -O %i.t2 -Cai=1
capture_one_hour_tcp.pcap
yanj3@force ~/AS3/task3 % cat 0.t2 1.t2 2.t2 >> capture_one_hour_tcp.t2
yanj3@force ~/AS3/task3 % t2_top -Sp -n 10 < capture_one_hour_tcp.t2 >
capture_one_hour_tcp_top10_packets.t2
```

Note: Just display some of the results, more info can be seen in the "capture_one_hour_tcp_top10_packets.t2" file in the AS3.zip.

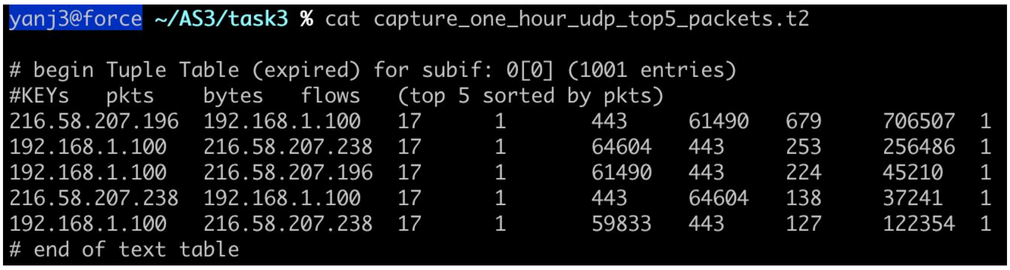


```
yanj3@force ~/AS3/task3 % cat capture_one_hour_tcp_top10_packets.t2

# begin Tuple Table (expired) for subif: 0[0] (608 entries)
#KEYs    pkts     bytes    flows    (top 10 sorted by pkts)
82.94.201.162   192.168.1.100   6       1       443     60577   1789    2664787 1
62.204.4.40     192.168.1.100   6       1       443     60529   1050    1521282 1
192.168.1.100   82.94.201.162   6       1       60577   443     1006    55526   1
131.207.96.28   192.168.1.100   6       1       443     60342   868     742181  1
131.207.96.28   192.168.1.100   6       1       443     60341   868     708332  1
192.168.1.100   131.207.96.28   6       1       60341   443     803     51300   1
62.204.4.40     192.168.1.100   6       1       443     60517   793     1127608 1
192.168.1.100   131.207.96.28   6       1       60342   443     785     50050   1
62.204.4.40     192.168.1.100   6       1       443     60401   558     813046  1
192.168.1.100   62.204.4.40     6       1       60529   443     481     40426   1
# end of text table
```

Extract the "UDP" packets from the "capture_one_hour.pcap" file and write them to a new file "cap_one_hour_udp.pcap", then use "crl_flow" and "t2_top" cmds.

