UWE Bristol University of the West of England UWEcyber Gold Award in association with National Cyber Security Centre Department for Digital, Culture, Media & Sport Academic Centre of Excellence in Cyber Security Education

UFCFEL-15-3 Security Data Analytics and Visualisation

Portfolio Assignment 1: Visualisation for Network Traffic Analysis (2022)

The completion of this worksheet is worth a **maximum of 20 marks** towards your portfolio assignment for the UFCFEL-15-3 Security Data Analytics and Visualisation (SDAV) module.

Brief

You have been asked to examine a sample of network traffic to investigate suspicious activity on some of the company workstations. The company directors need to be able to understand this data. Your task is to **produce a series of different visual representations to describe and understand the characteristics of the data, based on the task questions below**. You should use the Matplotlib documentation and the Pandas documentation to learn about the library functionality, as well as other online resources.

Assessment and Marking

For each question you will see the maximum number of marks you may be awarded for a complete answer in brackets.

- **Task 1:** Plot a Line Chart that shows "Minutes" on the x-axis, and "Total Number of Packets" sent on the y-axis. (3)
- **Task 2:** Plot a Line Chart that shows "Minutes" on the x-axis, and "Total Packet Length" sent on the y-axis. (3)
- **Task 3:** Display a Bar Chart that shows "Protocol" on the x-axis, and "Count" on the y-axis. (2)
- **Task 4:** Display a Scatter Chart that shows the association between Source and Destination data. (2)
- **Task 5:** Filter the data so that only 10.x.x.x Source addresses are included in a new DataFrame. (1)
- **(Advanced) Task 6:** Display a Node Link Diagram for this new DataFrame. (3)
- **(Advanced) Task 7:** For each Protocol type contained in this Dataframe, create a new Column and assign whether the Protocol usage is True or False. (3)
- **(Advanced) Task 8:** Show a Multi-Line Chart that shows the Total Packet Length Per Protocol. (3)

This assignment should be submitted as as PDF to your Blackboard portfolio submission as per the instructions in the assignment specification available on Blackboard. A copy of your work should also be provided via a UWE Gitlab repository, with an accessible link provided with your portfolio.

Contact

Questions about this assignment should be directed to your module leader (Phil.Legg@uwe.ac.uk). You can use the Blackboard Q&A feature to ask questions related to this module and this assignment, as well as the on-site teaching sessions.

In [1]:

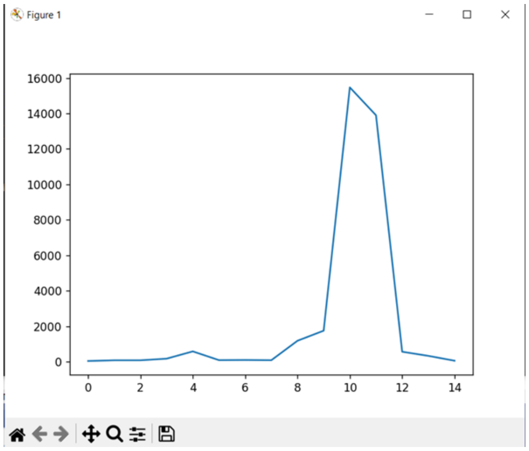
```
### Load in the libraries and the data
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
import seaborn as sns

# The following line is useful before each plot to increase the default size that it is rendered at:
# plt.figure(figsize=(20,10))

data = pd.read_csv('./T1_data/2022-task1_data.csv')
data
```

Out [1]:

Task 1: Plot a Line Chart that shows "Minutes" on the x-axis, and "Total Number of Packets" sent on the y-axis. (3)

Hint: The Time column could be grouped by minute by changing the precision of how time is measured.

```
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
import seaborn as sns

data = pd.read_csv('./T1_data/2022-task1_data.csv')
data

x=data['Minutes'].value_counts().sort_index()
print(x)
plt.plot(x)
plt.show()
```



In [2]:

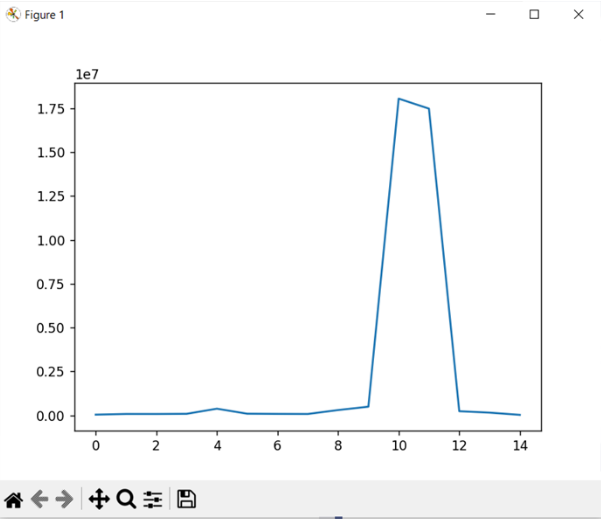
```
# ANSWER
```

Task 2: Plot a Line Chart that shows "Minutes" on the x-axis, and "Total Packet Length" sent on the y-axis. (3)

Hint: Group you data by "Time" and then you can take the sum of the Length column.

