Below are answers to the questions outlined in the assignment description. I primarily used the Python library pandas, but also used the library numpy for one question to make things a little easier.

My python scripts were written with Jupyter Notebooks. In my Blackboard submission, I have included the python notebook file as well as the python notebook file converted to a regular python file.

1. Average size of packets across all traffic captured.
   1. The average size of packets across all packets captured in the dataset is **768.18 bytes.**
   2. I calculated this number **by dividing the total sum of the number of bytes by the total sum of the number of packets. This yields the average size per packet.**
2. Complementary Cumulative Probability Distribution (CCDF)
   1. Flow durations Graphs:

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

* 1. Flow size graph (# of bytes):

Chart

Description automatically generatedChart, line chart

Description automatically generated

* 1. Flow size graph (# of packets):

Chart

Description automatically generatedChart, line chart

Description automatically generated

* 1. **Graph main features** – The main features of all of the graphs seem to be almost the same. The CCDF is very large at 1.0 probability at the beginning, then quickly tapers off to 0. From the logarithmic scale distribution, we can see that probability reduces, then tapers off evenly before dropping again to 0.
  2. **Logarithmic scale** – Graphing the CCDF with a logarithmic scale is very useful because it is a lot easier to see the change over time of the probability distribution. The logarithmic scale graph better highlights the changes over time.

1. Summary of traffic through router
   1. Table of top-10 port numbers by sender traffic volume

A screenshot of a cell phone

Description automatically generated

* 1. Likely applications for sender traffic –
     1. 80 – HTTP World Wide Web
     2. 443 – HTTP protocol over TLS/SSL
     3. 53 – Domain name server (DNS)
     4. 0 – reserved tcp/udp packets
     5. 25 – Simple Mail Transfer Protocol (SMTP)
     6. 22 – Secure Shell (SSH)
     7. 1935 – Macromedia Flash Communications Server MX
     8. 3074 – Xbox game port
     9. 3389 – MS WBT Server
     10. 2128 – Net Steward Control
  2. Table of top-10 port numbers by receiver traffic volume

A screenshot of a cell phone

Description automatically generated

* 1. Likely applications responsible for receiver traffic –
     1. 80 – HTTP World Wide Web
     2. 443 – HTTP protocol over TLS/SSL
     3. 53 – Domain name server (DNS)
     4. 445 – Microsoft-DS
     5. 25 – Simple Mail Transfer Protocol (SMTP)
     6. 123 – Network Time Protocol
     7. 1935 – Macromedia Flash Communications Server MX
     8. 3074 – Xbox game port
     9. 2048 – dls-monitor
     10. 0 – reserved tcp/udp packets
  2. **Significant differences** – Some significant differences I see are some unique top-10 ports among senders and receivers. Some unique services in Sender traffic are port 22 (Secure Shell), port 3389 (MS WBT Server), and port 2128 (Net Steward Control). Some unique services in receiver traffic include port 445 (Microsoft DS), port 123 (Network Time Protocol), and port 2048 (dls-monitor)

1. Traffic volumes based on source IP prefix
   1. Including source mask length 0:
      1. Percentage of traffic from most popular 0.1% of IP Prefixes: **59%**
      2. Percentage of traffic from most popular 1% of IP Prefixes: **82%**
      3. Percentage of traffic from most popular 10% of IP Prefixes: **98%**
   2. Percentage of traffic with source mask length 0: **43%**
   3. Excluding source mask length 0:
      1. Percentage of traffic from most popular 0.1% of IP Prefixes: **38%**
      2. Percentage of traffic from most popular 1% of IP Prefixes: **64%**
      3. Percentage of traffic from most popular 10% of IP Prefixes: **95%**
   4. Institute A with 128.112.0.0/16 address block:
      1. Percentage of traffic sent by A: **0.7%**
      2. Percentage of data sent to A: **2.2%**

Python Script (Jupyter Notebook converted to python file):

# To add a new cell, type '# %%'

# To add a new markdown cell, type '# %% [markdown]'

# %% [markdown]

# # Assignment # 5: Netflow Packet Analysis

# %%

# Necessary imports

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import csv

# %%

# Read network data into dataframe

network\_data = pd.read\_csv('Netflow\_dataset.csv')