```
yanj3@force ~/AS3/task3 % /usr/sbin/tcpdump -ntt -r capture_one_hour.pcap udp -w
capture_one_hour_udp.pcap
yanj3@force ~/AS3/task3 % time crl_flow -Ci=3600 -cl -Tf60 -O %i.t2 -Cai=1
capture_one_hour_udp.pcap
crl_flow -Ci=3600 -cl -Tf60 -O %i.t2 -Cai=1 capture_one_hour_udp.pcap  0.02s user
0.01s system 25% cpu 0.080 total
yanj3@force ~/AS3/task3 % cat 0.t2 1.t2 2.t2 >> capture_one_hour_udp.t2
yanj3@force ~/AS3/task3 % t2_top -Sp -n 5 < capture_one_hour_udp.t2 >
capture_one_hour_udp_top5_packets.t2
```

Note: Just display some of the results, more info can be seen in the “capture_one_hour_udp_top5_packets.t2” file in the AS3.zip.



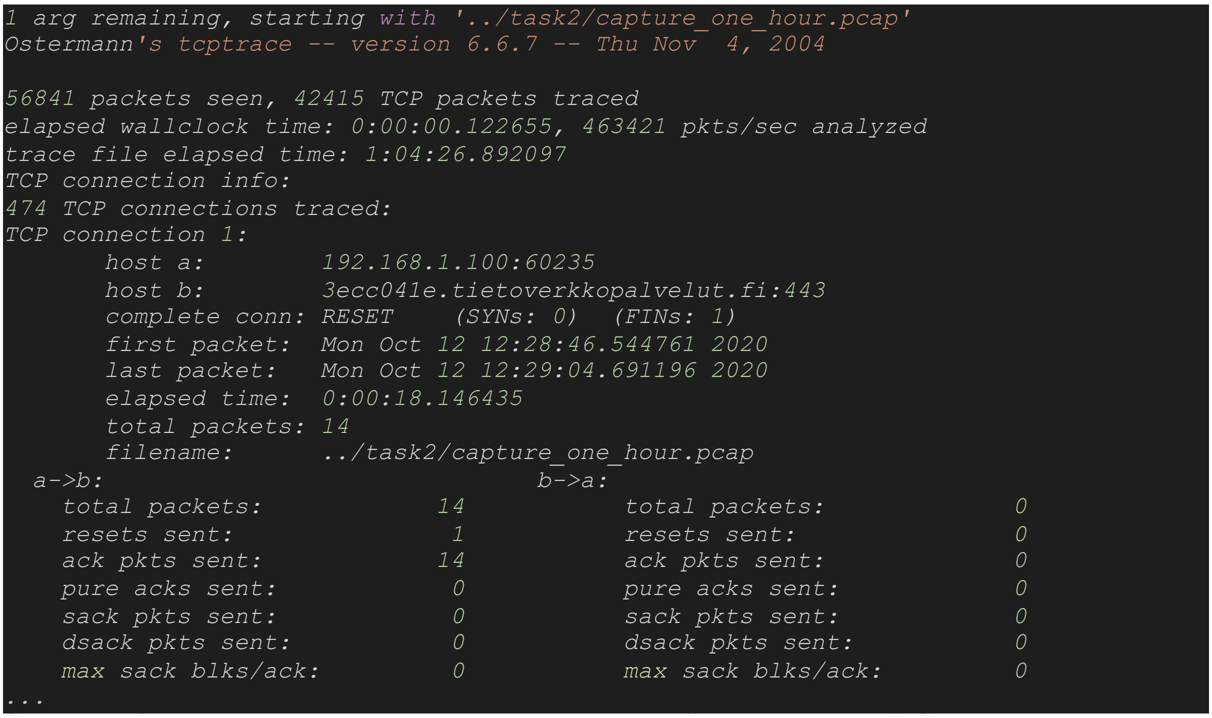
```
yanj3@force ~/AS3/task3 % cat capture_one_hour_udp_top5_packets.t2

# begin Tuple Table (expired) for subif: 0[0] (1001 entries)
#KEYs    pkts     bytes    flows    (top 5 sorted by pkts)
216.58.207.196  192.168.1.100   17      1       443     61490   679     706507  1
192.168.1.100   216.58.207.238  17      1       64604   443     253     256486  1
192.168.1.100   216.58.207.196  17      1       61490   443     224     45210   1
216.58.207.238  192.168.1.100   17      1       443     64604   138     37241   1
192.168.1.100   216.58.207.238  17      1       59833   443     127     122354  1
# end of text table
```

VI Top 10 fastest TCP connections

Get the TCP throughout info from “capture_one_hour.pcap” through tcptrace tool.

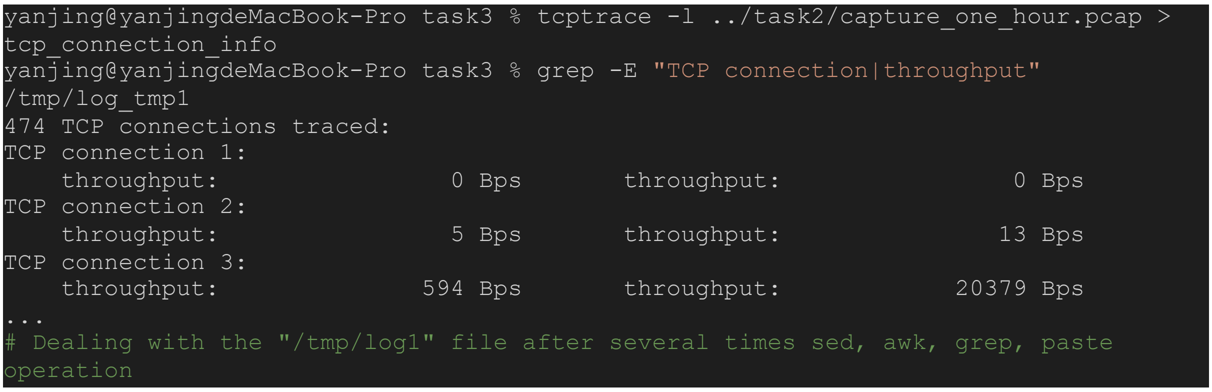
More info about “tcp_connection_info”, “top10_fastest_tcp_connection.csv” file in the AS3.zip.

In “tcp_connection_info” file, we can see the following:

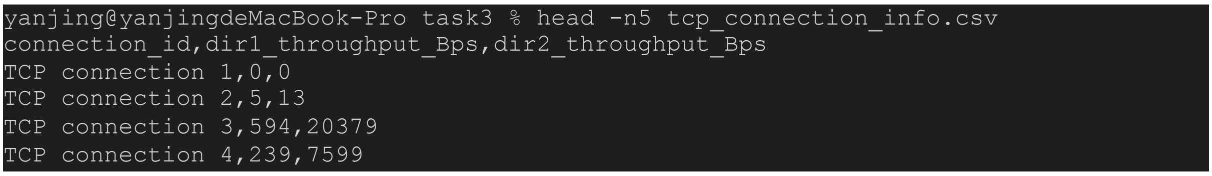


