Student Name : ___Nguyen An Loc______

Group : ______SCSI___________

Date : _____15/10/2025________

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.

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	4228	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	Percentage (%)
1	6	TCP	137707	77.698723
2	17	UDP	36852	20.793085

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 and/or Private Ports
4	15000	2103	Hypack Data Aquisition
5	8160	1354	Patrol

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

1. *Top 10 communication pairs*

	src_IP	Source Organisation	dst_IP	Destination Organisation	# of packets
0	130.14.250.7	NLM-ETHER	103.37.198.100	A-STAR-AS-AP	3739
1	171.67.77.19	NETBLK-SUNET	192.122.131.36	A-STAR-AS-AP	2656
2	129.99.230.54	NAS-NET	137.132.22.74	NUSNET	2097
3	137.132.228.42	NUSNET	137.131.17.212	ORACLE-4	1553
4	155.69.252.133	NTUNET1	138.75.242.36	M1LIMITED-SG	1475
5	13.107.4.50	MSFT	210.48.222.13	IIUM-MY	1289
6	155.69.203.215	NTUNET1	74.125.10.7	GOOGLE	1257
7	192.170.233.203	UOC-NET3	202.6.241.101	A-STAR-NET	1196
8	202.21.159.246	RPNET	104.146.199.27	O365	1143
9	155.69.59.250	NTUNET1	175.156.89.102	M1Net	1068

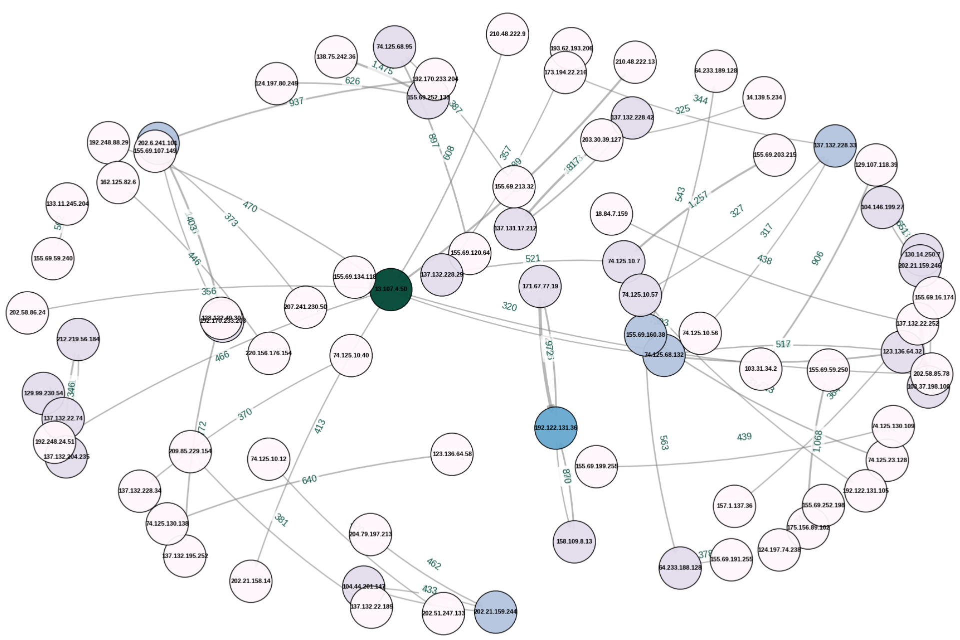
The top 10 communication pairs indicate that most data exchanges occur between research and academic networks such as *NTUNET1, NUSNET, NAS-NET, and A-STAR-AS-AP*. *A*STAR* appears as the main recipient, showing strong research collaboration with regional universities. *NTUNET1* frequently connects with multiple institutions and cloud providers like Microsoft and Google, reflecting heavy use of cloud-based services. Overall, the traffic pattern highlights that a few key institutional & cloud links dominate the network's total packet flow.

2. *Visualizing the communication between 60 IP hosts*

The directed-graph illustrates communication among 100 IP hosts, where each node represents a host and arrows indicate data flow directions. Edge labels show the number of packets exchanged, revealing that a few nodes act as central hubs with heavier traffic. Hosts such as 13.107.4.50 and 192.122.131.36 show higher connectivity, linking to many others. The overall structure highlights a clustered yet interconnected network, where major academic and

service nodes dominate packet transmission across the system.

Visualization of Communication between 60 IP Hosts
(Edge labels = No. of Packets)



EXERCISE 4F: SOFTWARE CODE

```
%pip install ipwhois
%pip install networkx

import pandas as pd
import numpy as np
from ipwhois import IPWhois
import networkx as nx
import matplotlib.pyplot as plt
import plotly.express as px
```

```python
# Define column format as per Table 1 in the manual
# There is an extra comma leading to a csv with 1 additional column
format = ['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',
        'udp_src_port', 'udp_dst_port', 'tcp_flags',
        'packet_size', 'IP_size', 'sampling_rate', 'extra']

# Read CSV file
log_df = pd.read_csv('Data_2.csv', header=None, names=format)

# Remove extra blank column
log_df.drop('extra', axis=1, inplace=True)

# Preview first 10 rows
log_df.head(10)
```