In [3]:

```
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
import seaborn as sns

data = pd.read_csv('./T1_data/2022-task1_data.csv')
data

data1 = pd.DataFrame({"Minutes": data['Minutes'].tolist(),           # Create pandas
DataFrame
                     "Length": data['Length'].tolist()})
print(data1)
x1=data1.groupby('Minutes').sum()
print(x1)
plt.plot(x1)
plt.show()
```

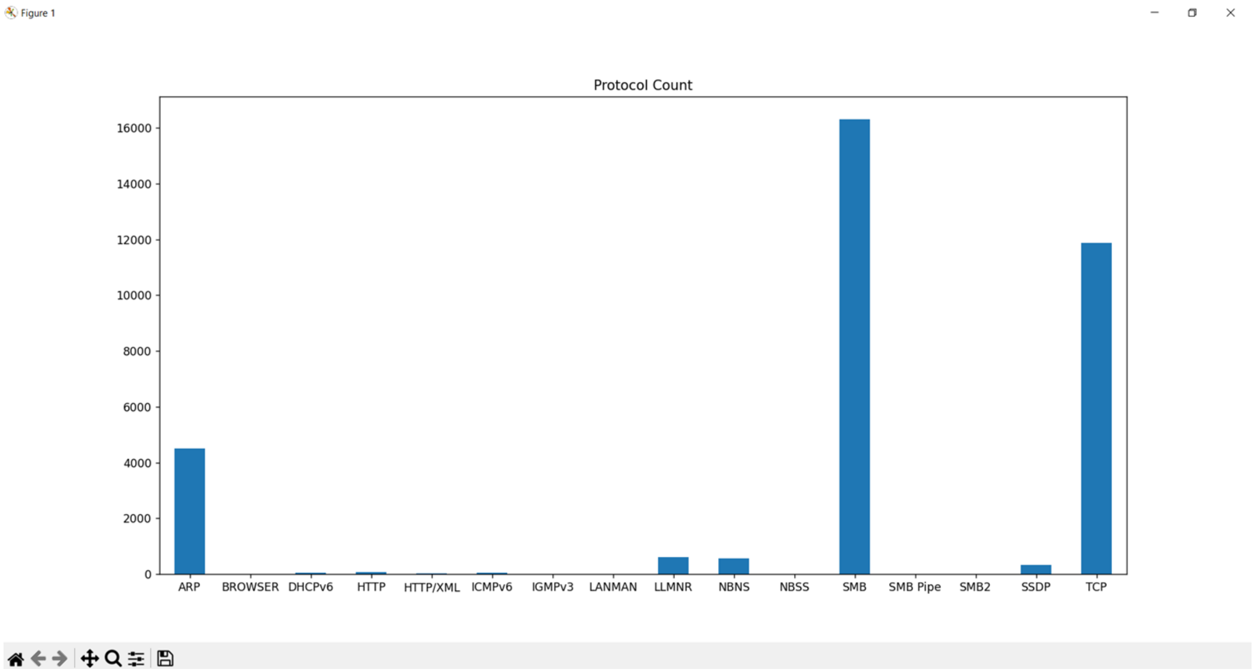


Task 3: Display a Bar Chart that shows "Protocol" on the x-axis, and "Count" on the y-axis. (2)

Hint: Search the pandas documentation for creating a Bar Chart from a DataFrame column.

In [4]:

```
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
import seaborn as sns

data = pd.read_csv('./T1_data/2022-task1_data.csv')
data

x=data['Protocol'].value_counts().sort_index()
print(x)
x.plot(kind='bar',rot=0,title="Protocol Count",figsize=(20,10)).grid(False)
plt.show()
```