```
1 arg remaining, starting with '../task2/capture_one_hour.pcap'
Ostermann's tcptrace -- version 6.6.7 -- Thu Nov  4, 2004

56841 packets seen, 42415 TCP packets traced
elapsed wallclock time: 0:00:00.122655, 463421 pkts/sec analyzed
trace file elapsed time: 1:04:26.892097
TCP connection info:
474 TCP connections traced:
TCP connection 1:
       host a:        192.168.1.100:60235
       host b:        3ecc041e.tietoverkkopalvelut.fi:443
       complete conn: RESET    (SYNs: 0)   (FINs: 1)
       first packet:  Mon Oct 12 12:28:46.544761 2020
       last packet:   Mon Oct 12 12:29:04.691196 2020
       elapsed time:  0:00:18.146435
       total packets: 14
       filename:      ../task2/capture_one_hour.pcap
  a->b:                              b->a:
    total packets:            14           total packets:              0
    resets sent:               1           resets sent:                0
    ack pkts sent:            14           ack pkts sent:              0
    pure acks sent:            0           pure acks sent:             0
    sack pkts sent:            0           sack pkts sent:             0
    dsack pkts sent:           0           dsack pkts sent:            0
    max sack blks/ack:         0           max sack blks/ack:          0
...
```

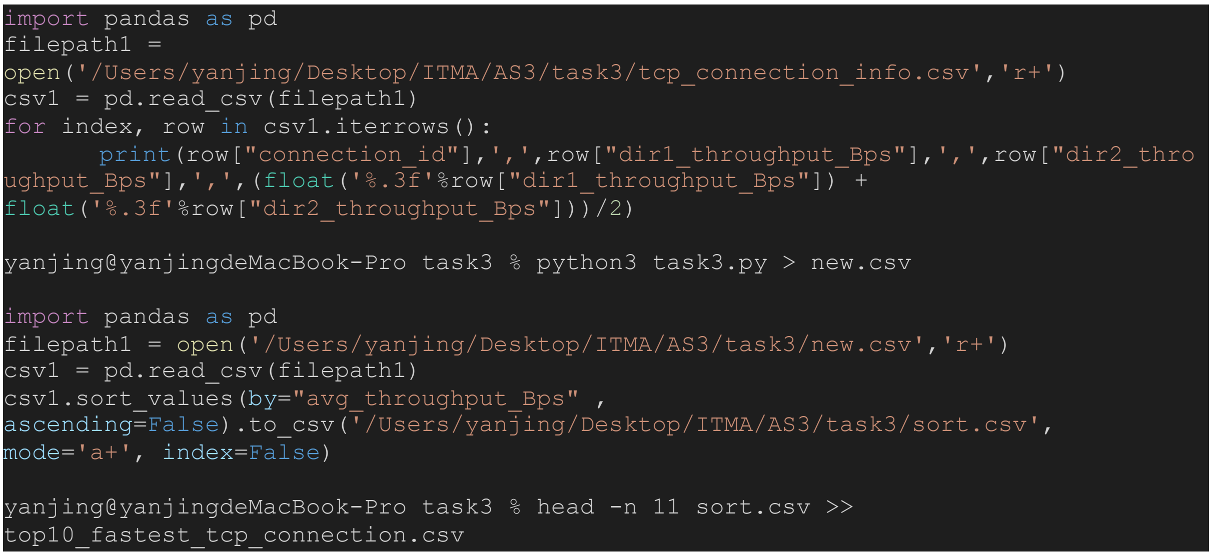
So In the following analysis, I did not display other info, only display “TCP connection ID” and other necessary info. As for the host info, packets info and so on, they can be check in “tcp_connection_info” file.



