Telecommunication Software

First practical exercise

Full name: **Mithin kumar Ananthula**

ID: **241AEM014**

Riga 2024

Contents

Task 1

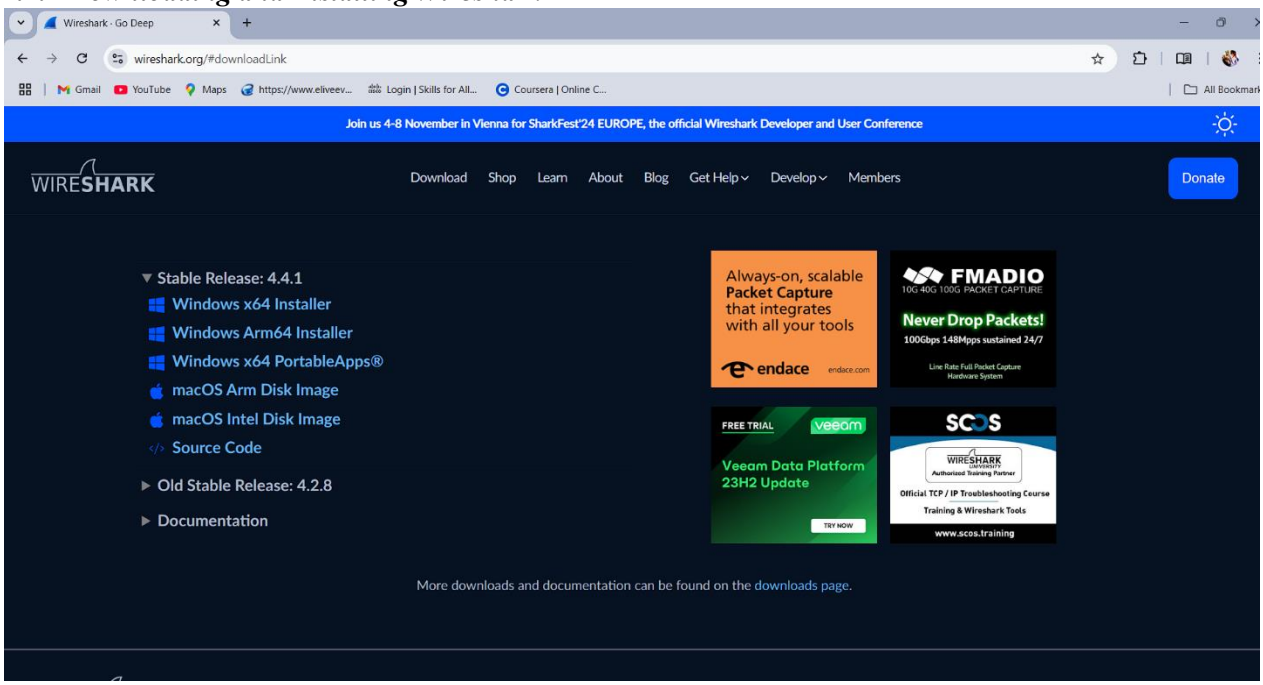
Finish Packet Sniffing and Wireshark Guidebook 02

2.1 Objective

The main of this practical work is to acquire skills of monitoring and analysis of networks using sniffing programs Wireshark and Network Miner. The lab assignment for this chapter is a warm-up tasting of the wireshark software. In this lab, we retrieve a web page and then, using Wireshark, capture packets.

