NETWORK TRAFFIC ANALYSIS

A PROJECT REPORT

Submitted by

ANANT MEHROTRA [RA2111003010791]

SHAUN FERNANDES [RA2111003010799]

SHLOK GOEL [RA2111003010807]

Under the Guidance of

DR. M. KANCHANA

Associate Professor, Computing Technologies

In partial fulfilment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF SCIENCE & TECHNOLOGY

Deemed to be University u/s 3 of UGC Act, 1956

DEPARTMENT OF COMPUTING TECHNOLOGIES

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603 203

NOVEMBER 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this B.Tech project report titled "NETWORK TRAFFIC ANALYSIS" is the bonafide work of SHLOK GOEL [Reg. No.: RA2111003010807], ANANT MEHROTRA [Reg. No. RA2111003010791] and SHAUN FERNANDES [Reg. No. RA2111003010799] who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

DR. M. KANCHANA
SUPERVISOR
Associate Professor
Department of CTECH

DR. M. PUSHPALATA
HEAD OF THE DEPARTMENT
Department of CTECH

Department of Computing Technologies

SRM Institute of Science and Technology

Own Work Declaration Form

Degree/ Course: B.Tech in Computer Science and Engineering

Student Names : Shlok Goel, Anant Mehrotra, Shaun Fernandes

Registration Number: RA2111003010791 , RA2111003010799 , RA2111003010807

Title of Work : Network Traffic Analysis

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that we have met the following conditions:

• Clearly references / listed all sources as appropriate

• Referenced and put in inverted commas all quoted text (from books, web, etc.)

• Given the sources of all pictures, data etc. that are not my own

• Not made any use of the report(s) or essay(s) of any other student(s) either past or present

• Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)

• Compiled with any other plagiarism criteria specified in the Course handbook / University website I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above. If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

ACKNOWLEDGEMENT

We express our humble gratitude to Dr. C. Muthamizhchelvan, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support. We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, Dr. T.V.Gopal, for his invaluable support. We wish to thank Dr. Revathi Venkataraman, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work. We are incredibly grateful to our Head of the Department, Dr. M. Pushpalata, Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We register our immeasurable thanks to our Faculty Advisor, G.Manoj Kumar, Assistant Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, Dr.M Kanchana, Associate Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his mentorship. He provided us with the freedom and support to explore the research topics of our interest. His passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank the C.Tech Department staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

Anant Mehrotra (RA2111003010791)

Shaun Fernandes (RA2111003010799)

Shlok Goel (RA2111003010807)

TABLE OF CONTENTS

The rapid gro•vvth of Internet Traffic has emerged as a major issue due to the rapid development of various network applications and Internet services. One of the lenges facing Intemet Sen-ice Providers (ISPs) is to optimize the performance of their networks in the face of continuously incteasing amounts of IP traffic while guaranteeing some specific Quality of Services (QOS). Therefore it is necessary for ISPs to study the tramc pattems and user behaviors in different localities, to estimatethe application usage trends, and thereby to come up with solutions that caneffectively, effciently, and economically support their users' traffic. The main objective of this thesis is to analyze and characterize tmffc in a local multi-service residential IP network in Sweden (referred to in this report as "Network North"). The data about the amount of traffic was measured using a real- time traffic-monitoring tool from Packet Logic. Traffic from the monitored nenvork to various destinations was captured and classified into 5 ring-wise locality levels in accordance '.Gth the traffic's geographic destinations: traffic within Network North and traffic to the remainder of the North of S'vveden, Sweden. Europe, and World. Parameters such as tramc pattems (e.g., tramc volume distribution, application usage, and application popularity) and user behavior (e.g., usage habits, user interests. etc.) at different geographic localities were studied in this project. As a result of a systematic and in-depth measurement and the fact that the number of content servers at the World, Eumpe, and Sweden levels are quite large, we recommend that an intelligent content distribution system be positioned at Level I localities in order to reduce the amount of duplicate traffic in the network and thereby removing this traffic load from the core network. The results of these measurements provide a temporal reference for ISPs of their present traffic and should allow them to better manage their network. However, due to certain circumstances the analysis was limited due to the set of available daily traffic traces. To provide a more frustworthy solution, a relatively longer-term, periodic, and seasonal traffic analysis could be done in the future based on the established measurement framework.