```
yanjing@yanjingdeMacBook-Pro task3 % tcptrace -l ../task2/capture_one_hour.pcap >
tcp_connection_info
yanjing@yanjingdeMacBook-Pro task3 % grep -E "TCP connection|throughput"
/tmp/log_tmp1
474 TCP connections traced:
TCP connection 1:
    throughput:                 0 Bps       throughput:                 0 Bps
TCP connection 2:
    throughput:                 5 Bps       throughput:                13 Bps
TCP connection 3:
    throughput:               594 Bps       throughput:             20379 Bps
...
# Dealing with the "/tmp/log1" file after several times sed, awk, grep, paste
operation
```

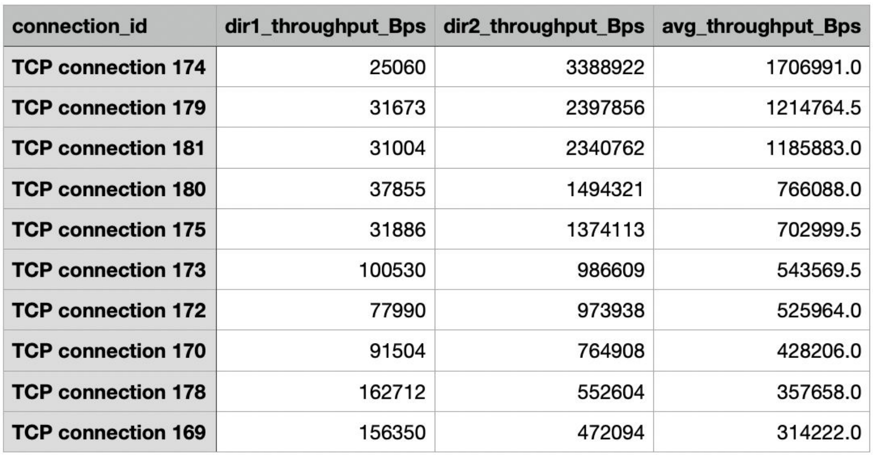


```
yanjing@yanjingdeMacBook-Pro task3 % head -n5 tcp_connection_info.csv
connection_id,dir1_throughput_Bps,dir2_throughput_Bps
TCP connection 1,0,0
TCP connection 2,5,13
TCP connection 3,594,20379
TCP connection 4,239,7599
```



```
import pandas as pd
filepath1 =
open('/Users/yanjing/Desktop/ITMA/AS3/task3/tcp_connection_info.csv','r+')
csv1 = pd.read_csv(filepath1)
for index, row in csv1.iterrows():
       print(row["connection_id"],',',row["dir1_throughput_Bps"],',',row["dir2_thro
ughput_Bps"],',',(float('%.3f'%row["dir1_throughput_Bps"]) +
float('%.3f'%row["dir2_throughput_Bps"]))/2)

yanjing@yanjingdeMacBook-Pro task3 % python3 task3.py > new.csv

import pandas as pd
filepath1 = open('/Users/yanjing/Desktop/ITMA/AS3/task3/new.csv','r+')
csv1 = pd.read_csv(filepath1)
csv1.sort_values(by="avg_throughput_Bps" ,
ascending=False).to_csv('/Users/yanjing/Desktop/ITMA/AS3/task3/sort.csv',
mode='a+', index=False)