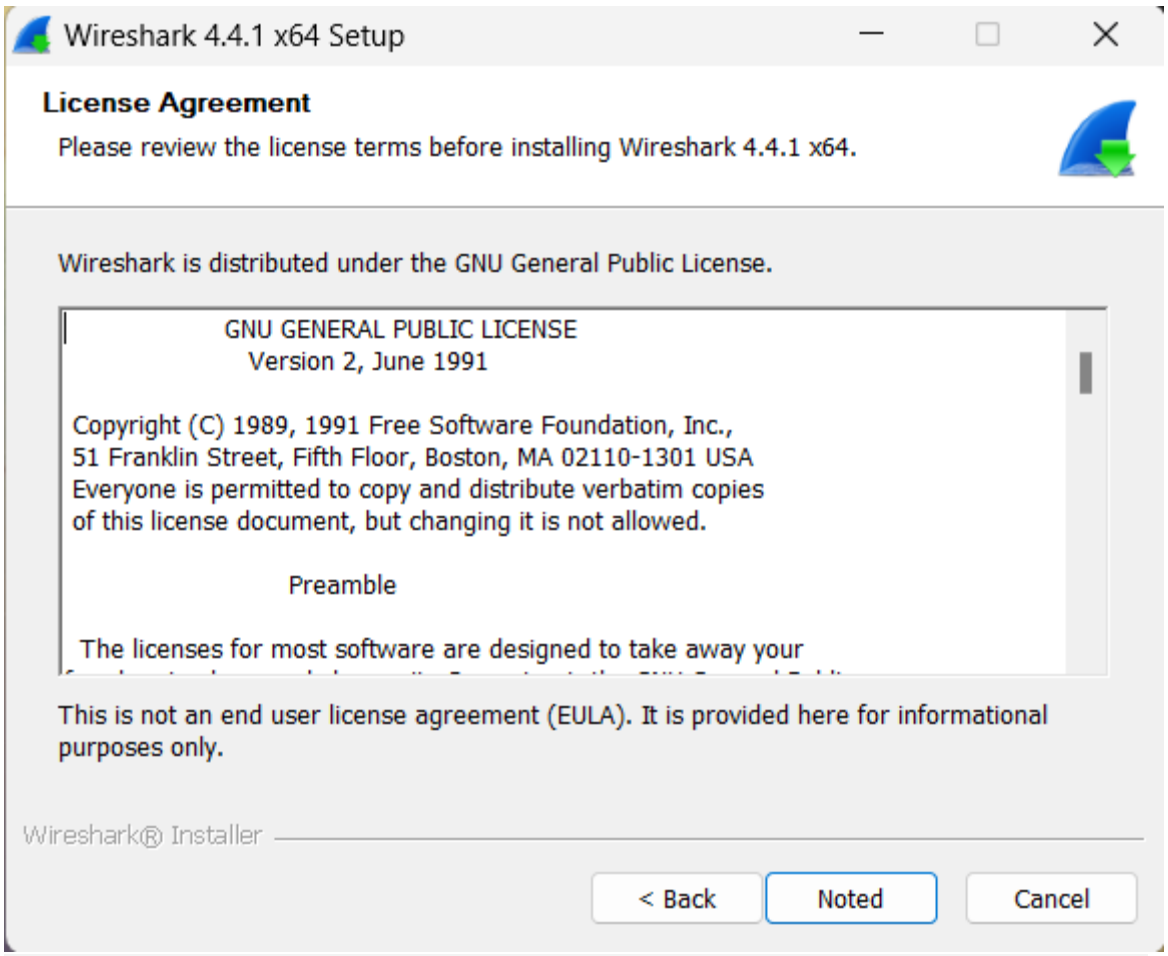
2.2 Lab Assignment

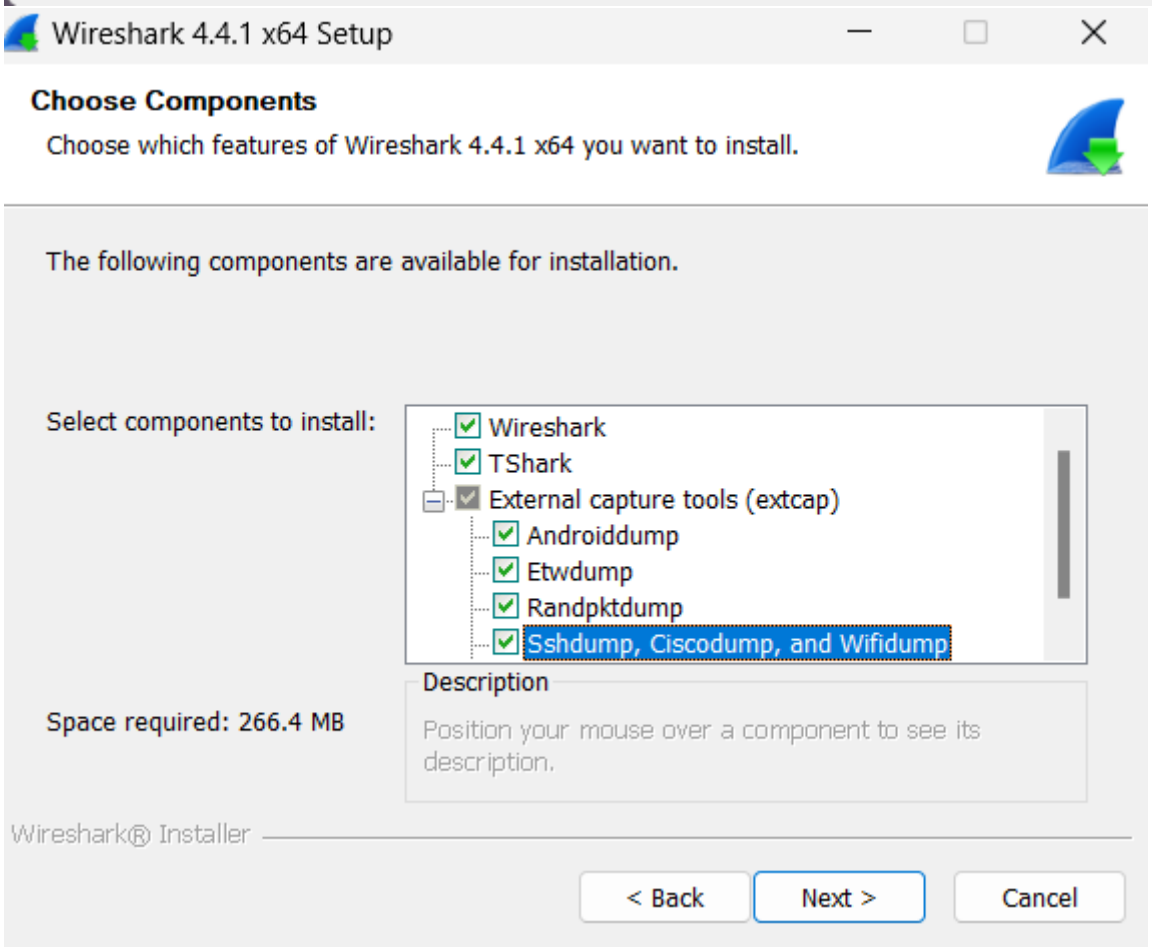
2.2.1 Downloading and Installing Wireshark