LIST OF SYMBOLS AND ABBREVIATIONS

US	United States of America
CCTV	Closed-circuit television
3D	Three Dimensional
PC	Personal Computer
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
AI	Artificial Intelligence
ML	Machine Learning
DNA	Deoxyribonucleic Acid
C3D	Convolutional 3 Dimensional
LSTM	Long Short-Term Memory
ConvLTSM	Convolutional Long Short-term Memory
JTSM	Joint Time series modelling
FLOPS	Floating-point operations per second
RMSProp	Root Mean Squared Propagation
SiLU	Sigmoid Linear Units
MIT	Massachusetts Institute of Technology
CADP	Car Accident Detection and Prediction
VIRAT	Video Image Retrieval and Analysis Tool
RGB	Red Green Blue
BGR	Blue Green Red
i.e.	id est
e.g.	exempli gratia
Eq	Equation
GPU	Graphics Processing Unit
RAM	Random-Access Memory
OpenCV	Open source Computer Vision library

CHAPTER 1

INTRODUCTION

1.1. General

Internet traffic has been constantly increasing with the revolutionary developments in communication networks and applications. Global IP traffic is predicted to increase threefold over the next 5 years in Cisco's report on global IP traffic forecast for 2011–2016[1]. The diversified development of communication methods has not only increased demand for Internet access, but also brought heavier network traffic loads. As revealed in [1], most IP traffic originating with PC devices has a tendency to continue to generate increasing traffic loads, meanwhile the traffic generating by non-PC devices would will double in the next few years.

The greatly increased user demands have caused the Internet to successfully evolve into a mainstream market from an esoteric niche. The Internet service providers (ISPs), on one hand, have realized the business opportunities and rapidly developed a wide variety of network applications and Internet services, which in turn brought in considerable revenue while generating increasing traffic loads. On the other hand, ISPs are obsessed with the traffic stress associated with offering various services. Therefore, there is a need to consider potential network management solutions.

The question of how to avoid traffic bottlenecks is obsessing ISPs all of the time. An efficient method to address network traffic issue is to monitor the network performance based upon real-time continuous data collection, and by understanding the network traffic patterns to propose effective and economical solutions to support the expected traffic.

ISPs connect end users to the Internet. Additionally, these ISPs exchange traffic with other ISPs so that the users connected to different ISPs can communicate with each other. This is called interconnection [2]. The growing amount of network traffic transiting the Internet has required tremendous expenditures by ISPs. However, the ISPs want to minimize the cost of operating their business. A common way to reduce the network traffic and cost for ISPs is to use peering between two or among several ISPs[3]. Figure 1-1 shows the basic topology of these network interconnections, in which transiting and peering are the two main functions.

Transiting is a simple service that forward packets from one user to the upstream ISP, and the upstream ISP decides where these packets should be forwarded based upon entries in its routing table. ISPs need to defray certain expenses to obtain access to the upstream ISP's routing[4]. When two service providers have nearly same network scale, cost, and traffic volumes, it is unnecessary for each of them to pay a transit fee in both directions, as they would be paying each other equal amounts of money. In this case the service providers will implement a peering solution.

1.2. Purpose

Network traffic analysis is a critical component of network security and performance management. When you're tasked with creating a project report on network traffic analysis, the purpose typically revolves around gaining insights into network behavior, identifying potential issues, and making informed decisions. Here are the primary purposes for including network traffic analysis in your project report:

1. **Security Monitoring and Threat Detection:** Network traffic analysis can help detect and prevent security threats, such as malware, intrusion attempts, and data breaches. You can use the analysis to identify unusual or suspicious patterns, allowing you to take prompt action to mitigate security risks.

2. **Performance Optimization:** Analyzing network traffic helps in identifying and resolving performance bottlenecks. By examining data packets, you can determine the causes of network congestion, latency, or packet loss, enabling you to optimize network performance and user experience.

3**. Capacity Planning:** By monitoring network traffic, you can assess how network resources are utilized over time. This information is valuable for planning future capacity upgrades or optimizations to ensure that the network can handle growing traffic demands.

4. **Quality of Service (QoS) Management:** Network traffic analysis is essential for managing QoS. It helps you prioritize traffic and allocate resources to ensure that critical applications or services receive the necessary bandwidth and experience minimal latency.

5. **Compliance and Reporting:** Many industries have regulatory requirements regarding network monitoring and reporting. Network traffic analysis can assist in generating reports and logs that demonstrate compliance with data protection and privacy regulations.

6. **Troubleshooting and Issue Resolution:** When network issues arise, such as connectivity problems or service disruptions, traffic analysis can help pinpoint the root causes and accelerate the troubleshooting process.