yanjing@yanjingdeMacBook-Pro task3 % head -n 11 sort.csv >>
top10_fastest_tcp_connection.csv
```

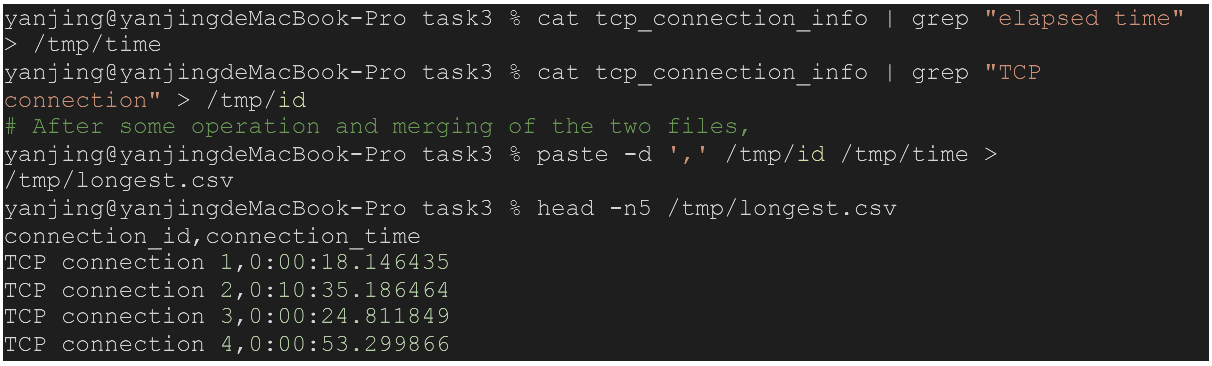
top10_fastest_tcp_connection



connection_id	dir1_throughput_Bps	dir2_throughput_Bps	avg_throughput_Bps
TCP connection 174	25060	3388922	1706991.0
TCP connection 179	31673	2397856	1214764.5
TCP connection 181	31004	2340762	1185883.0
TCP connection 180	37855	1494321	766088.0
TCP connection 175	31886	1374113	702999.5
TCP connection 173	100530	986609	543569.5
TCP connection 172	77990	973938	525964.0
TCP connection 170	91504	764908	428206.0
TCP connection 178	162712	552604	357658.0
TCP connection 169	156350	472094	314222.0

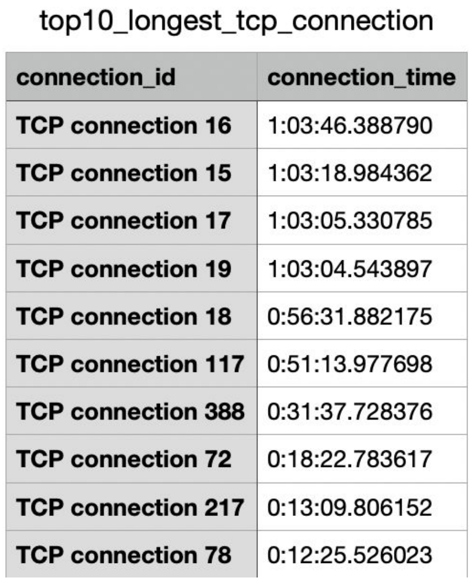
VII Top 10 longest (by time) TCP connections

Reuse the *“tcp_connection_info” file generated by the previous question “VI”*



```
yanjing@yanjingdeMacBook-Pro task3 % cat tcp_connection_info | grep "elapsed time"
> /tmp/time
yanjing@yanjingdeMacBook-Pro task3 % cat tcp_connection_info | grep "TCP
connection" > /tmp/id
# After some operation and merging of the two files,
yanjing@yanjingdeMacBook-Pro task3 % paste -d ',' /tmp/id /tmp/time >
/tmp/longest.csv
yanjing@yanjingdeMacBook-Pro task3 % head -n5 /tmp/longest.csv
connection_id,connection_time
TCP connection 1,0:00:18.146435
TCP connection 2,0:10:35.186464
TCP connection 3,0:00:24.811849
TCP connection 4,0:00:53.299866
```