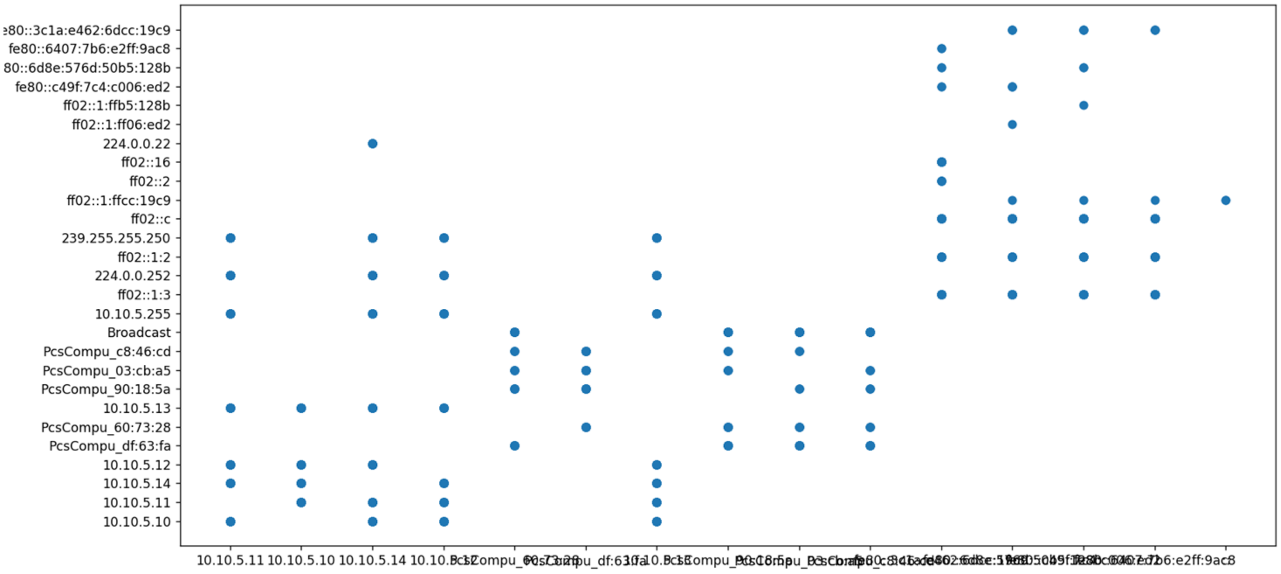


Task 4: Display a Scatter Chart that shows the association between Source and Destination data. (2)

Hint: Matplotlib has a scatterplot function that takes ***x*** *and* ***y*** *as inputs*

In [5]:

```
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
import seaborn as sns

data = pd.read_csv('./T1_data/2022-task1_data.csv')
data
x=data['Source'].tolist()
y=data['Destination'].tolist()
plt.scatter(x, y)
plt.show()
```



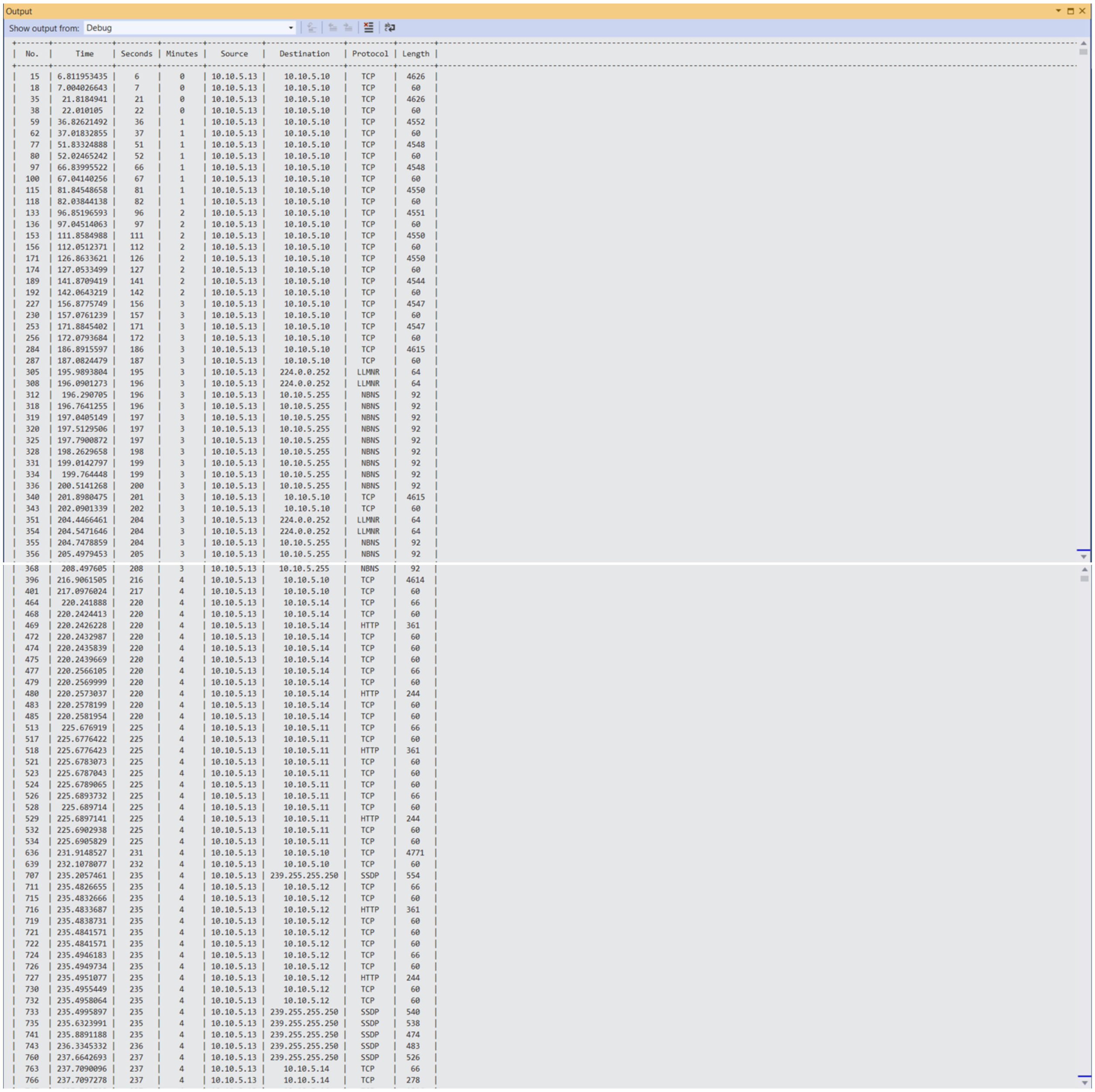
Task 5: Filter the data so that only 10.x.x.x Source addresses are included in a new DataFrame. (1)