4A: TOP TALKERS AND LISTENERS

```python
# Find organizations name based on IP address
def find_org(ip):
    ip = IPWhois(ip)
    result = ip.lookup_rdap()
    return result.get('network', {}).get('name')
# Find top 5 talkers (source IPs with most packets sent)
top_talkers = log_df['src_IP'].value_counts().nlargest(5).to_frame()

# Lookup organisations
org = []
for ip in top_talkers.index:
    org.append(find_org(ip))

top_talkers['Organisations'] = org
top_talkers = top_talkers.reset_index().rename(columns = {'src_IP':'IP 
Address',
                                                        'count':'# of 
packets'})
top_talkers
# Find top 5 listeners (destination IPs with most packets received)
top_listener = log_df['dst_IP'].value_counts().nlargest(5).to_frame()

org = []
for ip in top_listener.index:
    org.append(find_org(ip))

top_listener['Organisations'] = org
top_listener = top_listener.reset_index().rename(columns = {'dst_IP':'IP 
Address',
                                                          'count':'# of 
packets'})

top_listener
```

4B: TRANSPORT PROTOCOL

```
packet_df = log_df['IP_protocol'].value_counts().to_frame()
packet_df = packet_df.reset_index().rename(columns={'IP_protocol':'Header Value',
                                                    'count':'# of packets'})

# Calculate protocol percentages
percentage = []
for i in range(len(packet_df)):
    percentage.append(packet_df['# of packets'][i] * 100 / len(log_df))

packet_df['Percentage'] = percentage
packet_df
# Given IP_Protocol 6 = TCP, IP_Protocol 17 = UDP
tcp_packet_df = packet_df.loc[packet_df['Header Value'] == 6]
udp_packet_df = packet_df.loc[packet_df['Header Value'] == 17]
frames = [tcp_packet_df, udp_packet_df]
final_df = pd.concat(frames)
final_df
```

4C: APPLICATIONS PROTOCOL

```
# Top 5 destination ports
dest_port_df = log_df['udp_dst_port'].value_counts().nlargest(5).to_frame()
dest_port_df =
dest_port_df.reset_index().rename(columns={'udp_dst_port':'Destination IP port number',
                                                          'count':'# of packets',})
dest_port_df
```

4D: TRAFFIC

```
# Sum up the total IP data size in bytes
total_traffic = sum(log_df['IP_size'])

# bytes → MB
total_traffic_MB = total_traffic / (2**20)

# Multiply sampling rate (2048) to get total real traffic
total_traffic_MB *= 2048

print(f"Total Traffic (MB) = {total_traffic_MB:.3f} MB")
```

4E: ADDITIONAL ANALYSIS

1. **Top 10 communication pairs**

```
# Top 10 unique communication pairs
unique_comm_pairs_df = (
    log_df.groupby(['src_IP', 'dst_IP'])
    .size()
    .sort_values(ascending = False)
    .to_frame(name = '# of packets')
    .reset_index()
)
top_comm_df = unique_comm_pairs_df.head(10)
top_comm_df

# Lists for organisation lookups
src_org = []
dst_org = []

for i in range(10):
    src_org.append(find_org(top_comm_df['src_IP'][i]))
    dst_org.append(find_org(top_comm_df['dst_IP'][i]))

top_comm_df['Source Organisation'] = src_org
top_comm_df['Destination Organisation'] = dst_org

top_comm_df = top_comm_df.reindex(
    columns = [
        'src_IP',
        'Source Organisation',
        'dst_IP',
        'Destination Organisation',
        '# of packets'
    ]
)
# Display
top_comm_df
```

2. Visualization

```
# Top 60 communication pairs
subset_df = unique_comm_pairs_df.head(60)

# Directed graph with packet count
net = nx.from_pandas_edgelist(
    subset_df,
    source = "src_IP",
    target = "dst_IP",
    edge_attr = "# of packets",
    create_using = nx.DiGraph()
)

plt.figure(figsize = (20, 14))
plt.style.use("seaborn-v0_8-whitegrid")

# Layout
```

```python
layout = nx.spring_layout(net, k = 1.3, iterations = 140, seed = 24)

# Node size
fixed_node_size = 3000

# Compute packet weights
edges = net.edges()
weights = [net[u][v].get("# of packets", 1) for u, v in edges]
max_w = max(weights) if weights else 1

# Compute degree centrality for coloring
centrality = nx.degree_centrality(net)
color_values = [centrality[node] for node in net.nodes()]

# Draw network edges
nx.draw_networkx_edges(
    net, layout,
    edge_color = "#7D7D7D",
    alpha = 0.55,
    arrows = True,
    arrowsize = 16,
    width = [1.2 + (w / max_w) * 4 for w in weights],
    connectionstyle = "arc3,rad=0.07",
    min_source_margin = 18,
    min_target_margin = 18
)

# --- Draw nodes (teal-purple gradient instead of yellow-red) ---
nx.draw_networkx_nodes(
    net, layout,
    node_size = fixed_node_size,
    node_color = color_values,
    cmap = "PuBuGn",    # distinct color map
    edgecolors = "black",
    linewidths = 1.1,
    alpha = 0.95
)

# Add IP address labels
nx.draw_networkx_labels(
    net, layout,
    font_size = 8,
    font_color = "black",
    font_weight = "semibold"
)

# Create and draw edge labels (packet counts)
edge_labels = {(u, v): f"{net[u][v]['# of packets']:,}" for u, v in
net.edges()}
label_pos = {n: (x, y + 0.03) for n, (x, y) in layout.items()}
```

```
nx.draw_networkx_edge_labels(
    net, label_pos,
    edge_labels = edge_labels,
    font_size = 12,
    font_color = "#0B5345",  # dark green
    bbox = dict(boxstyle = "round,pad=0.25", facecolor = "white", alpha =
0.7, edgecolor = "none")
)

plt.title(
    "Visualization of Communication between 60 IP Hosts\n(Edge labels = No.
of Packets)",
    fontsize = 30,
    fontweight = "bold",
    pad = 25
)
plt.axis("off")
plt.tight_layout()
plt.show()
```