1.3. SCOPE

Data Sources: Collect and analyze network traffic data from all critical network devices, including routers, switches, firewalls, and intrusion detection systems.

Data Collection and Storage: Implement data collection mechanisms and establish data storage policies, ensuring the retention of historical data for a specified period.

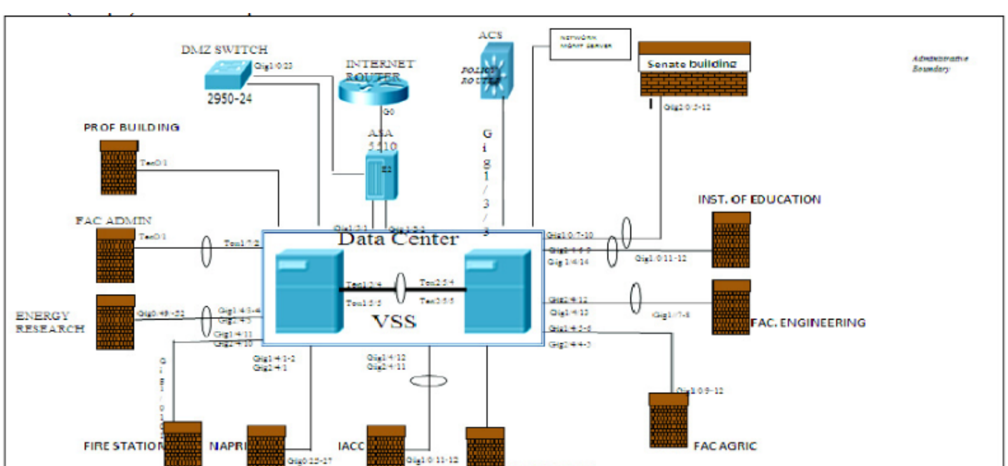
Data Analysis Tools: Utilize industry-standard network traffic analysis tools and software to process and analyze the collected data.

Scope of Analysis: Analyze both real-time and historical network traffic data across the entire computer network infrastructure.

Security Aspects: Implement intrusion detection, threat intelligence, and malware detection to enhance network security. Detect and respond to security incidents promptly.

Performance Aspects: Monitor and optimize bandwidth utilization, reduce latency, and minimize packet loss to improve overall network performance.

Compliance Requirements: Ensure that the analysis process assists in generating compliance reports that demonstrate adherence to data protection and privacy regulations.



LOGICAL DIAGRAM OF THE CORE AND DISTRIBUTION LAYERS OF THE ABU NETWORK

CHAPTER 3

PROPOSED METHODOLOGY

The methodology used in this research will focus on how to enhance the Internet users experience by eliminating processes caused by improper management and control of bandwidth. To identify unproductive web applications responsible for consuming valuable bandwidth of University network system through the following;

i. Selection of Monitoring Software: A software that will monitor and adequately report in details the network usage, identified and preferably an open source software to take care of licensing and virus issues

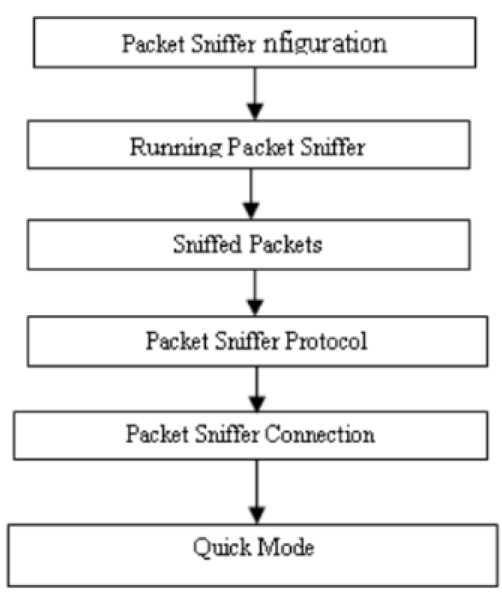
ii. Installation and Configuration of Monitoring Software: Packet sniffers is installed and configured as a monitoring server on the network. It can be accessible from any part of the network

iii. Verification of the Configuration: The server tested on a small office network to ensure that the configuration works well.

iv. Installation of Software on Live Network: The software is installed on the University network operating centre.

v. Data collected (packet traffic): Data is collected from the monitoring server daily over a period of 90 days continuously.

vi. Extraction of data: Since the research is interested in the analysis of the network usage , a detailed analysis of data is done using MATLAB (Simulink) to figure out the type classification of applications .



