# WEEK 9 LAB: SORTING

CS 162

## Background

Packet capture is the process in which a network analyst collects data on the packets being transmitted across a network between devices so that they can analyze the dataset for interesting or alarming features. The following bullets are things that the network analyst might want from such a packet capture while performing an analysis:

- IP addresses (IPs) that do the most/least talking by numbers of connections.
- IPs that do the most/least talking by the total number of bytes.
- Protocols handling the most/least talking by the numbers of connections.
- Protocols handling the most/least talking by the total number of bytes.
- The most and least common connections (source IP/destination IP pairs).
- The connections (source IP/destination IP pairs) moving the most data.

The file pcap.csv.zip contains a compressed packet capture session from a real network. First, you will need to unzip it (it is a big file). You can view it if you'd like, but since it's a large file with many many many rows, your text editor or even Microsoft Excel, will likely tell you that it will only show you a preview.

About the data inside… Each connection observed is given a unique numerical ID, a time in seconds after the capture began that the connection began, the IP of the source, the IP of the destination, the type of protocol used, and the length, in bytes, of the communication.

Here are the first few lines of that file. Your program will need to process any file of this kind of format, not just the one example being provided to you.

```
"No.","Time","Source","Destination","Protocol","Length"
"1","0.000000","HP_61:aa:c9","HP_61:aa:c9","LLC","54"
"2","0.561324","192.168.1.10","198.32.64.12","DNS","65"
"3","0.562358","192.168.1.1","192.168.1.10","ICMP","70"
"4","1.499948","HP_61:aa:c9","HP_61:aa:c9","LLC","54"
"5","1.571299","192.168.1.10","193.0.14.129","DNS","65"
"6","1.571392","192.168.1.10","192.112.36.4","DNS","65"
"7","1.571466","192.168.1.10","192.112.36.4","DNS","65"
"8","1.571543","192.168.1.10","192.5.5.241","DNS","65"
"9","1.572648","192.168.1.1","192.168.1.10","ICMP","70"
"10","1.853023","Cisco_04:41:bc","Cisco_04:41:bc","LOOP","118"
"11","2.581377","192.168.1.10","192.203.230.10","DNS","65"
```

## Instructions and Requirements

Your task is to develop a network analysis program with the following requirements:

1. Create a main method (in a file named program.py) that has a menu which allows the user to first input the filename of the PCAP csv file. It should then present the user with menu options corresponding to requirements 3 – 8 below. Your main method should then instantiate a NetworkAnalysis object with the filename (see requirement #2). It should then use that object to invoke the method corresponding to the option chosen by

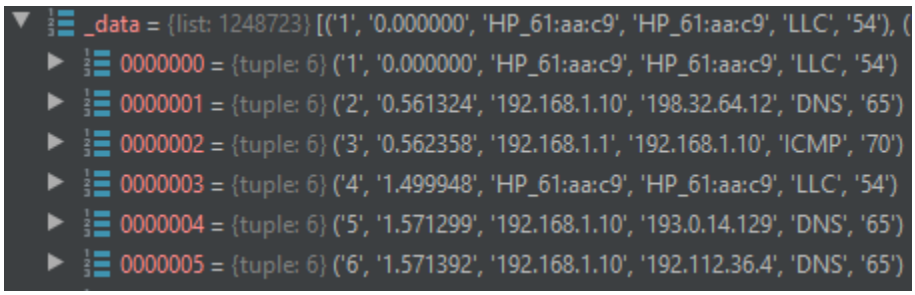
the user. Continually present the menu until the user chooses to quit the program by entering the option to quit.

```
NETWORK ANALYZER

Enter in the PCAP filename: pcap.csv
Choose analysis option:
(0) IPs by Connection
(1) IPs by Bytes
(2) Protocols by Connections
(3) Protocols by Bytes
(4) Connections by Connections
(5) Connections by Bytes
(6) Quit
 --> |
```

2. Make a class called **NetworkAnalysis** (network_analysis.py). It should have a constructor which takes in the filename of the .csv file and it should read and load the data into a class instance variable named _data. You may read the file in manually or use the csv module (recommended). _data should be a list of tuples, where each tuple corresponds to a line in the file and contains 6 items. Remember to skip the first line which contains the column headers. You can do this by using the next(file) function with the file object as a parameter.