Hint: Retrieve all rows where the Source string starts with 10.

In [6]:

```
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
import seaborn as sns
from prettytable import PrettyTable
import re

data = pd.read_csv('./T1_data/2022-task1_data.csv')
data

def valid_IP_Address(sample_str):
    ''' Returns True if given string is a
        valid IP Address, else returns False'''
    result = True
    match_obj = re.search( r"^(\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})$", sample_str)
    if  match_obj is None:
        result = False
    else:
        for value in match_obj.groups():
            if int(value) > 255:
                result = False
                break
    return result

def filter_ip(ip):
    if(valid_IP_Address(ip)):
        ip_new = list(map(int, ip.strip().split('.')[:2]))
        if ip_new[0] == 10:
            return True
cols = data.columns
new_List2= []
```

```
for i in range(len(cols)):
    new_List2.append(data.columns[i]) # Add header
myTable=PrettyTable(new_List2)

my_list = data['Source'].tolist()
my_list = list(set(my_list)) #get Unique list

new_list3=[]
for line in my_list:
    if  filter_ip(line):
        index1=data[data['Source'] == line].index
        for line2 in index1:
           myTable.add_row(data.loc[line2])
           new_list3.append(data.loc[line2])
print(myTable)

data1 = pd.DataFrame(new_list3, columns =["No.", "Time", "Seconds",
"Minutes","Source","Destination","Protocol","Length","Info"])
```