```
import pandas as pd
filepath1 = open('/tmp/longest.csv','r+')
csv1 = pd.read_csv(filepath1)
csv1.sort_values(by="connection_time" , ascending=False).to_csv('/tmp/sort.csv',
mode='a+', index=False)

yanjing@yanjingdeMacBook-Pro task3 % head -n11 /tmp/sort.csv >
top10_longest_tcp_connection.csv
```



top10_longest_tcp_connection

connection_id	connection_time
TCP connection 16	1:03:46.388790
TCP connection 15	1:03:18.984362
TCP connection 17	1:03:05.330785
TCP connection 19	1:03:04.543897
TCP connection 18	0:56:31.882175
TCP connection 117	0:51:13.977698
TCP connection 388	0:31:37.728376
TCP connection 72	0:18:22.783617
TCP connection 217	0:13:09.806152
TCP connection 78	0:12:25.526023

VIII Did byte and packet count top hosts differ?

Yes, they differ a little, but if the packets transferred among hosts are more, which also means that there is a high possibility the bytes transferred are more, too.

Report, task 4

So first of all, I filtered the ping packets through “ICMP” protocol and then filtered the iperf3 packets through “9200” port (as I used “iperf3 -c bouygues.iperf.fr -p 9200” above) with the following cmds.
More info about the two pcap files (capture_fif_min_icmp.pcap and capture_fif_min_iperf3.pcap) can be seen in the AS3.zip.

```
yanjing@yanjingdeMacBook-Pro task2 % tcpdump -ntt -r capture_fif_min.pcap icmp -w
capture_fif_min_icmp.pcap
yanjing@yanjingdeMacBook-Pro task2 % tcpdump -ntt -r capture_fif_min.pcap port 9200
-w capture_fif_min_iperf3.pcap
```

I How much was traffic that was not iperf or ping traffic?

Obtain the number of packets in each pcap file through the following cmds, then we know that 192973 packets in “capture_fif_min.pcap”, and there are 56 ICMP packets in “capture_fif_min_icmp.pcap”, 165433 iperf3 packets in “capture_fif_min_iperf3.pcap”.
So there are 27484 packets not iperf or ping traffic.

```
yanjing@yanjingdeMacBook-Pro task2 % tcpdump -r capture_fif_min.pcap | wc -l
reading from file capture_fif_min.pcap, link-type EN10MB (Ethernet)
 192973
```

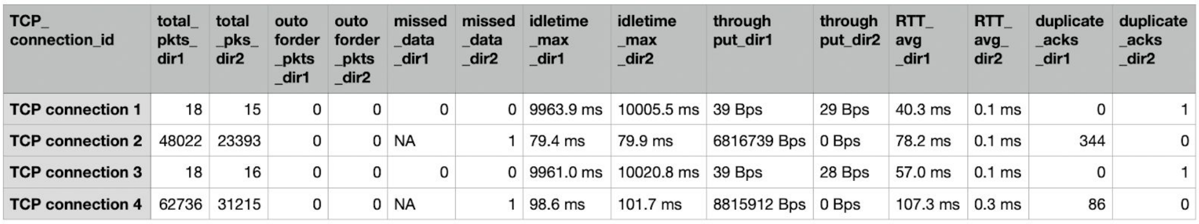
```
yanjing@yanjingdeMacBook-Pro task2 % tcpdump -r capture_fif_min_icmp.pcap | wc -l
reading from file capture_fif_min_icmp.pcap, link-type EN10MB (Ethernet)
     56
yanjing@yanjingdeMacBook-Pro task2 % tcpdump -r capture_fif_min_iperf3.pcap | wc -l
reading from file capture_fif_min_iperf3.pcap, link-type EN10MB (Ethernet)
 165433
```

II Compare iperf results from active and passive measurements. Provide a table.

Extract the needed info from “capture_fif_min_iper3.pcap” file through tcptrace tool, after several times’ “awk”, “sed”, “grep”, ”paste” operation, got the following table.
In “TASK3”, I have shown the info outputted by “tcptrace”, there are two directions, so in the following table, use dir1 and dir2 to distinguish them.