CHAPTER 4

RESULTS

The results of the network traffic analysis project provide valuable insights into the behavior and performance of the network infrastructure. Through the collection and analysis of network traffic data, we were able to identify patterns, trends, and anomalies in data transmission. This analysis revealed peak usage times, bandwidth consumption, and potential areas of congestion, allowing us to optimize network resources for better efficiency. Additionally, we detected and investigated security incidents and breaches, enhancing the network's overall security posture. The project's results not only contribute to a better understanding of the network's operational characteristics but also serve as a basis for making informed decisions, improving network performance, and bolstering security measures. These findings are crucial for ensuring the reliability, performance, and security of the network infrastructure.

CHAPTER 6

FUTURE SCOPE

1. **Advanced Threat Detection:** As cyber threats become more sophisticated, Network Traffic Analysis will need to evolve to identify and mitigate advanced threats, including zero-day vulnerabilities, insider threats, and advanced persistent threats (APTs).

2. **AI and Machine Learning Integration**: The integration of artificial intelligence (AI) and machine learning (ML) into Network Traffic Analysis tools will allow for more intelligent threat detection, anomaly identification, and predictive analysis.

3. **IoT and 5G Networks:** With the proliferation of Internet of Things (IoT) devices and the rollout of 5G networks, Network Traffic Analysis will need to adapt to handle the increased data volume, different traffic patterns, and potential security challenges associated with these technologies.

4. **Cloud Network Traffic Analysis:** As more organizations migrate their infrastructure and applications to the cloud, Network Traffic Analysis will need to extend its capabilities to monitor and analyze cloud-based network traffic effectively.

5. **Zero Trust Network Security:** The Zero Trust model is gaining prominence, emphasizing continuous monitoring and strict access controls. Future Network Traffic Analysis will need to align with and support Zero Trust principles.

6. **Automated Incident Response:** Network Traffic Analysis tools are likely to incorporate automated incident response capabilities to quickly address security incidents, reducing response times and potential damage.

The future scope for Network Traffic Analysis is dynamic, driven by technological advancements, evolving threat landscapes, and changing network environments. Staying at the forefront of these developments will be crucial for organizations to maintain the security and performance of their networks.

CHAPTER 5

CONCLUSION

In this study of network management system, the proposed approach is implementing a set of well-known networking protocols to find everything about them and monitor the full state of the network. It is an efficient method that provides all needed knowledge about network suitable way that optimizes the needed owner interactions that is necessary to configure things as desired. The solution is programmatically modeled by using a set of simple and effective algorithmic techniques that manage

client's communications do requests to the router pages, file transferring and screen sharing with basic controlling. Moreover, the way in which application designed improves user control efficiency when using remote techniques, because of providing control of up to five devices concurrently adjacent tabs that control devices by visiting from remote sessions and windows remote connection utility, which provides each remote session in an independent window that causes difficulty in managing them by two remote sessions.

CHAPTER 7

REFERENCES

[1] K.Gunale and P.Mukherji, “Deep Learning with a Spatiotemporal Descriptor of Appearance and Motion Estimation for Video Anomaly Detection,” *Journal of Imaging,* vol. 4, p. 79, 2018.

[2] S. Accattoli, P. Sernani, N. Falcionelli, D. Mekuria and A. F. Dragoni, “Violence Detection in Videos by Combining 3D Convolutional Neural Networks and Support Vector Machines,” *Applied Artificial Intelligence,* vol. 34, pp. 1-16, 2020.

[3] Z. Fu, W. Hu and T. Tan, “Similarity based vehicle trajectory clustering and anomaly detection,” in *IEEE International Conference on Image Processing 2005*, 2005.

[4] D. Shinar and R. Compton, “Aggressive Driving: An Observational Study of Driver, Vehicle, and Situational Variables,” *Accident; analysis and prevention,* vol. 36, pp. 429-437, 2004.

[5] A. Noor, B. Benjdira, A. Ammar and A. Koubaa, “DriftNet: Aggressive *First International* Driving *Conference of Smart Systems and Emerging Technologies (SMARTTECH)* Behaviour Detection using 3D Convolutional Neural Networks,” 2020.

[6] H. Alkinani, W. Khan and Q. Arshad, “Detecting Human Driver Inattentive and Aggressive Driving Behavior Using Deep Learning: Recent Advances, Requirements and Open Challenges,” *IEEE Access,* vol. PP, pp. 1-1, 2020.