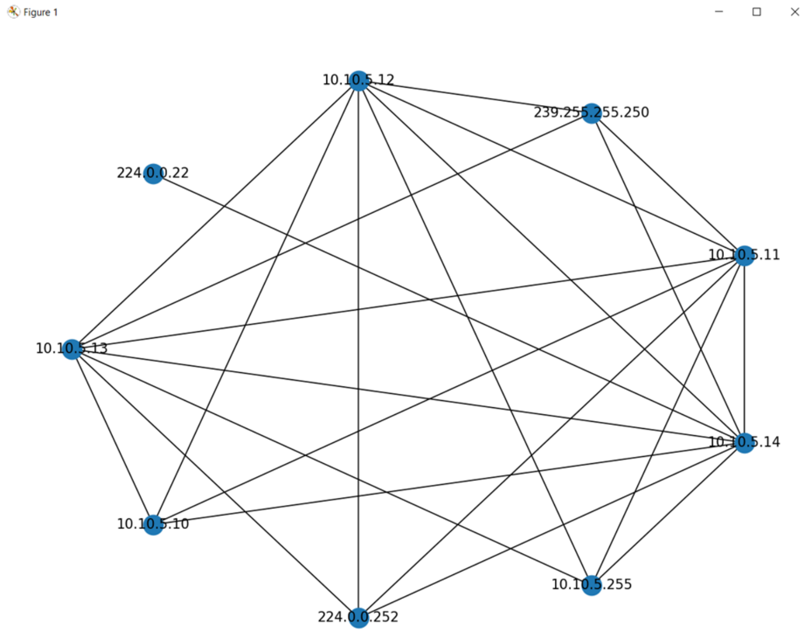
```
+-------+-------------+---------+---------+------------+-----------------+----------+--------+
|  No.  |     Time    | Seconds | Minutes |   Source   |   Destination   | Protocol | Length |
+-------+-------------+---------+---------+------------+-----------------+----------+--------+
|   15  | 6.811953435 |    6    |    0    | 10.10.5.13 |    10.10.5.10   |   TCP    |  4626  |
|   18  | 7.004026643 |    7    |    0    | 10.10.5.13 |    10.10.5.10   |   TCP    |   60   |
|   35  |  21.8184941 |   21    |    0    | 10.10.5.13 |    10.10.5.10   |   TCP    |  4626  |
|   38  |  22.010105  |   22    |    0    | 10.10.5.13 |    10.10.5.10   |   TCP    |   60   |
|   59  | 36.82621492 |   36    |    1    | 10.10.5.13 |    10.10.5.10   |   TCP    |  4552  |
|   62  | 37.01832855 |   37    |    1    | 10.10.5.13 |    10.10.5.10   |   TCP    |   60   |
|   77  | 51.83324888 |   51    |    1    | 10.10.5.13 |    10.10.5.10   |   TCP    |  4548  |
|   80  | 52.02465242 |   52    |    1    | 10.10.5.13 |    10.10.5.10   |   TCP    |   60   |
|   97  | 66.83995522 |   66    |    1    | 10.10.5.13 |    10.10.5.10   |   TCP    |  4548  |
|  100  | 67.04140256 |   67    |    1    | 10.10.5.13 |    10.10.5.10   |   TCP    |   60   |
|  115  | 81.84548658 |   81    |    1    | 10.10.5.13 |    10.10.5.10   |   TCP    |  4550  |
|  118  | 82.03844138 |   82    |    1    | 10.10.5.13 |    10.10.5.10   |   TCP    |   60   |
|  133  | 96.85196593 |   96    |    2    | 10.10.5.13 |    10.10.5.10   |   TCP    |  4551  |
|  136  | 97.04514063 |   97    |    2    | 10.10.5.13 |    10.10.5.10   |   TCP    |   60   |
|  153  | 111.8584988 |   111   |    2    | 10.10.5.13 |    10.10.5.10   |   TCP    |  4550  |
|  156  | 112.0512371 |   112   |    2    | 10.10.5.13 |    10.10.5.10   |   TCP    |   60   |
|  171  | 126.8633621 |   126   |    2    | 10.10.5.13 |    10.10.5.10   |   TCP    |  4550  |
|  174  | 127.0533499 |   127   |    2    | 10.10.5.13 |    10.10.5.10   |   TCP    |   60   |
|  189  | 141.8709419 |   141   |    2    | 10.10.5.13 |    10.10.5.10   |   TCP    |  4544  |
|  192  | 142.0643219 |   142   |    2    | 10.10.5.13 |    10.10.5.10   |   TCP    |   60   |
|  227  | 156.8775749 |   156   |    3    | 10.10.5.13 |    10.10.5.10   |   TCP    |  4547  |
|  230  | 157.0761239 |   157   |    3    | 10.10.5.13 |    10.10.5.10   |   TCP    |   60   |
|  253  | 171.8845402 |   171   |    3    | 10.10.5.13 |    10.10.5.10   |   TCP    |  4547  |
|  256  | 172.0793684 |   172   |    3    | 10.10.5.13 |    10.10.5.10   |   TCP    |   60   |
|  284  | 186.8915597 |   186   |    3    | 10.10.5.13 |    10.10.5.10   |   TCP    |  4615  |
|  287  | 187.0824479 |   187   |    3    | 10.10.5.13 |    10.10.5.10   |   TCP    |   60   |
|  305  | 195.9893804 |   195   |    3    | 10.10.5.13 |   224.0.0.252   |  LLMNR   |   64   |
|  308  | 196.0901273 |   196   |    3    | 10.10.5.13 |   224.0.0.252   |  LLMNR   |   64   |
|  312  |  196.290705 |   196   |    3    | 10.10.5.13 |   10.10.5.255   |   NBNS   |   92   |
|  318  | 196.7641255 |   196   |    3    | 10.10.5.13 |   10.10.5.255   |   NBNS   |   92   |
|  319  | 197.0405149 |   197   |    3    | 10.10.5.13 |   10.10.5.255   |   NBNS   |   92   |
|  320  | 197.5129506 |   197   |    3    | 10.10.5.13 |   10.10.5.255   |   NBNS   |   92   |
|  325  | 197.7900872 |   197   |    3    | 10.10.5.13 |   10.10.5.255   |   NBNS   |   92   |
|  328  | 198.2629658 |   198   |    3    | 10.10.5.13 |   10.10.5.255   |   NBNS   |   92   |
|  331  | 199.0142797 |   199   |    3    | 10.10.5.13 |   10.10.5.255   |   NBNS   |   92   |
|  334  |  199.764448 |   199   |    3    | 10.10.5.13 |   10.10.5.255   |   NBNS   |   92   |
|  336  | 200.5141268 |   200   |    3    | 10.10.5.13 |   10.10.5.255   |   NBNS   |   92   |
|  340  | 201.8980475 |   201   |    3    | 10.10.5.13 |    10.10.5.10   |   TCP    |  4615  |
|  343  | 202.0901339 |   202   |    3    | 10.10.5.13 |    10.10.5.10   |   TCP    |   60   |
|  351  | 204.4466461 |   204   |    3    | 10.10.5.13 |   224.0.0.252   |  LLMNR   |   64   |
|  354  | 204.5471646 |   204   |    3    | 10.10.5.13 |   224.0.0.252   |  LLMNR   |   64   |
|  355  | 204.7478859 |   204   |    3    | 10.10.5.13 |   10.10.5.255   |   NBNS   |   92   |
|  356  | 205.4979453 |   205   |    3    | 10.10.5.13 |   10.10.5.255   |   NBNS   |   92   |
