APPLICATION-BASED NETWORK TRAFFIC GENERATOR FOR NETWORKING AI MODEL DEVELOPMENT

Thesis

Submitted to

The School of Engineering of the

UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for

The Degree of

Master of Science in Computer Engineering

By

Khalil Ibrahim D Alsulami

Dayton, Ohio

May, 2021

APPLICATION-BASED NETWORK TRAFFIC GENERATOR FOR NETWORKING

AI MODEL DEVELOPMENT

Name: Alsulami, Khalil Ibrahim D

APPROVED BY:

Feng Ye, Ph.D.
Advisory Committee Chairman
Assistant Professor, Electrical and Computer Engineering

Tarek M. Taha, Ph.D.
Committee Member
Professor, Electrical and Computer Engineering

John S. Loomis, Ph.D.
Committee Member
Professor Emeritus, Electrical and Computer Engineering

Robert J. Wilkens, Ph.D., P.E.
Associate Dean for Research and Innovation
Professor
School of Engineering

Eddy M. Rojas, Ph.D., M.A., P.E.
Dean, School of Engineering

ABSTRACT

APPLICATION-BASED NETWORK TRAFFIC GENERATOR FOR NETWORKING AI MODEL DEVELOPMENT

Name: Alsulami, Khalil Ibrahim D
University of Dayton

Advisor: Dr. Feng Ye

The growing demands for communication and complex network infrastructure relay on overcoming the network measurement and management challenges. Lately, artificial intelligence (AI) algorithms have considered to improve the network system, e.g., AI-based network traffic classification, traffic prediction, intrusion detection system, etc. Most of the development of networking AI models require abundant traffic data samples to have a proper measuring or managing. However, such databases are rare to be found publicly. To counter this issue, we develop a real-time network traffic generator to be used by network AI models. This network traffic generator has a data enabler that reads data from real applications and establishes packet payload database and a traffic pattern database. The packet payload database has the data packets of real application, where network traffic generator locates the payload in the capture file (PCAP). The other database is traffic pattern database that contains the traffic patterns of a real application. This database depends on the timestamp in each packet and the number of packets in the traffic sample to form a traffic database. The network traffic generator has a built-in network simulator that allows to mimic the real application network traffic flows using these databases to simulate the real-traffic application. The simulator provides a configurable network environment as an interface. To assess our work, we tested the network traffic generator on two network AI models based on simulated traffic, i.e., AI classification model, and AI traffic prediction.

The simulation performance and the evaluation result showed improvement in networking AI models using the proposed network traffic generator, which reduce time consuming and data efficiency challenges.

ACKNOWLEDGMENTS

My words cannot express my deepest thanks for my thesis advisor Dr. Feng Ye of the school of engineering at University of Dayton. The channel to Prof. Professor Yi's channel was always open whenever I ran into a problem or had a question about my research or writing. He consistently allowed this paper to be own minded, but steered me to the right direction whenever thought be needed. I will be always grateful for the time and the effort I had under his supervision.

I would like to thank the Committee members Dr. Tariq Taha and Dr. John Loomis for their generosity in providing their time, support and effort in reviewing this document. I offer my thanks to the head of the department, Dr. Eric J. Blaster and the faculty at the School of Engineering at University of Dayton for their dedication and devotion that have had an important role in my Master's education. My thanks extend to my colleague Mr. Jielun Zhang, PH.D student at the University of Dayton, for his support on networking AI evaluation results.

Finally, my deepest and sincerest gratitude to my father and my mother for their unlimited support and love. I am grateful to my sisters and brothers for always being there for me I am grateful to His Majesty King Salman bin Abdulaziz Al Saud, Custodian of the Two Holy Mosques in the Kingdom of Saudi Arabia, and His Majesty Crown Prince Mohammed bin Salman. I forever owe them all for giving me the opportunity and the experience that have made me who I am.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

Networking artificial intelligence (AI) models are in a challenging race with the growth of network hardware and software machinery [1] [2] [3]. The challenges lie in the lack of available or accessible databases for researchers to enable them to manage and analyze the data [4] [5]. Therefore, there are three main reasons of these challenges. The first reason is the unreliability of the databases on which AI research is based. This is due to the change in network patterns and their continuous updating from time to time. Since there is no essential link between updated network traffic and the existing network traffic databases, it is difficult to predict the readings and results precisely [6] [7]. Therefore, all currently available databases need to be updated regularly for developing networking AI models. The second reason is the lack of an open source generator providing AI researchers with access to up-to-date databases. All the resources that can be used in this type of research either are for commercial purposes or are not available for development purposes. The final reason is that the current available generators are based on particular project or subject that does not serve all aspects of networking AI models.

Given that having updated databases consumes time and effort for networking AI researchers [8] [9], we propose to study the needs of researchers in the networking AI field by joining their previous and current researches[10] [11]. In specific, we focus on reducing the gap and creating a bridge that would allow databases to be directly linked to real-traffic network. The proposed model design allows users to create real-time databases that linked directly with networking AI models, where it has much flexibility in building databases and the ability to analyze and simulate them.

The first level of the proposed model design begins by connecting the generator to a main data source to build an instantaneous database. We design a packet capture file that contains the network traffic to be the main data source. The second level is configuration, in which the network tools, devices, and communication links needed to be structure. For this purpose, network simulator 3 (NS3) libraries are applied as the foundation to the design implementation. NS3 is an open source general purpose network simulator [12]. These libraries can build any type of network design by adding the required communication links, IPs, sockets, etc. It also has functions that allow the user to analyze and simulate the network traffic. For this, we buid a configurator structure that directs the user to set up the network connections without configuring the source codes. The third level is simulator, where the generator reads the input and converts it into a network environment to start transmits packets between network devices. The last level is the output data format, which are needed for networking AI models, e.g., type, size, etc. In addition, the generator is also able to generate as much database length as needed from the main data source. After using this design in a number of networking AI models, the evaluation results demonstrate that the generator can lead to the same results that networking AI models developed from a database collected from real life.

CHAPTER II

RELATED WORK AND BACKGROUND

2.1 Recent Advancement in Network Measurement

The communication and network technologies are developing rapidly [13] [14], which increase the challenge of measuring or managing the network [15] [16]. Although, the involvement of artificial intelligence (AI) has improved the way of measuring and managing the network [17] [18]. However, AI algorithms and deep learning (DL) mostly depends on a huge datasets [19] [20]. Unfortunately, the networking AI developers have few datasets to reach, which made controlling network traffic more challenging [21].

To fulfill the data-hungry needs, we develop a traffic generator enabler based on real network applications for supporting and enhancing networking AI model development [22] [23]. The network traffic classifier, for instance, is one of the interesting networking AI model areas for developers [24] [25]. Authors in [24] [26] [27] used neural networks such as Multilayer Perceptron (MLP) and Convolutional Neural Network (CNN) to build an encrypted network traffic classifiers that use the same dataset (ISCX2012 [28]). Network traffic prediction has subjected to AI model too [29] [30] [31]. Authors in [29] [30] have used long short-term memory (LSTM) neural network to propose AI-based traffic prediction scheme. Clearly, the volume of incoming traffic can be predicted precisely to grant time for network resource allocation. Not only that, AI has a major role in network detection system design [25]. However, latest works show that most of them were using the same datasets. Likewise, the autonomous allocation of resources in virtual networks [31], the virtual network embedding [8], and the network function virtualization in next-generation networks [9].

2.2 Networking AI Models in Network Measurement

AI Algorithms show an encouraging achievements for the modern network [32] [33]. Nonetheless, the development of networking AI model is restrained by updated and adequate datasets [34]. Although network researchers and commercial provide traffic generators, they are based on unreal-life or non-adjustable internet applications [35] [36] [37] [38]. Others are too general to support Networking AI models [12]. Therefore, the motivation of the traffic generator drove this work to insure dataset's flexibility and reliability that mimic a real network traffic and build an AI Algorithm pool to the traffic generator. With the help of NS-3, we developed a real application that enable network traffic generator to moderate the issue of data-hunger in networking AI model development, e.g. the aforementioned algorithms in [24] [30].

The network traffic generator developed to simulate the network traffic on real application packets that followed by the corresponding traffic patterns. The simulated traffic flows can be bridged by the existing AI models[24] [26] [39] [40] [41] [42] to create an interface with the generator. For more flexibility, users have the ability to configure the real application datasets to generate network traffic randomly based on their needs. In addition, the network traffic generator can mimic and expands traffic samples to provide as much as needed datasets and traffic.

Couple of AI-based algorithms were used to implement the network traffic generator, e.g. DL-based packet classification and DL-based traffic volume prediction. The performance of the network traffic generator was measured using different AI models to comprehensively evaluate the result. In both networking AI models the evaluation result shows that DL-based classification model that trained by the network traffic generator was accurately classify the traffic comparing by the that captured in real-life. The LSTM-based prediction model, which trained by the generated traffic, has an accurate predictions using the same databases that generated by the network traffic generator. The way of having simulation process before starting the data in any networking AI models has a significant impact of measuring and managing the dataset. Although there are several applications that networking AI models are used to measuring traffic, the network traffic generator provides a managing and measuring tools to collect, simulate and modifying data traffic.

CHAPTER III

NETWORK TRAFFIC GENERATOR ARCHITECTURE

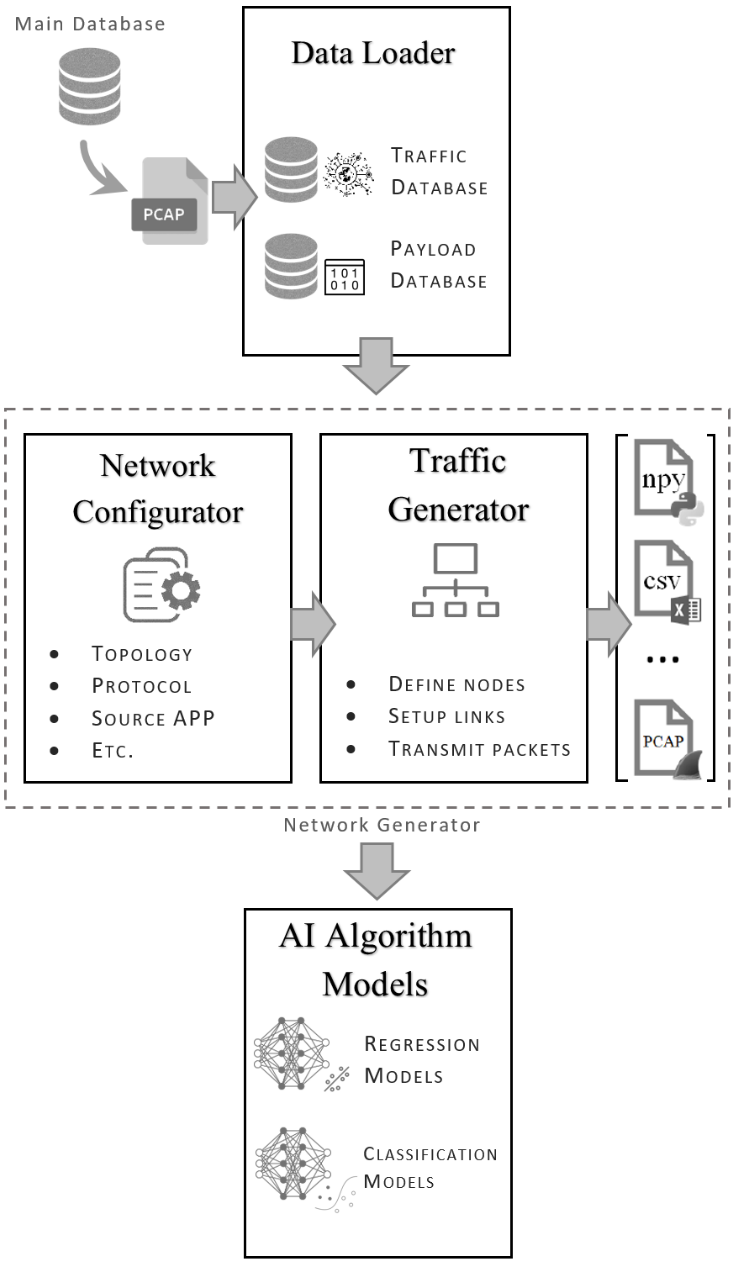


Figure 3.1: Overview of the proposed network traffic generator architecture.