```
yanjing@yanjingdeMacBook-Pro task4 % tcptrace -lrW capture_fif_min_iperf3.pcap >
/tmp/log
yanjing@yanjingdeMacBook-Pro task4 % cat /tmp/log | grep "total packets"
yanjing@yanjingdeMacBook-Pro task4 % cat /tmp/log | grep "outoforder pkts"
yanjing@yanjingdeMacBook-Pro task4 % cat /tmp/log | grep "missed data"
yanjing@yanjingdeMacBook-Pro task4 % cat /tmp/log | grep "idletime max"
yanjing@yanjingdeMacBook-Pro task4 % cat /tmp/log | grep "throughput"
yanjing@yanjingdeMacBook-Pro task4 % cat /tmp/log | grep "RTT avg"
yanjing@yanjingdeMacBook-Pro task4 % cat /tmp/log | grep "duplicate acks"
```

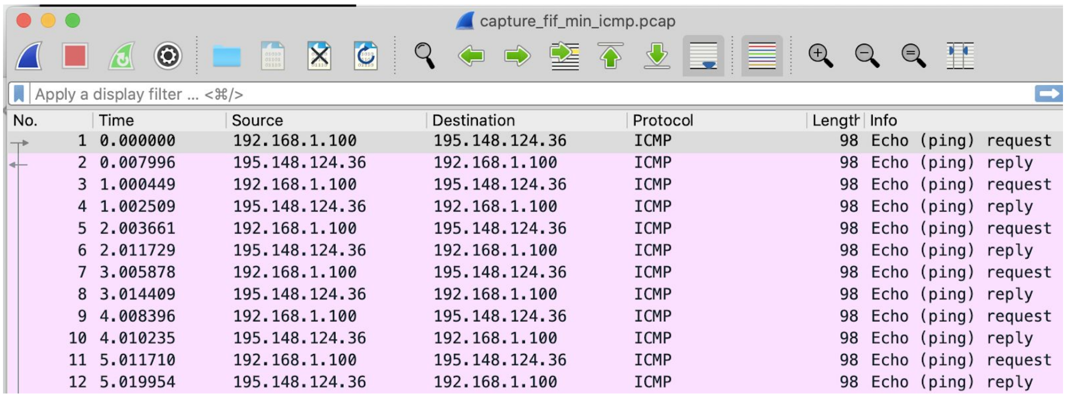
iperf3_info



TCP_ connection_id	total_ pkts_ dir1	total _pks_ dir2	outo forder _pkts _dir1	outo forder _pkts _dir2	missed _data _dir1	missed _data _dir2	idletime _max _dir1	idletime _max _dir2	through put_dir1	through put_dir2	RTT_ avg _dir1	RTT_ avg_ dir2	duplicate _acks _dir1	duplicate _acks _dir2
TCP connection 1	18	15	0	0	0	0	9963.9 ms	10005.5 ms	39 Bps	29 Bps	40.3 ms	0.1 ms	0	1
TCP connection 2	48022	23393	0	0	NA	1	79.4 ms	79.9 ms	6816739 Bps	0 Bps	78.2 ms	0.1 ms	344	0
TCP connection 3	18	16	0	0	0	0	9961.0 ms	10020.8 ms	39 Bps	28 Bps	57.0 ms	0.1 ms	0	1
TCP connection 4	62736	31215	0	0	NA	1	98.6 ms	101.7 ms	8815912 Bps	0 Bps	107.3 ms	0.3 ms	86	0

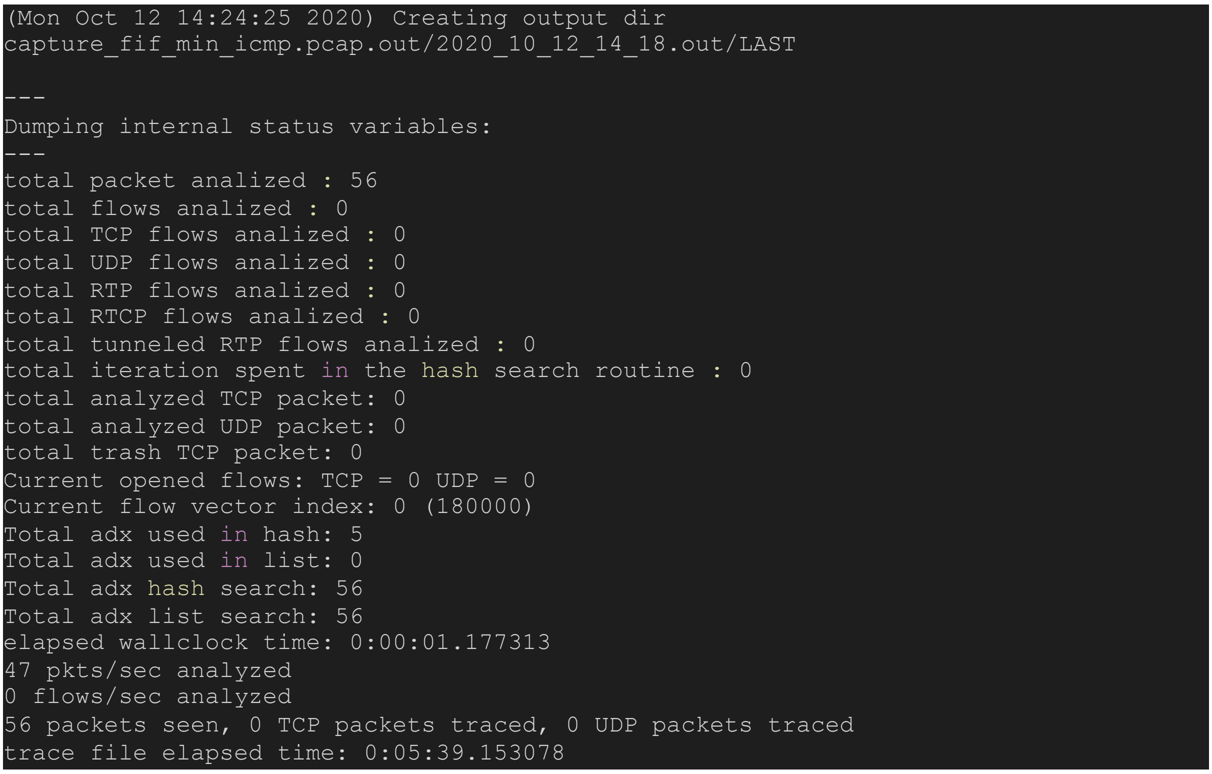
III Compare ping results from active and passive measurements. Provide a table.

After using “tstat” cmd to analyze the “capture_fif_min_icmp.pcap” (There are ICMP packets can be seen through WireShare) file, can not see any info related to delay, packet loss or throughput in the output.