|  368  |  208.497605 |   208   |    3    | 10.10.5.13 |   10.10.5.255   |   NBNS   |   92   |
|  396  | 216.9061505 |   216   |    4    | 10.10.5.13 |    10.10.5.10   |   TCP    |  4614  |
|  401  | 217.0976024 |   217   |    4    | 10.10.5.13 |    10.10.5.10   |   TCP    |   60   |
|  464  |  220.241888 |   220   |    4    | 10.10.5.13 |    10.10.5.14   |   TCP    |   66   |
|  468  | 220.2424413 |   220   |    4    | 10.10.5.13 |    10.10.5.14   |   TCP    |   60   |
|  469  | 220.2426228 |   220   |    4    | 10.10.5.13 |    10.10.5.14   |   HTTP   |  361   |
|  472  | 220.2432987 |   220   |    4    | 10.10.5.13 |    10.10.5.14   |   TCP    |   60   |
|  474  | 220.2435839 |   220   |    4    | 10.10.5.13 |    10.10.5.14   |   TCP    |   60   |
|  475  | 220.2439669 |   220   |    4    | 10.10.5.13 |    10.10.5.14   |   TCP    |   60   |
|  477  | 220.2566105 |   220   |    4    | 10.10.5.13 |    10.10.5.14   |   TCP    |   66   |
|  479  | 220.2569999 |   220   |    4    | 10.10.5.13 |    10.10.5.14   |   TCP    |   60   |
|  480  | 220.2573037 |   220   |    4    | 10.10.5.13 |    10.10.5.14   |   HTTP   |  244   |
|  483  | 220.2578199 |   220   |    4    | 10.10.5.13 |    10.10.5.14   |   TCP    |   60   |
|  485  | 220.2581954 |   220   |    4    | 10.10.5.13 |    10.10.5.14   |   TCP    |   60   |
|  513  |  225.676919 |   225   |    4    | 10.10.5.13 |    10.10.5.11   |   TCP    |   66   |
|  517  | 225.6776422 |   225   |    4    | 10.10.5.13 |    10.10.5.11   |   TCP    |   60   |
|  518  | 225.6776423 |   225   |    4    | 10.10.5.13 |    10.10.5.11   |   HTTP   |  361   |
|  521  | 225.6783073 |   225   |    4    | 10.10.5.13 |    10.10.5.11   |   TCP    |   60   |
|  523  | 225.6787043 |   225   |    4    | 10.10.5.13 |    10.10.5.11   |   TCP    |   60   |
|  524  | 225.6789065 |   225   |    4    | 10.10.5.13 |    10.10.5.11   |   TCP    |   60   |
|  526  | 225.6893732 |   225   |    4    | 10.10.5.13 |    10.10.5.11   |   TCP    |   66   |
|  528  |  225.689714 |   225   |    4    | 10.10.5.13 |    10.10.5.11   |   TCP    |   60   |
|  529  | 225.6897141 |   225   |    4    | 10.10.5.13 |    10.10.5.11   |   HTTP   |  244   |
|  532  | 225.6902938 |   225   |    4    | 10.10.5.13 |    10.10.5.11   |   TCP    |   60   |
|  534  | 225.6905829 |   225   |    4    | 10.10.5.13 |    10.10.5.11   |   TCP    |   60   |
|  636  | 231.9148527 |   231   |    4    | 10.10.5.13 |    10.10.5.10   |   TCP    |  4771  |
|  639  | 232.1078077 |   232   |    4    | 10.10.5.13 |    10.10.5.10   |   TCP    |   60   |
|  707  | 235.2057461 |   235   |    4    | 10.10.5.13 | 239.255.255.250 |   SSDP   |  554   |
|  711  | 235.4826655 |   235   |    4    | 10.10.5.13 |    10.10.5.12   |   TCP    |   66   |
|  715  | 235.4832666 |   235   |    4    | 10.10.5.13 |    10.10.5.12   |   TCP    |   60   |
|  716  | 235.4833687 |   235   |    4    | 10.10.5.13 |    10.10.5.12   |   HTTP   |  361   |
|  719  | 235.4838731 |   235   |    4    | 10.10.5.13 |    10.10.5.12   |   TCP    |   60   |
|  721  | 235.4841571 |   235   |    4    | 10.10.5.13 |    10.10.5.12   |   TCP    |   60   |
|  722  | 235.4841571 |   235   |    4    | 10.10.5.13 |    10.10.5.12   |   TCP    |   60   |
|  724  | 235.4946183 |   235   |    4    | 10.10.5.13 |    10.10.5.12   |   TCP    |   66   |
|  726  | 235.4949734 |   235   |    4    | 10.10.5.13 |    10.10.5.12   |   TCP    |   60   |
|  727  | 235.4951077 |   235   |    4    | 10.10.5.13 |    10.10.5.12   |   HTTP   |  244   |
|  730  | 235.4955449 |   235   |    4    | 10.10.5.13 |    10.10.5.12   |   TCP    |   60   |
|  732  | 235.4958064 |   235   |    4    | 10.10.5.13 |    10.10.5.12   |   TCP    |   60   |
|  733  | 235.4995897 |   235   |    4    | 10.10.5.13 | 239.255.255.250 |   SSDP   |  540   |
|  735  | 235.6323991 |   235   |    4    | 10.10.5.13 | 239.255.255.250 |   SSDP   |  538   |
|  741  | 235.8891188 |   235   |    4    | 10.10.5.13 | 239.255.255.250 |   SSDP   |  474   |
|  743  | 236.3345332 |   236   |    4    | 10.10.5.13 | 239.255.255.250 |   SSDP   |  483   |
|  760  | 237.6642693 |   237   |    4    | 10.10.5.13 | 239.255.255.250 |   SSDP   |  526   |
|  763  | 237.7090096 |   237   |    4    | 10.10.5.13 |    10.10.5.14   |   TCP    |   66   |
|  766  | 237.7097278 |   237   |    4    | 10.10.5.13 |    10.10.5.14   |   TCP    |  278   |
```