# %% [markdown]

# ## a) Average Packet sizes

# %%

# a. Average size of the packets across all the traffic captured in the dataset

num\_packets = network\_data['dpkts'].sum()

num\_bytes = network\_data['doctets'].sum()

avg\_packet\_size = num\_bytes/num\_packets

print("Average Packet size:", avg\_packet\_size)

# %% [markdown]

# ## b) Complementary Cumulative Probability Distribution (CCDF)

# %%

# Calculate flow durations

network\_data['duration'] = network\_data['last'] - network\_data['first']

# %%

# Plot CCDF of durations

plt.hist(network\_data['duration'], bins=50, density=True, histtype='step', cumulative=-1)

# Set labels

plt.xlabel('Duration')

plt.ylabel('Probability')

plt.title('Duration CCDF')

plt.show()

# %%

# Plot CCDF of durations with log scale

plt.hist(network\_data['duration'], bins=50, density=True, histtype='step', cumulative=-1, log=True)

# Set labels

plt.xlabel('Duration')

plt.ylabel('Probability')

plt.title('Duration CCDF (log scale)')

plt.show()

# %%

# Plot CCDF of durations with log scale

plt.hist(network\_data['doctets'], bins=50, density=True, histtype='step', cumulative=-1)

# Set labels

plt.xlabel('# of bytes')

plt.ylabel('Probability')

plt.title('# of bytes CCDF')

plt.show()

# %%

# Plot CCDF of durations with log scale

plt.hist(network\_data['doctets'], bins=50, density=True, histtype='step', cumulative=-1, log=True)

# Set labels

plt.xlabel('# of bytes')

plt.ylabel('Probability')

plt.title('# of bytes CCDF (log scale)')

plt.show()

# %%

# Plot CCDF of durations with log scale

plt.hist(network\_data['dpkts'], bins=50, density=True, histtype='step', cumulative=-1)

# Set labels

plt.xlabel('# of packets')

plt.ylabel('Probability')

plt.title('# of packets CCDF')

plt.show()

# %%

# Plot CCDF of durations with log scale

plt.hist(network\_data['dpkts'], bins=50, density=True, histtype='step', cumulative=-1, log=True)

# Set labels

plt.xlabel('# of packets')

plt.ylabel('Probability')

plt.title('# of packets CCDF (log scale)')

plt.show()

# %% [markdown]

# ## c) Kind of traffic flowing through router

# %%

# We only need these 2 colums

sender\_traffic = network\_data[['srcport', 'doctets']]

# Create table of port frequencies

src\_port\_frequency = sender\_traffic['srcport'].value\_counts().head(10).to\_frame(name='frequency')

# Create table of port usage in bytes

src\_port\_bytes = sender\_traffic.groupby(['srcport']).sum()

# Add percentage column to port usage by byte

src\_port\_bytes['doctets\_perc'] = (src\_port\_bytes['doctets'] / src\_port\_bytes['doctets'].sum()) \* 100

# Merge tables

src\_port\_frequency.merge(src\_port\_bytes, left\_index=True, right\_on='srcport')

# %%

# We only need these 2 colums

receiver\_traffic = network\_data[['dstport', 'doctets']]

# Create table of port frequencies

dst\_port\_frequency = receiver\_traffic['dstport'].value\_counts().head(10).to\_frame(name='frequency')

# Create table of port usage in bytes

dst\_port\_bytes = receiver\_traffic.groupby(['dstport']).sum()

# Add percentage column to port usage by byte

dst\_port\_bytes['doctets\_perc'] = (dst\_port\_bytes['doctets'] / dst\_port\_bytes['doctets'].sum()) \* 100

# Merge tables

dst\_port\_frequency.merge(dst\_port\_bytes, left\_index=True, right\_on='dstport')