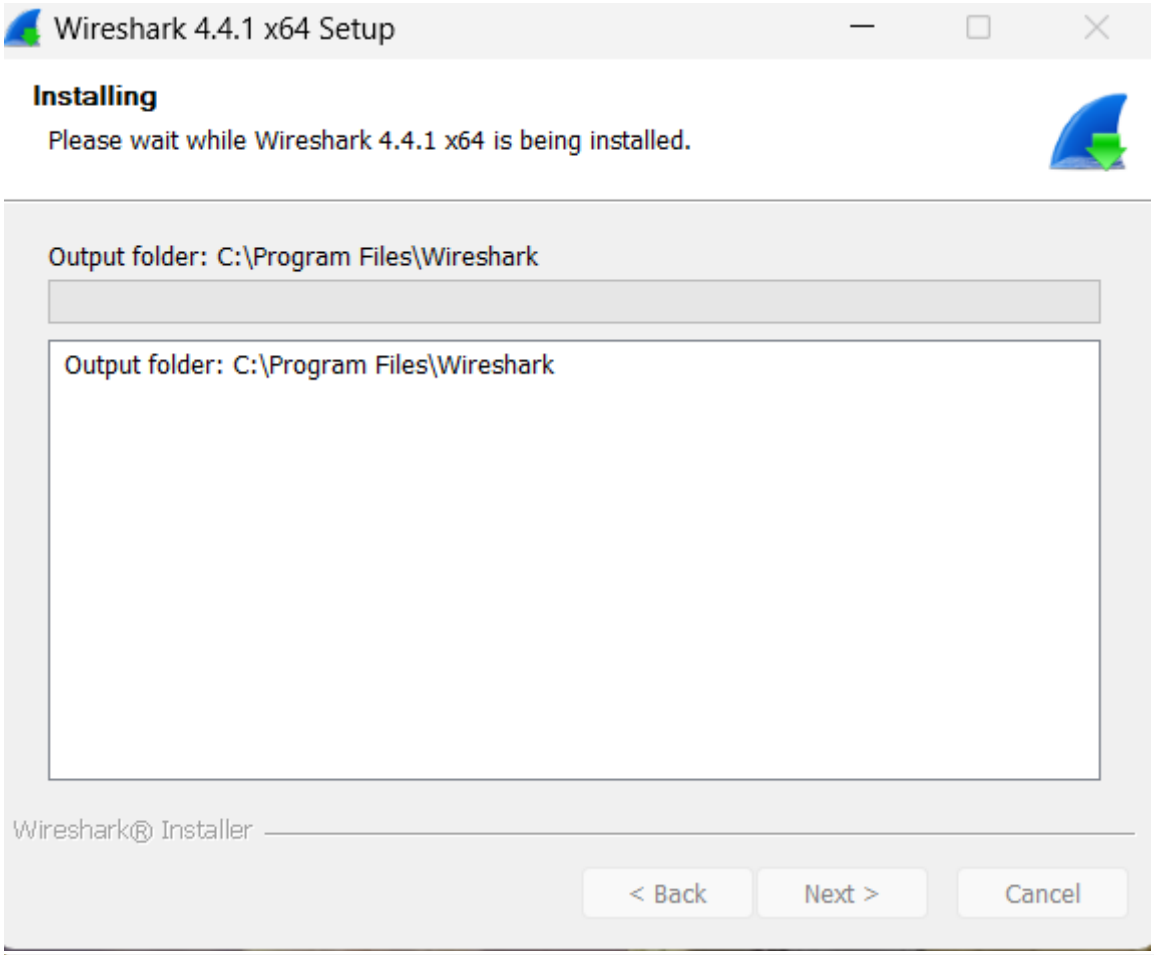


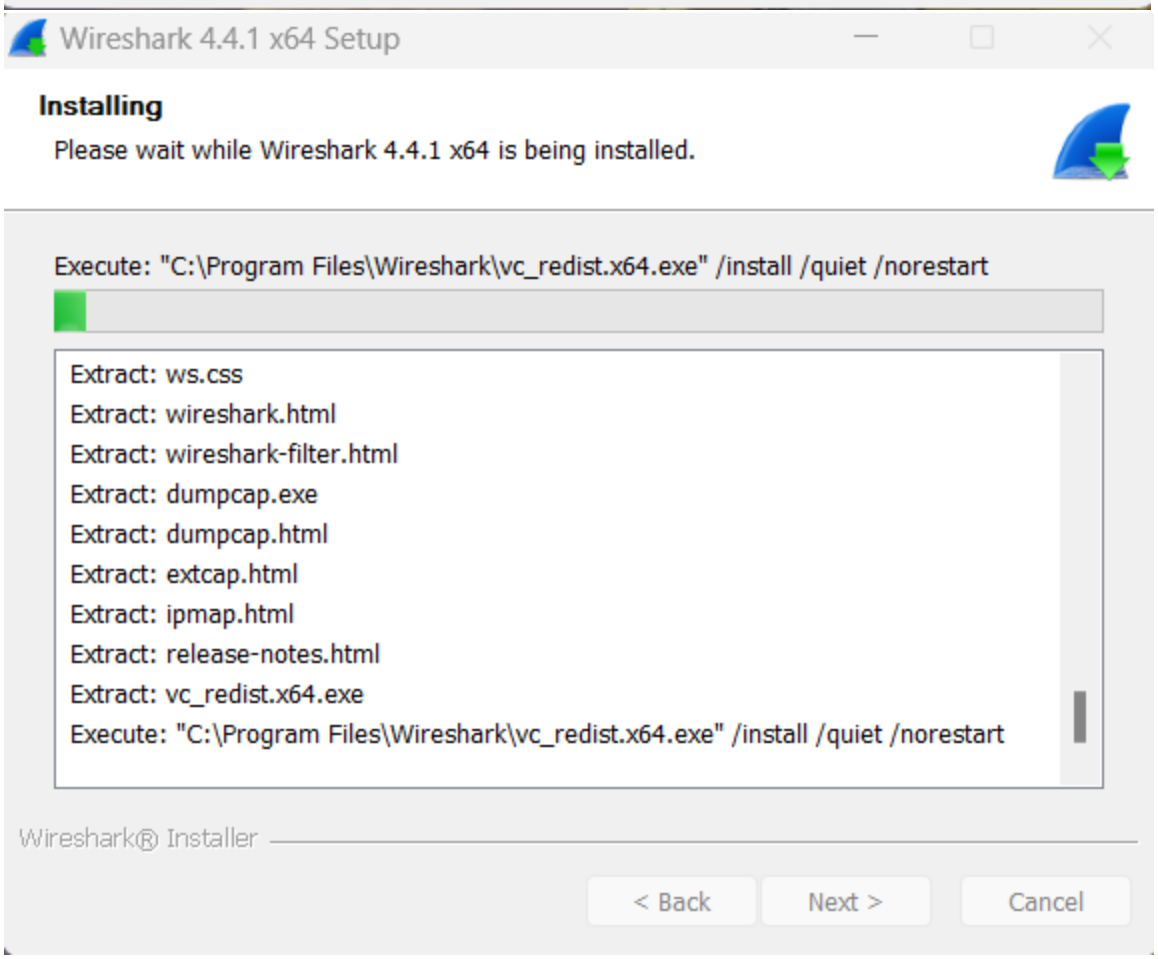
Visiting official web site for downloading the application