```
No.  Time      Source          Destination     Protocol  Length Info
  1  0.000000  192.168.1.100   195.148.124.36  ICMP      98 Echo (ping) request
  2  0.007996  195.148.124.36  192.168.1.100   ICMP      98 Echo (ping) reply
  3  1.000449  192.168.1.100   195.148.124.36  ICMP      98 Echo (ping) request
  4  1.002509  195.148.124.36  192.168.1.100   ICMP      98 Echo (ping) reply
  5  2.003661  192.168.1.100   195.148.124.36  ICMP      98 Echo (ping) request
  6  2.011729  195.148.124.36  192.168.1.100   ICMP      98 Echo (ping) reply
  7  3.005878  192.168.1.100   195.148.124.36  ICMP      98 Echo (ping) request
  8  3.014409  195.148.124.36  192.168.1.100   ICMP      98 Echo (ping) reply
  9  4.008396  192.168.1.100   195.148.124.36  ICMP      98 Echo (ping) request
 10  4.010235  195.148.124.36  192.168.1.100   ICMP      98 Echo (ping) reply
 11  5.011710  192.168.1.100   195.148.124.36  ICMP      98 Echo (ping) request
 12  5.019954  195.148.124.36  192.168.1.100   ICMP      98 Echo (ping) reply
```

```
yanj3@force ~/AS3/task4 % tstat -N net.conf -H histo.conf capture_fif_min_icmp.pcap
TNG tstat-3.1.1 (Hindenburg flavor) -- Tue May 17 14:41:53 CEST 2016

[Mon Oct 12 14:18:46 2020] created new outdir
capture_fif_min_icmp.pcap.out/2020_10_12_14_18.out
```



```
(Mon Oct 12 14:24:25 2020) Creating output dir
capture_fif_min_icmp.pcap.out/2020_10_12_14_18.out/LAST

---
Dumping internal status variables:
---
total packet analized : 56
total flows analized : 0
total TCP flows analized : 0
total UDP flows analized : 0
total RTP flows analized : 0
total RTCP flows analized : 0
total tunneled RTP flows analized : 0
total iteration spent in the hash search routine : 0
total analyzed TCP packet: 0
total analyzed UDP packet: 0
total trash TCP packet: 0
Current opened flows: TCP = 0 UDP = 0
Current flow vector index: 0 (180000)
Total adx used in hash: 5
Total adx used in list: 0
Total adx hash search: 56
Total adx list search: 56
elapsed wallclock time: 0:00:01.177313
47 pkts/sec analyzed
0 flows/sec analyzed
56 packets seen, 0 TCP packets traced, 0 UDP packets traced
trace file elapsed time: 0:05:39.153078
```