(Advanced) Task 6: Display a Node Link Diagram for this new DataFrame. (3)

Hint: Look at the NetworkX library: https://networkx.org/ and the online course notes.

In [7]:

```
import pylab as plt
from matplotlib.pyplot import figure
df1 = data1[['Source', 'Destination']]
G = nx.Graph()
G = nx.from_pandas_edgelist(df1, 'Source','Destination')
figure(figsize=(10, 8))
nx.draw_shell(G, with_labels=True)
plt.show()
```

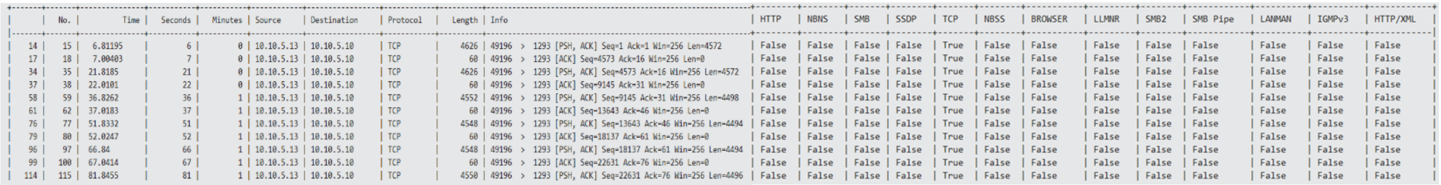


(Advanced) Task 7: For each Protocol type contained in this Dataframe, create a new Column and assign whether the Protocol usage is True or False (3)

Hint: Get a list of unique protocol values, assign each value to be a new column where the Protocol column is equal to the Protocol name.