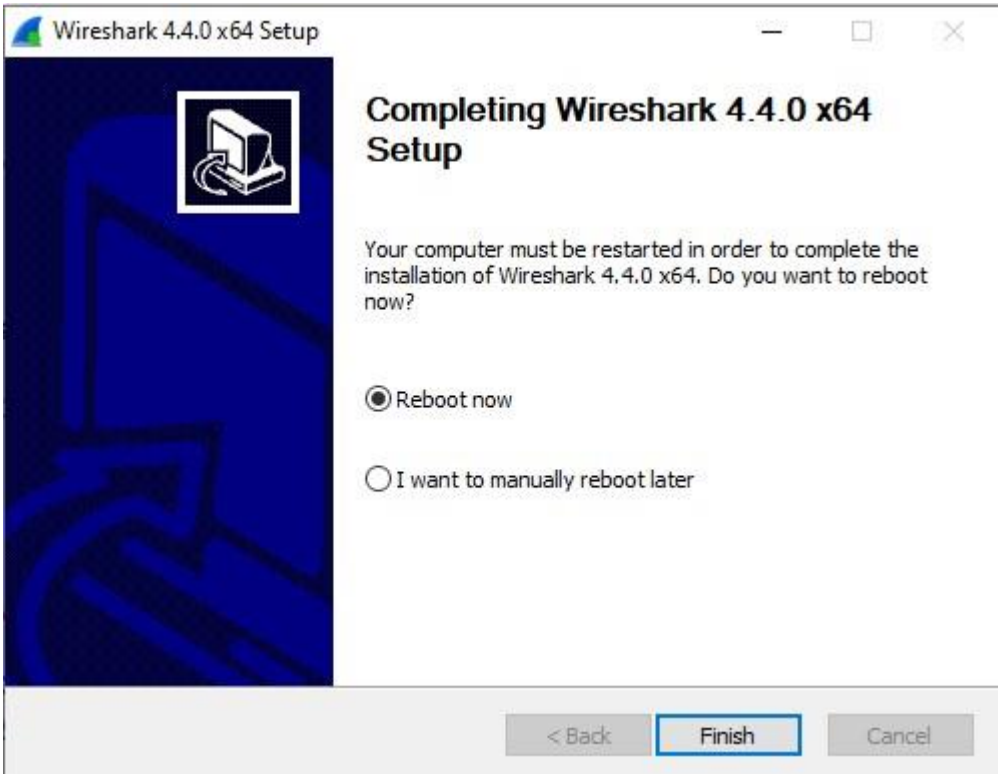
Installing the application process.


Wireshark 4.4.1 x64 Setup
License Agreement
Please review the license terms before installing Wireshark 4.4.1 x64.
Wireshark is distributed under the GNU General Public License.
GNU GENERAL PUBLIC LICENSE
Version 2, June 1991
Copyright (C) 1989, 1991 Free Software Foundation, Inc.,
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
Preamble
The licenses for most software are designed to take away your
This is not an end user license agreement (EULA). It is provided here for informational purposes only.
Wireshark® Installer
< Back
Noted
Cancel


Wireshark 4.4.1 x64 Setup
Choose Components
Choose which features of Wireshark 4.4.1 x64 you want to install.
The following components are available for installation.
Select components to install:
Wireshark
TShark
External capture tools (extcap)
Androiddump
Etwdump
Randpktdump
Sshdump, Ciscodump, and Wifidump
Description
Position your mouse over a component to see its description.
Space required: 266.4 MB
Wireshark® Installer
< Back
Next >
Cancel