3.1 Overview

As shown in Fig. 3.1, our proposed network traffic generator consists of four stages, i.e., Data Loader, Network Configurator, Traffic Generator and an Interface that bridges the Network AI Models. The data loader pulls in a real-life network traffic from real applications to build the datasets. Then, the configurator builds the network environment based on user's requirements. The traffic generator is to generate the traffic according to the pre-defined configurations, which include network topology, protocol, bandwidth, payload, etc. At the same time, the AI interface registers the traffic that is been simulated in different format to be an input to the AI-based algorithms. These components are comprehensively explained as follow.

3.2 Network Data Packet Payload Database

The first proposal component for the traffic generator is the data loader. From real applications, the data loader downloads the main database. Basically, the data loader reads the main database that contains the real application network traffic, which is stored into Packet Capture file format (PCAP file). This file holds raw data that consists of packets, IPs, sockets, payloads, ports, etc.

Fig. 3.2 shows the general structure of a PCAP includes the global header, the record headers and the data of each packet. Global header's data size is 24 bytes, which represents, e.g., magic number, the version numbers, GMT to local correction, accuracy of timestamps, etc. Data loader read this part of the PCAP file for time correction and accuracy to provide a reliable data form. The rest of the file size holds the network traffic into packets, where each packet has two parts. The first part is the record header with 16 bytes data size,

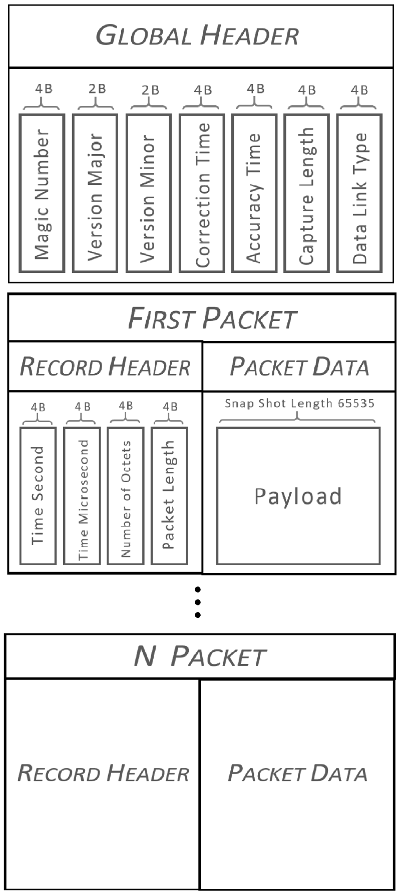
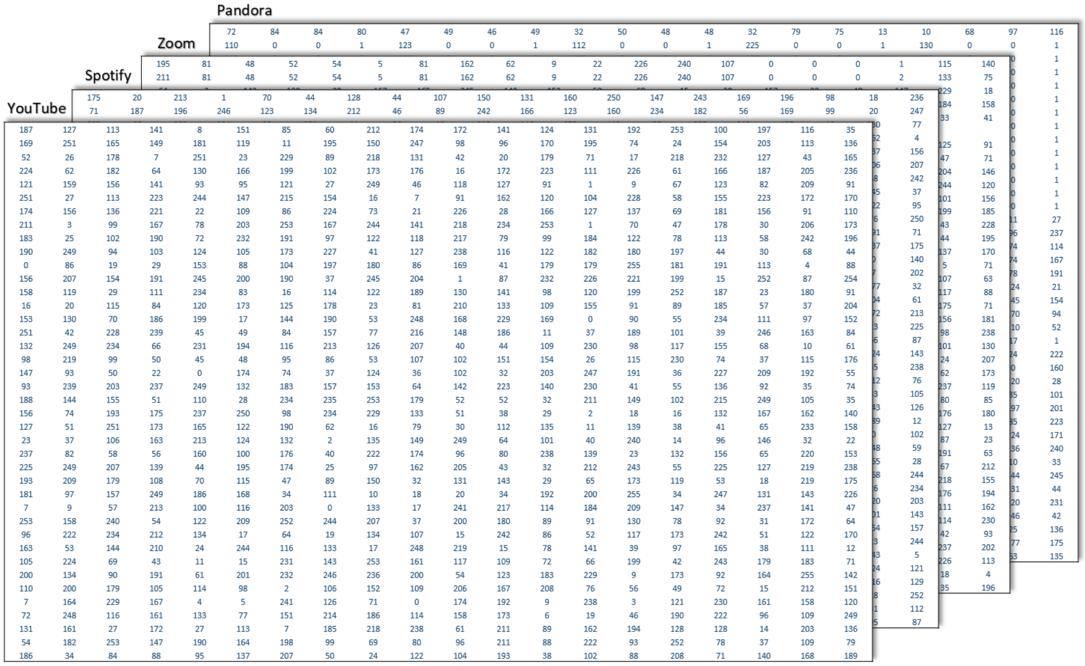


Figure 3.2: PCAP file structure overview.

which includes packet's timestamp in seconds, packet's timestamp in microseconds, and the number of octets of the saved packet. The following packet's data size is where the actual data contents are stored, which known as the payload.

The payload database contains the data of each packets that is been extracted from the real application PCAP file. In particular, the data loader revise the payload of each packet to clip down the IP and socket headers, which contains the addressing and routing information. Note that the PCAP stores data in the form of bitstreams, which can be converted into a character, hexadecimal, or decimal form. The maximum transmission unit



Pandora

72	84	84	80	47	49	46	49	32	50	48	48	32	79	75	13	10	68	97	116
110	0	0	1	123	0	0	1	112	0	0	1	225	0	0	1	130	0	0	1

Zoom

195	81	48	52	54	5	81	162	62	9	22	226	240	107	0	0	0	1	115	140
211	81	48	52	54	5	81	162	62	9	22	226	240	107	0	0	0	2	133	75

Spotify

175	20	213	1	70	44	128	44	107	150	131	160	250	147	243	169	196	98	18	236
71	187	196	246	123	134	212	46	89	242	166	123	160	234	182	56	169	99	20	247

YouTube

187	127	113	141	8	151	85	60	212	174	172	141	124	131	192	253	100	197	116	35
169	251	165	149	181	119	11	195	150	247	98	96	170	195	74	24	154	203	113	136
52	26	178	7	251	23	229	89	218	131	42	20	179	71	17	218	232	127	43	165
224	62	182	64	130	166	199	102	173	176	16	172	223	111	226	61	166	187	205	236
121	159	156	141	93	95	121	27	249	46	118	127	91	1	9	67	123	82	209	91
251	27	113	223	244	147	215	154	16	7	91	162	120	104	228	58	155	223	172	170
174	156	136	221	22	109	86	224	73	21	226	28	166	127	137	69	181	156	91	110
211	3	99	167	78	203	253	167	244	141	218	234	253	1	70	47	178	30	206	173
183	25	102	190	72	232	191	97	122	118	217	79	99	184	122	78	113	58	242	196
190	249	94	103	124	105	173	227	41	127	238	116	122	182	180	197	44	30	68	44
0	86	19	29	153	88	104	197	180	86	169	41	179	179	255	181	191	113	4	88
156	207	154	191	245	200	190	37	245	204	1	87	232	226	221	199	15	252	87	254
158	119	29	111	234	83	16	114	122	189	130	141	98	120	199	252	187	23	180	91
16	20	115	84	120	173	125	178	23	81	210	133	109	155	91	89	185	57	37	204
153	130	70	186	199	17	144	190	53	248	168	229	169	0	90	55	234	111	97	152
251	42	228	239	45	49	84	157	77	216	148	186	11	37	189	101	39	246	163	84
132	249	234	66	231	194	116	213	126	207	40	44	109	230	98	117	155	68	10	61
98	219	99	50	45	48	95	86	53	107	102	151	154	26	115	230	74	37	115	176
147	93	50	22	0	174	74	37	124	36	102	32	203	247	191	36	227	209	192	55
93	239	203	237	249	132	183	157	153	64	142	223	140	230	41	55	136	92	35	74
188	144	155	51	110	28	234	235	253	179	52	52	32	211	149	102	215	249	105	35
156	74	193	175	237	250	98	234	229	133	51	38	29	2	18	16	132	167	162	140
127	51	251	173	165	122	190	62	16	79	30	112	135	11	139	38	41	65	233	158
23	37	106	163	213	124	132	2	135	149	249	64	101	40	240	14	96	146	32	22
237	82	58	56	160	100	176	40	222	174	96	80	238	139	23	132	156	65	220	153
225	249	207	139	44	195	174	25	97	162	205	43	32	212	243	55	225	127	219	238
193	209	179	108	70	115	47	89	150	32	131	143	29	65	173	119	53	18	219	175
181	97	157	249	186	168	34	111	10	18	20	34	192	200	255	34	247	131	143	226
7	9	57	213	100	116	203	0	133	17	241	217	114	184	209	147	34	237	141	47
253	158	240	54	122	209	252	244	207	37	200	180	89	91	130	78	92	31	172	64
96	222	234	212	134	17	64	19	134	107	15	242	86	52	117	173	242	51	122	170
163	53	144	210	24	244	116	133	17	248	219	15	78	141	39	97	165	38	111	12
105	224	69	43	11	15	231	143	253	161	117	109	72	66	199	42	243	179	183	71
200	134	90	191	61	201	232	246	236	200	54	123	183	229	9	173	92	164	255	142
110	200	179	105	114	98	2	106	152	109	206	167	208	76	56	49	72	15	212	151
7	164	229	167	4	5	241	126	71	0	174	192	9	238	3	121	230	161	158	120
72	248	116	161	133	77	151	214	186	114	158	173	6	19	46	190	222	96	109	249
131	161	27	172	27	113	7	185	218	238	61	211	89	162	194	128	128	14	203	136
54	182	253	147	190	164	198	99	69	80	96	211	88	222	93	252	78	37	109	79
186	34	84	88	95	137	207	50	24	122	104	193	38	102	88	208	71	140	168	189

Figure 3.3: Multi-traffic database samples.

for Ethernet is 1500 bytes, and most of packets are less than 1480 bytes [43], which led the interface pads the packets with zeros to a fixed length of 1480 bytes. Fig. 3.3 shows several payload database examples, where each eight bit-stream represent a decimal value from 0 to 255. Each row defines a data packet that is zero padded to 1480 bytes.

3.3 Network Traffic Database

The real-time network traffic represents the statistics of network traffic flows. Fig. 3.4 shows an example traffic flow statistics of YouTube video streaming captured by Wireshark. In order to mimic the real traffic, the data loader must extract each packet with its time flag. Therefore, the traffic database designed to characterize the timeline of each packet. Since the PCAP contains the timestamp for each packet, the data loader calculates every packet's timestamp by taking the difference between the packet's intervals. This calculated values stored in the traffic database based on the number of packets in the given unit of time, e.g., seconds, milliseconds, microseconds, which also known as the packet rate. During the traffic generation, the traffic database acts as packet's schedule to imitate the real-traffic. For that, the network traffic generator generats the exact network traffic for each application. However, this is not the only way to take advantage of the traffic database. The data loader provides an option to imitate the real-traffic in different timeline.