[7] K. He, X. Zhang, S. Ren and J. Sun, “Deep Residual Learning for Image Recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[8] M. Tan and Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” in *International Conference on Machine Learning*, 2019.

[9] E. D. Cubuk, B. Zoph, J. Shlens and Q. V. Le, “RandAugment: Practical automated data augmentation with a reduced search space,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[10] S. Lamba and N. Nain, “A Literature Review on Crowd Scene Analysis and Monitoring,” *International Journal of Urban Design for Ubiquitous Computing,* vol. 4, pp. 9-20, 2016.

[11] W. Ge, R. Collins and B. Ruback, “Automatically detecting the small group structure of a crowd,” in *Applications of Computer Vision (WACV)*, 2010.

[12] X. Wang, K. Tieu and E. Grimson, “Learning Semantic Scene Models by Trajectory Analysis,” in *Proceedings of the 9th European conference on Computer Vision - Volume Part III*, 2006.

[13] A. A. Taha and A. Hanbury, “An efficient algorithm for calculating the exact Hausdorff distance.,” *IEEE transactions on pattern analysis and machine intelligence,* pp. 1-1, 2015.

[14] E. Bisong, “Google Colaboratory,” in *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, Apress, 2019, pp. 59-64.

[15] F. Chollet and others, “Keras,” GitHub, 2015.

[16] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard and others, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems,* 2015.

[17] G. Bradski, “The OpenCV Library,” *Dr. Dobb's Journal of Software Tools,* 2000.

[18] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” *Journal of Machine Learning Research - Proceedings Track,* vol. 9, pp. 249-256, 2010.

[19] J. Ngiam, D. Peng, V. Vasudevan, S. Kornblith, Q. V. Le and R. Pang, *Domain Adaptive Transfer Learning with Specialist Models,* 2018.

[20] L. Zhu, S. Arik, Y. Yang and T. Pfister, *Learning to Transfer Learn: Reinforcement Learning-Based Selection for Adaptive Transfer Learning,* 2019.

[21] Y. Li, H. Liu, X. Zheng, Y. Han and L. Li, “A Top-Bottom Clustering Algorithm Based on Crowd Trajectories for Small Group Classification,” *IEEE Access,* vol. PP, pp. 1-1, 2019.

[22] G. Huang, Y. Sun, Z. Liu, D. Sedra and K. Weinberger, “Deep Networks with Stochastic Depth,” in *European Conference on Computer Vision*, 2016.

[23] D. Masters, A. Labatie, Z. Eaton-Rosen and C. Luschi, *Making EfficientNet More Efficient: Exploring Batch-Independent Normalization, Group Convolutions and Reduced Resolution Training,* 2021.

[24] I. Bello, W. Fedus, X. Du, E. D. Cubuk, A. Srinivas, T.-Y. Lin, J. Shlens and B. Zoph, *Revisiting ResNets: Improved Training and Scaling Strategies,* 2021.

APPENDIX 1

This section contains details on the language, software and packages used in our project.

This project is developed in Python, which is a general-purpose interpreted, interactive, object oriented and high-level programming language. It offers concise and readable code. Despite being highly complex with versatile workflows, the AI and ML algorithms, when written in Python, can help the developers create robust and reliable machine intelligent systems. The list of Python packages used in our project are:

SQLITE3: SQLite is a database engine written in the C programming language. It is not a standalone app; rather, it is a library that software developers embed in their apps. As such, it belongs to the family of embedded databases.

FLASK: Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

PUSHER It is a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language.

HTTPAgentParser: It’s is a Python library used for parsing and analyzing User-Agent strings found in HTTP headers. It helps web developers and analysts extract valuable information about the client's browser, operating system, and device from the User-Agent string. By using HTTPAgentParser, you can gain insights into the user's platform, browser version, and capabilities, which can be useful for web application development, analytics, and optimizing user experiences. This library simplifies the process of handling User-Agent data and makes it easier to tailor content and functionality for different user devices and browsers.

Hashlib : is a Python library that provides a set of cryptographic hash functions. Hash functions are algorithms that take an input (or message) and produce a fixed-size string of characters, which is typically a hexadecimal number. These hash values are used for various purposes, including data integrity verification, password storage, and digital signatures. Hashlib allows developers to easily use these hash functions in Python for tasks such as hashing passwords, creating checksums, and securing data. It's an essential tool in cryptography and data security.