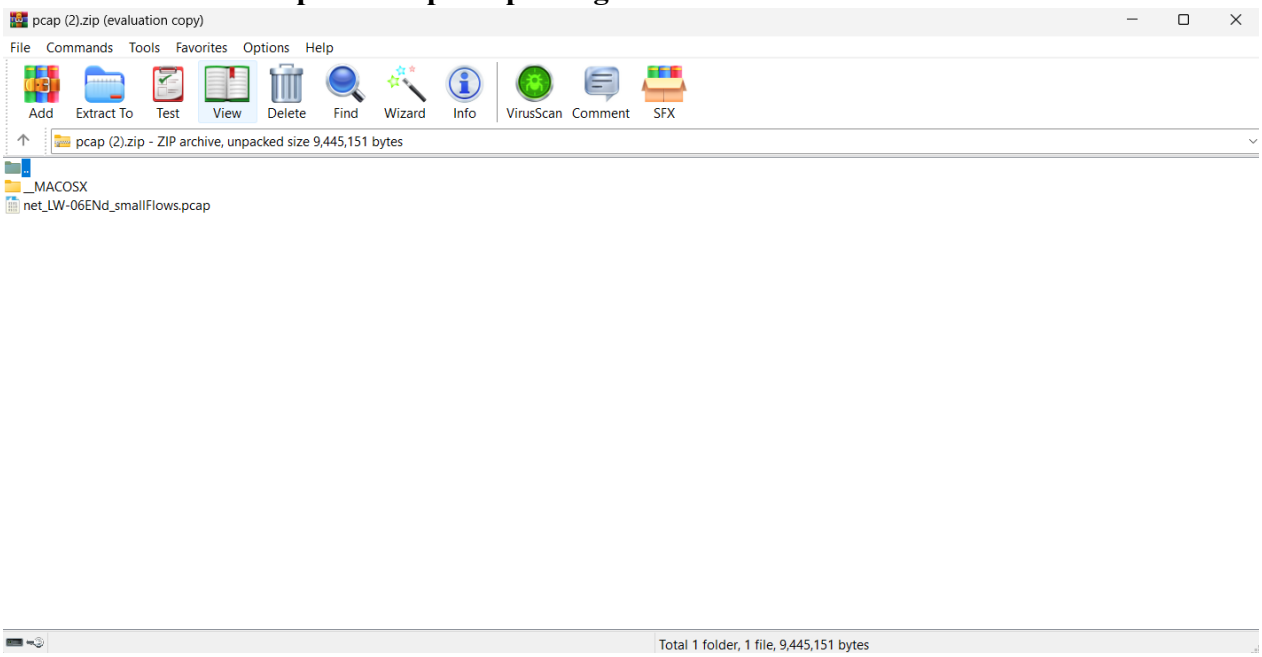

Wireshark 4.4.1 x64 Setup
Installing
Please wait while Wireshark 4.4.1 x64 is being installed.
Output folder: C:\Program Files\Wireshark
Output folder: C:\Program Files\Wireshark
Wireshark® Installer
< Back
Next >
Cancel


Wireshark 4.4.1 x64 Setup
Installing
Please wait while Wireshark 4.4.1 x64 is being installed.
Execute: "C:\Program Files\Wireshark\vc_redist.x64.exe" /install /quiet /norestart
Extract: ws.css
Extract: wireshark.html
Extract: wireshark-filter.html
Extract: dumpcap.exe
Extract: dumpcap.html
Extract: extcap.html
Extract: ipmap.html
Extract: release-notes.html
Extract: vc_redist.x64.exe
Execute: "C:\Program Files\Wireshark\vc_redist.x64.exe" /install /quiet /norestart
Wireshark® Installer
< Back
Next >
Cancel



Installed successfully to my Windows system.