In [8]:

```
from tabulate import tabulate
protocol = data1['Protocol'].tolist()
protocol = list(set(protocol))

for i in protocol:
   data1[i]='False'
   index2=data1[data1['Protocol'] == i].index
   for j in index2:
       data1.loc[j,i]='True'
print(tabulate(data1, headers='keys', tablefmt='psql'))
data1.to_csv('Files/new_dataset.csv')
```



```
+-----+-------+----------+-----------+-----------+------------+---------------+------------+----------+-------------------------------------------------------+--------+--------+-------+--------+-------+--------+-----------+---------+--------+------------+----------+----------+------------+
|     |   No. |     Time |   Seconds |   Minutes | Source     | Destination   | Protocol   |   Length | Info                                                  | HTTP   | NBNS   | SMB   | SSDP   | TCP   | NBSS   | BROWSER   | LLMNR   | SMB2   | SMB Pipe   | LANMAN   | IGMPv3   | HTTP/XML   |
|-----+-------+----------+-----------+-----------+------------+---------------+------------+----------+-------------------------------------------------------+--------+--------+-------+--------+-------+--------+-----------+---------+--------+------------+----------+----------+------------|
|  14 |    15 |  6.81195 |         6 |         0 | 10.10.5.13 | 10.10.5.10    | TCP        |     4626 | 49196  >  1293 [PSH, ACK] Seq=1 Ack=1 Win=256 Len=4572 | False  | False  | False | False  | True  | False  | False     | False   | False  | False      | False    | False    | False      |
|  17 |    18 |  7.00403 |         7 |         0 | 10.10.5.13 | 10.10.5.10    | TCP        |       60 | 49196  >  1293 [ACK] Seq=4573 Ack=16 Win=256 Len=0   | False  | False  | False | False  | True  | False  | False     | False   | False  | False      | False    | False    | False      |
|  34 |    35 | 21.8185  |        21 |         0 | 10.10.5.13 | 10.10.5.10    | TCP        |     4626 | 49196  >  1293 [PSH, ACK] Seq=4573 Ack=16 Win=256 Len=4572 | False  | False  | False | False  | True  | False  | False     | False   | False  | False      | False    | False    | False      |
|  37 |    38 | 22.0101  |        22 |         0 | 10.10.5.13 | 10.10.5.10    | TCP        |       60 | 49196  >  1293 [ACK] Seq=9145 Ack=31 Win=256 Len=0   | False  | False  | False | False  | True  | False  | False     | False   | False  | False      | False    | False    | False      |
|  58 |    59 | 36.8262  |        36 |         1 | 10.10.5.13 | 10.10.5.10    | TCP        |     4552 | 49196  >  1293 [PSH, ACK] Seq=9145 Ack=31 Win=256 Len=4498 | False  | False  | False | False  | True  | False  | False     | False   | False  | False      | False    | False    | False      |
|  61 |    62 | 37.0183  |        37 |         1 | 10.10.5.13 | 10.10.5.10    | TCP        |       60 | 49196  >  1293 [ACK] Seq=13643 Ack=46 Win=256 Len=0  | False  | False  | False | False  | True  | False  | False     | False   | False  | False      | False    | False    | False      |
|  76 |    77 | 51.8332  |        51 |         1 | 10.10.5.13 | 10.10.5.10    | TCP        |     4548 | 49196  >  1293 [PSH, ACK] Seq=13643 Ack=46 Win=256 Len=4494 | False  | False  | False | False  | True  | False  | False     | False   | False  | False      | False    | False    | False      |
|  79 |    80 | 52.0247  |        52 |         1 | 10.10.5.13 | 10.10.5.10    | TCP        |       60 | 49196  >  1293 [ACK] Seq=18137 Ack=61 Win=256 Len=0  | False  | False  | False | False  | True  | False  | False     | False   | False  | False      | False    | False    | False      |
|  96 |    97 | 66.84    |        66 |         1 | 10.10.5.13 | 10.10.5.10    | TCP        |     4548 | 49196  >  1293 [PSH, ACK] Seq=18137 Ack=61 Win=256 Len=4494 | False  | False  | False | False  | True  | False  | False     | False   | False  | False      | False    | False    | False      |
|  99 |   100 | 67.0414  |        67 |         1 | 10.10.5.13 | 10.10.5.10    | TCP        |       60 | 49196  >  1293 [ACK] Seq=22631 Ack=76 Win=256 Len=0  | False  | False  | False | False  | True  | False  | False     | False   | False  | False      | False    | False    | False      |
| 114 |   115 | 81.8455  |        81 |         1 | 10.10.5.13 | 10.10.5.10    | TCP        |     4550 | 49196  >  1293 [PSH, ACK] Seq=22631 Ack=76 Win=256 Len=4496 | False  | False  | False | False  | True  | False  | False     | False   | False  | False      | False    | False    | False      |
```

(Advanced) Task 8: Show a Multi-Line Chart that shows the Total Packet Length Per Protocol. (3)

Hint: Think about how you did this in Task 1 and Task 2, and recall that plt.plot can be used to append to a plot.

In [9]:

```
protocol2 = data1['Protocol'].tolist()
protocol2 = list(set(protocol2))
fig=plt.figure()
ax=plt.subplot(111)

for i in protocol2:
   x2=data1[data1['Protocol'] == i]
  # plt.plot(x2.Length,x2.Protocol)
   ax.plot(x2.Length,x2.Protocol)
   print(x2)
plt.show()
```

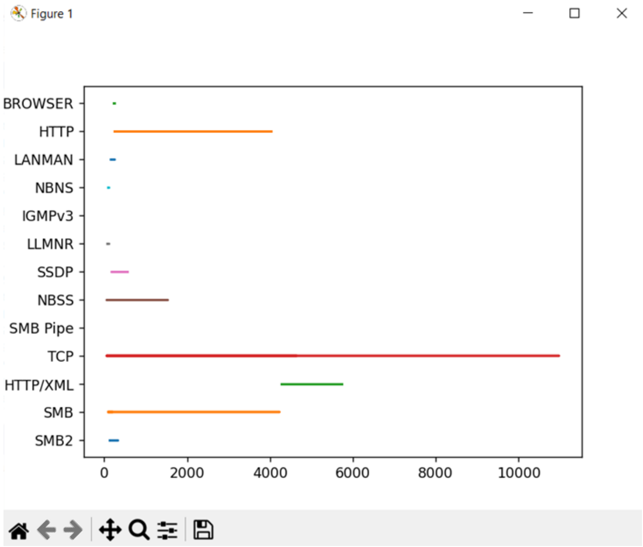


Figure 1
BROWSER
HTTP
LANMAN
NBNS
IGMPv3
LLMNR
SSDP
NBSS
SMB Pipe
TCP
HTTP/XML
SMB
SMB2
0
2000
4000
6000
8000
10000