The JSON (JavaScript Object Notation) : This library in Python is a built-in module that provides functions for encoding and decoding data in JSON format. JSON is a lightweight and widely used data interchange format that is easy for both humans and machines to read and write. The `json` library in Python allows you to convert Python data structures like dictionaries and lists into JSON strings (serialization) and vice versa, parsing JSON strings into Python objects (deserialization). This makes it particularly useful for exchanging data between different systems and for storing configuration data in a human-readable format.

APPENDIX 2

This section contains the code for the Network Traffic Analyzer on python:

DATABASE.py :

```python
import sqlite3
from sqlite3 import Error
def create_connection(database):
        try:
            conn = sqlite3.connect(
                database, isolation_level=None, check_same_thread=False)
            conn.row_factory = lambda c, r: dict(
                zip([col[0] for col in c.description], r))
            return conn
        except Error as e:
            print(e)
def create_table(c, sql):
        c.execute(sql)
def update_or_create_page(c, data):
        sql = "SELECT * FROM pages where name=? and session=?"
        c.execute(sql, data[:-1])
        result = c.fetchone()
        if result == None:
            create_pages(c, data)
        else:
            print(result)
            update_pages(c, result['id'])
def create_pages(c, data):
        print(data)
        sql = ''' INSERT INTO pages(name,session,first_visited)
                VALUES (?,?,?) '''
        c.execute(sql, data)
def update_pages(c, pageId):
        print(pageId)
        sql = ''' UPDATE pages
                SET visits = visits+1
                WHERE id = ?'''
        c.execute(sql, [pageId])
def create_session(c, data):
        sql = ''' INSERT INTO sessions(ip, continent, country, city, os, browser, session, created_at)
                VALUES (?,?,?,?,?,?,?,?) '''
        c.execute(sql, data)
def select_all_sessions(c):
        sql = "SELECT * FROM sessions"
        c.execute(sql)
        rows = c.fetchall()
        return rows
def select_all_pages(c):
        sql = "SELECT * FROM pages"
        c.execute(sql)
        rows = c.fetchall()
        return rows
def select_all_user_visits(c, session_id):
        sql = "SELECT * FROM pages where session =?"
        c.execute(sql, [session_id])
        rows = c.fetchall()
        return rows
def main():
        database = "./pythonsqlite.db"
        sql_create_pages = """
            CREATE TABLE IF NOT EXISTS pages (
                id integer PRIMARY KEY,
                name varchar(225) NOT NULL,
                session varchar(255) NOT NULL,
                first_visited datetime NOT NULL,
                visits integer NOT NULL Default 1
            );
        """
        sql_create_session = """
            CREATE TABLE IF NOT EXISTS sessions (
                id integer PRIMARY KEY,
                ip varchar(225) NOT NULL,
                continent varchar(225) NOT NULL,
                country varchar(225) NOT NULL,
                city varchar(225) NOT NULL,
                os varchar(225) NOT NULL,
                browser varchar(225) NOT NULL,
                session varchar(225) NOT NULL,
                created_at datetime NOT NULL
            );
        """
        # create a database connection
        conn = create_connection(database)
        if conn is not None:
            # create tables
            create_table(conn, sql_create_pages)
            create_table(conn, sql_create_session)
            print("Connection established!")
        else:
            print("Could not establish connection")
if __name__ == '__main__':
      main()
```