# %% [markdown]

# ## d) Traffic volumes based on source IP prefix

# %%

def top\_percents(src\_addr\_volume):

dec\_perc = 0.001

for dec\_perc in [0.001, 0.01, 0.1]:

percentage = src\_addr\_volume.iloc[int(len(src\_addr\_volume) \* dec\_perc)]['doctets\_cumperc']

print("Top", dec\_perc \* 100, "\b% of IP addresses:", percentage, "\b% of all bytes")

dec\_perc \*= 10

# %%

# Get source IP addresses bytes used by each

src\_addr\_volume = network\_data[['srcaddr', 'doctets']].groupby('srcaddr').sum().sort\_values(by='doctets', ascending=False)

# Add percentage of total bytes that each IP address uses

src\_addr\_volume['doctets\_perc'] = (src\_addr\_volume['doctets'] / src\_addr\_volume['doctets'].sum()) \* 100

# A cummulative percentage column

src\_addr\_volume['doctets\_cumperc'] = src\_addr\_volume['doctets\_perc'].cumsum()

# Get byte percentage by top 0.1%, 1%, and 10% of IP addresses

top\_percents(src\_addr\_volume)

# %%

# Get byte volume by source mask

src\_mask\_volume = network\_data[['src\_mask', 'doctets']].groupby('src\_mask').sum().sort\_index()

# Add percentage of whole bytes for each mask

src\_mask\_volume['doctets\_perc'] = (src\_mask\_volume['doctets'] / src\_mask\_volume['doctets'].sum()) \* 100

# Get the 0 mask entry

mask\_length\_zero\_perc = src\_mask\_volume.query('src\_mask == 0')['doctets\_perc'][0]

# Print the percentage

print("Percentage of traffic with source mask of 0:", mask\_length\_zero\_perc, "\b%")

# %%

# Get source IP addresses bytes used by each, but exclude 0 src\_mask

src\_addr\_volume = network\_data[['srcaddr', 'doctets', 'src\_mask']].query('src\_mask != 0').groupby('srcaddr').sum().sort\_values(by='doctets', ascending=False)

# Add percentage of total bytes that each IP address uses

src\_addr\_volume['doctets\_perc'] = (src\_addr\_volume['doctets'] / src\_addr\_volume['doctets'].sum()) \* 100

# A cummulative percentage column

src\_addr\_volume['doctets\_cumperc'] = src\_addr\_volume['doctets\_perc'].cumsum()

# Get byte percentage by top 0.1%, 1%, and 10% of IP addresses

print("Excluding 0 source masks...")

top\_percents(src\_addr\_volume)

# %% [markdown]

# ## e) Institution with 128.112.0.0/16 address block

# %%

# Grab source addresses, # of packets, and # of bytes

inst\_data = network\_data[['srcaddr', 'dpkts', 'doctets']]

# True if source address is in address block 128.112

inst\_data['from\_institution'] = inst\_data['srcaddr'].str.startswith('128.112')

# Find # of packets and # of bytes sent by in and out of institution

inst\_data = inst\_data.groupby('from\_institution').sum()

inst\_data['dpkts\_perc'] = (inst\_data['dpkts'] / inst\_data['dpkts'].sum()) \* 100

inst\_data['doctets\_perc'] = (inst\_data['doctets'] / inst\_data['doctets'].sum()) \* 100

print('Traffic sent from institution...')

inst\_data

# %%

# Grab source addresses, # of packets, and # of bytes

inst\_data = network\_data[['dstaddr', 'dpkts', 'doctets']]

# True if source address is in address block 128.112

inst\_data['to\_institution'] = inst\_data['dstaddr'].str.startswith('128.112')

# Find # of packets and # of bytes sent by in and out of institution

inst\_data = inst\_data.groupby('to\_institution').sum()

inst\_data['dpkts\_perc'] = (inst\_data['dpkts'] / inst\_data['dpkts'].sum()) \* 100

inst\_data['doctets\_perc'] = (inst\_data['doctets'] / inst\_data['doctets'].sum()) \* 100

print('Traffic Sent to institution...')

inst\_data

Execution Sample:

**Included as separate PDF attachment, filename:** macik\_erik\_assign5\_execution.pdf

References:

Volz, B. (2020). Service Name and Transport Protocol Port Number Registry. Retrieved November 28, 2020, from https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml