Student Name : Toh Kok Soon

Group : SCSI

Date : 15/10/2025

LAB 4: ANALZING NETWORK DATA LOG

You are provided with the data file, in .csv format, in the working directory. Write the program to extract the following informations.

EXERCISE 4A: TOP TALKERS AND LISTENERS

One of the most commonly used function in analyzing data log is finding out the IP address of the hosts that send out large amount of packet and hosts that receive large number of packets, usually know as TOP TALKERS and LISTENERS. Based on the IP address we can obtained the organization who owns the IP address.

List the TOP 5 TALKERS

Rank	IP address	# of packets	Organisation
1	13.107.4.50	5960	MSFT
2	130.14.250.7	4034	NLM-ETHER
3	155.69.160.38	3866	NTUNET1
4	171.67.77.19	2656	NETBLK-SUNET
5	155.69.199.255	2587	NTUNET1

TOP 5 LISTENERS

Rank	IP address	# of packets	Organisation
1	137.132.228.33	5908	NUSNET
2	192.122.131.36	4662	A-STAR-AS-AP
3	202.51.247.133	4288	NUSGP
4	137.132.228.29	4022	NUSNET
5	103.37.198.100	3741	A-STAR-AS-AP

EXERCISE 4B: TRANSPORT PROTOCOL

Using the IP protocol type attribute, determine the percentage of TCP and UDP protocol

	Header value	Transport layer protocol	# of packets
1	6	TCP	137707
2	17	UDP	36852
3			

TCP: 78.89%
UDP: 21.11%

EXERCISE 4C: APPLICATIONS PROTOCOL

Using the Destination IP port number determine the most frequently used application protocol. (For finding the service given the port number https://www.adminsub.net/tcp-udp-port-finder/)

Rank	Destination IP port number	# of packets	Service
1	443	43208	HTTPS
2	80	11018	HTTP
3	50930	2450	Dynamic Port
4	15000	2103	Unknown Port
5	8160	1354	Unknown (Patrol by search)

EXERCISE 4D: TRAFFIC

The traffic intensity is an important parameter that a network engineer needs to monitor closely to determine if there is congestion. You would use the IP packet size to calculate the estimated total traffic over the monitored period of 15 seconds. (Assume the sampling rate is 1 in 2048)

Total Traffic(MB)	331903.809 MB

EXERCISE 4E: ADDITIONAL ANALYSIS

Please append ONE page to provide additional analysis of the data and the insight it provides.
Examples include:
Top 5 communication pairs;
Visualization of communications between different IP hosts;
etc.

Please limit your results within one page (and any additional results that fall beyond one page limit will not be assessed).

EXERCISE 4F: SOFTWARE CODE

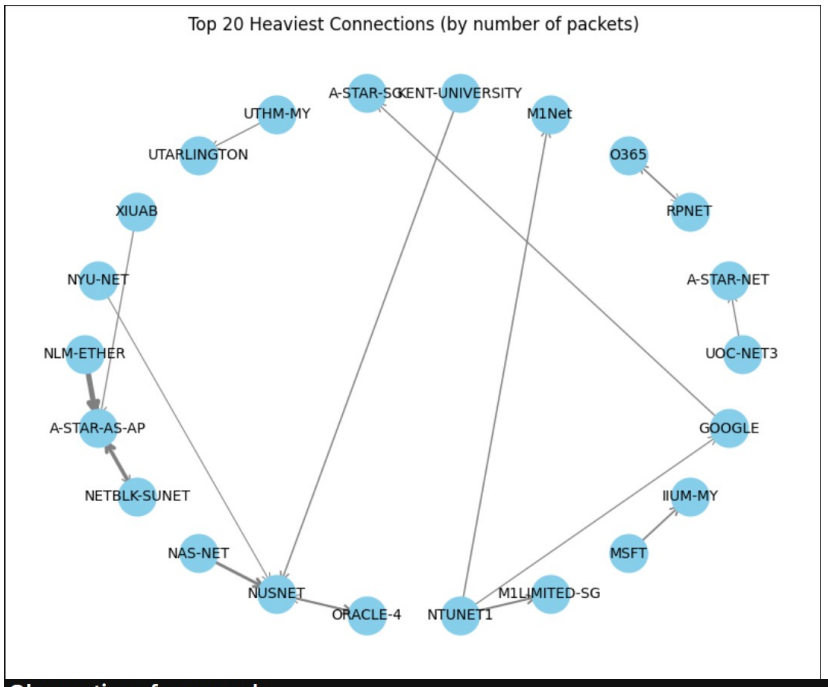
Please also submit your code to the NTULearn lab site.

Additional Analysis

Top 5 Connection Pairs

	src_IP	dst_IP	Number of Packets	From	To
0	130.14.250.7	103.37.198.100	3739	NLM-ETHER	A-STAR-AS-AP
1	171.67.77.19	192.122.131.36	2656	NETBLK-SUNET	A-STAR-AS-AP
2	129.99.230.54	137.132.22.74	2097	NAS-NET	NUSNET
3	137.132.228.42	137.131.17.212	1553	NUSNET	ORACLE-4
4	155.69.252.133	138.75.242.36	1475	NTUNET1	M1LIMITED-SG

Visualisation of Top 20 Heaviest Connections (by number of Packets)



Observations from graph

1. The graph shows most connection are mostly unidirectional
2. There's no obvious central hub
3. The graph is very sparse with some of the connection being only between the pair
4. A-STAR-AS-AP is the most heavy in traffic by number of packets which can be seen by the thickness of the connection
5. Most of these nodes seem to belong to universities, software services as well as telcoms

Insights

The observed network is likely to be a network of research organisation since it consists mostly of universities and research organisations. The many isolated pairs and dominance of unidirectional traffic suggest the observed network connections are mostly peer to peer. Futhermore, no obvious central hub could be seen from the graph, therefore reinforcing the idea that this is a mostly peer to peer network. In a client-server network, we would expect heavy traffic on one of the nodes in terms of both number of packets as well as nuber of connections.

Learnings

1. We can gain deeper insights into network through visualisation like network graphs
2. Whois lookup is slow and is a bottleneck when it comes to the analysis of the network log, most operation can be done straight from the IP address so we should only do lookups when necessary such as in the final visualisations, this can be further mitigated through caching in a dictionary map to avoid re-lookups.

CODE(Refer to .ipynb file for the actual implementation)

```
import pandas as pd
from ipwhois import IPWhois


# ## Helper Functions


# Identify the organisation through IP address
ip_org_mapping = {}

def get_organisation(ip_addr):
    if ip_addr in ip_org_mapping:
        return ip_org_mapping[ip_addr]
    else:
        ip = IPWhois(ip_addr)
        result = ip.lookup_rdap()
        ip_org_mapping[ip_addr] = result.get('network', {}).get('name')
    return result.get('network', {}).get('name')

# Identify common ports
def get_service_name(port):
    port_mapping = {
        20: "FTP Data",
        21: "FTP Control",
        22: "SSH",
        23: "Telnet",
        25: "SMTP",
        53: "DNS",
        67: "DHCP Server",
        68: "DHCP Client",
        69: "TFTP",
        80: "HTTP",
        110: "POP3",
        123: "NTP",
        143: "IMAP",
        161: "SNMP",
        194: "IRC",
        389: "LDAP",
        443: "HTTPS",
        465: "SMTPS",
        514: "Syslog",
        515: "LPD",
        587: "SMTP (Submission)",
        636: "LDAPS",
        993: "IMAPS",
        995: "POP3S",
        1433: "MS SQL",
        1521: "Oracle",
        1723: "PPTP",
        3306: "MySQL",
        3389: "RDP",
        5060: "SIP",
        5432: "PostgreSQL",
        5900: "VNC",
        6379: "Redis",
        8080: "HTTP-Alt",
        8443: "HTTPS-Alt",
        8888: "Alternative HTTP",
```

```
        9000: "Custom/Development",
        27017: "MongoDB",
        25565: "Minecraft"
    }

    if port in port_mapping:
        return port_mapping[port]
    else:
        return "Dynamic/Unknown Port"


# ## Read in Data


columns = [
    "Type",
    "sflow_agent_address",
    "inputPort",
    "outputPort",
    "src_MAC",
    "dst_MAC",
    "ethernet_type",
    "in_vlan",
    "out_vlan",
    "src_IP",
    "dst_IP",
    "IP_protocol",
    "ip_tos",
    "ip_ttl",
    "src_port",
    "dst_port",
    "tcp_flags",
    "packet_size",
    "IP_size",
    "sampling_rate"
]

df = pd.read_csv("Data_2.csv",header=None,names=columns)
df = df[df["Type"]=="FLOW"] # Filter to only FLOW type
print(df.dtypes)
df.head()


# ## Top 5 Talkers


top_talker = df.dropna(subset=['src_IP', 'dst_IP']) # Drop data with no src and dst ip
top_talker = top_talker['src_IP'].value_counts().head(5)

# Convert Back to dataframe
top_talker = top_talker.reset_index()
top_talker.columns = ['src_IP', 'Number of Packets'] # Rename column

# Get organisations
top_talker['organization'] = top_talker['src_IP'].apply(get_organisation)


# In[6]:
```

```
top_talker.head()


# ## Top 5 Listener

top_listener = df.dropna(subset=['src_IP', 'dst_IP']) # Drop data with no src and dst ip
top_listener = top_listener['dst_IP'].value_counts().head(5)

# Convert Back to dataframe
top_listener = top_listener.reset_index()
top_listener.columns = ['dst_IP', 'Number of Packets'] # Rename column

# Get organisations
top_listener['organization'] = top_listener['dst_IP'].apply(get_organisation)


top_listener.head()


# ## Top 5 Applications


top_apps = df['dst_port']
top_apps = top_apps.dropna()# Drop data with no dst port
top_apps = top_apps.value_counts().head(5)

# Convert Back to Dataframe
top_apps = top_apps.reset_index()
top_apps.columns = ['dst_port','Number of Packets']

# Get Service Name
top_apps["Service Name"] = top_apps['dst_port'].apply(get_service_name)


top_apps.head()


# ## Total Traffic

total_traffic = sum(df['IP_size'])
total_traffic_Mb = (total_traffic * 2048) / (1* pow(2, 20))
print(f"Total Traffic (MB) = {total_traffic_Mb:.3f} MB")


# ## Proportion of TCP and UDP packets

protocol_counts = df[(df['IP_protocol'] == 6) | (df['IP_protocol'] == 17)]
protocol_counts = protocol_counts['IP_protocol'].value_counts()

protocol_counts = protocol_counts.reset_index()
protocol_counts.columns = ['Protocol', 'Count']

protocol_map = {6: 'TCP', 17: 'UDP'}
protocol_counts['Protocol Name'] = protocol_counts['Protocol'].map(protocol_map)

protocol_counts.head()


total_count = protocol_counts['Count'].sum()
for index, row in protocol_counts.iterrows():
```

```
    proportion = row['Count'] / total_count
    print(f"{row['Protocol Name']}: {proportion:.2%}")


# # Additional Analysis

# ## Top 5 communication pair

top_communication_pair = df.dropna(subset=['src_IP', 'dst_IP']) # Drop data with no src and
dst ip

# Count occurrences of each src-dst pair
top_communication_pair = (
    top_communication_pair
    .groupby(['src_IP', 'dst_IP'])
    .size()
    .reset_index(name='Number of Packets')
    .sort_values(by='Number of Packets', ascending=False)
    .head(5)
)

top_communication_pair['From'] = top_communication_pair['src_IP'].apply(get_organisation)
top_communication_pair['To'] = top_communication_pair['dst_IP'].apply(get_organisation)

top_communication_pair = top_communication_pair.reset_index(drop=True)
top_communication_pair.head()


# ## Visualizing the communication between different IP hosts.

communication_pairs = df.dropna(subset=['src_IP', 'dst_IP']) # Drop data with no src and dst
ip

# Count occurrences of each src-dst pair
communication_pairs = (
    communication_pairs
    .groupby(['src_IP', 'dst_IP'])
    .size()
    .reset_index(name='Number of Packets')
    .sort_values(by='Number of Packets', ascending=False)
    .head(20)
)

communication_pairs.head()

communication_pairs['From'] = communication_pairs['src_IP'].apply(get_organisation)
communication_pairs['To'] = communication_pairs['dst_IP'].apply(get_organisation)

communication_pairs = communication_pairs.reset_index(drop=True)
communication_pairs.head()


import networkx as nx
import matplotlib.pyplot as plt

G = nx.DiGraph()

# Add edges
for _, row in communication_pairs.iterrows():
    src = row['From']
```

```
    dst = row['To']
    weight = row['Number of Packets']
    G.add_edge(src, dst, weight=weight)


plt.figure(figsize=(10, 8))

pos = nx.shell_layout(G)

nx.draw_networkx_nodes(G, pos, node_size=700, node_color='skyblue')

edges = G.edges(data=True)
nx.draw_networkx_edges(
    G, pos,
    edgelist=edges,
    width=[edge[2]['weight'] / 1000 for edge in edges],
    arrowstyle='->',
    arrowsize=15,
    edge_color='gray'
)

# Draw labels
nx.draw_networkx_labels(G, pos, font_size=10)

plt.title("Top 20 Heaviest Connections (by number of packets)")
plt.axis('off')
plt.show()
```