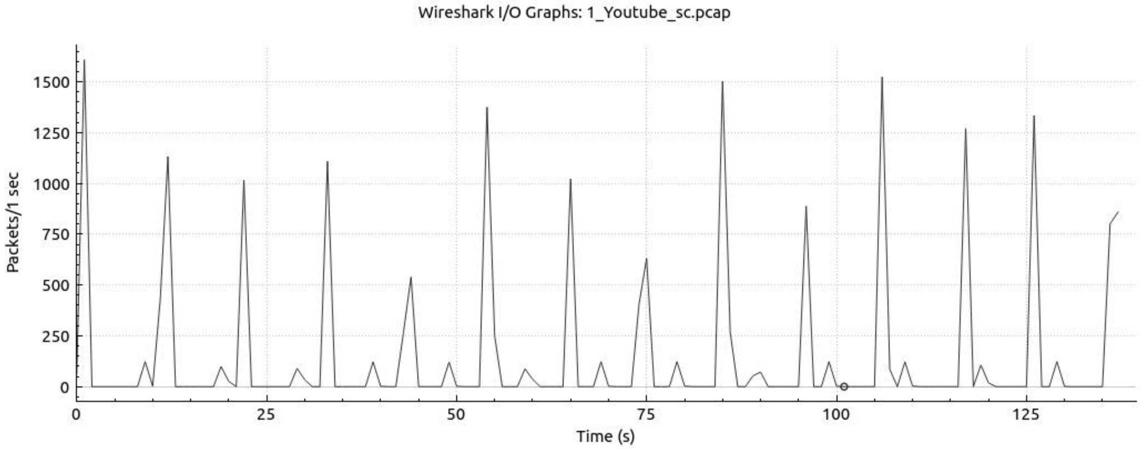


Figure 3.4: Sample clip of YouTube video traffic captured from Wireshark.

A productive way of generating the traffic that support the networking AI models is the ability of modifying the timeline of the network traffic. The network traffic generator designed to accept different timeline traffic parameter from the user to extend or compress the real-traffic. The idea is the generator can generates the same payload and traffic database in different timeline range from the original timeline. However, the network traffic in real applications are mostly random, which increases the challenge of preserving the real traffic and regenerate it in different shapes without compromising the traffic characteristics.

The proposed generator algorithm Double Index Sort (DIS). designed to navigate the real application sample to find the highest peaks in the traffic database sequence. Therefore, the data loader uses the algorithm to clip down the real-traffic into several segments. The segmentation process are based on finding the highest peaks and store the traffic values between the founded peak to the next peak. Upon request, the generator generates from the stored segmentations randomly as much as needed, e.g., seconds, minutes, hours.

```
input  : Two arrays $A[TrafficIndex]$
output: Array $Index[k]$
begin
    $Visited[TrafficIndex] = true$;
    for $i \leftarrow 0$ to Traffic Index do
        $B[HighestIndex] \leftarrow$ Selection Sort $A[i]$;
        $C[HighestIndex] \leftarrow i - HighestIndex$;
    end
    for $i \leftarrow 0$ to $HighestIndex$ do
        $S \leftarrow \infty$;
        $D \leftarrow \infty$;
        for $j \leftarrow 0$ to $HighestIndex$ do
            if $S > j$ and $D > C[j]$ and $Visited = false$ then
                $S \leftarrow j$;
                $D \leftarrow C[j]$;
            end
        end
        $Visited[S] = true$;
        $Index[k] = D$;
    end
end
```

Algorithm 1: Double Index Sort (DIS).

The DIS algorithm is proposed to find the segments in both periodic and pseudo-periodic. Although the generated traffic will mimic the real-traffic, the segmentation of each application will be different. The algorithm build to group the traffic starting from the highest peak to the end of the next highest peak, and store it into a segment. However, when the traffic's index has already joined into previous segments, the generator will skip the traffic to avoid traffic duplication. For Example, the YouTube traffic database in the Table 3.1 has 13 segments, where each segment has 9 to 11 time-line traffic. DIS separates each segments and store it in order to be regenerated by the user. Ultimately, the user can generate as much traffic time-line as needed from the real-time sample network traffic. In the YouTube sample, the DIS founds 13 segments, where each segments has different range of traffic values from N0 to Nn. The N0 contains the highest traffic value of each traffic segment. However, DIS adds traffics when two traffic values in-line. For example, segment 5, 6, 8, 9, and 11 has two traffic values more than zero at second N0 and N1. These segments will treat both seconds as one to insure that segments has the right highest peaks boundaries.

Table 3.1: A clip of the traffic database for YouTube.

Segments	$N0$	$N1$	$N2$	$N3$	$N4$	$N5$	$N6$	$N7$	$N8$	$N9$	$N10$
Seg1	1608	0	0	0	0	0	0	0	124	1	..
Seg2	1565	0	0	0	0	0	0	122	3	0	..
Seg3	1016	0	0	0	0	0	122	1	0	0	..
Seg4	1109	0	0	0	0	0	124	0	0	0	..
Seg5	331	474	0	0	0	0	122	1	0	0	0
Seg6	1472	153	0	0	0	123	1	0	0	0	0
Seg7	1022	0	0	0	124	1	0	0	0	..	..
Seg8	510	525	0	0	0	124	2	0	0	0	0
Seg9	1591	180	0	0	123	2	0	0	0	0	0
Seg10	888	0	0	124	1	0	0	0	0	0	..
Seg11	1609	1	0	124	1	0	0	0	0	0	0
Seg12	1270	0	123	1	0	0	0	0	0	..	..
Seg13	1334	0	0	124	0	0	0	0	0	0	..

3.4 Network Topology Configuration

The second proposal component for the traffic generator is the network configurator. The configurator is built on the Network Simulator (NS-3) [12], which is an open source simulation environment for networking. The configurator allows the user to pre-define the network environment. The configurable elements are listed in Table 3.2 for a flexible network configuration. By using an interface between the user and the simulator, the configurator reads the network specification that is pre-formatted to design the network environment. These customizable and pre-defined parameters are available for users as the following format to build the network devices specification Fig. 3.5.

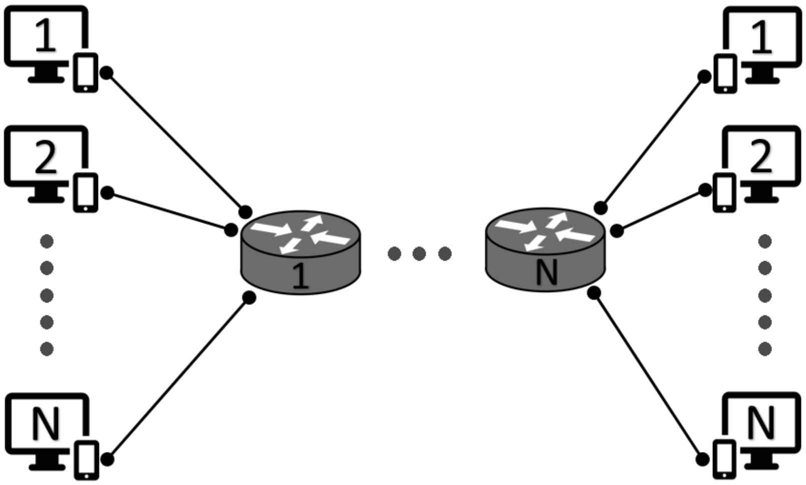


Figure 3.5: Network devices connection overview.

The configurator downloads these specifications or parameters to setup the communication links inputs, which will be translated into a network simulator. This simulator will define the network tools and set up the communication links. The configurator pre-defines ten range IPv4 addresses as an interface stacks. These IPs will be assigned to nodes accordingly between IP range from 10.1.1.0 to 10.1.10.0 during the simulation process. However, user can change or add extra IPs as needed for nodes. The overall network configuration simulator can generate an arbitrary user-defined traffic. It is worth to mention that even if the payload database or the traffic database are running out, the generator will keep generating traffic until reached the Sink End time that is defined by the user.

Table 3.2: Configuration setup of communication links.

Parameter	Description
Source	Network nodes 0 to N-1, where N is the total number of source nodes in the simulation.
Destination	Network nodes 0 to N-1, where N is the total number of destination nodes in the simulation.
Socket	The channel type between the source nodes to the destination nodes.
Bandwidth	The amount of data that is been allowed to be transmitted over the link, e.g., 100Mbps, 10Kbps, etc.
Delay	The communication delay over the link for the simulation, e.g., 1s, 10ms, 0.001ms, etc.
Port	A unique port number for the node to identify the transport protocol such as 8080.
Payload	The source application (APP) of the payload database, e.g., Youtube.txt, Zoom.txt, etc.
Sink Start	The start time of synchronizing the communication link in seconds.
Sink End	The end time of synchronizing the communication link in seconds.

3.5 Network Traffic Flow Configuration

The second proposal component for the traffic generator is the traffic generator. The traffic generator is the main component where customized traffic generation located. Accordingly, the generator builds the network environment, which includes the network devices and the communication setting following the configuration inputs Table 3.3.

3.5.1 Topology Setting

The setting of the topology are represted into a text file, where each line represented a communication link with its application, e.g., Table reftab:class4. Once the generator read these lines, it will build the network topology and setup the network environment for simulation. The generator then will setup the communication's pair links between the source and destination nodes. Substantially, the generator works as a traffic control to the network devices based on the given configuration by sending packets from source nodes to destination nodes carrying the selected payload and traffic databases.

Table 3.3: Input text file example of configuration setup of communication links.

APP No.	Src	Des.	Socket	BW	Delay	Port	Payload	Sink Start	Sink End
1	0	5	0	100Mbps	1ms	6	YouTube	0	200
2	1	5	1	100Mbps	1ms	8080	Netflix	201	400
3	2	5	1	100Mbps	1ms	8080	Spotify	401	600
4	3	6	0	100Mbps	1ms	6	Pandora	601	800
5	4	6	1	100Mbps	1ms	8080	Twitch	801	1000

The traffic generator includes an interface that provides default and manual setting. These options add more flexibility to generate and simulate the network traffic per user requirements. Both options are consistent with the networking AI models to have as much desired outcome. The default setting is built to randomly simulate the traffic, where payload and traffic database, source, distention, and protocol are based default inputs. However, the manual settings give the user the full control of the configuration, which includes databases, nodes, protocols, number of packets, timeline, etc. During the simulation, users will be notified about the configuration process that is been simulated by the generator including the payload and traffic database.

3.5.2 Traffic Statistic Settings

The network traffic generator provides a statistic setting with default and customize options. The default setting is built to read the pre-defined inputs, e.g., traffic database, payload database, source, destination, etc., and provide the result to the user. The customize opthion provides a platform for the user to pick the statistic setting such as number of packets, traffic timeline, number of sources, database type and format.

3.6 AI based Network Models

The outcome of the network traffic generator bridged with AI based network models to offer the support of performing and certifying the networking AI model development. To examine the network traffic generator, we bind the generator with two well-known networking AI algorithms that are used for measuring and managing the network.

The first AI algorithm that is used in the network traffic algorithm is classification model, which is based Multilayer Perceptron (MLP). MLP used as a neural network in Deep Learning, which contains the input layer, hidden layers, and the output layer. The input layer will download the needed files from the network traffic generator that contain the packet payload. The generator can provide the desire input format to the MLP in order to setup the second layer, which is the hidden layer, that compute and extract the input features using linear computation and non-linear activation. The output layer then defines the result of the traffic classification by calculating the categorical probabilities of each corresponding application class. Fig 3.6 shows an overview of the implemented MLP model

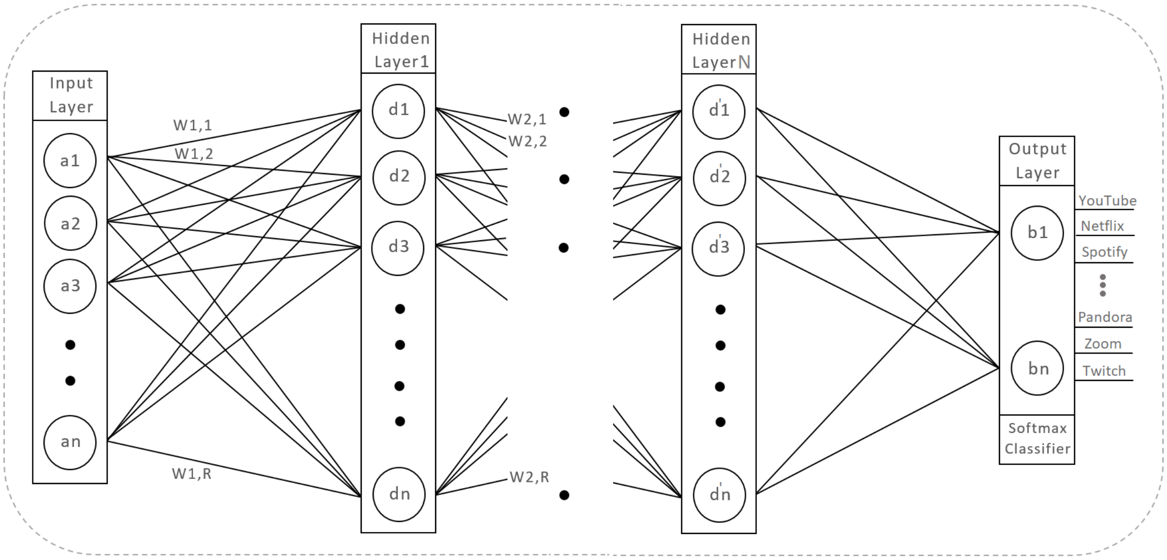


Figure 3.6: Overview of MLP based network traffic classifier.