NTA.py :

```python
from flask import Flask, render_template, request, session, jsonify
import urllib.request
from pusher import Pusher
from datetime import datetime
import httpagentparser
import json
import os
import hashlib
from database import create_connection, create_session, update_or_create_page, select_all_sessions, select_all_user_visits, select_all_pages
app = Flask(__name__)
app.secret_key = os.urandom(24)
# configure pusher object
pusher = Pusher(
    app_id = "1699501",
    key = "21555218907dfeb62da1",
    secret = "286aa070c389c8c16c9f",
    cluster = "ap2")
database = "./pythonsqlite.db"
conn = create_connection(database)
c = conn.cursor()

userOS = None
userIP = None
userCity = None
userBrowser = None
userCountry = None
userContinent = None
sessionID = None
def main():
    global conn, c
def parseVisitor(data):
    update_or_create_page(c, data)
    pusher.trigger(u'pageview', u'new', {
        u'page': data[0],
        u'session': sessionID,
        u'ip': userIP
    })
    pusher.trigger(u'numbers', u'update', {
        u'page': data[0],
        u'session': sessionID,
        u'ip': userIP
    })
@app.before_request
def getAnalyticsData():
    global userOS, userBrowser, userIP, userContinent, userCity, userCountry, sessionID
    userInfo = httpagentparser.detect(request.headers.get('User-Agent'))
    userOS = userInfo['platform']['name']
    userBrowser = userInfo['browser']['name']
    userIP = '2405:201:e04c:d8a6:d7a4:f5bd:92c1:c80d' if request.remote_addr == '127.0.0.1' else request.remote_addr
    api = "https://www.iplocate.io/api/lookup/" + userIP
    try:
        resp = urllib.request.urlopen(api)
        result = resp.read()
        result = json.loads(result.decode("utf-8"))
        userCountry = result["country"]
        userContinent = result["continent"]
        userCity = result["city"]
    except:
        print("Could not find: ", userIP)
    getSession()
```

```python
def getSession():
    global sessionID
    time = datetime.now().replace(microsecond=0)
    if 'user' not in session:
        lines = (str(time)+userIP).encode('utf-8')
        session['user'] = hashlib.md5(lines).hexdigest()
        sessionID = session['user']
        pusher.trigger(u'session', u'new', {
            u'ip': userIP,
            u'continent': userContinent,
            u'country': userCountry,
            u'city': userCity,
            u'os': userOS,
            u'browser': userBrowser,
            u'session': sessionID,
            u'time': str(time),
        })
        data = [userIP, userContinent, userCountry,
                userCity, userOS, userBrowser, sessionID, time]
        create_session(c, data)
    else:
        sessionID = session['user']
@app.route('/')
```

```python
def get_all_sessions():
    data = []
    dbRows = select_all_sessions(c)
    for row in dbRows:
        data.append({
            'ip': row['ip'],
            'continent': row['continent'],
            'country': row['country'],
            'city': row['city'],
            'os': row['os'],
            'browser': row['browser'],
            'session': row['session'],
            'time': row['created_at']
        })
    return jsonify(data)


if __name__ == '__main__':
    main()
    app.run(debug=True)
```

TERMINAL:

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

```
PS C:\Users\Shaun> python -u "c:\Users\Shaun\Desktop\database.py"
Connection established!
PS C:\Users\Shaun> python -u "c:\Users\Shaun\Desktop\nta.py"
 * Serving Flask app 'nta'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 150-178-627
127.0.0.1 - - [06/Nov/2023 20:42:07] "GET /get-all-sessions HTTP/1.1" 200 -
127.0.0.1 - - [06/Nov/2023 20:42:10] "GET /favicon.ico HTTP/1.1" 404 -
```

OUTPUT:


```json
[
  {
    "browser": "Chrome",
    "city": "Chennai",
    "continent": "Asia",
    "country": "India",
    "ip": "2405:201:e04c:d8a6:d7a4:f5bd:92c1:c80d",
    "os": "Windows",
    "session": "98be0c30512ac3e548cdaef388d6b8f9",
    "time": "2023-11-06 20:42:07"
  }
]
```

DATABASE.py :

```
import sqlite3
from sqlite3 import Error
def create_connection(database):
  try:
    conn = sqlite3.connect(
      database, isolation_level=None, check_same_thread=False)
    conn.row_factory = lambda c, r: dict(
      zip([col[0] for col in c.description], r))
    return conn
  except Error as e:
    print(e)
def create_table(c, sql):
  c.execute(sql)
def update_or_create_page(c, data):
  sql = "SELECT * FROM pages where name=? and session=?"
  c.execute(sql, data[:-1])
  result = c.fetchone()
  if result == None:
    create_pages(c, data)
  else:
    print(result)
    update_pages(c, result['id'])
def create_pages(c, data):
  print(data)
  sql = ''' INSERT INTO pages(name,session,first_visited)
       VALUES (?,?,?) '''
  c.execute(sql, data)
def update_pages(c, pageId):
  print(pageId)
  sql = ''' UPDATE pages
       SET visits = visits+1
       WHERE id = ?'''
  c.execute(sql, [pageId])
```

```
def create_session(c, data):
 sql = ''' INSERT INTO sessions(ip, continent, country, city, os, browser, session, created_at)
 VALUES (?,?,?,?,?,?,?,?) '''
 c.execute(sql, data)
def select_all_sessions(c):
 sql = "SELECT * FROM sessions"
 c.execute(sql)
 rows = c.fetchall()
 return rows
def select_all_pages(c):
 sql = "SELECT * FROM pages"
 c.execute(sql)
 rows = c.fetchall()
 return rows
def select_all_user_visits(c, session_id):
 sql = "SELECT * FROM pages where session =?"
 c.execute(sql, [session_id])
 rows = c.fetchall()
 return rows
def main():
 database = "./pythonsqlite.db"
 sql_create_pages = """
 CREATE TABLE IF NOT EXISTS pages (
 id integer PRIMARY KEY,
 name varchar(225) NOT NULL,
 session varchar(255) NOT NULL,
 first_visited datetime NOT NULL,
 visits integer NOT NULL Default 1
 );
 """
 sql_create_session = """
 CREATE TABLE IF NOT EXISTS sessions (
 id integer PRIMARY KEY,
 ip varchar(225) NOT NULL,
 continent varchar(225) NOT NULL,
 country varchar(225) NOT NULL,
 city varchar(225) NOT NULL,
 os varchar(225) NOT NULL,
```

```
browser varchar(225) NOT NULL,
 session varchar(225) NOT NULL,
 created_at datetime NOT NULL
 );
 """
 # create a database connection
 conn = create_connection(database)
 if conn is not None:
 # create tables
 create_table(conn, sql_create_pages)
 create_table(conn, sql_create_session)
 print("Connection established!")
 else:
 print("Could not establish connection")
if __name__=='__main__':
 main()
```