2.2.2 Download and open example capturing file



Extracting ".prep" file to understand the process

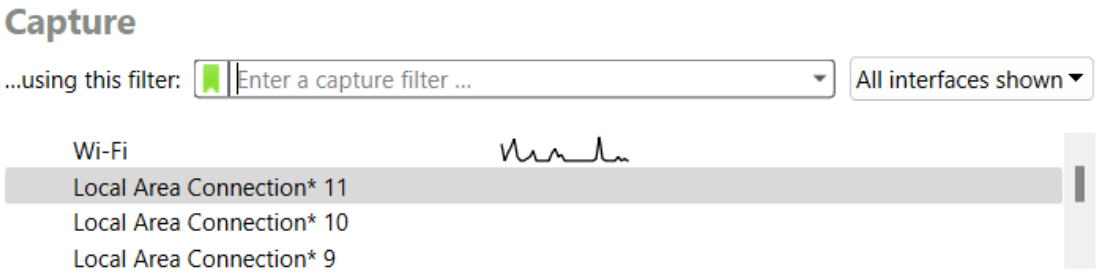
2.3 Report

Student Name	Student ID	Date
Mithin Kumar Ananthula	214AEM014	22.09.2024

Mainly used wireless_connection.pcap file to monitor and analyze

2.3.1 Capture file properties

#	Parameter	Value
1	Time of capture, S	5.774
2	Packets	9
3	Bytes, MiB	607
4	Average packet size, B	67
5	Average packets per seconds, pps	1.6
6	Average bytes per seconds, B/s	105
7	Determine the relative network load L for control period T by formula L = (Traffic [Mbits] / T [sec]) / (Bandwidth [Mbits/sec]) Bandwidth = 100 Mbits/sec L **Given Data** • **Time of capture, TTT** = 5.774 seconds • **Bytes, MiB** = 607 MiB • **Bandwidth** = 100 Mbits/sec The relative network load approximately L= 8.60	



As my laptop has no ethernet port to it I could not do the Ethernet Traffic Distribution by Protocols and Ethernet Traffic distribution by node.

2.3.2 Ethernet Traffic Distribution by Protocols:
In Wireshark, Ethernet Traffic Distribution by Protocols provides a detailed breakdown of captured traffic by protocol type, helping users analyze the composition of network activity. This can be accessed via **Statistics > Protocol Hierarchy**, which displays the percentage of total traffic, number of packets, and bytes associated with each protocol (e.g., TCP, UDP, ICMP, ARP). Users can apply filters to focus on specific protocols or visualize the data using **IO Graphs**. This analysis helps identify dominant protocols, optimize bandwidth usage, and detect anomalies like excessive ARP traffic or unusual protocol activity. It's an essential tool for network troubleshooting and performance monitoring.

2.3.4 Ethernet Traffic distribution by node:
In Wireshark, Ethernet Traffic Distribution by Node provides insights into how traffic is distributed among devices (nodes) in the network. This can be viewed through Statistics > Endpoints, which displays a list of nodes with their IP/MAC addresses, packet counts, and traffic volumes (bytes sent and received). Users can identify the top talkers and listeners, analyze communication patterns, and filter traffic by specific nodes. Additionally, the Statistics > Conversations tool shows bidirectional traffic between nodes, helping

identify heavy connections or potential bottlenecks. This analysis is crucial for monitoring network performance, pinpointing misbehaving devices, and ensuring balanced traffic distribution.

2.3.4 Display Filters

simple search filters (Display Filters) using AND, OR, NO to display packets from (to) a specific node generated by ICMP, DNS, ARP requests (responses) when accessing any server of your choice

#	Display Filter	Description
1	icmp	Displays all ICMP packets. This filter will show all ICMP requests & responses.
2	arp	Displays ARP requests and replies
3	dns	Displays all DNS packets, including requests and responses.
4	arp.opcode == 1	Displays all ARP requests. ARP requests have an opcode value of 1.
5	icmp || dns	Displays packets that are either ICMP or DNS.
6	icmp && !(arp)	Displays ICMP packets but excludes ARP packets.
7	ip	Displays all IP traffic
8	tcp	Displays only TCP traffic
9	udp	Displays UDP traffic

2.3.5 Network Problem Analyze

Analyze the 5 note/warning/error problems existing on the network. Find and read information about network problems on the Internet. For initial data use the Analyze/Expert Information.

#	Expert Information	Severity	Your short description1
1	Connection reset (RST)	Warning	Indicates that a TCP connection was abruptly closed by the remote host. This can occur due to server issues, network problems, or application errors.
2	TCP keep-alive segment	Note	A TCP keep-alive segment is used to check if the connection is still active. It is sent periodically to maintain the connection and detect broken connections.
3	Packet loss	Error	Occurs when packets of data are lost during transmission, leading to degraded network performance and potential data corruption. This can be caused by network congestion, hardware issues, or signal interference.
4	High latency	Error	Refers to delays in network packet transmission. High latency can result in slow response times and poor performance for applications and services. Causes may include network congestion, long distances between nodes, or routing issues.
5	Duplicate packets	Warning	Happens when the same packet is received more than once. This can result from network congestion, retransmissions, or packet loss. Duplicate packets can lead to unnecessary processing and potential data duplication.

Task 2

Python Numpy, Pandas and Matplotlib

Dataset overview

The dataset contains metrics related to telecommunication traffic, such as source Source (nw_src) and destination (nw_dst) IP addresses Source (tp_src) and destination (tp_dst) ports It also categorizes traffic into types like FTP, WWW, VOIP, and others.
Analysis of the dataset shows that FTP traffic handles the largest volume, with the highest number of forward packets, reverse packet sizes, and duration.