In details, If a is an input payload packets, and b is the corresponding of the application then the classification function is $f : a \to b$. To clarify, the neural network is can be introduced as the following:

$$d_i = \sigma(d_{i-1} \cdot w_i^T + v_i), i \geq 2, \tag{3.1}$$

where i is the hidden layer index, and d_i is the potential vector in the i_{th} layer, w_i and v_i are the weight vector and the alignment in the i_{th} layer, and σ is the activation non-linear function. For the output layer, the Softmax function $S(.)$ compute the cross entropy values as follows:

$$\mathbf{p_i} = S(d_J) = \frac{\exp d_J^i}{\sum_{i=1}^{T} \exp d_J^i}, \tag{3.2}$$

Where J is the last hidden layer index, and T is the total number of classes in scope of applications. The maximum element of the probability vector $\mathbf{p} = p_1, p_2, ..., p_m$ will be classified as the foloowing result., i.e,

$$q = \arg\max_{p_i \in \mathbf{p}} \tag{3.3}$$

The second AI algorithm that is used in the network traffic algorithm is the short-term traffic prediction Fig. 3.7. It is a traffic prediction model that developed based on LSTM neural network. LSTM has the ability to preserves memory using the additional gates. It has an input layer, a hidden layer with memory, and an output layer. The memory cell are participating three operation gates the input gates, the forget gate and the output gate. The state value of long or shor-time stores into LSTM cells. Fig. 3.8 shows the structure of one single LSTM cell.

The following equations explains each part of the forget gate, current state, memory cell, and the output gate, where w and v are the weight matrices associated with corresponding gates or components.

Forget Gate:

$$f_t = \sigma(w_f[h_{t-1}, x_t] + v_f), \tag{3.4}$$

Input Gate:

$$i_t = \sigma(w_i[h_{t-1}, x_t] + v_i), \tag{3.5}$$

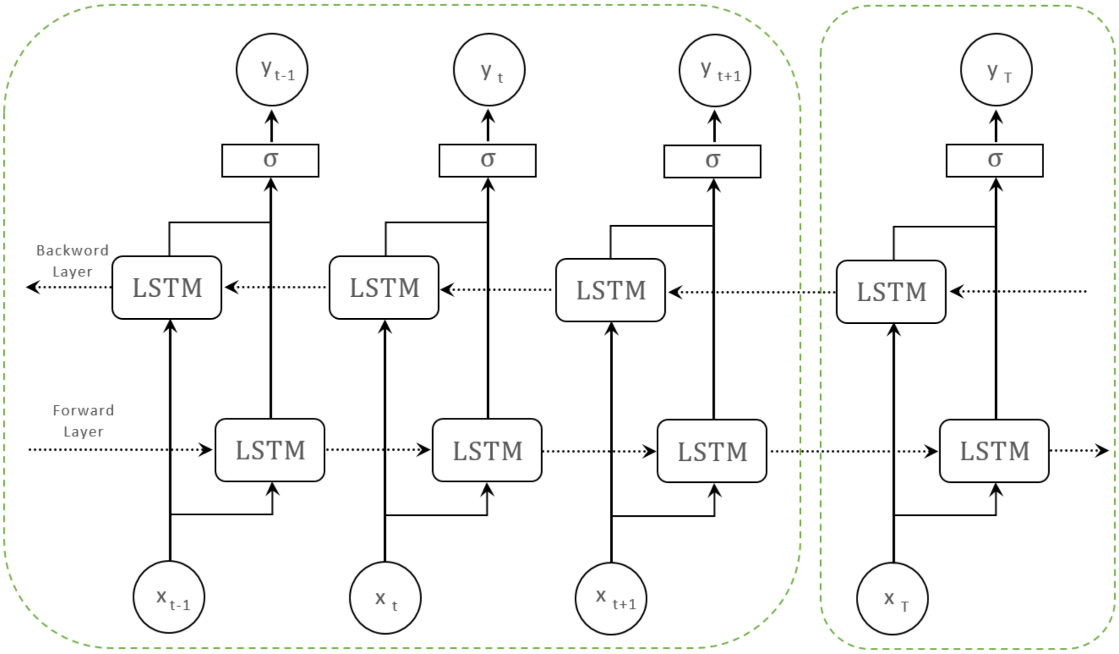


Figure 3.7: Overview of basic LSTM system.

Current State:

$$\hat{c}_t = tanh(w_c[h_{t-1}, x_t] + v_c), \tag{3.6}$$

$$MemoryCell : c_t = (f_t \cdot c_{t-1}) + (i_t \cdot \hat{c}_t), \tag{3.7}$$

Output Gate:

$$o_t = \sigma(w_o[h_{t-1}, x_t] + v_o), \tag{3.8}$$

$$h_t = o_t \cdot tanh(c_t), \tag{3.9}$$

The feature of the network traffic generator is the variousity of data-shape that support the networking AI models. During the research, we were involved with several networking AI models to fulfill data requirements, e.g., format, size, and form, which serve the regression tasks in networking, e.g., traffic prediction. Therefore, the generator can produce a list of data in file format as shown in the Table 3.4.

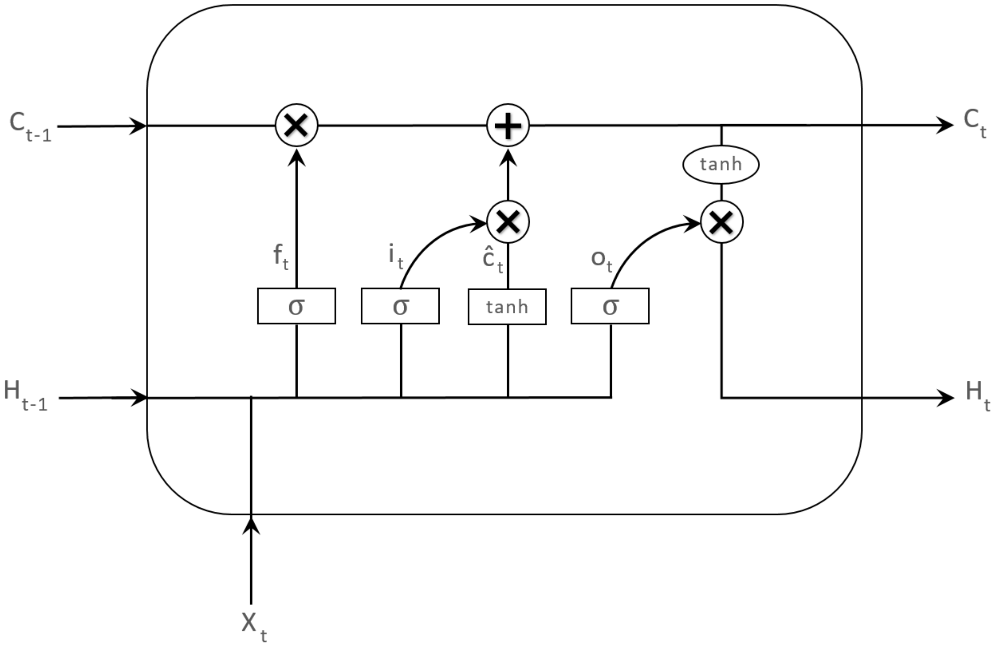


Figure 3.8: Overview of single LSTM memory cell.

Table 3.4: Type of data file format for networking AI models.

File Format	Description
PCAP	Similar to the real application file with different IPs, sockets, and/or datasize.
NPY	A 3D matrix file that hold the payload in decimal value for training purposes.
CSV	Contain the Payload and the traffic Database.
XML	Contains the simulation process.

CHAPTER IV

EVALUATION RESULTS

4.1 Evaluation Settings

The relevant of the network traffic generator to the networking AI models is a combination between two major elements the data and network structures, the material that were used has to be efficient and effective. In practice, a set of network libraries that predefine the PCAP structure were used to restore the data before being stored in separate databases. However, the network traffic generator was built on LINUX system using NS-3 libraries called ns-allinone-3.29, where both set of libraries knowingly used because of the desirable features variety conjunction, interoperability, memory management, debugging, C++ coding and object oriented concept.

4.2 Database

the traffic database and the payload database are what the traffic generator depends on for generating a real-time traffic. Networking AI models relay on the accuracy of these two database as inputs to their models. Therefore, throughout the research the generator has tested tens of traffic samples from different applications. Table 4.1 shows different applications with different traffic length. In this section, both databases are explicitly explained to show the effectiveness of them.

4.2.1 Payload Database

Each packets contains a payload, where the data stored, as previously shown in Fig. 3.2. These data are represented in bit-stream 0s and 1s. In order to be reloaded into a database,

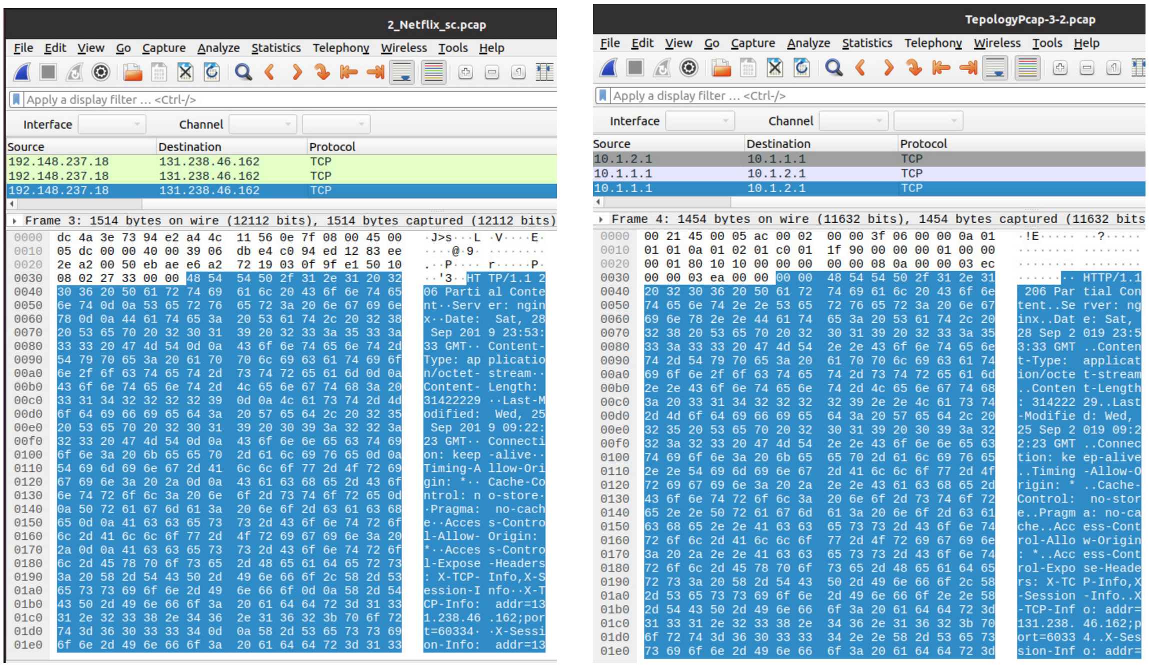
Table 4.1: Collected dataset applications overview

Application Name	Protocol	Sampled Records / Total Records
Youtube Video	UDP	20,000/23,863
Netflix	TCP	20,000/102,513
Spotify	TCP	20,000/318,462
YouTube Music	UDP	20,000/51,952
Pandora	TCP	20,000/61,865
Amazon Prime Video	TCP	20,000/3,242,785
Twitch	TCP	20,000/363,744
Discord	UDP	20,000/219,427
Zoom	UDP	20,000/255,492
Tiktok	UDP	20,000/506,950

the network traffic generator reads the PCAP file structure until locating the starts of the payload. Then, the generator stores every eight bits into decimal number. However, most network traffic applications, e.g., Wireshark, interpreted the payload into an ASCCI code form, e.g., Netflix traffic in Fig. 4.1a. Although ASCCI does not include the most of the character, the byte that does not match any character will stored as dots (.). To compare the network traffic generator with these application, the Fig. 4.1b shows the matches between the payload from the Wireshark application and the generator payload database. In addition, we can see the simulation process has reflected to the traffic protocols, e.g., IPs, packet length, etc. However, the payload that highlighted in both figures are the same.

4.2.2 Traffic database

In each packet, the timestamp defines the arrival time for a packet as shown in Fig. 3.2. The PCAP file has two part of defining the timestamp. The first timestamp is a second, which represented in Little-endian format in a form of (UNIX time) or also known as Epoch. For example, where years, days, hours, minutes and seconds are interpreted from Epoch time


2_Netflix_sc.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Interface
Channel

```
Source           Destination      Protocol
192.148.237.18   131.238.46.162   TCP
192.148.237.18   131.238.46.162   TCP
192.148.237.18   131.238.46.162   TCP

Frame 3: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
0000  dc 4a 3e 73 94 e2 a4 4c  11 56 0e 7f 08 00 45 00   ·J>s···L ·V····E·
0010  05 dc 00 00 40 00 39 06  db e4 c0 94 ed 12 83 ee   ····@·9· ········
0020  2e a2 00 50 eb ae e6 a2  72 19 03 0f 9f e1 50 10   .··P···· r·····P·
0030  08 02 27 33 00 00 48 54  54 50 2f 31 2e 31 20 32   ··'3··HT TP/1.1 2
0040  30 36 20 50 61 72 74 69  61 6c 20 43 6f 6e 74 65   06 Parti al Conte
0050  6e 74 0d 0a 53 65 72 76  65 72 3a 20 6e 67 69 6e   nt··Serv er: ngin
0060  78 0d 0a 44 61 74 65 3a  20 53 61 74 2c 20 32 38   x··Date:  Sat, 28
0070  20 53 65 70 20 32 30 31  39 20 32 33 3a 35 33 3a    Sep 201 9 23:53:
0080  33 33 20 47 4d 54 0d 0a  43 6f 6e 74 65 6e 74 2d   33 GMT·· Content-
0090  54 79 70 65 3a 20 61 70  70 6c 69 63 61 74 69 6f   Type: ap plicatio
00a0  6e 2f 6f 63 74 65 74 2d  73 74 72 65 61 6d 0d 0a   n/octet- stream··
00b0  43 6f 6e 74 65 6e 74 2d  4c 65 6e 67 74 68 3a 20   Content- Length: 
00c0  33 31 34 32 32 32 32 39  0d 0a 4c 61 73 74 2d 4d   31422229 ··Last-M
00d0  6f 64 69 66 69 65 64 3a  20 57 65 64 2c 20 32 35   odified:  Wed, 25
00e0  20 53 65 70 20 32 30 31  39 20 30 39 3a 32 32 3a    Sep 201 9 09:22:
00f0  32 33 20 47 4d 54 0d 0a  43 6f 6e 6e 65 63 74 69   23 GMT·· Connecti
0100  6f 6e 3a 20 6b 65 65 70  2d 61 6c 69 76 65 0d 0a   on: keep -alive··
0110  54 69 6d 69 6e 67 2d 41  6c 6c 6f 77 2d 4f 72 69   Timing-A llow-Ori
0120  67 69 6e 3a 20 2a 0d 0a  43 61 63 68 65 2d 43 6f   gin: *·· Cache-Co
0130  6e 74 72 6f 6c 3a 20 6e  6f 2d 73 74 6f 72 65 0d   ntrol: n o-store·
0140  0a 50 72 61 67 6d 61 3a  20 6e 6f 2d 63 61 63 68   ·Pragma:  no-cach
0150  65 0d 0a 41 63 63 65 73  73 2d 43 6f 6e 74 72 6f   e··Acces s-Contro
0160  6c 2d 41 6c 6c 6f 77 2d  4f 72 69 67 69 6e 3a 20   l-Allow- Origin: 
0170  2a 0d 0a 41 63 63 65 73  73 2d 43 6f 6e 74 72 6f   *··Acces s-Contro
0180  6c 2d 45 78 70 6f 73 65  2d 48 65 61 64 65 72 73   l-Expose -Headers
0190  3a 20 58 2d 54 43 50 2d  49 6e 66 6f 2c 58 2d 53   : X-TCP- Info,X-S
01a0  65 73 73 69 6f 6e 2d 49  6e 66 6f 0d 0a 58 2d 54   ession-I nfo··X-T
01b0  43 50 2d 49 6e 66 6f 3a  20 61 64 64 72 3d 31 33   CP-Info:  addr=13
01c0  31 2e 32 33 38 2e 34 36  2e 31 36 32 3b 70 6f 72   1.238.46 .162;por
01d0  74 3d 36 30 33 33 34 0d  0a 58 2d 53 65 73 73 69   t=60334· ·X-Sessi
01e0  6f 6e 2d 49 6e 66 6f 3a  20 61 64 64 72 3d 31 33   on-Info:  addr=13
```

(a) Wireshark real-traffic sample (Netflix).

TepologyPcap-3-2.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Interface
Channel

```
Source      Destination   Protocol
10.1.2.1    10.1.1.1      TCP
10.1.1.1    10.1.2.1      TCP
10.1.1.1    10.1.2.1      TCP

Frame 4: 1454 bytes on wire (11632 bits), 1454 bytes captured (11632 bits
0000  00 21 45 00 05 ac 00 02  00 00 3f 06 00 00 0a 01   ·!E····· ··?·····
0010  01 01 0a 01 02 01 c0 01  1f 90 00 00 00 01 00 00   ········ ········
0020  00 01 80 10 10 00 00 00  00 00 08 0a 00 00 03 ec   ········ ········
0030  00 00 03 ea 00 00 00 00  48 54 54 50 2f 31 2e 31   ········ HTTP/1.1
0040  20 32 30 36 20 50 61 72  74 69 61 6c 20 43 6f 6e    206 Par tial Con
0050  74 65 6e 74 2e 2e 53 65  72 76 65 72 3a 20 6e 67   tent..Se rver: ng
0060  69 6e 78 2e 2e 44 61 74  65 3a 20 53 61 74 2c 20   inx..Dat e: Sat, 
0070  32 38 20 53 65 70 20 32  30 31 39 20 32 33 3a 35   28 Sep 2 019 23:5
0080  33 3a 33 33 20 47 4d 54  2e 2e 43 6f 6e 74 65 6e   3:33 GMT ..Conten
0090  74 2d 54 79 70 65 3a 20  61 70 70 6c 69 63 61 74   t-Type:  applicat
00a0  69 6f 6e 2f 6f 63 74 65  74 2d 73 74 72 65 61 6d   ion/octe t-stream
00b0  2e 2e 43 6f 6e 74 65 6e  74 2d 4c 65 6e 67 74 68   ..Conten t-Length
00c0  3a 20 33 31 34 32 32 32  32 39 2e 2e 4c 61 73 74   : 314222 29..Last
00d0  2d 4d 6f 64 69 66 69 65  64 3a 20 57 65 64 2c 20   -Modifie d: Wed, 
00e0  32 35 20 53 65 70 20 32  30 31 39 20 30 39 3a 32   25 Sep 2 019 09:2
00f0  32 3a 32 33 20 47 4d 54  2e 2e 43 6f 6e 6e 65 63   2:23 GMT ..Connec
0100  74 69 6f 6e 3a 20 6b 65  65 70 2d 61 6c 69 76 65   tion: ke ep-alive
0110  2e 2e 54 69 6d 69 6e 67  2d 41 6c 6c 6f 77 2d 4f   ..Timing -Allow-O
0120  72 69 67 69 6e 3a 20 2a  2e 2e 43 61 63 68 65 2d   rigin: * ..Cache-
0130  43 6f 6e 74 72 6f 6c 3a  20 6e 6f 2d 73 74 6f 72   Control:  no-stor
0140  65 2e 2e 50 72 61 67 6d  61 3a 20 6e 6f 2d 63 61   e..Pragm a: no-ca
0150  63 68 65 2e 2e 41 63 63  65 73 73 2d 43 6f 6e 74   che..Acc ess-Cont
0160  72 6f 6c 2d 41 6c 6c 6f  77 2d 4f 72 69 67 69 6e   rol-Allo w-Origin
0170  3a 20 2a 2e 2e 41 63 63  65 73 73 2d 43 6f 6e 74   : *..Acc ess-Cont
0180  72 6f 6c 2d 45 78 70 6f  73 65 2d 48 65 61 64 65   rol-Expo se-Heade
0190  72 73 3a 20 58 2d 54 43  50 2d 49 6e 66 6f 2c 58   rs: X-TC P-Info,X
01a0  2d 53 65 73 73 69 6f 6e  2d 49 6e 66 6f 2e 2e 58   -Session -Info..X
01b0  2d 54 43 50 2d 49 6e 66  6f 3a 20 61 64 64 72 3d   -TCP-Inf o: addr=
01c0  31 33 31 2e 32 33 38 2e  34 36 2e 31 36 32 3b 70   131.238. 46.162;p
01d0  6f 72 74 3d 36 30 33 33  34 2e 2e 58 2d 53 65 73   ort=6033 4..X-Ses
01e0  73 69 6f 6e 2d 49 6e 66  6f 3a 20 61 64 64 72 3d   sion-Inf o: addr=
```

(b) Wireshark generated-traffic sample (Netflix).

Figure 4.1: Real-traffic vs. generated-traffic in Wireshark (Netflix).

1613268875 to Sunday 14 February 2021 02:14:35. The second timestamp is a microsecond that gave the exact arrival time of the packet. Therefore, the traffic database measures the arrival time for each packet and store the number of packet at the same arrival time. Eventually, the network traffic generator will be capable of generates the traffic database with the number of packet at any given time. To evaluate the results of the PCAP file, Fig. 4.2, Fig. 4.3, and Fig. 4.4 compare the real application network traffic and the traffic database.

4.3 Traffic Flow Simulation

The network traffic generator has several features to assist the networking AI models by analyzing the datasets before being used. Where a user can simulate the data using

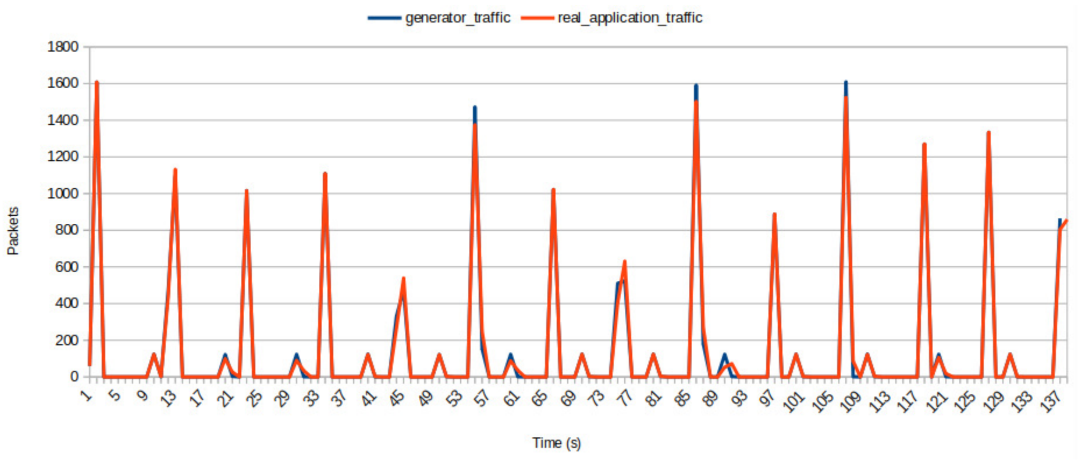


Figure 4.2: Raw traffic statistics and data stored in the traffic database (YouTube).

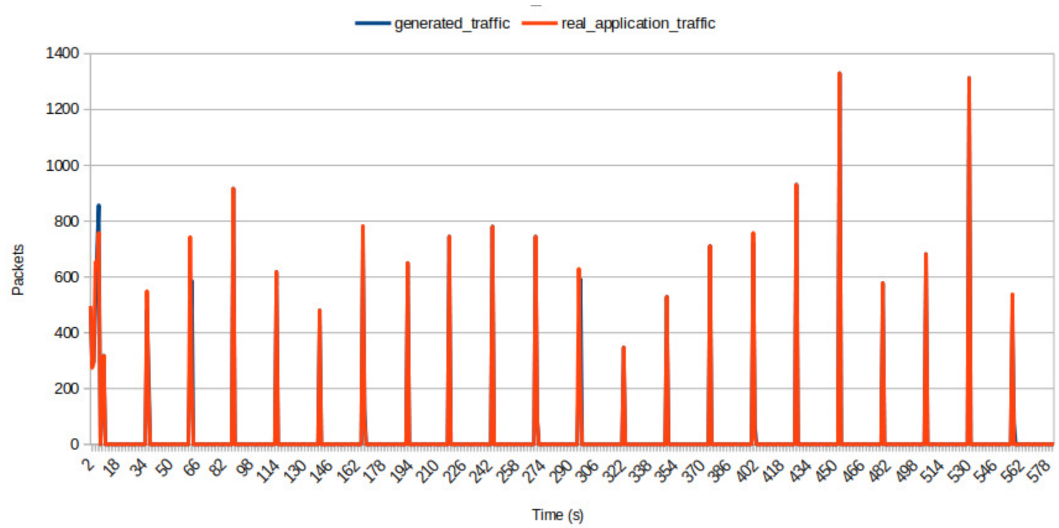


Figure 4.3: Raw traffic statistics and data stored in the traffic database (Netflix).

one or more applications at the same time. In addition to the possibility of generating an unlimited number of databases using a sample of data. These additional features have been studied according to the needs of networking AI.

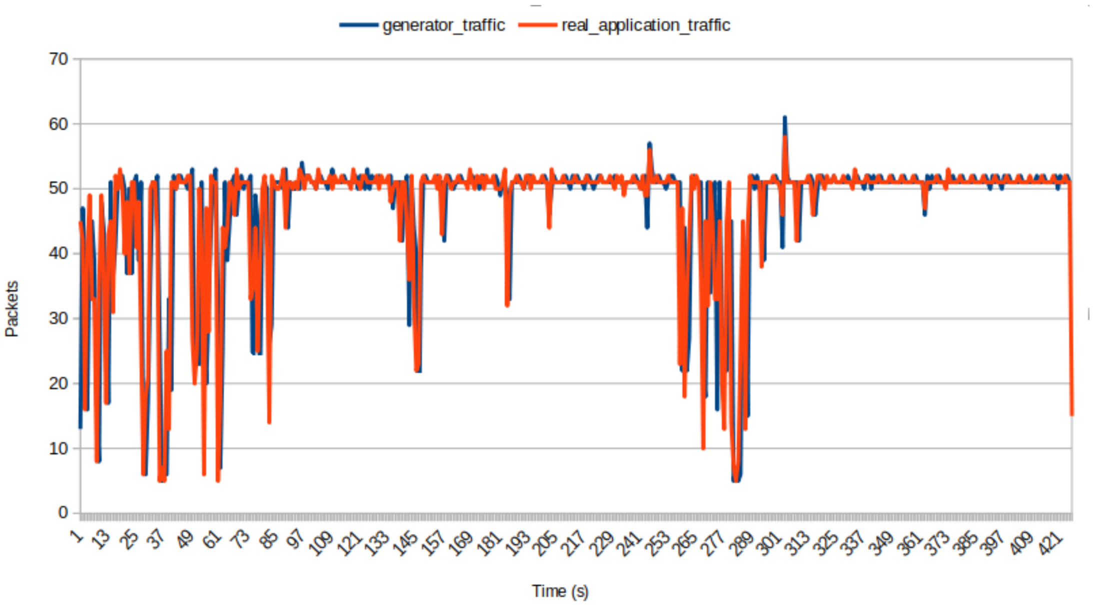


Figure 4.4: Raw traffic statistics and simulated traffic flows (Discord).

4.3.1 Network Topology Setting

The network traffic generator designed to be able to accept one or more applications to simulate network performance by providing inputs manually. Through the input text file, the user can specify one application at each node to be the source of database transmission, e.g., YouTube app at node A, and Netflix app at node B. The user then can specify the destination node, which will act as a receiver. When starting the simulation, the generator simultaneously sends the data from the payload database to the destination. By analyzing the generator output, Fig. 4.5 from Network Animation Software shows that each communication link is handled individually by providing an independent IP, socket, and payload. This is due to the presence of a router built inside the generator to schedule the data transmission between nodes.

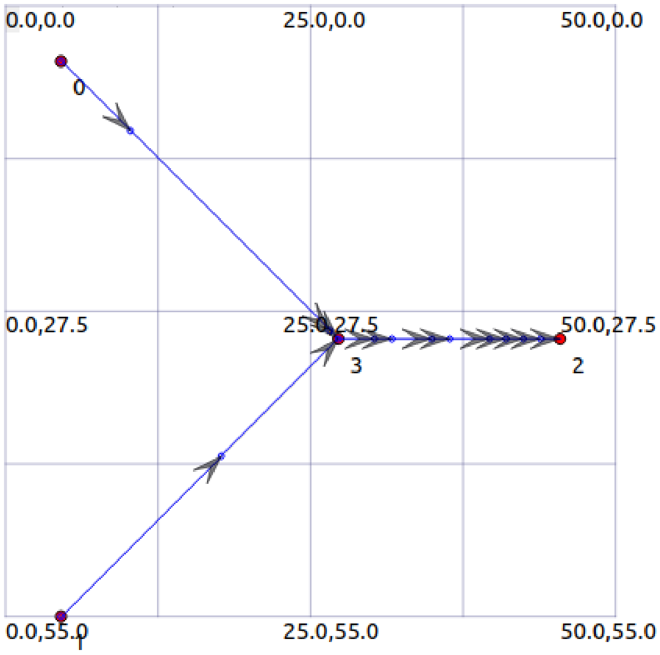
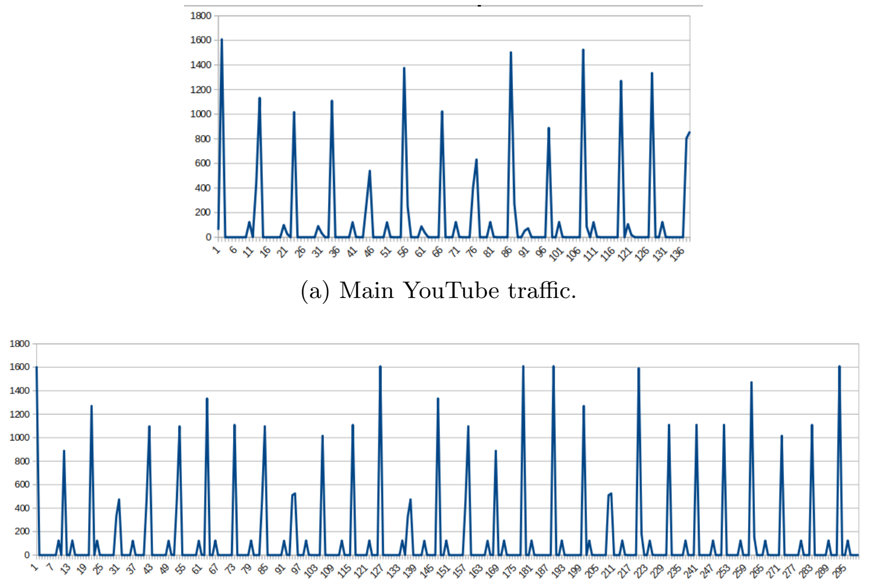


Figure 4.5: NetAnim application: Two different application transmit to one destination.

4.3.2 Traffic Flow Generation

Another feature of the generator is the ability to synthesize the traffic and simulate the traffic databases. This feature relies on finding the highest peaks in the traffic application's database and clipping it into several parts using DIS algorithm that separates the dataset from highest to highest peaks and store them into segments. These segments can be generated as requested by the user by specifying the traffic time-line.

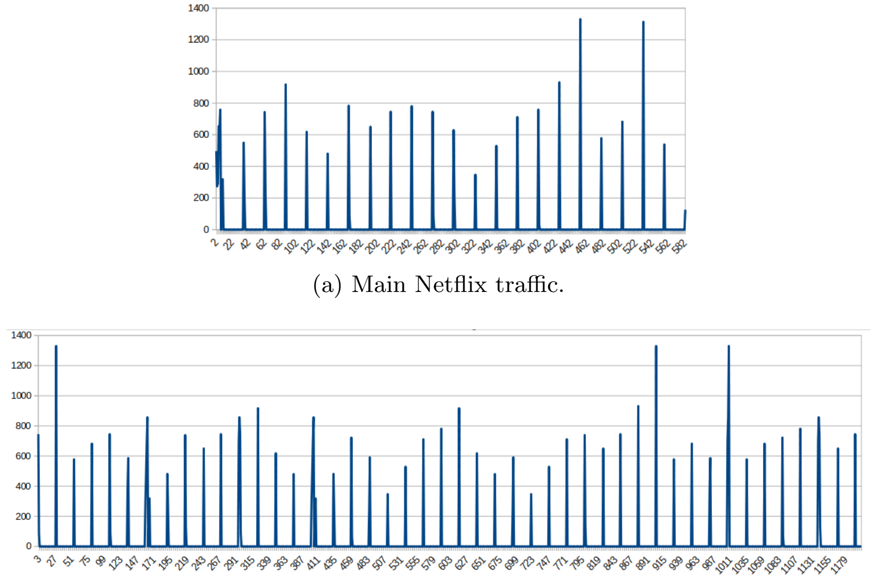
By Appling DIS algorithm to several traffic databases such as YouTube (Fig. 4.6a), Netflix (Fig. 4.7a), AMZprime (Fig. 4.8a), and Discord (Fig. 4.9a) , the generated traffic were imitating the real traffic randomly by doubling the timeline, e.g., YouTube (Fig. 4.6b), Netflix (Fig. 4.7b), AMZprime (Fig. 4.8b), and Discord (Fig. 4.9b).



(a) Main YouTube traffic.

(b) Double YouTube traffic timeline.

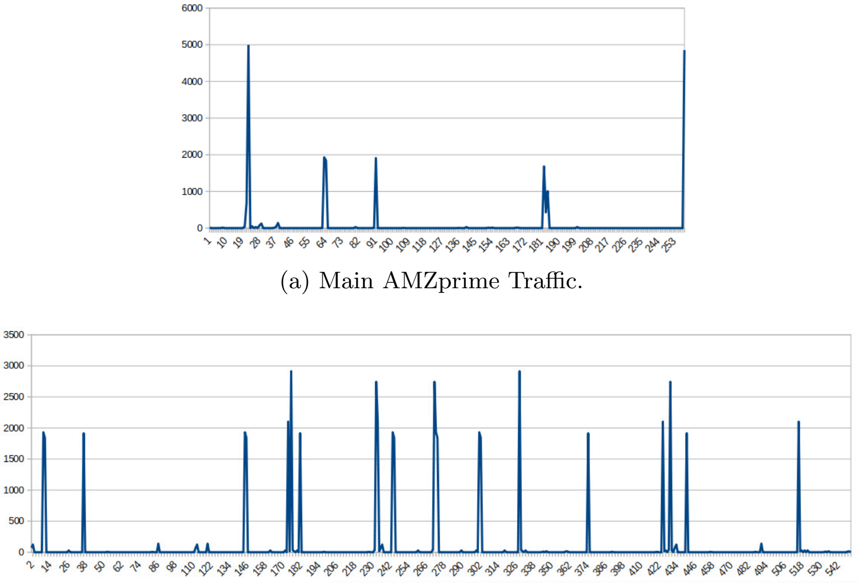
Figure 4.6: Real-time YouTube traffic vs. extended real-time Youtube traffic.



(a) Main Netflix traffic.

(b) Double Netflix traffic timeline.

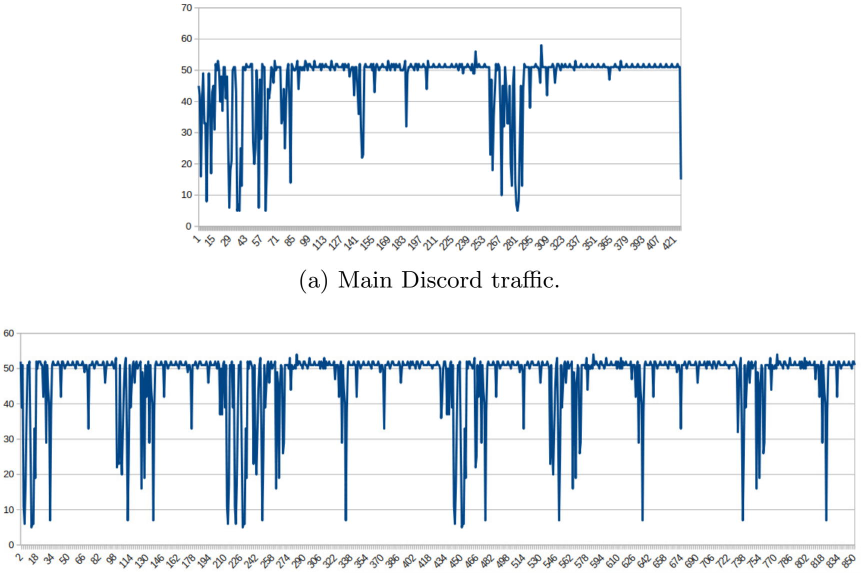
Figure 4.7: Real-time Netflix traffic vs. extended real-time Netflix traffic.



(a) Main AMZprime Traffic.

(b) Double AMZprime traffic timeline.

Figure 4.8: Real-time AMZprime traffic vs. extended real-time AMZprime Traffic.



(a) Main Discord traffic.

(b) Double Discord traffic timeline.

Figure 4.9: Real-time Discord traffic vs. extended real-time Discord traffic.

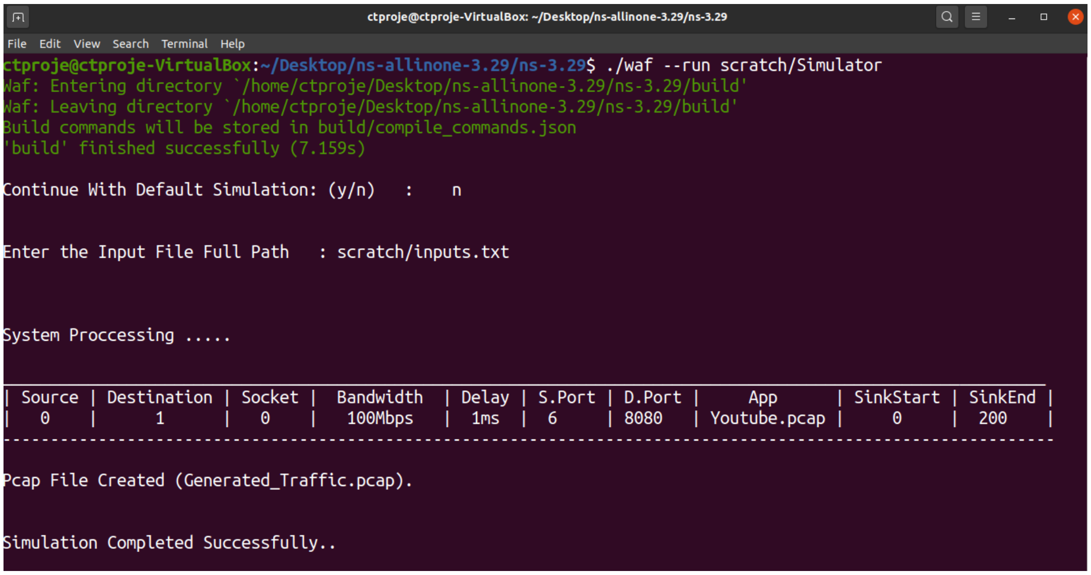
4.4 Networking AI Models Implementation

Network traffic generator's output provides ready-to-go datasets. Networking AI models use the outcome from the generator traffic flow as inputs to these models, which eliminate the lost time of data implementation and data correction. Therefore, the network traffic generator provides a measurement and managements tools to the AI models to handle the dataset requirements. For that, we implement the network traffic generator to different AI models to measure and evaluate its performance.

4.4.1 Performance Measurement

To measure the effectiveness of the proposed network traffic generator, the beginning of the network traffic generator starts by giving the option for the default simulation. The communication links then will be settled between the network devices. The default option will transmit the default databases that is used as an example of the simulation process. However, the input text file can setup the configuration as needed for this demonstration.

After setting up the configuration, the traffic generates from ten different applications separately by using the traffic and the payload databases highlighted in the Table 4.1. For each and individual application, the generator read the main database and provide inputs to the AI algorithms. For example, the user interface in Fig. 4.10 shows the network traffic configurator for the YouTube video transmitted from node 0 as a transmitter to node 1 as a receiver. The socket 0 indicates that the socket is UDP as a communication link protocol. The demonstration indicated that the bandwidth setting is 100 Mbps, which mimicking the normal life network traffic. The generator in this example will keep generate the traffic until it reached the EndSink as defined by the user. Since networking AI models are built in different program language, e.g., C, C++, Python, etc., the network traffic generator produces different types of database format such as NPY file, CSV file. The AI algorithms read these files as an input to their models.



ctproje@ctproje-VirtualBox: ~/Desktop/ns-allinone-3.29/ns-3.29
File Edit View Search Terminal Help

```
ctproje@ctproje-VirtualBox:~/Desktop/ns-allinone-3.29/ns-3.29$ ./waf --run scratch/Simulator
Waf: Entering directory `/home/ctproje/Desktop/ns-allinone-3.29/ns-3.29/build'
Waf: Leaving directory `/home/ctproje/Desktop/ns-allinone-3.29/ns-3.29/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (7.159s)

Continue With Default Simulation: (y/n)   :    n


Enter the Input File Full Path   : scratch/inputs.txt



System Proccessing .....

_______________________________________________________________________________________________________
| Source | Destination | Socket |  Bandwidth  | Delay | S.Port | D.Port |     App      | SinkStart | SinkEnd |
|   0    |      1      |   0    |   100Mbps   |  1ms  |  6     | 8080   | Youtube.pcap |     0     |  200    |
--------------------------------------------------------------------------------------------------------

Pcap File Created (Generated_Traffic.pcap).


Simulation Completed Successfully..
```

Figure 4.10: Network traffic configrator user interface.

4.4.2 Performance Evaluation

The evaluation performance for network AI classifier and AI-based traffic prediction model shows an accurate results using the network traffic genrator. First, Fig. 4.11 shows the performance of the classification of the proposed DL-based traffic classification function that attached to the traffic generator. The classification function uses 10,000 training data that sampled from the generated traffic, and 10,000 testing data are sampled from the real-traffic.

Note that both datasets are alternately exclusive for the generality of the evaluation results. Diagonal Confusion Matrix displays the number of packets correctly classified, where rating accuracy is also presented for each application. The confusion matrix shows that the average accuracy packets from all classes of applications, e.g., YouTube, Netflix, and Spotify, is 99.9.

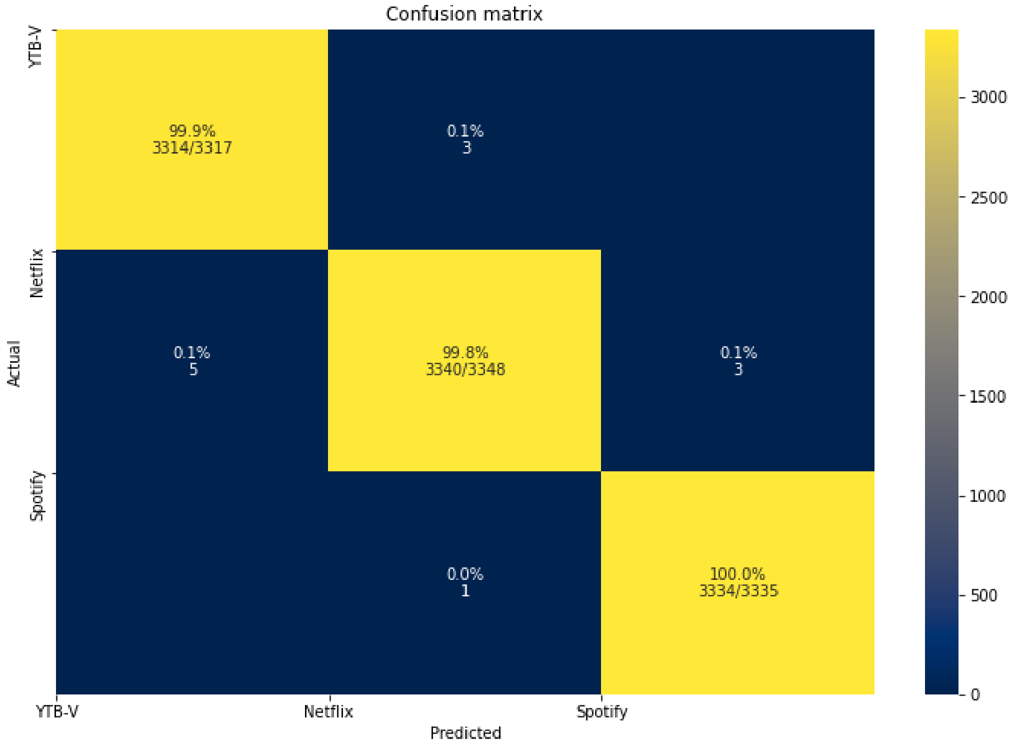


Figure 4.11: Results of the testing network traffic classifier.

The second performance evaluation is the AI-based traffic prediction model. Two scenarios are implemented in which test and training datasets are sampled from various sources. Fig. 4.12 shows the YouTube video traffic forecast results in the first scenario, in which the training data and test data are sampled equally from the generated traffic. On the left side of the yellow line, the traffic generated used to train the prediction model while the pilot traffic on the right side.

The evaluation result shows that a forecasting model that has been trained by the traffic generated can effectively predict the traffic generated in future periods in MAE of 38.66 on average. In the second scenario, Fig. 4.13 illustrates the traffic prediction results, where training data is sampled from the YouTube video traffic generated while the test data is captured from the real-traffic. The predicted patterns illustrate the effectiveness of the generated traffic in training the AI model, such that a prediction model that has been trained by the generated traffic can help predict a real-life traffic.

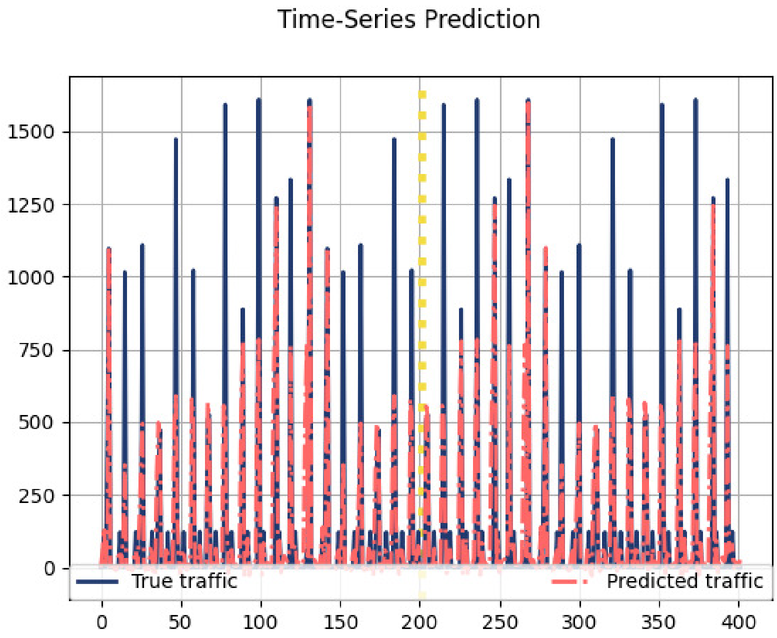


Figure 4.12: Traffic prediction using netowrk traffic generator of YouTube video traffic.