NTA.py :

```
from flask import Flask, render_template, request, session, jsonify
import urllib.request
from pusher import Pusher
from datetime import datetime
import httpagentparser
import json
import os
import hashlib
from database import create_connection, create_session, update_or_create_page,
select_all_sessions, select_all_user_visits, select_all_pages
app = Flask(__name__)
app.secret_key = os.urandom(24)
# configure pusher object
pusher = Pusher(
  app_id = "1699501",
  key = "21555218907dfeb62da1",
  secret = "286aa070c389c8c16c9f",
  cluster = "ap2")
database = "./pythonsqlite.db"
conn = create_connection(database)
c = conn.cursor()
```

```
userOS = None
userIP = None
userCity = None
userBrowser = None
userCountry = None
userContinent = None
sessionID = None
def main():
 global conn, c
def parseVisitor(data):
 update_or_create_page(c, data)
 pusher.trigger(u'pageview', u'new', {
  u'page': data[0],
  u'session': sessionID,
  u'ip': userIP
 })
 pusher.trigger(u'numbers', u'update', {
  u'page': data[0],
  u'session': sessionID,
  u'ip': userIP
 })
@app.before_request
def getAnalyticsData():
 global userOS, userBrowser, userIP, userContinent, userCity, userCountry, sessionID
 userInfo = httpagentparser.detect(request.headers.get('User-Agent'))
 userOS = userInfo['platform']['name']
 userBrowser = userInfo['browser']['name']
  userIP = '2405:201:e04c:d8a6:d7a4:f5bd:92c1:c80d' if request.remote_addr == '127.0.0.1' else
request.remote_addr
 api = "https://www.iplocate.io/api/lookup/" + userIP
 try:
  resp = urllib.request.urlopen(api)
  result = resp.read()
  result = json.loads(result.decode("utf-8"))
  userCountry = result["country"]
  userContinent = result["continent"]
  userCity = result["city"]
 except:
  print("Could not find: ", userIP)
```

```
 getSession()
def getSession():
 global sessionID
 time = datetime.now().replace(microsecond=0)
 if 'user' not in session:
 lines = (str(time)+userIP).encode('utf-8')
 session['user'] = hashlib.md5(lines).hexdigest()
 sessionID = session['user']
 pusher.trigger(u'session', u'new', {
 u'ip': userIP,
 u'continent': userContinent,
 u'country': userCountry,
 u'city': userCity,
 u'os': userOS,
 u'browser': userBrowser,
 u'session': sessionID,
 u'time': str(time),
 })
 data = [userIP, userContinent, userCountry,
 userCity, userOS, userBrowser, sessionID, time]
 create_session(c, data)
 else:
 sessionID = session['user']
@app.route('/')
def index():
  data = ['home', sessionID, str(datetime.now().replace(microsecond=0))]
  parseVisitor(data)
  return f'User data: {data}'
@app.route('/get-all-sessions')
def get_all_sessions():
  data = []
  dbRows = select_all_sessions(c)
  for row in dbRows:
    data.append({
      'ip': row['ip'],
      'continent': row['continent'],
      'country': row['country'],
      'city': row['city'],
      'os': row['os'],
      'browser': row['browser'],
      'session': row['session'],
      'time': row['created_at']
    })
  return jsonify(data)
if __name__ == '__main__':
  main()
  app.run(debug=True)
```