   Here is a debug snapshot example of what _data should look like:




3. An instance method called **ips_by_connections()**, which returns a list of tuples, where the first element of the tuple is the connecting (not the connected-to) IP, and the second element is the number of connections for that IP in the file. This list should be sorted, from fewest to most connections.

   Here is a snapshot of the expected output (at least the beginning and end):

0 ('85.166.214.187', 1)
1 ('96.249.244.237', 1)
2 ('213.154.172.143', 1)
3 ('69.22.67.159', 1)
…
977 ('HP_61:aa:c9', 56082)
978 ('172.16.112.100', 61071)

4. An instance method called **ips_by_bytes()**, which does the same, but sorted by number of bytes. The second number in the tuple should be the total number of bytes.

   Here is a snapshot of the expected output (at least the beginning and end):

   0 ('Oracle_11:34:bf', 42)
   1 ('128.42.30.180', 60)
   2 ('4e:d7:9d:30:30:66', 60)
   3 ('FujitsuT_89:58:22', 60)
   …
   977 ('172.16.112.100', 14260820)
   978 ('173.14.243.233', 22024758)

5. An instance method called **protocols_by_connections()**, which returns a list of tuples, where the first element of the tuple is the protocol, and the second element is the number of connections for that protocol in the file. This list should be sorted, from fewest to most connections.

   Here is a snapshot of the expected output (at least the beginning and end):

   0 ('DB-LSP', 1)
   1 ('Socks', 1)
   2 ('ANSI C12.22', 1)
   3 ('Rlogin', 1)
   …
   65 ('TELNET', 293419)
   66 ('TCP', 630301)

6. An instance method called **protocols_by_bytes()**, which does the same, but sorted by number of bytes. The second number in the tuple should be the total number of bytes.

   Here is a snapshot of the expected output (at least the beginning and end):

   0 ('Rlogin', 55)
   1 ('DB-LSP', 60)
   2 ('RTSP', 153)
   3 ('EAPOL', 161)
   …

65 ('HTTP', 32976812)
66 ('TCP', 213941036)

7. An instance method called **connections_by_connections()**, in which the tuples are, in order, the sending IP, the receiving IP, and the number of connections.

   Here is a snapshot of the expected output (at least the beginning and end):

   0 (('221.226.84.106', '192.168.221.128'), 1)
   1 (('54.164.148.246', '192.168.221.128'), 1)
   2 (('172.16.118.1', '192.168.1.20'), 1)
   …
   4336 (('172.16.112.20', '172.16.112.100'), 37033)
   4337 (('HP_61:aa:c9', 'HP_61:aa:c9'), 55978)

8. An instance method called **connections_by_bytes()**, in which the tuples are, in order, the sending IP, the receiving IP, and the number of bytes.

   Here is a snapshot of the expected output (at least the beginning and end):

   0 (('be:08:f1:2f:fa:d4', '86:79:13:59:87:b5'), 42)
   1 (('Oracle_83:4a:82', 'Oracle_09:b9:49'), 42)
   2 (('Oracle_11:34:bf', 'Cisco_04:41:bc'), 42)
   3 (('Vmware_84:86:5f', 'Vmware_34:9c:9d'), 42)
   …
   4336 (('172.16.112.194', '194.7.248.153'), 7311430)
   4337 (('173.14.243.233', '192.168.221.128'), 22024758)

9. All sorting in requirements 3 – 8 should be done using **QuickSort** (use the base recursive implementation provided in ZyBooks). Add the QuickSort methods to your NetworkAnalysis class as "private" methods (put an underscore in front of the method names). You will have to modify the QuickSort implementation methods slightly to support sorting a list of tuples instead of individual data elements. To do this, QuickSort will need an additional parameter that tells it which tuple index it should use for comparing.

## Hints

- Dictionaries make great containers when you are wanting to count things. You can use the "thing" you are counting as the key and the count as the value.
- Tuples are like lists but are immutable.
- Requirements 3 – 8 require sorting. Filter the results first, and then send them through QuickSort.
- Many more hints and examples will be given in lecture!