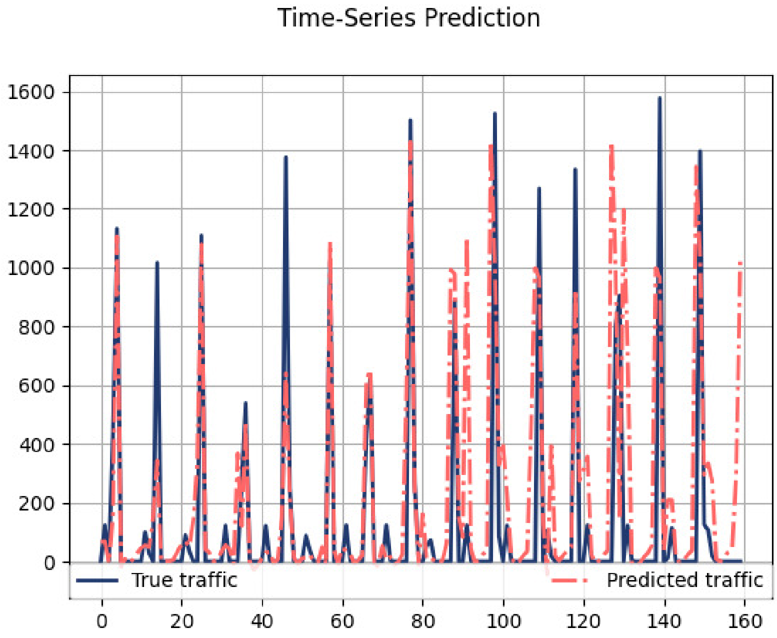


Figure 4.13: Traffic prediction for real-life of YouTube video traffic.

CHAPTER V

CONCLUSION AND FUTURE WORK

Although network AI models experience some difficulties to analyzing, measuring and managing the network due to data-hungry at real-time databases, it is the efficient way to measure the effectiveness of future network functionality. Even though the networking AI development depends on the easiness, accuracy, and quality of creating instantaneous network databases, the network traffic generator helped to step up further for the AI algorithms. Therefore, the impact of the network traffic generator on the mechanism of providing databases made these algorithms in a sustainable state to obtain the largest number of real-time data with least time and the most quality. The evaluation results demonstrated that the use of the network traffic generator on the networking AI models such as classification and traffic prediction performed a high accuracy comparing by the late studies.

In the future work, The network traffic generator can be developed to read multiple number of applications at the same time and separate their traffic to generate specific application traffic. Another future vision is to link the network traffic generator with networking AI models directly without the need to generate files to serve those models. Also, one of the aspects that I seek in the future is to replace the source of the main databases from being reading through a PCAP file to being capturing directly from the real network.

BIBLIOGRAPHY

[1] K. Lim, S. Soh, and S. Rai, *Computer Communication Network Upgrade for Optimal Capacity Related Reliability.* Asia-Pacific Conference on Communications, pp. 1102–1107, 2005.

[2] S. Adibi, *Traffic Classification Packet , Flow , and Application-based Approaches.* IJACSA - International Journal of Advanced Computer Science and Applications, vol. 1, pp. 6–15, 2010.

[3] Z. Aouini, A. Kortebi, and Y. Ghamri-Doudane, *Traffic monitoring in home networks: Enhancing diagnosis and performance tracking.* in 2016 13th International Conference on New Technologies for Distributed Systems (NOTERE), pp. 1–6, 2016.

[4] R. Roy and B. Mukherjee, *Managing Traffic Growth in Telecom Mesh Networks (Invited Paper).* Proceedings of 17th International Conference on Computer Communications and Networks, pp. 1-6, 2008.

[5] J. Xu and K. Wu, *Living with Artificial Intelligence: A Paradigm Shift toward Future Network Traffic Control.* in IEEE Network, vol. 32, no. 6, pp. 92-99, 2018.

[6] P. Amaral, J. Dinis, P. Pinto, L. Bernardo, J. Tavares, and H. S. Mamede, *Machine learning in software Defined Networks: Data collection and traffic classification.* in IEEE International Conference on Network Protocols (ICNP), pp. 1–5, 2016.

[7] M. W. et al, *Machine Learning for Networking: Workflow, Advances, and Opportunities.* IEEE Network, 2018.

[8] H. Yao, X. Chen, M. Li, P. Zhang, and L. Wang, *A novel reinforcement learning algorithm for virtual network embedding.* Neurocomputing, vol. 284, pp. 1–9, 2018.

[9] Z. Li, Z. Ge, A. Mahimkar, J. Wang, B. Y. Zhao, H. Zheng, J. Emmons, and L. Ogden, *Predictive analysis in network function virtualization.* in Proceedings of the Internet Measurement Conference 2018, pp. 161–167, 2018.

[10] B. M. et al, *Routing or Computing? The Paradigm Shift Towards Intelligent Computer Network Packet Transmission Based on Deep Learning.* IEEE Trans. Computers, vol. 66, no. 11, pp. 1946–60, 2017.

[11] Z. M. F. et al, *State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems.* IEEE Commun. Surveys and Tutorials, vol. 19, no. 4, pp. 2432–55, 2017.

[12] G. Carneiro, *Ns-3: Network simulator 3.* in UTM Lab Meeting April, vol. 20, pp. 4–5, 2010.

[13] R. Hofstede, B. T. P. Celeda, I. Drago, R. Sadre, A. Sperotto, and A. Pras, *Flow monitoring explained: From packet capture to data analysis with netflow and ipfix.* IEEE Communications Surveys Tutorials, vol. 16, no. 4, pp. 2037–2064, 2014.

[14] V. Sciancalepore, F. Z. Yousaf, and X. C. Perez, *z-TORCH: An automated NFV orchestration and monitoring solution.* IEEE Transactions on Network and Service Management, vol. 15, no. 4, pp. 1292–1306, 2018.

[15] Z. M. Fadlullah, B. M. F. Tan and, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, *State-of-the-art deep learning: Evolving machine intelligence toward tomorrows intelligent network traffic control systems.* IEEE Communications Surveys Tutorials, vol. 19, no. 4, pp. 2432–2455, 2017.

[16] Z. S. et al, *Traffic Engineering in Software-Defined Networking: Measurement and Management.* IEEE Access, vol. 4, pp. 3246–56, 2016.

[17] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y. Liang, , and D. I. Kim, *Applications of deep reinforcement learning in communications and networking: A survey.* IEEE Communications Surveys Tutorials, vol. 21, no. 4, pp. 3133–3174, 2019.

[18] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, *Artificial neural networks-based machine learning for wireless networks: A tutorial.* IEEE Communications Surveys Tutorials, vol. 21, no. 4, pp. 3039–3071, 2019.

[19] C. L. I. et al, *The Big-Data-Driven Intelligent Wireless Network: Architecture, Use Cases, Solutions, and Future Trends.* IEEE Vehic. Tech. Mag., vol. 12, no. 4, Dec. , pp. 20–29, 2017.

[20] P. Wang, X. Chen, F. Ye, and Z. Sun, *A survey of techniques for mobile service encrypted traffic classification using deep learning.* IEEE Access, vol. 7, pp. 54024–54033, 2019.

[21] L. Cong and W. Yong-Hao, *Strategy of Data Manage Center Network Traffic Scheduling Based on SDN.* 2016 International Conference on Intelligent Transportation, Big Data and Smart City (ICITBS), pp. 29-34, 2016.

[22] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, *Deep packet: a novel approach for encrypted traffic classification using deep learning.* Soft Comput., vol. 24, no. 3, pp. 1999–2012, 2020.

[23] S. Rahman, T. Ahmed, M. Huynh, M. Tornatore, and B. Mukherjee, *Auto-scaling VNFs using machine learning to improve QoS and reduce cost.* in IEEE International Conference on Communications (ICC), pp. 1–6, 2018.

[24] P. Wang, F. Ye, X. Chen, and Y. Qian, *Datanet: Deep learning based encrypted network traffic classification in sdn home gateway.* IEEE Access, vol. 6, pp. 55380–55391, 2018.

[25] J. Zhang, F. Li, and F. Ye, *An ensemble-based network intrusion detection scheme with bayesian deep learning.* ICC 2020-2020 IEEE International Conference on Communications, pp. 1–6, 2020.

[26] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, *Deep packet: A novel approach for encrypted traffic classification using deep learning.* Soft Computing, vol. 24, no. 3, pp. 1999–2012, 2020.

[27] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, *End-to-end encrypted traffic classification with one-dimensional convolution neural networks.* IEEE International Conference on Intelligence and Security Informatics (ISI). IEEE, pp. 43–48, 2017.

[28] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, *Toward developing a systematic approach to generate benchmark datasets for intrusion detection.* computers and security, vol. 31, no. 3, pp. 357–374, 2012.

[29] D. Szostak and K. Walkowiak, *Machine learning methods for traffic prediction in dynamic optical networks with service chains.* 21st International Conference on Transparent Optical Networks (ICTON), pp. 1–4, 2019.

[30] J. Zhang, F. Ye, and Y. Qian, *Intelligent and application-aware network traffic prediction in smart access gateways.*, 2020.

[31] R. Mijumbi, J.-L. Gorricho, J. Serrat, M. Claeys, J. Famaey, and F. D. Turck, *Neural network-based autonomous allocation of resources in virtual networks.* European Conference on Networks and Communications (EuCNC). IEEE, pp. 1–6, 2014.

[32] F. T. et al, *On Removing Routing Protocol from Future Wireless Networks: A Real-Time Deep Learning Approach for Intelligent Traffic Control.* IEEE Wireless Commun., vol. 25, no. 1, Feb. , pp. 154–60, 2018.

[33] C. Z. et al, *Deep Learning in Mobile and Wireless Networking: A Survey.* arXiv: 1803.04311v1, 2018.

[34] S. W. Y. Fu, X. H. C. Wang, and S. McLaughlin, *Artificial Intelligence to Manage Network Traffic of 5G Wireless Networks.* in IEEE Network, vol. 32, no. 6, pp. 58-64, 2018.

[35] C. Javali and G. Revadigar, *Network web traffic generator for cyber range exercises.* in 2019 IEEE 44th Conference on Local Computer Networks (LCN), pp. 308–315, 2019.

[36] R. Fujdiak, V. Uher, P. Mlynek, P. Blazek, J. Slacik, V. Sedlacek, J. Misurec, M. Volkova, and P. Chmelar, *Ip traffic generator using container virtualization technology.* in 2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), pp. 1–6, 2018.

[37] P. Wette and H. Karl, *Dct2gen: A traffic generator for data centers.* Computer Communications, vol. 80, pp. 45–58, 2016.

[38] M. F. Mokbel, L. Alarabi, J. Bao, A. Eldawy, A. Magdy, M. Sarwat, E. Waytas, and S. Yackel, *Mntg: An extensible web-based traffic generator.* in International Symposium on Spatial and Temporal Databases. Springer, pp. 38–55, 2013.

[39] J. Zhang, F. Li, H. Wu, and F. Ye, *Autonomous model update scheme for deep learning based network traffic classifiers.* IEEE Global Communications Conference (GLOBECOM), pp. 1–6, 2019.

[40] G. E. H. S. Osindero and Y. W. Teh, *A fast learning algorithm for deep belief Nets.* Neural computation, vol. 18, no. 7, pp. 1527–1554, 2006.

[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks.* in Advances in neural information processing systems, pp. 1097–1105, 2012.

[42] S. Hochreiter and J. Schmidhuber, *Long Short-Term Memory.* Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[43] D. Murray, T. Koziniec, K. Lee, and M. Dixon, *Large mtus and internet performance.* in 2012 IEEE 13th International Conference on High Performance Switching and Routing. IEEE, pp. 82–87, 2012.