Traffic Summary by Category

ICMP and VOIP traffic have relatively high packet counts and sizes but are smaller than FTP. WWW traffic shows moderate metrics, while DNS and P2P traffic exhibit significantly lower activity levels.
Correlation analysis reveals a strong positive relationship between forward packet counts and forward byte counts, suggesting that higher packet numbers correspond to increased data volumes. Reverse traffic metrics, such as packet size and duration, also show moderate correlations with forward traffic, reflecting interdependent network behaviors.

WWW traffic is moderate in terms of packet count and size, while DNS and P2P have relatively smaller traffic. Here's a summary of the most relevant metrics:

Category	Forward Packet	Reverse Size Packets	Reverse Duration
FTP	37578.36	1.7 million	23487.34
ICMP	4807.45	429k	4902.44
VOIP	3590.93	316k	4255.72
WWW	1883.44	208k	2072.54
DNS	841.06	48k	511.24
P2P	221.00	9k	155.16

Correlation Analysis:

Using a correlation matrix, we analyzed the relationship between various traffic metrics:

Metrics like forward packet count (forward_pc) and reverse packet size show moderate to high correlation, implying that an increase in forward traffic is often matched by increased reverse traffic.

Forward byte count (forward_bc) shows a strong positive correlation with forward packet count, suggesting that higher packet counts naturally lead to a higher volume of bytes transmitted.

Visualization:

A heatmap was generated to visually represent the correlation between metrics, with values ranging from -1 to 1. Strong correlations are indicated by colors on the scale, highlighting key relationships between traffic metrics.

This analysis provides a clear understanding of how different types of telecommunication traffic behave in terms of data transfer, packet sizes, and durations, with correlations showing how these metrics are interrelated.

Code:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt


file_path = 'SDN_traffic.csv'  # Adjust this to your file location

data = pd.read_csv(file_path)


print("Dataset Preview:")

print(data.head())

category_group = data.groupby('category').mean(numeric_only=True)


print("\nTraffic Summary by Category (forward_pc, reverse_size_packets, reverse_duration):")

print(category_group[['forward_pc', 'reverse_size_packets',
'reverse_duration']].sort_values(by='forward_pc', ascending=False))


correlation_matrix = data[['forward_pc', 'forward_bc', 'reverse_size_packets', 'reverse_duration']].corr()

print("\nCorrelation Matrix:")

print(correlation_matrix)


plt.figure(figsize=(8, 6))

plt.title('Correlation Matrix of Traffic Metrics', fontsize=14)

plt.imshow(correlation_matrix, cmap='coolwarm', interpolation='none')

plt.colorbar(label='Correlation coefficient')


for i in range(len(correlation_matrix)):

    for j in range(len(correlation_matrix.columns)):

        plt.text(j, i, f"{correlation_matrix.iloc[i, j]:.2f}", ha="center", va="center", color="black")
```

plt.xticks(range(len(correlation_matrix.columns)), correlation_matrix.columns, rotation=45)

plt.yticks(range(len(correlation_matrix.columns)), correlation_matrix.columns)

plt.tight_layout()

plt.show()

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

file_path = 'SDN_traffic.csv'  # Adjust this to your file location
data = pd.read_csv(file_path)

print("Dataset Preview:")
print(data.head())

category_group = data.groupby('category').mean(numeric_only=True)

print("\nTraffic Summary by Category (forward_pc, reverse_size_packets, reverse_duration):")
print(category_group[['forward_pc', 'reverse_size_packets', 'reverse_duration']].sort_values(by=

correlation_matrix = data[['forward_pc', 'forward_bc', 'reverse_size_packets', 'reverse_duration
print("\nCorrelation Matrix:")
print(correlation_matrix)


plt.figure(figsize=(8, 6))
plt.title('Correlation Matrix of Traffic Metrics', fontsize=14)
plt.imshow(correlation_matrix, cmap='coolwarm', interpolation='none')
plt.colorbar(label='Correlation coefficient')

for i in range(len(correlation_matrix)):
    for j in range(len(correlation_matrix.columns)):
        plt.text(j, i, f"{correlation_matrix.iloc[i, j]:.2f}", ha="center", va="center", color='

plt.xticks(range(len(correlation_matrix.columns)), correlation_matrix.columns, rotation=45)
plt.yticks(range(len(correlation_matrix.columns)), correlation_matrix.columns)
plt.tight_layout()
plt.show()
```

Correlation Matrix: