NOVEL TIME-SERIES BASED DDOS ATTACK DETECTION SCHEMES FOR TRADITIONAL NETWORKS AND SOFTWARE DEFINED NETWORKS

by

Ramin Fuladi

B.S., Electrical and Electronics Engineering, Amirkabir University of Technology (Tehran Polytechnic)-Iran, 2004

M.S., Department of Electrical and Electronics Engineering, Boğaziçi University, 2014

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University
2021

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my advisor, Prof. Emin Anarım, for his great support and understanding in this long journey. He has that limitless kindness, encouragement and patience for me and this study. Without his supervision and great expertise this thesis would not have been possible.

Orhan Ermiş was with me on this tough road since the beginning of my thesis studies. His positive approach, motivation and help keep me going on. I would like to thank him for being a great mentor and a friend to me.

My sincere gratitudes goes to Prof. Mutlu Koca and Assit. Prof. Tunçer Baykaş for their invaluable comments and suggestions during my PhD study. I am also thankful to Prof. M. Ufuk Çağlayan and Assoc. Prof. Şerif Bahtiyar for their participation in thesis committee and their constructive comments.

My special thanks should be given to my friends: Khadija Hanifi, Cemil Eren Kayataş, Murad Ismet Arsan, Tayyab Waqar, Muhammad Ali Nagaria, Alireza Ahmadi Asl, Gholamreza Ilkhani, Derya Erhan and my colleagues at Ericsson Research Turkey and Arçelik A.Ş. I would like to extend my thanks to the staff of Boğaziçi Electrical and Electronics Department for their support during my PhD study.

Finally, and very importantly, I wish to thank my beloved mother, Akhtar Narenjiyeh, and my brother, Daniel F Fouladi, for their support in all aspects of my personal and academic life.

ABSTRACT

NOVEL TIME-SERIES BASED DDOS ATTACK DETECTION SCHEMES FOR TRADITIONAL NETWORKS AND SOFTWARE DEFINED NETWORKS

Distributed Denial of Service (DDoS) attacks are always one of the most significant threats for computer networks since they affect the user satisfaction by degrading the availability of on-line services. Although some countermeasures such as Intrusion Detection Systems (IDSs) provide effective mechanisms to discriminate various types of DDoS attacks, they become impotent of detection when bogus packets similar to normal ones are dispatched by the attacker. One promising approach for the DDoS detection in traditional networks is to use the time-series representation of the network traffic while analyzing the incoming packets. Particularly, discriminating features are extracted from the representation of the traffic flow in order to be used with several data analytic techniques such as statistical measures or machine learning algorithms. In this thesis, we first improve the previous works in the literature for the traditional networks by introducing three methods using frequency domain analysis and statistical measures. Later, we extend our findings for SDNs and we propose three different DDoS detection and countermeasure schemes for SDN by employing: (i) Auto-Regressive Integrated Moving Average and a dynamic thresholding method, (ii) Discrete Wavelet Transform and Auto-Encoder Networks, and (iii) Continuous Wavelet Transform and Convolutional Neural Network. Experimental results show that proposed schemes have high detection and low false alarm rates. Finally, we compare proposed schemes in terms of their attack detection performance and computational complexity cost analysis.

ÖZET

GELENEKSEL VE YAZILIM TABANLI AĞLAR İÇİN YENİ, ZAMAN SERİSİ BAĞLAMLI DDoS SALDIRI TESPİT ŞEMALARI

Dağıtık Hizmet Reddi (DDoS) saldırıları, çevrimiçi hizmetlerin kullanılabilirliğini azaltarak kullanıcı memnuniyetini etkilediği için bilgisayar ağları için her zaman en önemli tehdittir. İzinsiz Giriş Tespit Sistemleri (IDS'ler) gibi bazı karşı önlemler, çeşitli DDoS saldırılarını ayırt etmek için etkili mekanizmalar sağlasa da saldırgan tarafından normal paketlere benzer sahte paketler gönderildiğinde, tespit etmede yetersiz kalırlar. Gelen paketleri incelerken, ağ trafiğinin zaman serisi betimlemesinin kullanılması, geleneksel ağlarda DDoS tespiti için ümit vaadeden yaklaşımlardandır. Özellikle, istatistiksel ölçümler veya makine öğrenme algoritmaları gibi çeşitli veri analitiği teknikleriyle kullanılmak üzere trafik akışının betimlemesinden ayırt edici özellikler elde edilir. Bu tezde, ilk olarak, frekans alanı analizi ve istatistiksel ölçümler kullanı larak geleneksel ağlar için literatürdeki çalışmaları iyileştiren üç ayrı şema sunuyoruz. Daha sonra, Yazılım Tabanlı Ağlar (SDN) için bulgularımızı genişletiyoruz. SDN için aşağıdakileri kullanarak üç farklı DDoS tespit ve karşı önlem şeması öneriyoruz: (i) Zaman Serisi Özbağlanımlı Tümlenik Kayan Ortalamalı model ve dinamik bir eşikleme yöntemi, (ii) Ayrık Dalgacık Dönüşümü ve Oto Kodlayıcı Ağları ve (iii) Sürekli Dalgacık Dönüşümü ve Evrişimli Sinir Ağı. Deneysel sonuçlar, önerilen şemalarımızın yüksek algılama oranına ve düşük yanlış alarma sahip olduğunu göstermektedir. Son olarak, önerilen DDoS algılama ve karşı önlem şemalarını saldırı algılama performansı ve hesaplama karmaşıklığı maliyeti analizi açısından karşılaştırılmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

a	Order of AR terms in ARIMA model
$\mathfrak{a}$	Number of standard deviation, σ_{rate} to define threshold th_{rate}
A	Input matrix to the convolution layer of CNN
$\mathcal{A}$	Output matrix of convolution layer filters and input matrix A
$\acute{A}$	Dimension reduced matrix using PCA method
act_val	$USIP_{med}$ value that activates the *Detection* module
A_{DFT}	Matrix for DFT transform of the number of packets
A_{DWT}	Matrix for DWT transform of the number of packets
$avg_{hit-rate_t}$	Average hit rate instance at time interval t
b	Number of standard deviation, σ_d to define threshold th_{dist}
C	Label of class
$\mathcal{C}$	Covariance matrix
$counter$	Hash table counter
d	Number of samples shared with two windows in STFT
D	Over-complete dictionary with the size of $n \times m$
Des_{IP}	Destination IP address
$DestIP_{max}$	Destination IP address with the maximum occurrence
$\mathcal{D}_f$	Time-series of distance between two exponential filters
DIP_max	Destination IP address with the maximum occurrences
dur_t	Total duration in the flow table at time interval t
e	Number of epochs used during the training a neural network
e_0	First prediction error of ARIMA
$Eigvec_{max}$	Eigenvector of the maximum eigenvalue of time-series Υ
e_t	Prediction error of ARIMA at time interval t
Euc_dist	Set of Euclidean distances between each element of F and $\hat{F}$
$EXP\_f1(\alpha_1)$	Exponential filter, α_1 close to 1
$EXP\_f2(\alpha_2)$	Exponential filter, α_2 much less than 1

f	CNN convolution layer kernel filter
F	Set of feature vectors used to train the auto-encoder
$f(\alpha)$	Function of α
$flow\_count_t$	Total number of flows in the flow table
$Flow\_T$	A flow entry in a flow table of a switch
f_t	Feature vector at time interval t
$\hat{f}_t$	Output of auto-encoder at time interval t with f_t as the input
$\check{f}_t$	K-sparse vector obtained by solving $f_t \approx D\check{f}_t$
$\hat{\check{f}}_t$	Estimated value of $\check{f}_t$
$\|\check{f}_t\|_0$	l_0 pseudo-norm of $\check{f}_t$
$\mathcal{G}_h = \{g_{h_1}, \ldots, g_{h_\iota}\}$	Impulse response of high pass filter used in DWT
$\mathcal{G}_l = \{g_{l_1}, \ldots, g_{l_\iota}\}$	Impulse response of low pass filter used in DWT
H	Entropy of a random variable
$\mathcal{H}$	Hurst Exponent
H_D	Hash tables for extracting UDIP
$Hit_i, 1 \leq i \leq k$	Time-series of $avg_{hit-rate_t}$ used in training phase
H_l	Entropy value of l
H_S	Hash tables for extracting USIP
j	$j = \sqrt{-1}$
k	Number of samples for training
K	Number of non-zero coefficients in sparse coding
$kurt_l$	Kurtosis value of l
l	l^{th} sub-band of the DWT
ℓ	Number of differencing in ARIMA model
L	Set of sub-bands of the DWT
m	Number of atoms (columns) in dictionary D
$\mathcal{M}$	Time-series of the Rolling median for $\mathcal{D}_f$
max_level	Maximum number of DWT sub-band levels
min_dist	Minimum distance between each element and remains in $\mathcal{M}$
$Model\_X$	Training flag for USIP
$Model\_Y$	Training flag for NUDIP

n	Size of feature vector f_t
N	Size of time-series
O	Number of observations in SOM modeling
$P(.)$	Probability function
p_i	Possible probability of a random variable where $i \in [1, h]$
q	Order of MA terms in ARIMA model
Q	Number of neurons in SOM
r_t	Number of flow entries in the flow table at time t
S	Scale parameter of CWT
$score_{1,t}$	Anomaly score for x_t
$score_{2,t}$	Anomaly score for y_t
Sc_{qt}	Two-dimensional output of CWT at time interval t
skw_l	Skewness value of l
Src_{IP}	Source IP address
t	Sampling time interval
th_{med}	Threshold value for min_dist
th_{dist}	Threshold value of Euclidean distance of auto-encoder
th_{rate}	Threshold value of $avg_{hit-rate}$
V	V^{th} percentile
$\mathcal{V}$	Matrix of eigenvectors of j largest eigenvalues
Vx_l	Different percentile values of l where $x \in \{5, 25, 50, 75, 90\}$
w	Sliding window size
$\mathcal{X}$	Time-series of the number of unique source IP addresses
$X(f)$	Output of DFT where $f \in [0, N-1]$
x_t	USIP instance at time interval t
x_t	Output of exponential filter at time interval t
$\hat{x}_t$	Prediction of ARIMA at time interval t
$\mathcal{Y}$	Time-series of the number of normalized UDIP
y_i^{high}	Output of DWT high pass filter where $i \in [1, N]$
y_i^{low}	Output of DWT low pass filter where $i \in [1, N]$
y_t	UDIP instance at time interval t

Z	Time-series of ζ_t
$\mathcal{Z}$	Time-series of the Total number of packets
z_t	Number of packets instance at time interval t
$\mathcal{Z}_T$	Uni-variate time-series of z_t used in training phase
$\mathcal{U}_1$	Random Variable
$\mathcal{U}_2$	Random Variable
$USIP_{med_t}$	Median of $x_{N-w}, \ldots, x_w \in \mathcal{X}$ at time interval t
α	Smoothing constant of the exponential filter where $0 \leq \alpha \leq 1$
ε_{k}	k^{th} error item in ARIMA where $\mathrm{k} \in [t-q, t]$
ζ_t	Product output of (x_t, z_t) and v at time interval t
θ_j	j^{th} MA item in ARIMA where $j \in [1, q]$
ι	Size of impulse responses of the filters used in DWT
κ	Size of CNN convolution layer kernel
λ	Eignevalue of the covariance matrix $\mathcal{C}$
Λ_t	Lyapunov exponent at time interval t
μ	Mean value of a discrete variable
μ_d	Mean value of Euc_dist
μ_l	Mean value of l
μ_m	Mean value of min_dist
μ_{rate}	Mean value of Hit_T
μ_{u_1}	Mean value of $\mathcal{U}_1$
μ_{u_2}	Mean value of $\mathcal{U}_2$
ν	Eignenvector of the covariance matrix $\mathcal{C}$
ρ	Number of standard deviation, σ_m to define th_{med}
σ	Standard deviation of a discrete variable
σ^2	Variance of a discrete variable
σ_d	Standard deviation of Euc_dist
σ_l^2	Variance value of l
σ_m	Standard deviation of min_dist
σ_{rate}	Standard deviation of Hit_T

τ	Transition parameter of CWT
ϕ_i	i^{th} AR item in ARIMA where $i \in [1, a]$
$\Phi(t)$	Mother wavelet
Υ	Bi-variate time-series of USIP and totalPacket used in training phase

LIST OF ACRONYMS/ABBREVIATIONS

2D	Two Dimensional
AAR	Adaptive Auto-Regressive
ABF	Average of Bytes per Flow
ACC	Accuracy
ADF	Average of Duration per Flow
AI	Artificial Intelligence
AIC	Akaike's Information Criterion
AICc	Akaike's Information Criterion Corrected
ANN	Artificial Neural Network
APF	Average of Packet per Flow
AR	Auto-Regressive
ARIMA	Auto-Regressive Integrated Moving Average
AUC	Area Under Curve
BIC	Bayesian Information Criterion
BMP	Basic Matching Pursuit
BMU	Best-Matching Unit
BPNN	Back Propagation Neural Network
BR	Byte Rate
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CWT	Continuous Wavelet Transform
DDoS	Distributed Denial of Service
DFT	Discrete Fourier Transform
DNS	Domain Name System
DoS	Denial of Service
DT-CWT	Discrete Time-Continuous Wavelet Transform
DWT	Discrete Wavelet Transform
etsDstp	Entropy of Destination Port

etsProtocol	Entropy of Packet Protocol
etsSrcIP	Entropy of Source IP
etsSrcP	Entropy of Source Port
FP	False Positive
FPr	False Positive rate
FPSA	Flow Percentage with Small Amount packet
FT	Fourier Transform
GDP	Grow of Different Port
GSF	Grow of Single Flow
HTTP	Hypertext Transfer Protocol
IA	Idle timeout Adjustment
ICMP	Internet Control Message Protocol
ID	Identity Document
IDS	Intrusion Detection System
IoT	Internet of Thing
IP	Internet Protocol
IRC	Internet Relay Chat
KNN	K-Nearest Neighbor
KSVD	K-mean Singular Value Decomposition
LDOS	Low rate Denial of Service
LSTM	Long Short-Term Memory
MA	Moving Average
ML	Machine Learning
MLP	Multi Layer Perceptron
mMTC	Massive Machine Type Communications
MRA	Multi-Resolution Analysis
NB	Naive Bayes
NTP	Network Time Protocol
NUDIP	Normalized Unique Destination IP addresses
OF	OpenFlow
PCA	Principal Component Analysis

PDF	Probability Density Function
PPF	Percentage of Pair-Flow
QoS	Quality of Service
RL	Reinforcement Learning
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
RoQ	Reduction of Quality
RTT	Round Trip Time
SDN	Software Defined Networking
SFP	Symmetric Flow Percentage
SOM	Self-Organizing Map
SSH	Secure Socket Shell
STFT	Short Term Fourier Transform
SVM	Support Vector Machine
SW	Shapiro Wilk
TCP	Transmission Control Protocol
TN	True Negative
totalPacket	Total Number of Packets
TP	True Positive
TPr	True Positive Rate
UDIP	Unique Destination IP Addresses
UDP	User Datagram Protocol
USIP	Unique Source IP Addresses
VAFR	Variation Rate of Symmetric Flow

1. INTRODUCTION

With the increased popularity of online services, to ensure the availability of online services have become a significant issue due to the existence of Distributed Denial of Service (DDoS) attacks [1, 2]. In such attacks, an attacker forms an army of distributed and connected devices called bots to overwhelm the traffic of a targeted network by masquerading source IP addresses of those devices to attack the network in an untraceable manner. As a result, incoming packets to the network are categorized as legitimate traffic by the protection mechanisms [3]. One of the main defense mechanisms against DDoS attacks is to use Intrusion Detection Systems (IDS). In general, IDS are categorized based on the approach used for detecting the attacks, namely the signature-based IDS and anomaly-based IDS [4]. In the first approach, the detection mechanism is trained by the set of predetermined malicious traffic. Then, features of malicious traffic such as IP addresses of attackers, etc. are stored in a database to be used in the detection for subsequent attacks. When a new activity is detected, the database is queried to determine the activity is malicious or not. In anomaly-based detection, the detection mechanism of an IDS is trained using the pattern of a normal activity. Then, any abnormal activity is reported as the intrusion. Anomaly-based IDS have a significant advantage when compared with the signature-based ones because they are more successful in terms of detecting zero-day DDoS attacks. Therefore, in this thesis, our main focus is to study and improve the anomaly-based IDS approaches.

Anomaly-based IDS approaches are studied over the years for traditional networks [5–7]. An intelligent IDS for both signature-based and anomaly-based approaches is proposed in [5]. Another study in [6] uses entropy of time-series to discriminate DDoS attacks. In [7], authors propose the use of cluster modeling that jointly works with entropy of the time-series.

Software-Defined Networking (SDN) architectures are also vulnerable to DDoS attacks. In contrast to traditional networks, SDNs separate the control plan (con-

troller) from data plane (switches and routers). Although the centralization of the controller decreases the deployment cost, it makes the controller more vulnerable to DDoS attacks [8]. In addition, the attackers can target different elements of SDN, particularly southbound and northbound interfaces, switch hardware, and the controller [9]. Existing DDoS attack detection mechanisms for SDNs are generally based on statistical, Machine Learning (ML) or hybrid approaches. While statistical methods employ threshold values to discriminate attacks from normal traffic [10–15], ML techniques generalize the discriminating model by using training data consisting of one or more than one features obtained from the network traffic [16–27]. Compared to ML techniques, statistical methods are more efficient in terms of execution performance of the proposed technique; however, defining optimal thresholds for statistical methods is challenging.

1.1. Motivation

Most of the anomaly-based IDS approaches in the literature apply packet level analysis to detect any anomalies caused by DDoS attack samples [28]. In such methods, the payload of each arriving packet is inspected to find any suspicious changes in each element of the payloads (i.e., different status flags). However, the new generation DDoS attacks mimic the legitimate traffic payload and they make the traditional methods ineffective in detecting intrusions. Flow level analysis and representing DDoS attack features as time-series would be an alternative for existing methods. Moreover, data analytic methods are employed to obtain meaningful result on the time-series data. Therefore, our main motivation is to use time-series analysis of the network traffic together with the data analytic methods to detect DDoS attacks.

Existing solutions to detect DDoS attack for SDNs also utilize data analytic methods such as statistical measures and ML techniques approaches. Statistical measures require low computational effort and hardware. Those methods analyze statistical features to define optimum thresholds to separate DDoS attack traffic from normal ones. However, due to the dynamic characteristics of the network traffic in SDN, the use of

a constant value as the threshold can be inefficient while detecting the attacks. For this aim, in this thesis, we study the dynamic thresholding approach for DDoS attack detection in SDN.

ML techniques are another widely studied topic for the detection of DDoS attacks in SDN. These techniques may require more computational cost particularly while training the appropriate ML model for the network. The selection of appropriate features for generalizing the discriminating model is an important step while applying ML approaches. These features are generally extracted and selected from the statistics obtained from the network. Although there have been different approaches, the use of frequency-domain for feature extraction has promising results [29]. Furthermore, frequency-domain features differentiate between attack and normal traffic [30] and thus, DDoS attack samples can be easily identified by analyzing those features in frequency-domain. Therefore, in this thesis, we mostly focused on DDoS attack detection schemes based on frequency-domain features for SDNs.

1.2. Contribution

In this thesis, we propose novel DDoS detection schemes for both traditional networks and SDNs. In terms of the DDoS detection in traditional networks, we propose the following three schemes based on the time-series analysis together with the frequency-domain features:

- In the first scheme (Scheme 1), features extracted from Discrete Fourier Transform (DFT) [31] and Discrete Wavelet Transform (DWT) [32] are obtained and used by naïve Bayes classifier [33] to discriminate attack from normal traffic. Finally, the DDoS attack detection performance of naïve Bayes classifier with DFT, DWT and DFT+DWT as the feature set is compared with a thresholding method [34].
- In the second scheme (Scheme 2), we employ statistical features including periodicity, kurtosis, skewness [35] and Hurst exponent parameter [36] based on a time-series data to detect DDoS attacks. Those aforementioned features are com-

pared regarding the performance of discriminating attack from normal traffic.

- In the third scheme (Scheme 3), we use DFT and sparse coding algorithm [37] to extract features for separating attack from normal traffic in an anomaly-based approach. The obtained features are used by a Self-Organized Map (SOM) [38] lattice to generalize the normal behavior of the network.
- The result of three proposed methods are compared regarding the attack detection performance.

For the detection of DDoS attacks in SDN, we propose the following three novel time-series based DDoS attack detection and countermeasure schemes:

- The first scheme (Scheme A) uses dynamic thresholding approaches for DDoS attack detection. We use Unique Source IP addresses (USIP) and Normalized Unique Destination IP addresses with respect to total number of Packets (NUDIP) as characteristic features for detection. Then, we employ ARIMA model [39] and median-based dynamic thresholding approach, respectively to generate two anomaly binary scores to detect DDoS attacks.
- The second proposed scheme (Scheme B) is based on DWT and an ML approach. We use USIP and Total number of Packets (totalPacket) as characteristic features for detection. Then, we employ DWT and auto-encoder neural network [40] to detect anomaly caused by DDoS attack traffic.
- The third proposed scheme (Scheme C) is based on Continuous Wavelet Transform (CWT) [41] and an ML approach. We use USIP and NUDIP as characteristic features for detection. Then, we employ CWT and Convolutional Neural Network (CNN) [42] to classify traffic sample as a DDoS attack or a normal network traffic.
- Finally, we compare our DDoS attack detection and countermeasure schemes with each other in terms of their computational cost complexity analysis and detection performance in our simulations carried out by using GNS3 environment [43] and Mininet emulator [44].

Some of contributions of this thesis have been published in [45–48] or submitted for publication in [49, 50]

1.3. Thesis Outline

The rest of this dissertation is organized as follows:

In Chapter 2, the theory background of different concepts including DDoS attack, SDN architecture, time-series analysis and ML approaches are discussed. Moreover, the literature review corresponding to DDoS attack detection for traditional networking and SDNs is given.

In Chapter 3, three different time-series based DDoS attack detection schemes for traditional network architectures are proposed.

Three DDoS attack detection and countermeasure schemes for SDN are presented in Chapter 4, Chapter 5 and Chapter 6.

In Chapter 7, we compare the performance of the proposed DDoS attack defense schemes.

In Chapter 8, we conclude the thesis and present future research directions.

2. LITERATURE OVERVIEW

In this chapter, we explain the theory behind the DDoS attack and countermeasure schemes studied in this thesis. We first introduce the (D)DoS attacks that threaten the availability of computer networks. Later, we give a brief introduction related to those attacks and how they affect the Software-Defined Network (SDN) architectures. Then, we highlight the importance of time-series analysis to detect DDoS attacks. Finally, we overview several machine learning techniques that are applicable to DDoS attack detection.

2.1. Distributed Denial of Service (DDoS) Attack

Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks are the major security threats for the availability of computer networks by flooding the targeted online service with spurious packets. In general, DDoS attacks are the improved and more powerful form of DoS attack that the attacker use multiple compromised agents instead of a single device as in DoS attacks [51]. Therefore, in this thesis, we fully focus on DDoS attacks and countermeasure schemes against those attacks.

To initiate the DDoS attack, the attacker employs a large number of compromised devices and forces them to send fake packets toward the victim server. Then, the server tries to respond those fake packets with its limited resources. Due to the overloaded process pipeline of the server, it cannot properly serve its legitimate users. Eventually, this may force to deny all service requests coming from all users [52]. The denial of service imposed by the DDoS attacker can be carried out by either depleting the resource of the server such as the computational power, the memory size etc., or by sending a large number of illegal requests to the server [53].

In more details, the attacker employs compromised computers and devices called zombies / agents to initiate the DDoS attack. Those zombies are normal users in

the network with some security vulnerabilities. The attacker benefits from those vulnerabilities, penetrates to their system, creates a secure channel, installs an adversary software tool through the channel and takes the control of them to carry out the attack to the service, namely the victim server. The adversary software tool is responsible for coordinating all zombies. Once the attacker takes the control of all zombies, they begin to send bogus packets toward targeted service simultaneously. Suddenly, the victim server faces with a large volume of malicious traffic which it is unable to handle all at the same time [54].

Some specific characteristic features of the network changes significantly during the DDoS attacks such as the number of unique source IP addresses. Moreover, before launching the attack, the attacker needs to choose the underlying protocol such as TCP, UDP or ICMP. As a result, during the attack, a specific protocol type becomes dominant. On contrary to most of the intrusion types, which can be traced back easily, because of the spoofed source IP addresses, the tracing back of DDoS attack is exhausting and time consuming [55].

2.1.1. Distributed Denial of Service Attack Architecture

To carry out DDoS attacks, attackers employ various architecture models. Those architecture models can be categorized as agent-handler, Internet Relay Chat (IRC)-based and reflector [56], which are defined as follows:

2.1.1.1. Agent-handler Model. The architecture has three layers of client, handler and the agent [57]. The client is the mediator to make the connection between the attacker(s) and agent members possible. The attack directions from the attacker(s) are conveyed to the agent members by the client through the software tools which are called handlers. Master and daemon are the alternative names for handler and agent, respectively.

2.1.1.2. IRC-based Model. This architecture is similar to the agent-handler model; however, instead of using handlers, the client employs IRC systems to send attack directions to agent members [57]. The IRC system is a free online chatting environment. Various Internet users make use of the IRC system to communicate and chat with each other [58]. Therefore, the attacker can take advantage of this system to communicate with the agent members. The IRC legitimate ports are used by the attacker to send the commands to the agent members. As a result, the tracing back of the DDoS attack commands is much harder in this architecture compared to agent-handler. Furthermore, the attacker can hide its illegitimate packet traffic because of a large volume of traffic exchanging in the IRC system. Another advantage of IRC system is that the attacker can log on to the IRC system and see the list of all available agents [59], instead of keeping the track of available agents.

2.1.1.3. Reflector Model. This DDoS attack architecture is similar to the previously mentioned models. However, instead of commanding the army of zombies to send packets directly to the victim server, the attacker commands zombies to send packets to the users connected to the Internet with a spoofed source IP of the victim server. As a result, the victim faces with a large volume of packets from several systems. The attacker can also amplify the effect of this model by sending the packets to the broadcast address of the reflector networks [60].

2.1.2. Degree of Automation

The amount of attacker's intervention in the process of initiating the attack categorizes DDoS attacks into three groups, namely manual, semi-automatic, and automatic models. In the automatic model, the attacker scans all possible vulnerabilities of the agents' systems, penetrates the system, and finally embarks on the attack. In the semi-automatic model, the attacker just commands the start of the attack. Finally, in the manual model, the attacker carries out all aforementioned attack process manually. Because there is no sign of attackers in the automatic model, such type of DDoS model is hard to trace back; therefore, it is the desired form of attack utilized

by DDoS attackers [61].

2.1.3. Targets of DDoS Attacks

To initiate the DDoS attack, the attacker targets different components of the victim's system. According to the targeting component, DDoS attack is divided into two groups, which are called the bandwidth depletion (volumetric) and resource depletion attacks [62]. The goal of the bandwidth depletion attack is to consume the bandwidth of the victim server. It is fulfilled by flooding the server with a large volume of bogus packets; therefore, the bandwidth of the victim is used by attack packets and the server becomes unavailable for legitimate users. In the resource depletion, the attacker sends malformed packets, where the payload part of the packet filled by some wrong information, to the victim. During the analyzing of the received packets, the victim tries to cope with the malformed packets and the resources including CPU, memory etc., are assigned to those packets. As a result, the resources of the server are utilized with handling the packets and the server cannot respond to the requests coming from the legitimate users properly. Resource depletion is further classified as the protocol and application attacks. While the protocol attacks consume the processing capacity of the victim by targeting Layer 3 and Layer 4 protocol communications, the application attacks use the weakness of Layer 7 to consume finite resources such as disk space, memory, etc. [63].

2.1.4. Intrusion Detection Systems (IDS)

Intrusion Detection Systems (IDS) are employed as a defense mechanism to protect online services from DDoS attacks. IDS operates with either using signature-based detection or anomaly-based detection [4]. In the signature-based approach, the signature(s) of a set of known attacks are introduced to the system before the system starts its normal operation. The IDS compares the traffic signatures with this aforementioned set to detect potential similarities in the network traffic. On the other hand, in the anomaly-based approach, the detection system is trained with the pattern of the nor-

mal activity of the network. Then, the trained system is used for finding the abnormal behaviors of the network traffic. In general, anomaly-based IDS approaches provides better performance in terms of DDoS detection than signature-based IDS approaches.

The majority of existing IDS approaches implement packet level analysis to extract features of a network traffic. For instance, the approach proposed in [64] uses payload of incoming packets to obtain attributes which are different in normal and abnormal activities. However, in recent DDoS attacks, characteristics of the legitimate web service traffic are mimicked by attacking entities to make traditional methods ineffective. One of the best alternative approach for the detection of recent DDoS attacks is to use time-series-based representation of the network traffic and its characteristics, which is one of our main motivations in this thesis.

2.2. Time-Series Analysis

Time-series analysis is a statistical method applied on time-series data, which is a set of values (observations) sampled chronologically for every specific time interval. This time interval is called time-series frequency [65]. The time-series data can be either a collection of observations of a single variable or the collection of observations of different variables that are called univariate and multi-variate time-series data, respectively. A time-series data is characterized by different behaviors such as:

- mean-reverting or explosive behavior
- time trend
- seasonality
- structural breaks.

To find and estimate underlying model of the time-series data, the aforementioned characteristics need to be taken into account [66]. A time-series data is mean reverting if the observations fluctuate around a constant and time-invariant mean value. Sometimes, in addition to non-zero mean value, the time-series exhibits increasing or

decreasing trend which is proportionate to the time period. Seasonality is another important characteristic which occurs when time-series data exhibits regular and predictable patterns at time intervals. Any sudden changes in the time-series refers to the structural breaks characteristics. Although these sudden changes make the underlying model estimation much harder, it can be employed to detect anomalies in the data.

Furthermore, any time-series data can be also characterized by the statistical features. Those features provide more information related to the underlying probability distribution of the time-series which are used to discriminate two or more than two time-series data. In this thesis, we extract different statistical features from the time-series and employ them for DDoS attack detection. The definitions of employed statistical features based on a given time-series data $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ are as follows [35]:

Definition 1. *mean (μ): The mean of a given time-series $\mathcal{X}$ with N observations, is calculated as follows:*

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i. \tag{2.1}$$

Definition 2. *variance (σ^2): Variance is a measurement of the spread between observations in a data set. It measures how far each observation in the set is from the mean, μ. The square root of variance, σ, is called standard deviation. The variance is estimated as follow:*

$$\sigma^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu)^2. \tag{2.2}$$

Definition 3. *percentiles: Percentiles indicate the percentage of scores that fall below a particular value [67]; therefore, the V^{th} percentile is the lowest value that is greater than a certain percentage (V) of the values in a specific distribution. In this work, 5^{th}(V5), 25^{th}(V25), 50^{th}(V50), 75^{th}(V75) and 75^{th}(V90) percentiles are employed.*

Definition 4. *skewness (Skw): skewness is a measure of asymmetry of a distribution. It shows how the curve of distribution is distorted to the right or left compared to normal*

bell curved distribution. The skewness is estimated from the given data set is as below:

$$Skw = \frac{1}{N\sigma^3}\sum_{i=1}^{N}(x_i - \mu)^3. \tag{2.3}$$

Definition 5. *kurtosis (Kurt): Kurtosis is the degree of the sharpness of a distribution. Data exceeds at the tail of the distribution compared to normal bell curved distribution. It is a measure for the existence of extreme data at the tails of a distribution. If the kurtosis of a distribution is larger than that of the normal distribution, then the tail data exceed more than the normal one. On the other hand, a distribution with lower kurtosis has tail data that are less extreme than the tails of the normal one. The kurtosis is calculated from the given data as follows:*

$$Kurt = \frac{1}{N\sigma^4}\sum_{i=1}^{N}(x_i - \mu)^4. \tag{2.4}$$

Definition 6. *entropy (H): Entropy is a metric to find the amount of randomness of an attribute within a specific period of time. A high spread probability distribution has a high entropy value, while the entropy of concentrated probability distribution is low. The entropy of a random variable with the different values of* $\{p_1, p_2, ..., p_h\}$ *within a given time window is defined as shown in the following equation:*

$$H = \sum_{i=1}^{h} p_i log_2(\frac{1}{p_i}). \tag{2.5}$$

Definition 7. *Hurst exponent (*$\mathcal{H}$*): The Hurst exponent parameter is the measure for long-term memory of a time-series which shows how far that time-series deviates from random walk series [36]. The parameter shows the relative tendency of a time-series either to return to the mean value (mean-reverting pattern) or to keep a specific trend (trending pattern). The Hurst parameter resides between* 0 *and* 1*, which based on the value the corresponding time-series is classified into three categories as follows [36]:*

- $\mathcal{H} \leq 0.5$*: The time-series is mean-reverting (anti-persistent).*

- $\mathcal{H} = 0.5$*: The time-series is random walk.*
- $\mathcal{H} \geq 0.5$*: The time-series is trending (persistent).*

All explained parameters in this subsection are used for time-series model estimation. Time-series models are employed for different applications such as forecasting future outcomes, anomaly detection and etc. [68]. In this thesis, we use time-series models to distinguish DDoS attack samples from normal ones.

Two well-known approaches called time-domain analysis and frequency-domain analysis are mostly used to model the time-series [69]. In this thesis, we use exponential filter (smoothing) [70] and ARIMA model [39] for time-series model estimation in time-domain. On the other hand, Discrete Fourier Transform (DFT) [31], Discrete Wavelet Transform (DWT) [32] and Continuous Wavelet Transform (CWT) [41] are used to characterize time-series in frequency-domain. The aforementioned time and frequency domains approaches are briefly discussed as follows:

2.2.1. Exponential Filter

The exponential filter [70] is a weighted combination of the previous estimate, x_{t-1} (output), with the newest input data, x_t, with the sum of the weights equal to 1, so that the output matches the input at a steady state. This weight assignment is accomplished by using a smoothing constant α ($0 \leq \alpha \leq 1$) to calculate the smoothed value x_t at time t as follows:

$$\mathrm{x}_t = \alpha \mathrm{x}_{t-1} + (1 - \alpha) x_t, \tag{2.6}$$

where α controls the closeness of this new value using the recent observation. The function of α is defined as below:

$$f(\alpha) = \begin{cases} \mathrm{x}_t \cong x_t & \alpha \to 0 \\ \mathrm{x}_t \cong \mathrm{x}_{t-1} & \alpha \to 1 \end{cases}. \tag{2.7}$$

2.2.2. Auto Regressive Integrated Moving Average (ARIMA)

The ARIMA model [39] is used to forecast a time-series which can be made stationary by differencing if necessary. A stationary time-series data has the property that its statistical characteristics such as the mean and variance are constant over the time [71]. ARIMA is characterized by a three-tuple $\langle a, \ell, q \rangle$, where a is the number of Auto-Regressive (AR) terms, ℓ is the number of differences required for stationarity, and q is the number of Moving Average (MA) or lagged forecast errors terms. Let $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ be a stationary time-series, the general forecasting equation in terms of x is represented as:

$$\hat{x_t} = \theta_0 + \phi_1 x_{t-1} + \ldots + \phi_a x_{t-a} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \ldots - \theta_q \varepsilon_{t-q},$$

where ϕ_i $(i = 1, ..., a)$ and θ_j $(j = 1, ..., q)$ are AR and MA items, respectively. ε_{k} $(\mathrm{k} = t, t-1, ..., t-q)$ are named as error terms which are generally assumed to be independent, identically distributed variables sampled from a normal distribution with zero mean. It is worth to note that, depending on the characteristic of the series, each a, ℓ, and q can be equal to zero; therefore, the model may be simplified. For instance, ARIMA$\langle a, 0, 0 \rangle$, ARIMA $\langle 0, \ell, 0 \rangle$, ARIMA $\langle 0, 0, q \rangle$ and ARIMA $\langle a, 0, q \rangle$ are pure AR $\langle a \rangle$, random walk, MA $\langle q \rangle$ and ARMA $\langle a, q \rangle$ respectively. In this thesis, we employ ARIMA to model the normal behavior of the network and to forecast the expected upcoming observation value and then chaos theory is applied to find the anomaly samples.

2.2.2.1. Chaos Theorem. The error between the actual value and the estimated one is the key factor to detect DDoS attack samples. To assign anomaly score to the potential DDoS attack candidates in the network traffic, we analyze the chaotic behavior of the estimated error. Therefore, we employ the local Lyapunov exponent [72] as below:

$$\Lambda_t = \frac{1}{t} ln(|\frac{e_t}{e_0}|), \tag{2.8}$$

where e_0, e_t and Λ_t are the first prediction error, the t^{th} prediction error and the Lyapunov exponent at t^{th} time instance, respectively. A Positive Λ_t may correspond to the prediction error which could be caused by DDoS attack or by some legitimate traffic [73]. On the other hand, the negative Λ_t refers to the normal behavior. We use these values to distinguish the DDoS attack candidates from the normal ones.

2.2.3. Frequency Domain Analysis

In the frequency domain analysis, the Fourier Transform (FT) [74] is one of the most popular transform techniques to analyze signals. FT is applied to a discrete data by using the following steps. Let $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ be a discrete data, the Discrete Fourier Transform (DFT) [31] is defined as:

$$X(f) = \sum_{k=0}^{N-1} x_k e^{-j\frac{2\pi}{N}fk}, \ for\ 0 \leq f \leq N-1, \tag{2.9}$$

where $j = \sqrt{-1}$. The output of the transform is the sequence of N complex numbers, $X(f)$. According to the Euler's formula [75], the complex exponential function in Equation 2.9 is represented as:

$$e^{j\frac{2\pi}{N}fk} = cos(j\frac{2\pi}{N}fk) + jsin(j\frac{2\pi}{N}fk). \tag{2.10}$$

Therefore, $X(f)$ carries the information of phases and amplitudes of a set of complex sinusoidal functions with the frequency $\frac{f}{N}$ cycles per time unit. DFT cannot find the exact time of the frequency components if the time-series frequency characteristic varies by time. Thus, for those time-series, non-stationary series [76], Short Term Fourier Transform (STFT) [76] has been proposed. As shown in Figure 2.1, the main goal of STFT is to split the signal into small parts of stationary sub-signals. The original series is divided into several equally length windows with the size of w samples, where each window may share the last d data samples with the next window and then, DFT is applied for each window. The window-size of the STFT window is the trade off between the frequency resolution and time resolution, which corresponds that the

larger window-size we have, the more information about the frequency we get and the smaller window-size we select, the more information about the time we have. Therefore, the selection of the window-size is the critical point in STFT.

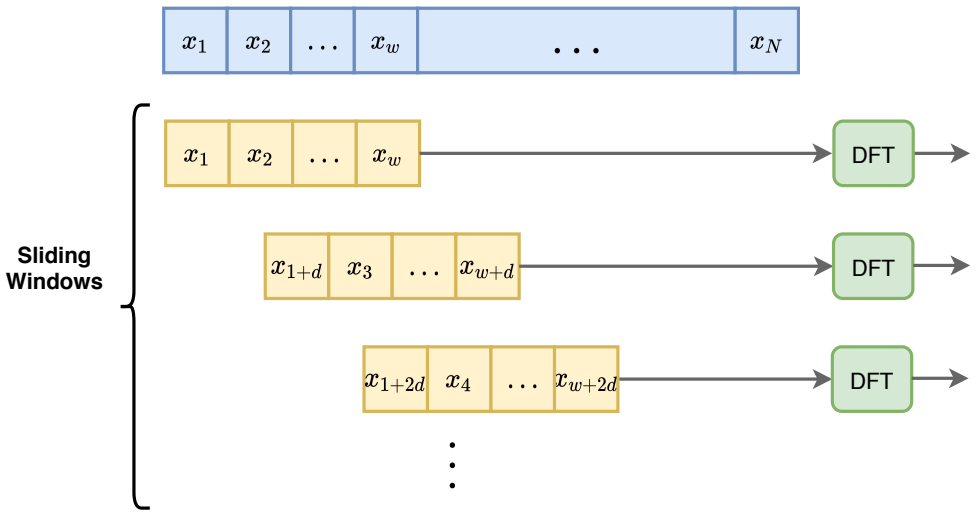


Figure 2.1. Short Term Fourier Transform (STFT).

On the other hand, wavelet transform can be considered as an alternative approach for analyzing a dynamical frequency spectrum. The idea of wavelet comes from Multi-Resolution Analysis (MRA) [77]. MRA achieves good time resolution at high frequencies and good frequency resolution at low frequencies. Therefore, wavelet transform is another method for analyzing the time-series at different frequencies with different resolutions. There are two types of wavelet, namely Continuous Wavelet Transform (CWT) [41] and Discrete Wavelet Transform (DWT) [32]. DWT operates in two steps called filtering and sub-sampling. The filtering step is achieved by applying two filters (low and high pass filters) on the input time-series. The sub-sampling is realized by down-sampling [31] the outputs of two filters by 2. While filters modify the resolution of the time-series, down-sampling process controls the scale of the time-series. The low pass and high pass filters are used to analyze the low and high frequencies [31]. The process begins by introducing the time-series, $\mathcal{X}$, to a half-band digital low pass filter with the impulse response of $\mathcal{G}_l = \{g_{l_1}, \ldots, g_{l_\iota}\}$ which removes

all frequencies above half of the maximum frequency in the time-series while keeps the scale of the time-series intact. Considering the Nyquist theorem [78], half of the output of the low pass filter can be discarded by applying down-sampling with the factor of two [32]. As a result, the scale of the time-series is now doubled. More formally, for all $x_i \in \mathcal{X}$ and $g_{l_j} \in \mathcal{G}_l$ the low pass filter y_i^{low} is calculated as follows:

$$y_i^{low} = \sum_{j=-\infty}^{\infty} g_{l_{2j-i}} x_i. \tag{2.11}$$

Simultaneously, $\mathcal{X}$ passes through a half-band high pass filter with the impulse response of $\mathcal{G}_h = \{g_{h_1}, \ldots, g_{h_\iota}\}$ and the half of the output is discarded. This process can be expressed as

$$y_i^{high} = \sum_{j=-\infty}^{\infty} g_{h_{2j-i}} x_i. \tag{2.12}$$

The above procedure, which is also known as sub-band coding [79], can be repeated for further decomposition by introducing the output of the low pass filter to the filters in the next level. The maximum number of sub-band levels depends on the number of data points in the time-series and the length of the filter. DWT runs for max_level until the time-series becomes shorter than the filter length (ι) for a given wavelet. This corresponds to:

$$max\_level = \lfloor log_2(\frac{N}{\iota - 1}) \rfloor. \tag{2.13}$$

CWT was proposed as the alternative approach to STFT to solve the resolution problem. The transform is similar to the method used for STFT which the original time-series is multiplied with the wavelet function, similar to the window function in the STFT and then transform is applied on each time-domain segment separately. CWT differs from the STFT method in two aspects:

(i) The Fourier transform of the segmented signal is not taken

(ii) The width of the window is changed as the transform is calculated for every spectral component.

The continuous wavelet transform is defined as follows:

$$CWT_x^{\psi} = \frac{1}{\sqrt{|S|}} \int x(t)\Psi^*(\frac{t-\tau}{S})dt, \tag{2.14}$$

where τ and S are the transition and scale parameters, respectively. The $\Psi(t)$ function is the mother wavelet. The term mother implies that during the transformation process other functions, $\Psi^*(\frac{t-\tau}{S})$, are derived from this main function. The transition, τ, term is used in the same sense as window term works in STFT. It corresponds to the location of the sliding window, as the window is shifted through the time-series. The scale parameter, S, can be considered as the inverse value of the frequency. Therefore, the large values of S are related to the low frequencies and the small values of S correspond to the high frequencies. Different wavelets such as the Morlet wavelet and the Mexican hat function can be employed as the mother wavelet. Then the computation starts with $S = 1$ and CWT is calculated for all Ss, smaller and larger than 1. Because the time-series is band-limited [80], a limited interval of Ss are enough to compute CWT. In general, the computation starts with $S = 1$ and continues for increasing values. It means that the transformation begins from the high frequency and proceeds toward lower frequencies. The CWT can be employed for discrete time-series as well which is sometimes called Discrete Time Continuous Wavelet Transform (DT-CWT).

2.3. Machine Learning

ML is a sub-field of Artificial Intelligence (AI). The goal of ML is to provide the machine (computer or any smart devices) with a bunch of data and the desired output to learn the underlying model between input data and the result [42]. The process is different from traditional programming in which the computer is fed with data and a number of lines of code to generate the output. ML methods are widely divided into two categories of supervised learning and unsupervised learning methods [42]. In

a supervised learning, the computer is provided with the data that are labeled with the corresponding desired outputs. On the other hand, in an unsupervised learning method, the computer is just fed with data without any labels and the computer tries to cluster the data into different groups of subsets which the elements of each subset share some common properties. The data provided to the ML algorithm can be considered as a table which each column is a variable (feature) and each row is just an observation of all features at a specific time. In this section we give a brief introduction related to ML methods used in this thesis.

2.3.1. Naïve Bayes Classification

In naïve Bayes classification method, which is based on Bayes' theorem, it is assumed that all features are strong (naïve) independent from each other [33]. Because naïve Bayes classification can be implemented easily without any complicated iterative parameters, it is efficient for very large dataset. Bayes theorem provides a way of calculating the posterior probability, $P(C|f_i)$, from $P(C)$, $P(f_i)$, and $P(f_i|C)$, where f_i is the i^{th} feature vector in a specific feature set, F, on a given class C. The corresponding equations used in naïve Bayes classification are defined as follows:

$$P(C|f) = \frac{P(f|C)P(C)}{P(f)} \tag{2.15}$$

and

$$P(C|F) = P(C)\prod_i P(f_i|C). \tag{2.16}$$

Considering the Bayes' equation, the likelihood probability ($\prod_i P(f_i|C)$) could be used as a score of class C. This score is used as a threshold to separate attack from normal traffic.

2.3.2. Self-Organizing Map (SOM)

Self-Organizing Map (SOM), which is also called Kohonen Map [38], is one of the unsupervised learning techniques. It is mostly used for clustering, dimension reduction and feature extraction [81]. Contrary to conventional neural networks, which apply back-propagation [40] with the gradient descent algorithm [82] to generalize the model, SOM employs competitive learning technique. SOM model comprises neurons where each neuron is connected to adjacent neurons by neighborhood relation. During the model generation, for each feature vector, f, from the training data, the distances (i.e., Euclidean distance), between all weight vectors and f are calculated. Then a neuron that its weight vector is the closest vector to f is chosen as the Best-Matching Unit (BMU). Once the BMU is selected, the weight vectors of the SOM are updated so that the BMU is moved towards the input vector. Finally, the model of SOM is generated in accordance with the training data. We refer curious reader to [38] for further details.

2.3.3. Auto-encoder Network

An auto-encoder is an unsupervised [83] neural network that replicates the input as the output [40] and basically operates as follows: first, the input data is compressed, and encoded in a lower dimension representation. Then, the output with the same dimension of the input is reconstructed from the encoded representation. The main goal of the auto-encoder neural network is to construct an output as close as possible to the input data by applying three sub-modules, namely, encoder, bottleneck (latent), decoder. The encoder is a set of layers used to encode the data to the lower dimensions. The bottleneck is the layer that represents the lowest possible dimension of the data. Finally, the decoder is a set of layers that constructs the output. Like all other neural network models, the auto-encoder neural network also requires a training phase to estimate the model parameters. In general, the back-propagation [40] setting is used in this estimation by minimizing the reconstruction loss. For more details about auto-encoder network, we refer the readers to [84].

2.3.4. Convolutional Neural Network (CNN)

Due to the high performance and accuracy neural networks have been utilized in ML applications. A simple Multi-Layer Perceptron (MLP) is a type of feed-forward neural network [85] which consists of three layers of nodes: input layer, a hidden layer and output layer. Except the input layer, each node uses a non-linear activation function. In order to train an MLP network, back-propagation technique is employed [40]. MLP has two disadvantages: first, because MLP is a fully connected network, the total number of parameters can grow to very high, when the input vector is large. Second MLP disregards the spatial information. Therefore, for data with spatial characteristic and with a large number of elements, such as images, MLP is deemed insufficient. In order to solve the mentioned problems, Convolutional Neural Network (CNN) has been put forward [85]. CNN is a feed-forward neural network which is generally used for data with spatial structure and relation such as images. The input data to CNN is represented as two-dimensional array (matrix) with one or more than one channel. CNN employs two main functions of convolution and pooling to reduce the input into the essential features and then uses those features for classification purpose. Each CNN consists of four basic blocks as follows:

(i) Convolution layer: In this type of the layer, the input matrix is convolved with a filter which is sometimes called kernel. Let $A_{n\times n}$ and $\mathrm{f}_{\kappa\times\kappa}$ be the input matrix and the aforementioned filter where $\kappa < n$. In order to apply convolution, the filter is passed over the input matrix. At each step, the dot product of the filter and the viewed part of the input matrix with the size of $\kappa \times \kappa$ are calculated and is recorded as the output of the convolution at that step. The process continues until the filter passes all over the input matrix. The output is a matrix, $\mathcal{A}$, with the new size of $n_1 \times n_1$, where $n_1 << n$.

(ii) Activation layer: The activation layer, which is a non-linear function, allows the network to train itself via back-propagation process during the training phase. The output of the convolution layer is passed to this non-linear function. The ReLu function [86] is typically used in this layer.

(iii) Pooling layer: The pooling layer down-samples and reduces the size of the matrix obtained from the previous layer. A filter is passed over the matrix and for each group of the values, one number is selected which is usually the maximum value in the group. The selection of the maximum value is so called max pooling [87]. This layer leverages the speed the training phase and allows the network to focus on the most important information.

(iv) Fully connected layer: This layer is as the same as MLP layers. The input is a one-dimensional vector representing the output of the previous layers. The output is a set of probabilities for different possible classes. The class with the highest probability is the chosen class as the output of the CNN which is usually carried out by using softmax function [42].

2.3.5. Dimension Reduction by Principal Component Analysis (PCA)

Principal Component Analysis (PCA) [88] is an unsupervised statistical learning method widely used in ML applications for dimension reduction. Let $A_{n\times u}$ be a matrix of u different variables and n samples for each variable. The first step of applying PCA is to find the corresponding covariance matrix of A. The covariance of two variables $\mathcal{U}_1 = \{u_{1_1}, \ldots, u_{1_n}\}$ and $\mathcal{U}_2 = \{u_{2_1}, \ldots, u_{2_n}\}$ is defined as:

$$COV(\mathcal{U}_1, \mathcal{U}_2) = \sum_{i=1}^{n} (u_{1_i} - \mu_{u_1})(u_{2_i} - \mu_{u_2}), \tag{2.17}$$

where, μ_{u_1} and μ_{u_2} are the mean values of $\mathcal{U}_1$ and $\mathcal{U}_2$ respectively. Using equation 2.17, the covariance matrix, $\mathcal{C}_{u\times u}$, of A is obtained. The next step is to find the eigenvectors and the eigenvalues of the matrix, $\mathcal{C}$. An eigenvector, ν and the corresponding eigenvalue, λ of a matrix $\mathcal{C}$ satisfy the equation below:

$$\mathcal{C}\nu = \lambda\nu, \tag{2.18}$$

where λ is a scalar. Once the eigenvectors and the corresponding eigenvalues of the C matrix are obtained, in order to reduce the dimension of the matrix A from $n \times u$

to $n \times j$, $j < u$, the eigenvectors are sorted by decreasing the eigenvalues and the first j largest eigenvectors are chosen to create a matrix, $\mathcal{V}_{u\times j}$. Finally, the dimension reduction is carried out as:

$$\acute{A}_{n\times j} = A_{n\times u}\mathcal{V}_{u\times j}. \tag{2.19}$$

2.3.6. Performance Metrics

In this thesis, we consider the problem of the separation attack traffic from normal one. Therefore, there are some parameters that should be taken into account while evaluating the performance of detection including true positive (TP), true negative (TN), false positive (FP) and false negative (FN) [42]. TP and TN are two metrics that reflect the success rate of true detection. While TP shows the number of attack samples detected by the algorithm, TN on the other hand, represents the number of benign samples which are correctly labeled as normal instances. FP and FN, in contrast, are used to show the failure of the method in correct detection. FP is related to the number of normal samples incorrectly classified as malicious samples and FN is the number of attack samples that are misplaced as normal.

Using the four previously mentioned parameters, we utilize five metrics including true positive rate (TPr), false positive rate (FPr), precision (PEr), $F1\_score$ and total accuracy (ACC). While $F1\_score$ gives a better measure of the incorrectly classified cases, ACC represents the ability of the system in true detection of both normal and DDoS attack. TPr, FPr, PEr, $F1\_score$ and ACC are defined as:

- True positive rate/ Recall: $TPr = \frac{TP}{TP+FN}$
- False positive (alarm) rate: $FPr = \frac{FP}{FP+TN}$
- Precision: $PEr = \frac{TP}{TP+FP}$
- F1_score: $F1\_score = \frac{2TP}{2TP+FP+FN}$
- Total accuracy: $ACC = \frac{TP+TN}{TP+TN+FP+FN}$

Furthermore, we employ Receiver Operating Characteristic, ROC, curve [89] to observe the performance of the scheme in separating attack from the normal traffic. The Area Under the Curve of ROC, AUC, represents the degree of separability. The ROC curve is plotted with TPr against the FPr where TPr is on y-axis and FPr is on the x-axis.

2.4. DDoS Attack Detection in Traditional Networks

There exist various studies that use time-series based methods to analyze traffic patterns to discover abnormalities caused by DDoS attacks in traditional networks [90–98]. ARIMA is one of the time-series based methods widely used in anomaly detection. A time-series based DDoS attack detection is proposed in [90]. ARIMA model is employed to predict the flow of packets for each second and if the estimated value exceeds a certain pre-defined threshold, an alarm is raised. Authors in [91], propose an ARIMA model time-series to detect DDoS attack. The number of packets in every one minute is sampled. Then number of packets in the following minute is estimated using ARIMA model. The repeatability of chaotic behavior and the enormous growth in the ratio of number of packets to the number of source IP addresses are used to identify attack. ML-based approaches are also employed to detect DDoS attack traffic. A real-time application layer DDoS attack detection based on time-series analysis is proposed in [92]. The time-series is generated based on the entropy of HTTP GET request per the number of source IP address. Adaptive Auto-Regressive model (AAR) [99] is employed to create time-series. Support Vector Machine (SVM) classifier [100] is implemented to classify attack. Twelve essential features which are appropriate for ML algorithms are extracted from network traffic in [93]. Four ML algorithms including Naïve Bayes, decision tree [101], SVM and MLP are applied in [94] to classify various types of DDoS attacks. Naïve Bayes is used in [95] to detect DDoS attack samples. The classifier is implemented in Weka tools [102] to analyze the network traffic. The proposed model achieves 99% accuracy. Naïve Bayes, random forest and MLP are employed in [96] to classify DDoS attack samples. MLP outperforms in classifying traffic samples. A hybrid neural network based on SOM is employed in an intrusion detection system

proposed by [97] for DDoS attack detection. An intrusion detection system based on a two-layer MLP is proposed by [98] to classify network traffic into six types of traffic. The proposed system achieves almost 91% accuracy.

Although ML approaches can yield higher accuracy compared to traditional approaches, choosing an appropriate set of features is the most important step when using an ML algorithm. There exist many feature selection and extraction methods such as statistical based, entropy based and etc [103]. Frequency domain analysis is a famous approach widely used in signal processing applications such as speech processing and vital signs signal analysis where it has proved its higher accuracy performance in feature extraction [104]. Various studies employ the capability of the frequency-domain based methods to analyze traffic patterns, and to discover abnormalities [30, 105–109]. Authors in [30] study the energy distribution of DDoS attack in frequency-domain. According to the study, the majority of DDoS attack energy resides in lower frequency band. Using the spectrum energy and simple thresholding method, authors in [105] separate Low-rate DoS attack (LDoS) from normal traffics. The spectral energy within the main lobe is used as the threshold value. Two stage Reduction of Quality (RoQ) attack detection method is proposed in [106]. At the first stage, wavelet analysis is used to detect potential attack and then as the next step, auto-correlation analysis is employed to extract attack characteristics. The power of spectrum energy in identifying normal TCP traffic is discussed in [107]. Wavelet filters are used in different anomaly scenarios in [108]. These filters are found to be quite effective at exposing the details of anomaly traffics. Authors use spectral analysis as one part of their detection of DoS attack method in [109].

In this thesis, we contribute to the previous studies by proposing three different DDoS attack detection schemes based on time-series analysis for traditional networks. We propose to use of statistical features including periodicity, skewness, kurtosis and Hurst parameter, obtained from the network traffic, to discriminate DDoS attack from normal traffic. Moreover, the features obtained from the frequency-domain representation of network traffic are employed in two proposed schemes to detect DDoS attack

samples. In the first scheme, the DWT and DFT features are introduced to the naïve Bayes classifier to separate attack from the normal traffic and in the second one, first, features are obtained from the frequency-domain by applying sparse coding algorithms and then introduced to SOM model to find anomalies caused by DDoS attack traffic.

2.5. DDoS Attack Detection in Software-Defined Networking

In this section, we discuss the concept of SDN and how SDN differs from traditional networks. Then, we mention the existing studies related to DDoS attack detection for SDNs.

2.5.1. Software Defined Networks

Software-defined networking (SDN) becomes more popular in IT worlds every year. Employing a centralized controller mechanism, SDN outperforms traditional networks in terms of the management and monitoring capabilities that they provide for the large-scale networks [110]. Each network architecture consists of three main planes, namely control, data, and management [111] which are defined in below:

Definition 8. *Control Plane: The control plane calculates and programs actions for the data plane. All routing decisions and other network functions such as quality of service are employed in this plane.*

Definition 9. *Data Plane: The data plane is responsible for forwarding the data packets according to the actions based on rules that are programmed into lookup tables by the control plane. The data forwarding should be carried out as fast as possible in this plane*

Definition 10. *Management Plane: All network devices can be accessed and managed in this plane. To modify the devices in the network, several tools such as SSH, telnet and the console port are used.*

In traditional networks, all three planes reside in the single network components such as switches or routers. Therefore, the burden of all processes that are carried out in each plane is on the network components. Additionally, those network components are provided by networking hardware vendors with the inflexible software, which cannot be modified by the user. Due to the increasing amount of network traffic and the dynamically changing network structures, the traditional networks cannot meet current networking requirements such as scalability, central control and management. Therefore, to deal with these problems, SDNs have replaced traditional networks for medium or large sized networks. Table 2.1 compares SDN and traditional networks in terms of various characteristics such as programmability, complexity, flexibility, etc., as the table indicates, SDN addresses existing issues in traditional networks [112].

Table 2.1. SDN vs. traditional network architecture [112].

Properties	**SDN**	**Traditional**
Programmability	✓	✗
Centralized control	✓	✗
Error-pron configuration	✗	✓
Complex control	✗	✓
Flexibility	✓	✗
Better performance	✓	✗
Easy implementation	✓	✗
Efficient configuration	✓	✗
Better management	✓	✗

While adapting the functionality of the traditional network architecture, SDN introduces new components to the networking infrastructure. Instead of having all three planes in a switch, SDN separates the control and data planes to diminish the burden on traditional networks caused by lack of a central control mechanism [113]. This separation between the two planes increases the flexibility of network administration during troubleshooting and monitoring. In addition to data and control planes,

SDN also comprises an application plane for software programs to provide efficient solutions for basic network functionalities such as load-balancing, intrusion detection, traffic monitoring, etc [114]. All the communications between the controller and the switches in SDN are defined via the specific protocols such as OpenFlow (OF) [115] and XMPP [116]. In this thesis, all communication between the controller and switches are carried out by using OF protocol; therefore, the following paragraphs give more details related to this protocol. The OF protocol was first proposed in 2008 [115]. The main idea is to divide the network into two separate parts of control plane and data plane, which communicate with each other through a secure channel. Packets are always handled as "flows" in an OF switch. When a new packet arrives at a switch, it is compared with the packets which are recorded as the entries inside the flow table of the switch. If it is matched with an instance in the flow table, the associated action is executed and the counter of the flow entry increments; otherwise, the new unknown packet is encapsulated into Packet-In massage and sent to the controller. A flow entry mainly includes three parts of header, counter and action. Header consists of different segments of a packet, such as IP addresses, ports, type of protocol, etc. When the new packet arrives, its field is compared against the header of the flow entry to find a match. The counter stores the information of the packets matching this flow entry. The action part determines the type of action if a match is found.

2.5.2. DDoS Attack Detection For SDNs

The centralized controller of the SDN, makes the system more vulnerable to DDoS attacks; therefore, DDoS attack detection methods for SDNs are studied widely in literature. In general, DDoS attack detection algorithms in the literature can be categorized as intrinsic and extrinsic [12]. While the former is related to the structural changes in SDN, the latter corresponds to the flow-based analysis. Since they provide better detection accuracy as far as the SDN environment is concerned, we focus on flow-based solutions in this work. Such solutions are also classified as either the statistical-based or the ML-based. Most of the works related to the statistical approach, are based on entropy concept. Entropy is the measure of randomness of an attribute within a

specific period of time. For instance, a random variable with a high spread probability distribution has high entropy. On the other hand, in order to differentiate the attack traffic from the normal one, a ML method needs to be fed by a set of training samples. Each sample consists of a set of features obtained from the network traffic data to generalize the discriminating model.

Due to the promising results of entropy-based methods in detecting anomalies in traditional networks [117–119], they have also been adopted by detection algorithms for SDN. The maximum entropy estimation method [120] is employed in [121] to estimate the benign traffic distribution to solve the security issues in home and office networks. The protocol type and the destination port numbers are used as the feature for maximum entropy estimation. The entropy of source and destination IP addresses and ports are employed in [122] to detect DDoS, worm propagation and port-scan attacks. Another entropy-based approach is proposed in [123]. The entropy of destination IP address at each switch is employed to find anomalies. In the case of any DDoS attack incident, alert information is sent to the controller. In [10], a threshold is obtained by using the entropy of the destination IP addresses. Later, this threshold is used to determine whether there is an attack or not. Particularly, if the entropy of the test sample is less than the defined threshold, it is labeled as an attack. In [12], an entropy-based method [124] is proposed for DDoS attack detection and mitigation. Different features from payload are extracted and used for the detection of the attack. The algorithm has three stages of nominal, preparatory and active mitigation. While the normal pattern baseline is obtained in an attack free scenario at nominal stage, the attack samples are identified at the preparatory stage. Later, the countermeasure against the attack traffic is carried out at the active mitigation stage. A combination of entropy and information distance is used in [11] to check the traffic for the DDoS attack detection. Another important example method [13] evaluates the entropy of different flow features (source IP, destination IP, source port, destination port, protocol and flow rate, etc.) while detecting DDoS attacks. A DDoS attack detection for SDNs using entropy and Kullback-Leibler divergence [125] is proposed in [14]. This method particularly analyzes the change in the source and destination IP addresses' probability distribu-

tions using Kullback-Leibler divergence and entropy values. In [15], an entropy-based (D)DoS detection in IoT-based SDNs is proposed and different flow features including source/destination IP addresses, source/destination ports are employed for detecting anomalies in the network traffic.

SDN-based DDoS attack detection methods also use ML algorithms for the detection procedure. Some features from the network flow are extracted and used by a ML model to differentiate malicious traffics from benign ones. In [126], six features including, Average of Packets per Flow (APF), Average of Bytes per Flow (ABF), Average of Duration per Flow (ADF), Percentage of Pair-Flow (PPF), Grows of Single-Flow (GSF) and Grows of Different Port (GDP) are extracted from the network traffic and introduced into SOM [38] to create a selective model for separating attacks from normal traffic. In [16], a DDoS detection algorithm based on entropy and ML approaches is proposed. In the first phase, lightweight, a method based on entropy is used to detect attack traffics. In the second phase, heavyweight, different ML approaches are employed to classify abnormal traffics. In [19], authors use different ML methods to detect DDoS attack in SDN. According to the result, SVM [100] algorithm has the best performance. In [20], a set of five features consisting of entropy of Source IP (etsSrcIP), entropy of Source Port (etsSrcP), entropy of Destination Port (etsDstP), entropy of packet Protocol (etsProtocol) and total number of Packets (totalPacket) is fed into an SOM to classify the traffic as normal or attack. In [18], SVM and Idle time out Adjustment (IA) are used to detect DDoS attacks and to classify the traffic. The entropy of source and destination IP addresses are used as the vector for SVM algorithm to generalize a specific DDoS attack detection model in [21]. Four different features including Byte Rate (BR), Symmetric Flows Percentage (SFP), Variation Rate of Asymmetric Flow (VAFR) and Flows Percentage with Small Amount packets (FPSA) are extracted in [22]. The features are used as a four-tuple feature by a back propagation neural network (BPNN) to discriminate attack from normal traffics. In [127], different features are collected from the SDN and used in ML algorithm to detect the DDoS attack. Four different ML algorithms including SVM, K-Nearest Neighbor (KNN) [128], Artificial Neural Network (ANN) [129] and Naïve Bayes [130],

are employed. KNN and NB outperform with respect to success rates. In [17], a stack auto-encoder based on deep learning model is employed to classify the data obtained from the OF switch. A deep learning-based DDoS detection and defense architecture is proposed in [23]. The detection part consists of multiple deep learning layers such as CNN, LSTM [131] and bidirectional RNN [132]. In [24], a deep learning model is proposed to detect DDoS attack in SDN. The statistics during the attack are analyzed and employed by the deep learning approach to classify the traffic into two classes, i.e., malicious and legitimate. Convolutional neural networks also present good attack detection accuracy as introduced in [26]. [25], different ensemble models including ensemble CNN, ensemble RNN, ensemble LSTM, and hybrid Reinforcement Learning (RL) [133] are used for DDoS detection in SDNs. The result shows that the ensemble CNN outperforms.

In this thesis, we contribute to previous studies by proposing three DDoS attack defense schemes for SDN architecture which are integrated into the controller of the SDN architecture. The first scheme benefits from time-series based ARIMA model and exponential smoothing method to address the constant threshold issue in the previous statistical-based approaches. The second proposed scheme employs the statistical features of DWT to find the normal pattern of the traffic in each switch. The auto-encoder neural network is used to generate the normal model. The third scheme uses CWT with the CNN architecture to discriminate attack flows from normal ones in each switch in the SDN network. The comparison between our proposed schemes and the previous studies is summarized in Table 2.2.

Table 2.2. DDoS attack detection for SDN state of the art.

Paper	**Method**	**Approach**	**Dynamic Threshold**	**Time Domain**	**Frequency Domain**
Mehdi et al., [121]	Statistical	Entropy	✗	✗	✗
Giotis et al., [122]	Statistical	Entropy	✗	✗	✗
Wang et al., [123]	Statistical	Entropy	✗	✗	✗
Mousavi et al., [10]	Statistical	Entropy	✗	✗	✗
Kalkan et al., [12]	Statistical	Entropy	✗	✗	✗
Sahoo et al., [11]	Statistical	Entropy	✗	✗	✗
Ahalawat et al., [13]	Statistical	Entropy	✗	✗	✗
Abdelazim et al., [14]	Statistical	Entropy	✗	✗	✗
Galeano et al., [15]	Statistical	Entropy	✗	✗	✗
Proposed scheme #1	Statistical	ARIMA, dynamic threshold	✓	✗	✗
Braga et al., [126]	ML	SOM	✗	✓	✗
Da Silva et al., [16]	Statistical+ ML	Entropy, K-means, SVM	✗	✓	✗
Meti et al., [19]	ML	SVM, Naïve Bayes, ANN	✗	✓	✗
Nam et al., [20]	ML	SOM	✗	✓	✗
Phan et al., [18]	ML	SVM	✗	✓	✗
Cui et al., [21]	ML	SVM	✗	✓	✗
Wang et al., [22]	ML	BPNN	✗	✓	✗
Polat et al., [127]	ML	SVM, KNN, ANN, Naïve Bayes	✗	✓	✗
Niyaz et al., [17]	ML	Stack auto-encoder	✗	✓	✗
Li et al., [23]	ML	CNN, LSTM, RNN	✗	✓	✗
Rasool et al., [24]	ML	ANN	✗	✓	✗
Zhao et al., [26]	ML	CNN	✗	✓	✗
Haider et al., [25]	ML	CNN, RNN, LSTM, RL	✗	✓	✗
Proposed scheme #2	Statistical+ ML	Hit-rate, Auto-encoder	✗	✗	✓
Proposed scheme #3	Statistical+ ML	CNN	✗	✗	✓

3. DDOS ATTACK DETECTION ON TRADITIONAL NETWORKS

In this chapter, we introduce our DDoS attack detection study results on traditional networks. For this respect, we propose three different schemes based on frequency domain analysis, statistical analysis and ML approaches. The detailed descriptions of our schemes are introduced in the following sections.

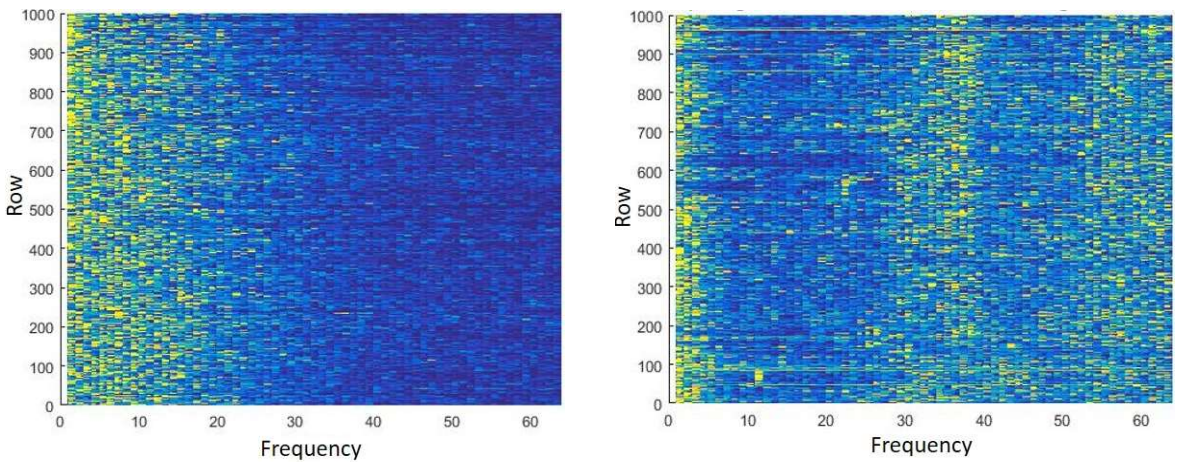
3.1. Frequency Based DDoS Attack Detection Approach Using Naïve Bayes Classification

In this study, we propose a DDoS attack detection method based on frequency domain analysis. The analysis is initiated by representing the number of arriving packets which are sampled for each specific time interval, in a time-series representation. Then, coefficients of Discrete Wavelet Transform [32] and absolute value of Discrete Fourier Transform [31] of the time-series are used as features to separate DDoS attack from normal traffic. Wavelet transform provides higher resolution information about frequency domain that increases the accuracy of the detection. Naïve Bayes classifier [33] is employed to classify attack and normal traffic. In order to compare the performance of the proposed method, results are compared with a simple thresholding classifier [34]. Attack samples extracted from real network traffic (namely booter) [134] and normal network traffic of Boğaziçi University [135] are used as attack and normal datasets, respectively. The method is implemented using MATLAB R2015a [136] and Weka 3.6 [102].

3.1.1. Proposed Method

To detect DDoS attacks, passive monitoring is used together with an offline dataset. A time-series data, which is represented as $\mathcal{Z} = \{z_1, \ldots, z_N\}$, is generated by sampling the number of packets for each time interval $t = 1$ ms. Then, $\mathcal{Z}$ is further

divided into $w = 128$-length windows that each window shares the last $d = 64$ values with the consecutive window. The absolute value of DFT and Haar wavelet [137] are applied on each window and associated coefficients are saved as a row in two separate matrices of A_{DFT} and A_{DWT}, respectively. Each row in each matrix is normalized with respect to the sum of all values in the related row; as a result, each element in the corresponding row can be considered as a probability value. The obtained matrices are employed as features by the naïve Bayes classifier for DDoS attack detection. Later, in the training phase, the naïve Bayes model is trained and the optimum threshold value is computed. For this purpose, the first 1000 rows of A_{DFT} and A_{DWT} are used in the training phase. Additionally, DFT and DWT thresholds are obtained for the thresholding method. Details of the scheme are given in the following sections.



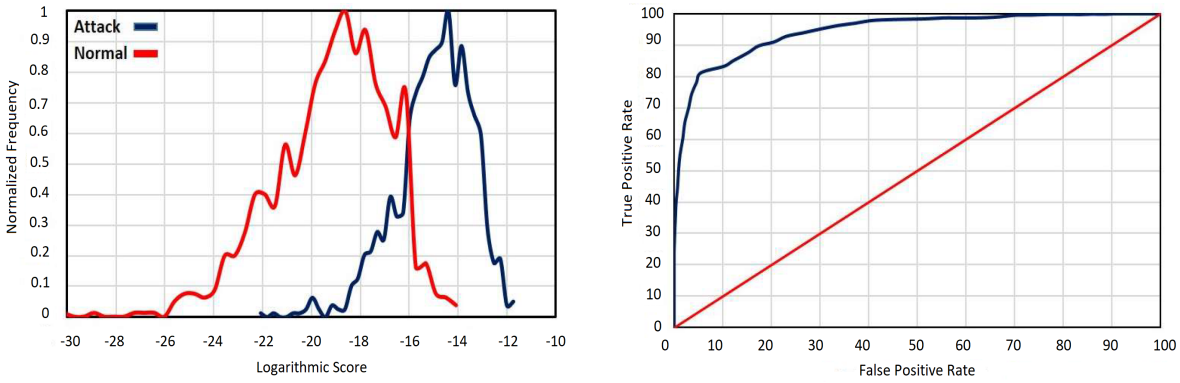
(a) Attack. (b) Normal.

Figure 3.1. Spectrogram of attack and normal traffic for DFT analysis, (a) Attack, (b) Normal.

3.1.2. DFT-based Threshold Selection

TCP protocol is the major protocol in Internet traffic and traffic / congestion mechanisms and Round-Trip Time (RTT) used in TCP protocol induce periodic patterns to the packet arrival on traffic flows; therefore, in the frequency domain the energy is distributed in various boundaries of frequency [138]. Moreover, the energy of DDoS attack resides in lower frequencies [30]. Figure 3.1 displays the spectrogram of training dataset of A_{DFT} matrices for both normal and DDoS traffics. Blue and

yellow correspond to the regions with lower and higher frequency energy, respectively. In contrast to the main energy of attack traffic, which resides in lower frequencies, the normal one is distributed in three different bounds of energy. Although all 64 columns of A_{DFT} can be used to estimate the score, by considering Figure 3.1, using just the first ten components of each row in A_{DFT} would be enough to separate attack traffic from normal one. Therefore, for each row, a probability score is obtained by the product of the first ten elements in the corresponding row. Figure 3.2 illustrates the normalized histogram of logarithm values of the attack and normal scores estimated using first ten components and corresponding Receiver Operating Characteristic (ROC) curve [89]. Considering both histogram and ROC, -17 is chosen as the threshold value to separate attack from the normal traffic in the thresholding method.



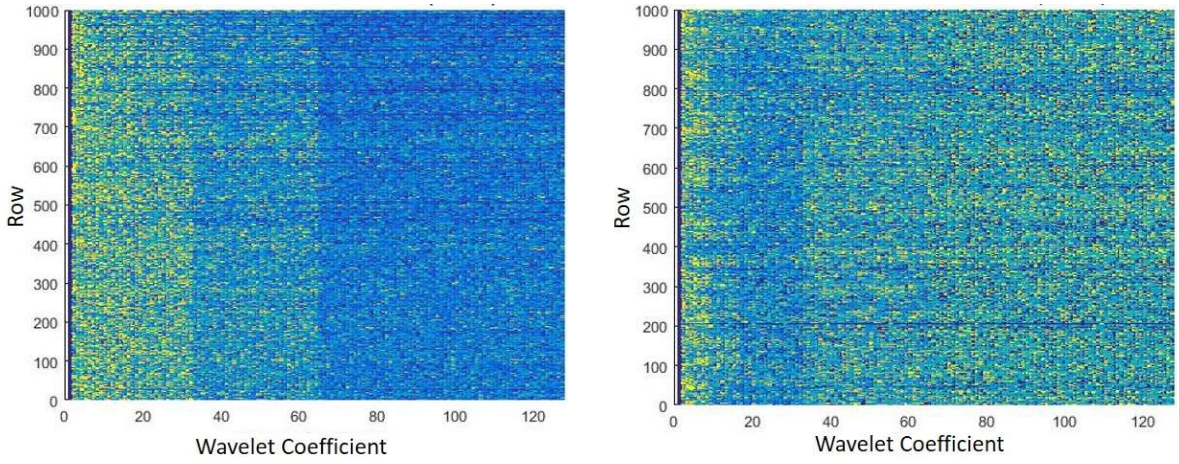
(a) Histogram. (b) ROC.

Figure 3.2. Thresholding classification using scores obtained by the first 10 frequency components of DFT in training dataset, (a) Histogram, (b) ROC.

3.1.3. DWT-based Threshold Selection

Wavelet transform not only gives information about the presence of frequency components in the signal but also provides more knowledge about their locations. Figure 3.3 displays the wavelet training dataset for both normal and attack traffics. Different bounds of energy are occupied by the wavelet coefficients of normal traffic except the boundary approximately between the 10^{th} and 30^{th} coefficients. DDoS attack energy bound again resides in lower frequencies. Because wavelet method analyzes frequency components in various scales, the resolution of energy distribution using

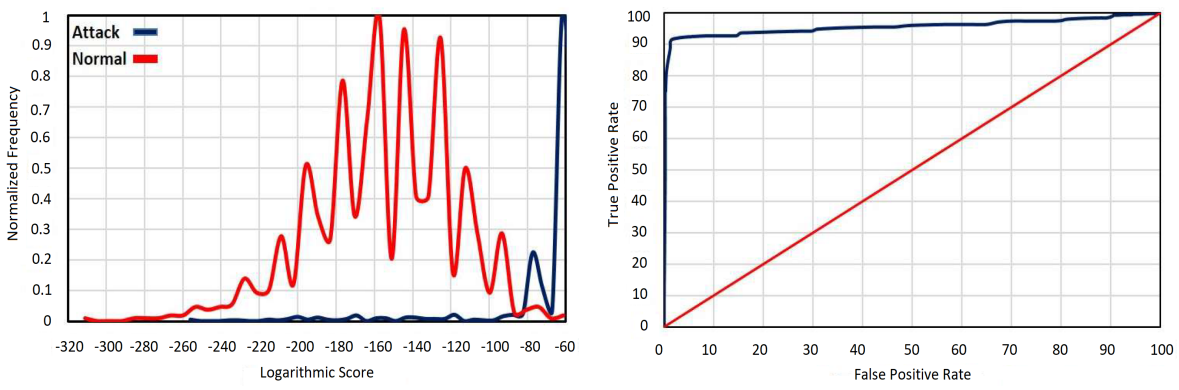
wavelet is more than that of DFT method. Taking Figure 3.3 into account, for this method, we just use the first 30 coefficients to generate the probability score in order to segregate between attack and normal traffic.



(a) Attack. (b) Normal.

Figure 3.3. Haar wavelet coefficients of attack and normal traffic for DWT analysis, (a) Attack, (b) Normal.

Figure 3.4 displays the normalized histogram of logarithm value of the attack and normal scores evaluated by using first 30 components and corresponding ROC. Considering both histogram and ROC, -63 is chosen as the threshold value to separate attack from normal traffic in the thresholding method.



(a) Histogram. (b) ROC.

Figure 3.4. Thresholding classification using scores obtained by the first 30 frequency components of DWT in training dataset, (a) Histogram, (b) ROC.

Table 3.1. Confusion table for naïve Bayes and thresholding methods.

Classifier	Dataset		Attack	Normal	$TPr(\%)$	$FPr(\%)$	$ACC(\%)$	F1_score
Thresholding	**DFT**	**Attack**	2229	420	84.14	13.06	85.32	0.87
		Normal	253	1683				
Thresholding	**DWT**	**Attack**	1575	1074	59.46	0.05	76.55	0.75
		Normal	1	1935				
Naïve Bayes	**DFT**	**Attack**	2548	101	96.19	7.33	94.72	0.95
		Normal	142	1794				
Naïve Bayes	**DWT**	**Attack**	2303	346	86.94	4.18	90.64	0.92
		Normal	81	1855				
Naïve Bayes	**DFT+DWT**	**Attack**	2564	85	96.79	5.32	95.93	0.96
		Normal	103	1833				

3.1.4. Test Results

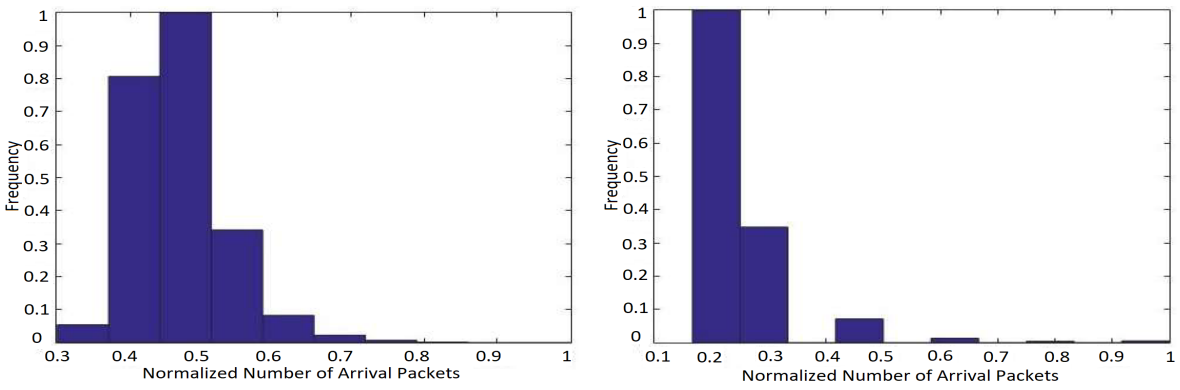
Once the training phase is completed, test data is fed to both naïve Bayes classifier and thresholding method. Test dataset consists of 1936 and 2649 samples of normal and attack traffics, respectively. Three different feature sets including DFT, DWT and DFT+DWT (combined feature) are provided to the naïve Bayes classifier. For the thresholding method, the first 10 elements and the first 30 elements of DFT and DWT are multiplied to obtain the corresponding scores, respectively. The logarithm of scores is taken and compared against threshold obtained during the training phase. The confusion tables of various experiments are summarized in Table 3.1. In the thresholding method, although the FPr of classifier using DWT feature is lower, the classifier categorizes more than 40% of attack samples as normal traffic which results in lower accuracy. On the other hand, by selecting -17 as the threshold value for the classifier based on DFT feature, we achieve the FPr about 13% which yields higher accuracy around 85% compared to 76% of DWT based classifier.

Like thresholding method, DWT-based naïve Bayes classifier has lowest FPr, but its accuracy is lower than those of DFT and combined methods. The combined feature results in lower FPr which improves the accuracy by 1.21% compared to the result of DFT feature alone. It indicates that although DWT feature results in low accuracy, it can be employed as a complimentary for DFT feature to improve the overall accuracy.

All in all, comparing two classifiers, naïve Bayes outperforms thresholding method for both features and the highest accuracy belongs to naïve Bayes classifier with combined feature.

3.2. Statistical Measures: Promising Features For Time-Series Based DDoS Attack Detection.

In this study, we extract four statistical measures, namely periodicity [139], kurtosis, skewness [35] and Hurst exponent [36] from the time series data of the network traffic to investigate their capabilities in separating the DDoS attack samples from the normal ones. For the attack traffic and the normal one, we use CAIDA 2007 and CAIDA 2008 datasets [140], respectively. All simulations are carried out by Matlab R2016a [136]. Passive monitoring is used to obtain data for both normal and DDoS traffic data. Also, to generate time-series $\mathcal{Z}$, the number of arrival packets for each time interval, $t = 1$ ms are counted. The time-series data, $\mathcal{Z}$, is further divided into $w = 1024$-length windows. Later we used periodicity, kurtosis, skewness and Hurst parameter values of each window to discriminate the attack traffic from normal one.



(a) Normal. (b) Attack.

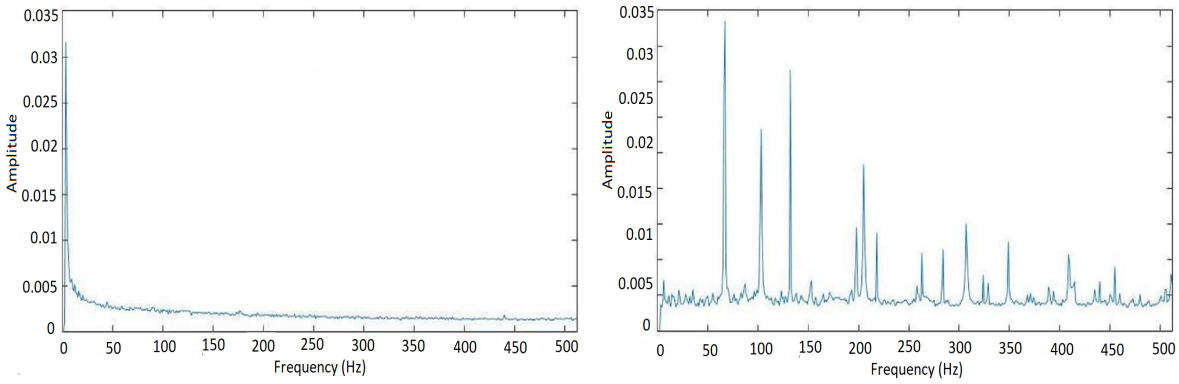
Figure 3.5. Normalized histogram of packets, (a) Normal, (b) Attack.

In total, 400 windows are obtained and they are divided into training ($85\% - 340$ windows) and test ($15\% - 60$ windows) datasets. Figure 3.5 represents the histogram of both normal and traffic number of packets ($\mathcal{Z}$). Both histograms are normalized to make the number of arrival packets reside in $[0, 1]$ interval. As illustrated in Figure

3.5, the normalized number of arrival packets follows different probability distribution characteristics for attack traffic and normal traffic; however, two histograms overlap with each other in some intervals which result in lower DDoS detection performance. Therefore, to improve the DDoS detection performance, one alternative solution could be to use the statistical features of $\mathcal{Z}$. Table 3.2 summarizes the statistical measures of both traffics. Those traffics are heavy tailed and right skewed. Table 3.2 confirms that both kurtosis and skewness can be used as discriminating features to separate attack from normal traffic. In order to use aforementioned features to separate attack from normal traffic, the thresholding method is employed and for each statistical measure, a threshold is obtained which the process of choosing the threshold is discussed in the following section.

Table 3.2. Statistical measures for normal and DDoS attack traffics.

	μ	σ	**Kurtosis**	**Skewness**
Normal	0.47	0.06	4.63	0.98
DDoS Attack	0.23	0.12	11.42	2.42



(a) Normal. (b) Attack.

Figure 3.6. Normalized averaged periodogram, (a) Normal, (b) Attack.

3.2.1. Discriminating Threshold Selection

The number of periodic components in a time-series can be employed to discriminate attack traffic from normal traffic. Frequency domain analysis gives more details

related to available periodic components. According to the Nyquist theorem [78], for time interval of $t = 1$ ms, the maximum frequency which can be realized, is 500 Hz. By $w = 1024$-length window, we achieve 0.488 Hz frequency resolution, distributed in 512 bins. Figure 3.6 displays the normalized average periodogram of training datasets. In contrast to the main energy of the normal traffic which resides in lower frequencies, the attack traffic is distributed in various bounds of frequencies. Moreover, it is also deduced that attack traffic has more periodic components than normal traffic. Figure 3.7 displays the normalized histogram of number of periodic components for both traffics. Both traffics share different number of periodic components; therefore, this feature would not be a good choice for DDoS attack detection.

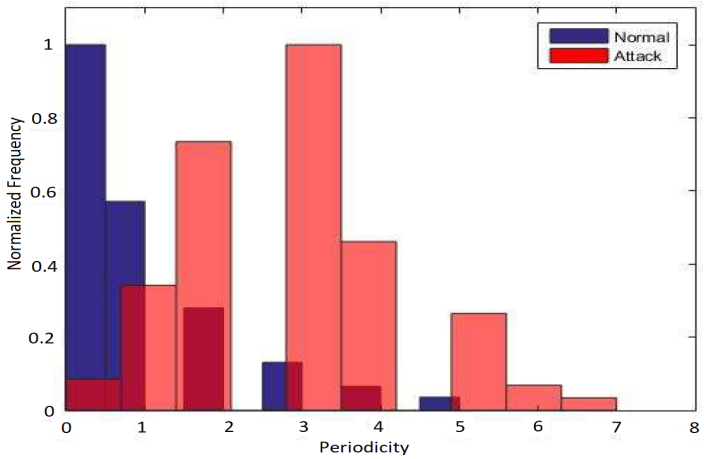


Figure 3.7. Normalized histogram of number of periodic components.

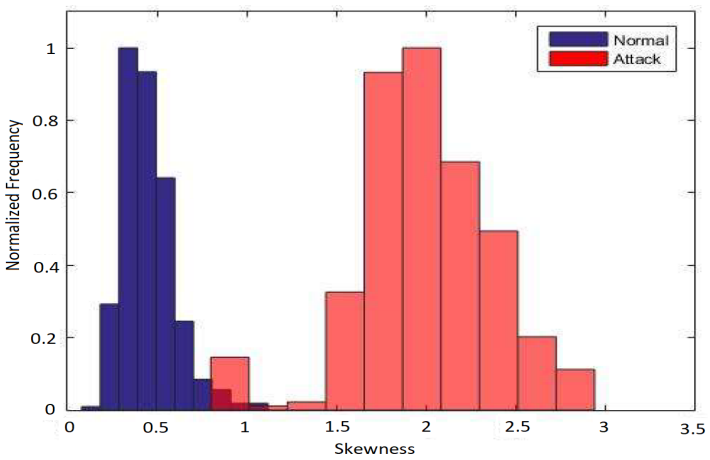


Figure 3.8. Normalized histogram of skewness.

Figure 3.8 displays the normalized histogram of skewness values for training windows. Although both traffics are right skewed, the attack traffic tends to be more asymmetric to the right than normal one which can be confirmed as illustrated in Figure 3.5. Figure 3.9 shows the normalized histogram of kurtosis value for both normal and attack traffics. Both traffics have heavy tailed distribution. While the kurtosis of normal traffic is distributed near the value of 4, the kurtosis of attack has a large variance. The Hurst exponent distribution for both traffics is illustrated in Figure 3.10. The attack traffic tends to have anti-persistent behavior; on the other hand, normal traffic has persistent behavior.

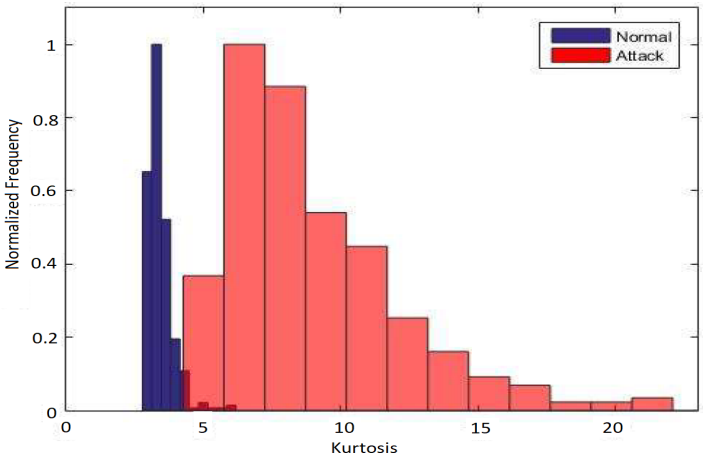


Figure 3.9. Normalized histogram of kurtosis.

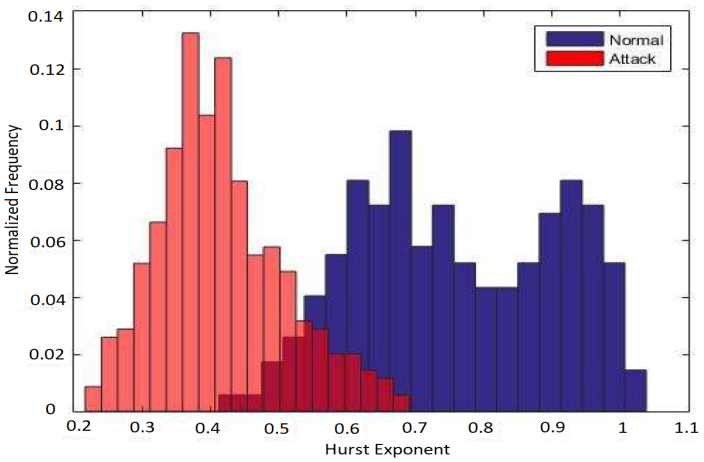


Figure 3.10. Normalized histogram of Hurst exponent.

As the performance criterion, we utilize the ROC curve. The discrimination accuracy of a model can be related to the area under ROC curve which is so called Area Under the Curve (AUC) [89]. Figure 3.11 displays the ROC of four different statistical parameters. The corresponding AUC and threshold values are summarized in Table 3.3.

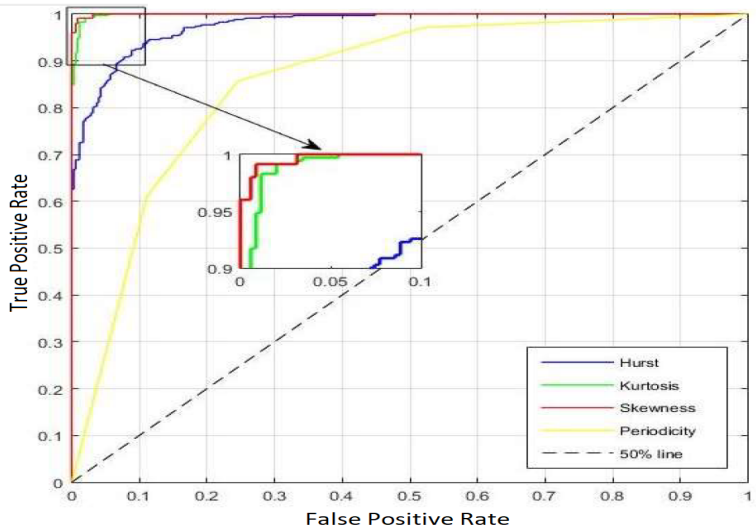


Figure 3.11. ROC of four statistical measures.

Table 3.3. AUC and threshold values for statistical measures.

	Period	**Kurtosis**	**Skewness**	**Hurst Exponent**
AUC (%)	85.66	99.86	99.95	97.96
Threshold value	2	4.5	1	0.57

3.2.2. Experimental Results

Once the threshold value for each statistical parameter is obtained, the test set is used to examine the performance of each measure in separating attack from normal traffic. If the values of kurtosis, skewness, and periodic components greater than those thresholds specified in Table 3.3, the test sample is considered as an attack; otherwise, it is classified as a normal instance. For Hurst parameter, values less than corresponding threshold are considered as the attack. Each test traffic consists of 60 sample windows.

Using thresholds from Table 3.3 and then carrying out binary hypothesis test, the confusion table for all measures is obtained which is summarized in Table 3.4. As given in Table 3.3, skewness outperforms all other measures. Although both histograms share a common area less than the threshold value of 1 in Figure 3.8, it can be considered to be the effect of outliers on attack traffic. Considering the histogram of number of packets is illustrated in Figure 3.5, the normal traffic tends to be more symmetric than does attack traffic; therefore, high performance of skewness in detection is logically expected. The worst performance belongs to the period parameter. The reason for this low performance can be inferred from Figure 3.7. It is clear that both normal and attack overlap each other in different bounds of histogram. Kurtosis and Hurst exponent also perform well enough in detecting DDoS attack. The test results agree with the outcomes of the training phase.

Table 3.4. Confusion table of statistical measures.

		Normal	**Attack**	TPr(%)	FPr(%)	ACC (%)	F1_score
Periodicity	**Normal**	46	14	80	23.33	78.33	0.79
	Attack	12	48				
Kurtosis	**Normal**	58	2	95	3.33	95.83	0.96
	Attack	3	57				
Skewness	**Normal**	60	0	96.67	0	98.33	0.98
	Attack	2	58				
Hurst exponent	**Normal**	55	5	91.66	8.33	91.66	0.92
	Attack	5	55				

3.3. Anomaly-Based DDoS Attack Detection by Using Sparse Coding and Frequency Domain

In this study, we propose an anomaly-based DDoS detection using sparse representation model and frequency domain analysis. To the best of our knowledge, this is the first jointly use of these two methods for DDoS detection. The proposed model uses the absolute value of DFT of the normal traffic to learn a sparse dictionary by employing K-SVD [141] algorithm which is a general form of K-means [142] cluster-

ing method. Moreover, the sparse dictionary is employed by Basic Matching Pursuit (BMP) algorithm to generate sparse coefficients of absolute value of DFT for both normal and attack traffics [143]. The normal behavior model is created by using sparse coefficients of the normal traffic and the Self-Organizing Map (SOM) lattice [38]. Finally, we evaluate the performance and the applicability of the proposed model by using CAIDA dataset [140]. Numerical evaluations show that the proposed model achieves high performance rates in terms of the detection performance.

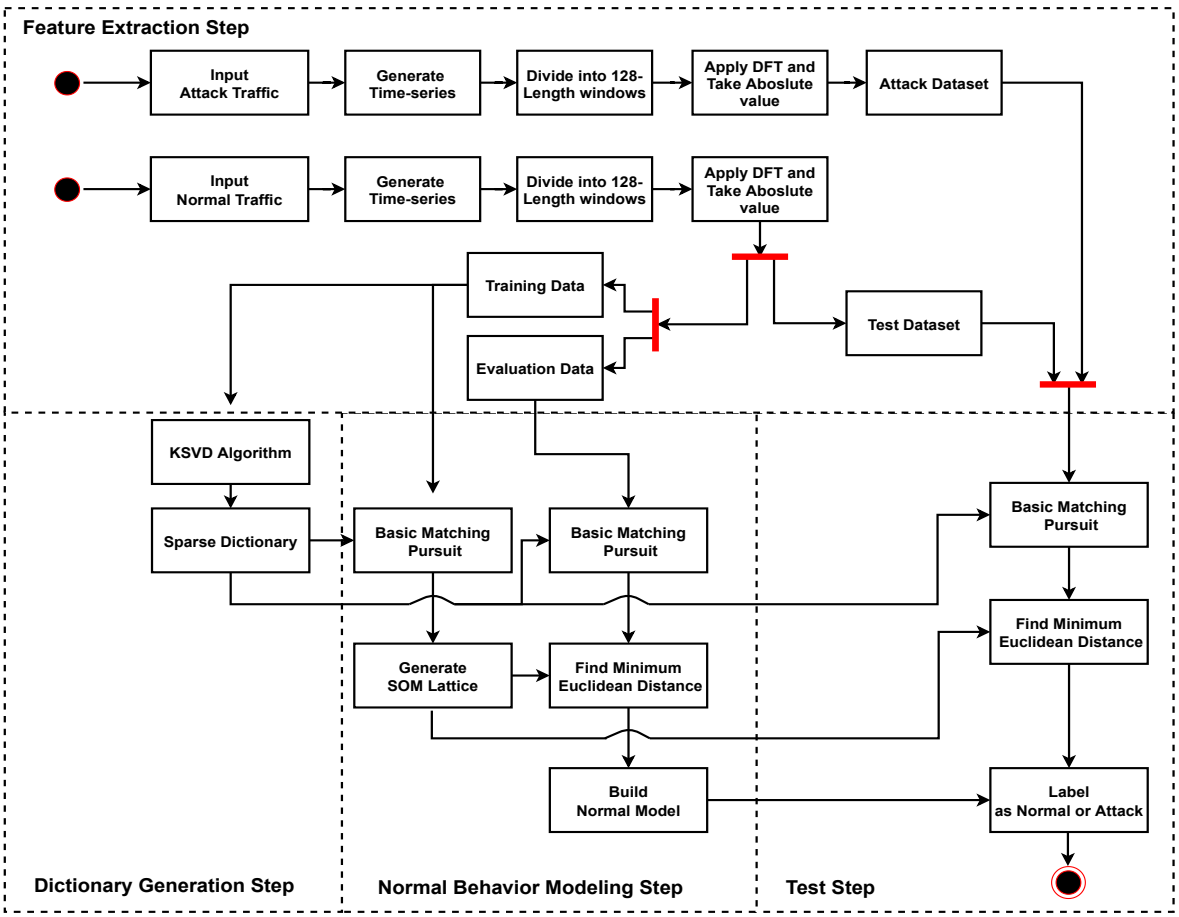


Figure 3.12. An overall view of the proposed scheme based on frequency-domain analysis and sparse coding.

3.3.1. The Proposed Model

In this study, we propose the improved version of DDoS detection approach discussed in Section 3.1 [45] by employing sparse coding to extract the most discriminating

features from the frequency domain. Moreover, the SOM model is used for creating the normal behavior of the system. The overall view of the proposed model is as illustrated in Figure 3.12. The proposed method operates in four steps, namely the Feature Extraction, Dictionary Generation, Normal Behavior Modeling and Test steps, which are defined as follows:

(i) Feature Extraction: DDoS and normal traffic samples from the input dataset are generated in this step. First, time windows are obtained by dividing the dataset into 128 ms windows. Then, for each window, absolute value of DFT is estimated and is labeled as DDoS sample or normal sample. Later, normal samples are divided as the training, the evaluation and the test parts in order to be used in the following modules.

(ii) Dictionary Generation: We employ K-SVD algorithm to generate the sparse dictionary of the normal samples. Dictionary is generated by applying K-SVD algorithm on the training part of normal samples that are already generated in the feature extraction module.

(iii) Normal Behavior Modeling: BMP algorithm is applied on both training and evaluation parts of normal samples to estimate the sparse coefficients. Then, SOM model based on coefficients of the training part is generated and the empirical distribution of the minimum Euclidean distance between neurons of SOM and coefficients of the evaluation part is used as the normal behavior of the system.

(iv) Test: BMP with the same dictionary of the previous part is executed on the test part of attack and normal traffics. In order to assign attack or normal label for each test sample, the minimum Euclidean distance between corresponding coefficients and the SOM lattice is calculated and compared with the normal behavior model.

3.3.1.1. Feature Extraction Step. The Feature Extraction step of the proposed model operates as follows. First, the number of all arriving packets are counted for each time interval, $t = 1$ ms, in order to generate time-series $\mathcal{Z}$. Later, the time-series, $\mathcal{Z}$, is divided into $w = 128$-length windows with $d = 64$-length overlapped values with

consecutive windows as shown in Figure 3.13. The output of DFT is a complex signal with the same length as input windows. Then, we use the absolute value of DFT as a feature. Since absolute value of DFT for a real signal is symmetric, we only consider one half of it, and therefore, for each window, we obtain 64 samples. Output samples of the feature extraction step are labeled as DDoS and normal traffic samples. The normal traffic samples are divided into three categories as training, evaluation and test samples to be used in the subsequent steps of the proposed model.

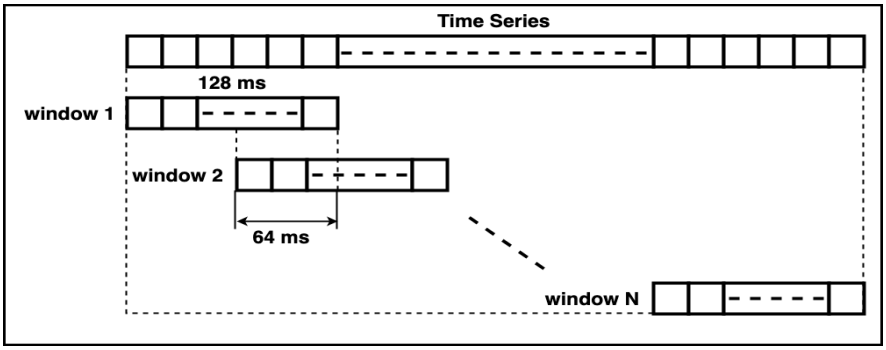


Figure 3.13. Process of dividing time-series into time windows.

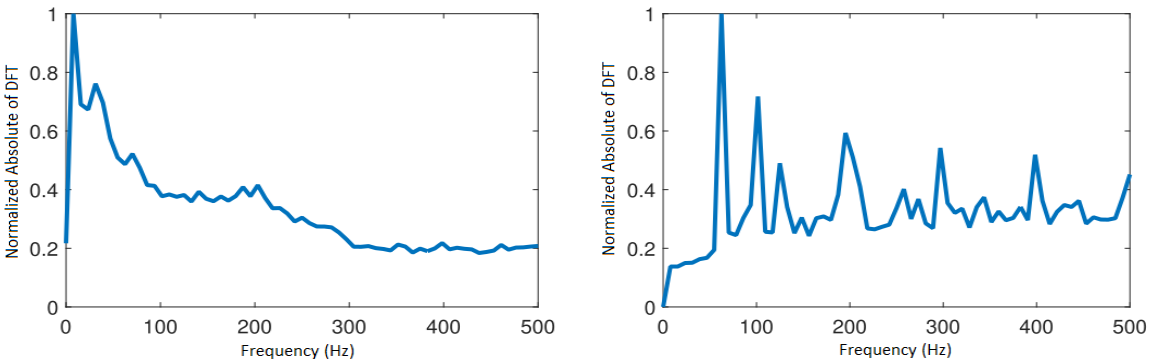
3.3.1.2. Dictionary Generation Step. The Dictionary Generation step operates on the training samples, which are obtained in the previous step. We employ K-SVD algorithm [144] on training samples to generate an over-complete dictionary D. This dictionary is used to estimate the sparse coefficients of the samples. In sparse coding, an observation $f_{t_{n\times 1}}$ at time interval t, can be modeled by $f_t \approx D\check{f}_t$ where $D_{n\times m}$ and $\check{f}_{t_{m\times 1}}$ are dictionary matrix, and $K-$sparse vector respectively and $n < m$. By using $K-$ sparse vector, we guarantee that only $K \ll n$ elements in $\check{f}_t$ become non-zero. The idea behind the sparse coding is to reconstruct f_t by using linear combination of K columns (or atoms) out of D. Then, $\check{f}_t$ can be approximated by solving the following optimization problem:

$$\hat{\check{f}}_t = \underset{\check{f}_t}{\operatorname{argmin}} \|\check{f}_t\|_0 \quad s.t. \quad f = D\check{f}_t, \tag{3.1}$$

where $\|\check{f}_t\|_0$ is the l_0 pseudo-norm that counts the number of non-zero elements in $\check{f}_t$ [144].

3.3.1.3. Normal Behavior Modeling Step. To generate the normal behavior model, first we obtain corresponding K number of coefficients by applying BMP algorithm on the training part of normal data using dictionary D. Then, SOM algorithm is applied on K coefficients to generate an SOM model. Finally, the evaluation part of normal traffic is processed in the BMP algorithm with the same D dictionary and the corresponding K coefficients are estimated. The set of the minimum Euclidean distance between evaluation part coefficients and the SOM model is used as the normal behavior of the system.

3.3.1.4. Test Step. The performance of the proposed model is analyzed in this step. Since the dictionary D is generated by using the normal traffic data, we expect to have different sparse coefficients when applying BMP algorithm on the attack samples using this dictionary. Both test part of normal traffic and DDoS attack are fed to BMP algorithm with D as the sparse dictionary. The minimum Euclidean distance between the coefficients of each sample and the SOM lattice is calculated and compared with the normal behavior model.



(a) Normal. (b) Attack.

Figure 3.14. Normalized absolute value of DFT, (a) Normal, (b) DDoS attack.

3.3.2. Discussions and Preliminary Results for CAIDA Dataset

In this section, we give the discussions and preliminary results based on the application of the proposed model on CAIDA dataset with respect to sparse coding, normal behavior modeling based on sparse coefficients and SOM. The number of packets

in each time intervals is the network feature which we employ in this work. The number of packets is counted per each 1 ms; therefore, according to the Nyquist theorem [78], the maximum frequency of received signal which can be realized for this sample rate, is 500 Hz. By 128 ms window, we achieve 7.8 Hz frequency resolution, distributed in 64 bins. Figure 3.14 displays an example of the normalized absolute value of DFT of number of packets for both normal and DDoS traffic of CAIDA dataset. Compared to the attack traffic in which has different dominant frequency bands, normal traffic tends to be a slow traffic. Because, both the attack traffic and the normal traffic share different number of periodic components, DFT feature would not be a good choice for the DDoS attack detection by itself and therefore, we use sparse coding for extracting discriminating features.

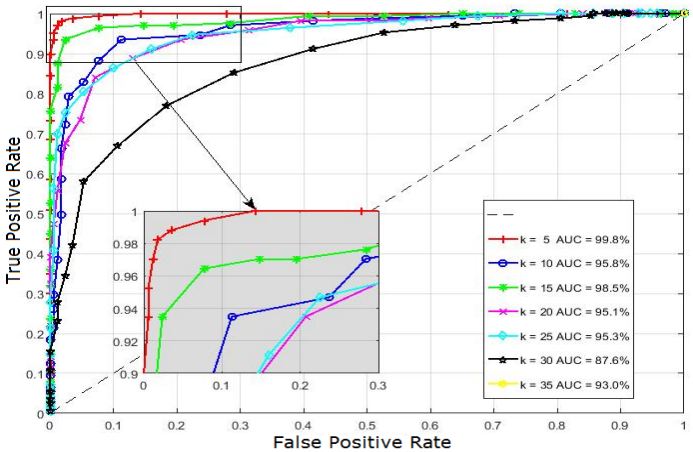


Figure 3.15. ROC for K={5, 10, 15, 20, 25, 30, 35} in sparse coding.

3.3.2.1. Sparse Coding by using K-SVD Algorithm for Dictionary Generation. As previously defined, an over-complete dictionary $D \in \mathbb{R}^{64\times256}$, is estimated by applying K-SVD algorithm on the training part of the normal traffic data. D is initialized by using Gaussian random matrix and then it is processed by K-SVD algorithm. This dictionary is used by matching pursuit method to find the $K = 5$ coefficients of sparse coding for absolute value in DFT of the number of packets. K and the size of the dictionary D are selected empirically and by applying embedded feature selection method [145]. Figure 3.15 displays ROC curves for different values of K. Considering this AUC metric, the

model with $K = 5$ outperforms compared to other values of K. By fixing K as 5, the performance of the model is analyzed by considering three different sizes of dictionary. Figure 3.16 shows the ROC results for different dictionary sizes. Dictionary with the size of 256 outperforms others with respect to AUC.

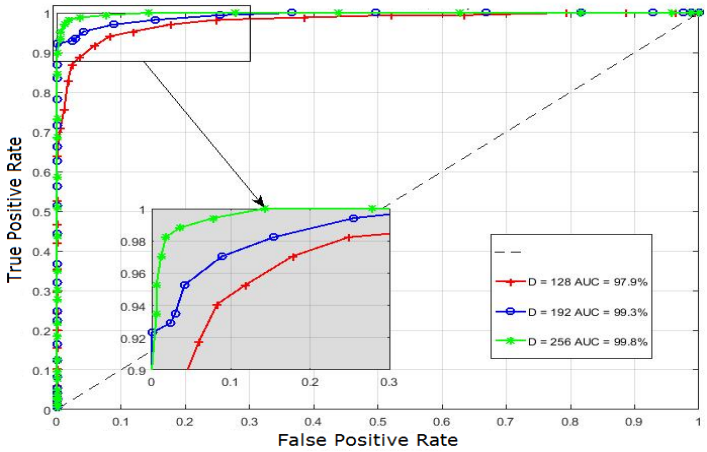
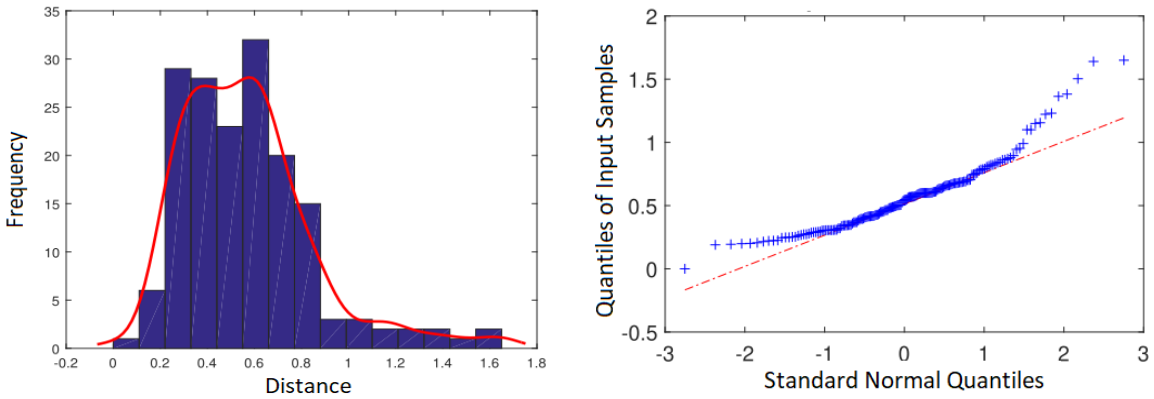


Figure 3.16. ROC for different dictionary with size of{128, 192, 256} and $K = 5$.

3.3.2.2. Normal Modeling Based on Sparse Coding Coefficients. In an anomaly detection approach, the model for the normal activity is obtained and used as the metric for spotting malicious activities. Any digression from normal behavior is considered as abnormality. One way to find the normal behavior of the network traffic is to utilize a learning algorithm. For this respect, we propose to use SOM algorithm as a learning algorithm in this work. Before modeling the normal behavior, the input data and the coefficients set obtained from basic matching pursuit are normalized with respect to the z-score of each observation. In order to create an SOM lattice, the size of the map should be specified. The number of neurons is determined by the number of observations in the training dataset using $Q \simeq 5 \times \sqrt{(O)}$ where Q and O are the number of neurons and the number of observations, respectively [146]. From this observation, the number of neurons is selected to be 144.

Later, the SOM model is trained by the coefficients of normal training dataset. The number of iterations is assigned approximately as 500 times of the product of

lattice dimension which is 72000 [38]. In the next step, the minimum Euclidean distance between each coefficient of evaluation part of normal data and the neurons of the SOM are calculated and empirical probability distribution of those distances is utilized as the normal behavior of the system. Figure 3.17 represents the empirical probability density and normal Q-Q plot of the minimum Euclidean distance of evaluation dataset from the neurons of the SOM model. From the Figure 3.17, we can infer that the model is heavy-tailed and right skewed.



(a) Empirical pdf. (b) Q-Q Plot.

Figure 3.17. (a) Empirical pdf and (b) Q-Q plot of minimum distance from neurons of SOM model.

Table 3.5. Statistical parameters of the empirical pdf of minimum distance from neurons of SOM model.

No	Statistics	Value
1	μ	0.5646
2	σ	0.2697
3	$\mu + 2 \times \sigma$	1.1264
4	$\mu + 3 \times \sigma$	1.4073
5	Kurtosis	5.54
6	Skewness	1.33
7	Shapiro-Wilk P-Value	4.8E-4

Table 3.5 summarizes some statistical parameters of the empirical distribution. The Shapiro-Wilk (SW) test($P - value < 0.05$) [147] and a visual inspection of the

empirical Probability Density Function (PDF) [78] and normal Q-Q plot (Figure 3.17), show that minimum distances are not normally distributed. Moreover, kurtosis ($5.54 > 3$) and skewness ($1.33 > 0$) indicate that distribution is fat-tailed (leptokurtic) [35] and skewed to the right, respectively. In the normal distribution extreme events are less likely than of fat-tailed one. This property of fat-tailed distribution should be taken into account during the threshold value estimation. Underestimating this parameter would increase FPr and decreases the DDoS detection performance.

3.3.3. DDoS Detection Performance

In order to test the DDoS detection performance of the proposed model, we use CAIDA intrusion detection evaluation dataset [140]. Normal dataset is divided into three subsets as training 796 (70%), evaluation 169 (15%) and test 169 (15%). 169 TCP-based DDoS attack samples are used in the test step. To analyze the performance of the proposed model, we use TP and FP.

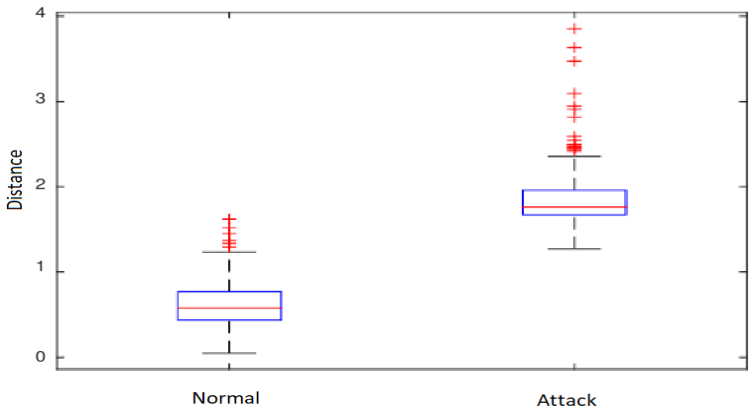


Figure 3.18. Boxplot chart of minimum distance from neurons of SOM.

The test dataset consists of normal and attack traffics, where each traffic has 169 samples. Five non-zero coefficients set of each test sample is estimated by using the dictionary D and BMP algorithm. In the next step, the minimum Euclidean distance of each coefficient set from neurons of SOM model is found. Figure 3.18 displays the boxplot chart of the corresponding distances set. There are some points which reside outside the fence of the boxplot chart of the normal distances. These data points are located inside the boundary of attack distances which yields false alarms during the

performance test. These distances are compared with the normal model which is the empirical distribution of distance of the normal evaluation part from the SOM model.

Table 3.6 summarizes the confusion table for different threshold values based on the standard deviation of the normal model. As the threshold value increases, the FP value decreases but simultaneously, the detection performance deteriorates (TP). Selecting the two standard deviations of the mean value as the threshold to separate normal and attack data results in the best detection rate (TP); on the other hand, FP has the worst value. Although the 5 standard deviation of the mean value has the best FP, the detection rate of the model decreases significantly. Selecting the 3 standard deviation of the mean value as the threshold, gives the best result which is the trade off between TP and FP. The proposed algorithm achieves 1.2% and 99.4% of FPr and TPr, respectively.

Table 3.6. Confusion table for different threshold values obtained from empirical pdf of minimum distance from neurons of SOM model.

Threshold		**Normal**	**Attack**	TPr(%)	FPr(%)	ACC (%)	F1_score
$2 \times \sigma$ (1.1264)	**Normal**	156	13	100	7.69	96.15	0.96
	Attack	0	169				
$3 \times \sigma$ (1.4073)	**Normal**	167	2	99.41	1.18	99.11	0.99
	Attack	1	168				
$4 \times \sigma$ (1.6434)	**Normal**	168	1	68.04	0.59	83.73	0.81
	Attack	54	115				
$5 \times \sigma$ (1.9131)	**Normal**	169	0	23.67	0	61.83	0.38
	Attack	129	40				

Table 3.7. Comparison between accuracy of proposed schemes for traditional networks.

Proposed Scheme	**Accuracy(%)**	F1_score
Scheme 3 described in Section 3.3 [47]	99.11	0.99
Scheme 2 described in Section 3.2 [46]	98.33	0.98
Scheme 1 described in Section 3.1 [45]	93.27	0.92

3.3.3.1. Performance Comparison of DDoS Detection of Scheme 1, 2 and 3. The performance of the proposed scheme in terms of detection accuracy and computational cost complexity is compared with previous works in this chapter. Table 3.7, compares the accuracy of the proposed methods for traditional networks, namely Scheme 1 [45], Scheme 2 [46] and Scheme 3 [47]. Because the dataset used in [45] is different, we re-simulate the method. According to the Table 3.7, Scheme 3 has the best detection accuracy. Therefore, we can conclude that using the features from frequency domain analysis and applying sparse coding as the feature extraction approach have a positive effect on the DDoS detection performance. Regarding to the computational complexity cost we use the following assumptions:

(i) $C = 2$, where C is the number of classes which can be Normal or DDoS attack.
(ii) $m >> w$, where m and w are the number of columns in the sparse dictionary and the size of window, respectively.
(iii) $m^2 > Q$, where Q is the number of neurons of SOM lattice.
(iv) DFT is based on Fast Fourier Transform (FFT) [148].
(v) Passive monitoring is employed.

Scheme 1 is based on DFT, DWT and naive Bayes classifier. DFT and DWT has the computational complexity cost of $O(wlogw)$ and $O(w)$, respectively. The computational complexity cost of naive Bayes classifier depends on the number of classes and the number of features, which is $O(wC) \approx O(w)$. Therefore, the overall computational complexity cost of Scheme 1 is $O(wlogw + w) \approx O(wlogw)$.

Scheme 2 obtains four statistical measures including kurtosis, skewness, the number of periodic components and Hurst exponent. The calculation of kurtosis and skewness has the computational complexity cost of $O(\frac{w}{2})$. Finding the number of periodic components and Hurst exponents have $O(wlogw)$ and $O(w^2)$ computational complexity cost, respectively. Finally the overall computational complexity cost of Scheme 2 is $O(\frac{w}{2} + wlogw + w^2) \approx O(w^2)$.

Scheme 3 is based on DFT, BMP and finding the minimum distance between neurons of SOM and features. DFT has the computational complexity cost of $O(wlogw)$. BMP depends on matrix (dictionary) and vector multiplication; therefore, the computational complexity cost of BMP is $O(m^2)$. The computational complexity cost of finding minimum distance of neurons of SOM and the feature vector is $O(Q)$. The overall computational complexity cost of Scheme 3 is $O(wlogw + Q + m^2) \approx O(m^2)$.

Comparing the computational complexity cost of three schemes, Scheme 1 has the best computational complexity cost which is followed by Scheme 2 and Scheme 3.

The three proposed schemes can be employed by IDS to find anomalous activities caused by DDoS attacks. Both host-based and network-based IDS can benefit from these schemes.

4. A DDOS ATTACK DETECTION AND DEFENSE SCHEME USING TIME-SERIES ANALYSIS FOR SDN

In this chapter, we propose a DDoS attack detection and defense scheme (Scheme A) based on statistics and time-series analysis. The contributions of this chapter are as follows:

(i) First, we extract key features from the flow table of OpenFlow (OF) switches to analyze their time-series representation. Therefore, the controller monitors each switch separately and applies the proposed algorithms to detect anomalies due to the DDoS attack event in particular switches. Such anomaly can be detected using the Unique Source IP addresses (USIP) feature, since during the attack phase this value increases significantly compared to the normal traffic.

(ii) In addition, when an attack occurs, it is possible to observe changes in the number of Unique Destination IP addresses (UDIP) feature; however, this change is not as significant as the change in the USIP feature. By considering the dramatic growth in Total Number of Packets (totalPacket) in the flow table during the attack, the number of UDIP value substantially reduces when it is normalized by the totalPacket value. Therefore, we employ NUDIP as the second feature to detect DDoS attacks.

(iii) The upcoming value of the USIP time-series is estimated by using an ARIMA model and the error of estimation is examined for chaotic behavior. Finally, according to the aforementioned method, a binary anomaly score is assigned to the traffic instance.

(iv) Another binary anomaly score is obtained using the dynamic threshold method based on the exponential filter and the NUDIP time-series. The product of two mentioned binary scores is used to identify DDoS attack in each switch. After detecting the attack samples, the scheme activates the countermeasure mechanism.

(v) The chaotic behavior and the dynamic threshold method resolve the problem of

constant threshold in the previous works. Moreover, by monitoring each switch, the source of the attack can be identified and prompt countermeasure would be applied by modifying the packet forwarding policy at that switch. In order to evaluate system performance in successful detecting of DDoS, we implement our scheme on an example SDN network via using Mininet environment [44].

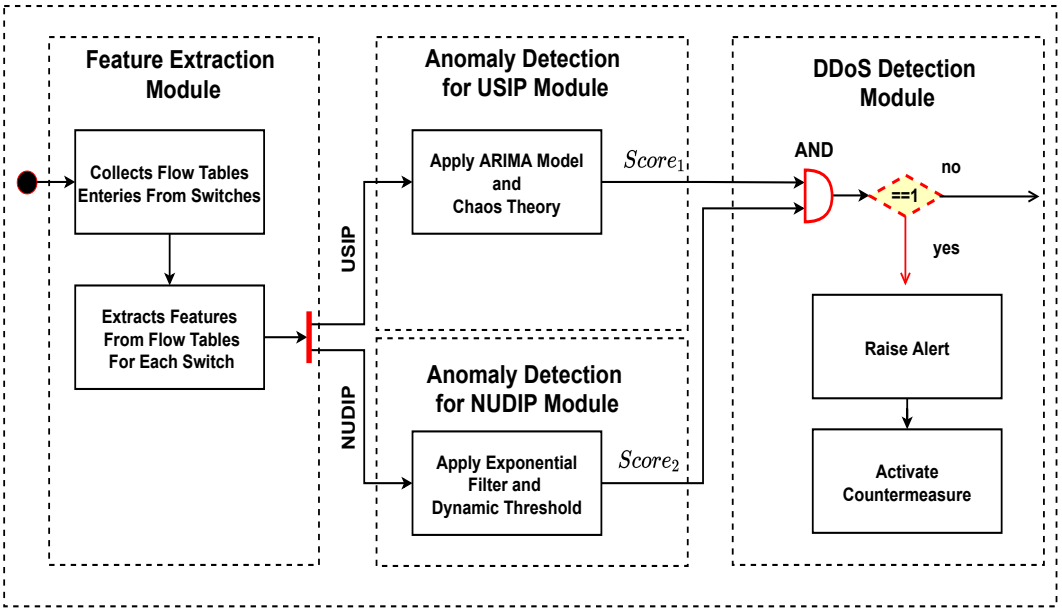


Figure 4.1. Proposed DDoS detection and defense scheme (Scheme A) for SDN.

4.1. DDoS Attack Detection and Defense

In this section, we introduce the proposed DDoS attack detection and defense scheme for SDN. The overall view of the scheme is as illustrated in Figure 4.1. The proposed scheme consists of four main modules, namely *Feature Extraction*, *Anomaly Detection for USIP*, *Anomaly Detection for NUDIP* and *DDoS Detection*. The proposed detection algorithm is illustrated in Figures 4.2 to 4.5. The controller uses the proposed scheme and monitors each switch separately to detect any anomalies related to DDoS attack traffic. First, *Feature Extraction* module extracts corresponding statistical features including USIP and NUDIP from the flow tables of the OF switch. While the USIP feature is used by the *Anomaly Detection for USIP* module to assign an anomaly score to the sample, the NUDIP feature is employed by *Anomaly Detection for NUDIP* module to release another anomaly score. The *Detection* module

decides whether there is an anomaly or not with respect to anomaly scores. In the case of any abnormal conditions, this module raises anomaly alarm and activates the countermeasure part. The symbols used in this work are listed in Table 4.1.

Table 4.1. Definitions of symbols used in Chapter 4.

Symbol	**Definition**	**Symbol**	**Definition**
H_S, H_D	Hash tables for extracting USIP and UDIP	*counter*	The hash table counter variable
r_t	Number of flow entries in the flow table at time interval t	k	Number of samples for training the normal models
$Model\_X$, $Model\_Y$	Training flags for USIP and NUDIP	$\mathcal{X} = \{x_1, \ldots, x_N\}$	Time-series of the number of unique source IP addresses
$\mathcal{Y} = \{y_1, \ldots, y_N\}$	Time-series of the number of normalized unique destination IP addresses	$\langle x_t, y_t \rangle$	USIP and UDIP instances at time interval t
$\langle a, \ell, q \rangle$	The order of ARIMA, Number of AR ,Differencing and MA terms	e_t	The prediction error of ARIMA at time interval t
Λ_t	Lyapunov exponent at time interval t	$0 \leq \alpha \leq 1$	Smoothing constant of the exponential filter
$f(\alpha)$	Function of α	$EXP\_f1(\alpha_1)$, $EXP\_f2(\alpha_2)$	Exponential filters, α_1 close to 1, α_2 much less than 1
$\mathcal{D}_f$	Time-series of the distance between the outputs of two exponential filters	$\mathcal{M}$	Time-series of the Rolling median for $\mathcal{D}_f$
w	Window size of the rolling median	min_dist	Minimum distance between each element and other elements in $\mathcal{M}$
μ_m	Mean value of min_dist	σ_m	Standard deviation of min_dist
ρ	The number of standard deviation, σ_m to define the threshold	th_{med}	The threshold value for min_dist
$score_{1,t}$, $score_{2,t}$	Anomaly score for x_t and y_t	$DestIP_{max}$	The destination IP address with the maximum occurrence

1:
2: $H_D(Des_{IP}, counter) \leftarrow 0$
3: $H_S(Src_{IP}, counter) \leftarrow 0$
4: **for** $\forall(Src_{IP}, Des_{IP}) \in Flow\_T$ **do**
5: **if** $Src_{IP} \notin H_S$ **then**
6: $H_S(Src_{IP}, counter) \leftarrow 1$
7: **else**
8: $H_S(Src_{IP}, counter) \leftarrow counter + 1$
9: **end if**
10: **if** $Des_{IP} \notin H_D$ **then**
11: $H_D(Des_{IP}, counter) \leftarrow 1$
12: **else**
13: $H_D(Des_{IP}, counter) \leftarrow counter + 1$
14: **end if**
15: **end for**
16: $x_t \leftarrow$ Number of non-empty elements in H_S /*USIP sample*/
17: $y_t \leftarrow$ Number of non-empty elements in H_D /*UDIP sample*/
18: z_t /*# of packets */
19: $y_t \leftarrow y_t/z_t$ /* NUDIP sample*/
20: **return** $H_D(Des_{IP}, occurrence),\ x_t,\ y_t$
21:

Figure 4.2. Algorithm 4.1, extract statistic features (USIP and NUDIP).

4.1.1. Feature Extraction

Due to the spoofed IP addresses and random generation of source IP addresses, the number of flow entries with the unique source IP addresses increase during the DDoS attack. Although the number of unique destination IP addresses may not change significantly during the attack compared to the normal condition, the normalized value of this variable with respect to the total number of packets in the flow table decreases [149]. Therefore, in this work, we consider the number of unique source

IP addresses and the normalized unique destination IP addresses as the key features for DDoS detection. In the *Feature Extraction* module, for each specific time interval t, we obtain statistics from the flow tables of each switch to extract aforementioned features. The value of time interval t may be chosen according to the corresponding network infrastructure and the traffic volume in the network. As shown in Algorithm 4.1 illustrated in Figure 4.2, two hash tables [150] namely H_S and H_D are used to find USIP and UDIP, respectively. If a source IP address is not listed in H_S, it is added to the table and the corresponding counter is set to 1. On the other hand, if the source IP address already exists in the table, the counter is incremented by one. The same procedure is applied for the destination IP addresses and the corresponding hash table H_D. Once all entries of the flow table are processed, the USIP feature of the sample, x_t, is obtained by counting the total number of non-empty elements of the H_S; also, the total number of non-empty elements of H_D are counted as UDIP and then it is divided by the total number of packets in the flow table to obtain normalized UDIP, denoted as y_t. These two features are treated as time-series and employed in DDoS detection process independently to inspect the possible instances of DDoS attack.

4.1.2. Anomaly Detection for USIP

In this module, by applying ARIMA and chaos theory on the x_t feature, a binary anomaly score ($score_{1,t}$) is obtained for each time interval t. ARIMA $\langle a, \ell, q \rangle$ model is employed to predict x_t as $\hat{x_t}$ [151]. In order to estimate the value of x_t, the ARIMA model is generated in advance. k samples of the USIP are stored in $\mathcal{X}$ to train the model. If $\mathcal{X}$ is a non-stationary time-series, differencing with $\ell \geq 1$ is applied. If necessary, to stabilize the variance of $\mathcal{X}$, Box-Cox transformation [152] can be used. Akaike's Information Criterion (AIC), Corrected AIC (AICc) and Bayesian Information Criterion (BIC) may be used for choosing the order of the model. Minimizing such criteria results in obtaining optimal model [153]. We use ARIMA model to generate the normal pattern of USIP; therefore, during the model generation $\mathcal{X}$ should not contain any attack instances [154].

1:
2: $Model\_X \leftarrow True$
3: $x\_count \leftarrow 0$
4: $\mathcal{X} \leftarrow []$
5: **for** each time interval t **do**
6: **if** $Model\_X$ **then**
7: $\mathcal{X}||x_t$, where || stands for concatenation
8: $x\_count++$
9: **if** $x\_count > k$ **then**
10: Estimate $ARIMA_{\langle a,\ell,q\rangle}$ using $\mathcal{X}$
11: $Model\_X \leftarrow False$
12: **end if**
13: **else**
14: $\hat{x_t} \leftarrow ARIMA_{\langle a,\ell,q\rangle}(\{x_{t-a-\ell}, \ldots, x_{t-1}\})$
15: $e_t \leftarrow |x_t - \hat{x}_t|$ /*Prediction error*/
16: $\Lambda_t \leftarrow \frac{1}{t} ln(|\frac{e_t}{e_0}|)$ /*Lyapunov Exponent*/
17: **if** $\Lambda_t < 0$ **then**
18: e_t is normal /*Normal traffic*/
19: $score_{1,t} \leftarrow 0$
20: $e_0 \leftarrow e_t$ /*Update e_0*/
21: **else**
22: e_t is chaotic /*DDoS candidate*/
23: $score_{1,t} \leftarrow 1$
24: **end if**
25: **end if**
26: **end for**
27:

Figure 4.3. Algorithm 4.2, anomaly score computation for Unique Source IP Addresses.

Once the model is generated, the training phase is completed. For each upcoming traffic sample, x_t, the estimated value, $\hat{x_t}$, is obtained by using the normal model. Then, an anomaly score is given by analyzing the chaotic behavior of the estimation error. The absolute value of the prediction error, e_t is calculated by:

$$e_t = |x_t - \hat{x_t}|. \tag{4.1}$$

In some time intervals, the prediction errors differ. Such heterogeneous outcomes might be raised by the anomaly of the USIP instance, x_t. Thus, in order to assign anomaly score, Lyapunov exponent, Λ_t of the error, e_t, is calculated and the positive value of the Λ_t may correspond to anomaly. If the Λ_t is negative, the denominator error term in Lyapunov equation (Equation 2.8), e_0, is replaced by e_t and the instance is labeled as normal ($score_{1,t} = 0$); otherwise, it is labeled as anomaly ($score_{1,t} = 1$). However, due to the similarity of USIP pattern, some normal traffic samples (i.e., flash crowd events) may be classified as attack candidates that also yields high false positive rate. Therefore, as introduced in the next section, we use NUDIP feature to increase the accuracy. Algorithm 4.2 illustrated in Figure 4.3, describes the procedure carried out in this module.

4.1.3. Anomaly Detection for NUDIP

Another anomaly score ($score_{2,t}$) is assigned for each sample by applying the adaptive thresholding method. The method uses the y_t feature of the sample. In order to generate the model, k samples of NUDIP are stored in $\mathcal{Y}$ as a time-series. $\mathcal{Y}$ is processed by two exponential filters namely EXP_f1 and EXP_f2. The corresponding smoothing constant of EXP_f1 filter, α_1, is chosen as close as possible to 1; hence, the output of this filter follows the trend in the $\mathcal{Y}$. On the other hand, EXP_f2 filter has α_2 much less than 1; thus, the output of the filter follows the change in $\mathcal{Y}$. The absolute difference between outputs of these two filters, $\mathcal{D}_f$ is used for extracting the discriminative feature. The difference is independent from the trend change; therefore, it almost fluctuates around zero, which reflects the dynamic change of the $\mathcal{Y}$.

1:
2: $Model\_Y \leftarrow True$
3: $y\_count \leftarrow 0$
4: $\mathcal{Y} \leftarrow []$
5: **for** each time interval t **do**
6: **if** $Model\_Y$ **then**
7: $\mathcal{Y}||y_t$
8: $y\_count++$
9: **if** $y\_count > k$ **then**
10: $\mathcal{D}_f \leftarrow [0]$
11: $\mathcal{M} \leftarrow []$
12: $min\_dist \leftarrow []$
13: $l_1 \leftarrow \mathcal{Y}[1]$
14: $h_1 \leftarrow \mathcal{Y}[1]$
15: **for** $\forall y_i \in \mathcal{Y}[2:]$ **do**
16: $l_i \leftarrow \alpha_1 l_{i-1} + (1-\alpha_1)y_i$
17: $h_i \leftarrow \alpha_2 h_{i-1} + (1-\alpha_2)y_i$
18: $\mathcal{D}_f|||h_i - l_i|$
19: **end for**
20: **for** $\forall i \in 1, 2, \ldots, k-w$ **do**
21: $\mathcal{M}||Med(D_{f_i}, \ldots, D_{f_{i+w}})$
22: **end for**
23: **for** $\forall m_i \in \mathcal{M}$ **do**
24: $min\_dist||min(|m_i - m_j|$ /* *where* $j \in 1, \ldots, k-w$) and $j \neq i$*/
25: **end for**
26:
27:
28: $\mu_m \leftarrow Mean(min\_dist)$
29: $\sigma_m \leftarrow StdDev(min\_dist)$
30: $th_{med} \leftarrow \mu_m + \rho \times \sigma_m$
31: $Model\_Y \leftarrow False$
32: **end if**
33: **else**
34: $l_t \leftarrow \alpha_1 l_{t-1} + (1-\alpha_1)y_t$
35: $h_t \leftarrow \alpha_2 h_{t-1} + (1-\alpha_2)y_t$
36: $D_{f_t} \leftarrow |h_t - l_t|$
37: $m_t \leftarrow Med(\{D_{f_{(t-w)}}, \ldots, D_{f_t}\})$
38: $dist_t \leftarrow min(|m_t - m_i|)$, for all $m_j \in \mathcal{M}$
39: **if** $dist_t < th_{med}$ **then**
40: $score_{2,t} \leftarrow 0$ /*Normal*/ /*Discard the first element of $\mathcal{M}$ and min_dist, then add the corresponding new values*/
41: $\mathcal{M} \leftarrow \mathcal{M}_2^{k-w}||m_t$
42: $min\_dist \leftarrow min\_dist_2^{k-w}||dist_t$
43: $\mu_m \leftarrow Mean(min\_dist)$
44: $\sigma_m \leftarrow StdDev(min\_dist)$
45: **else**
46: $score_{2,t} \leftarrow 1$ /*Attack*/
47: **end if**
48: **end if**
49: **end for**
50:

Figure 4.4. Algorithm 4.3, anomaly score computation for Normalized Unique Destination IP Addresses.

By applying rolling median with the window size of w on $\mathcal{D}_f$, a median time-series, $\mathcal{M}$, is generated. The minimum distance of each instance in $\mathcal{M}$ from the rest of samples in $\mathcal{M}$ are calculated and recorded in a new set, denoted as min_dist. The mean value, μ_m, and standard deviation, σ_m, of min_dist are calculated. Once the aforementioned parameters are calculated, the Model_Y is set to 'False' and anomaly scoring part is activated. For each y_t feature of upcoming traffic sample, using EXP_f1 and EXP_f2, the corresponding absolute difference, D_{f_t} is calculated. The median of $\{D_{f_{t-w}}, \ldots, D_{f_t}\}$ is calculated and the distances between median and each element in $\mathcal{M}$ is calculated and the minimum distance is obtained. If the minimum distance is less than the threshold value $th_{med} = \mu_m + \rho \times \sigma_m$, the instance is labeled as normal ($score_{2,t} = 0$), otherwise, it is labeled as the abnormal point ($score_{2,t} = 1$). Moreover, if the sample is normal, the first elements of $\mathcal{M}$ and min_dist are discarded and then the new median and corresponding minimum distance are appended respectively. As the result, μ_m, σ_m and th_{med} are updated. Algorithm 4.3 shown in Figure 4.4, illustrates the procedure of this module.

```
1:
2: for each time interval t do
3:    if ¬Model_X & ¬Model_Y then
4:       if score_{1,t} × score_{2,t} == 1 then
5:          DDoS attack is detected
6:       else
7:          Normal traffic is detected
8:       end if
9:    end if
10: end for
11:
```

Figure 4.5. Algorithm 4.4, DDoS detection based on ARIMA and dynamic thresholding.

4.1.4. DDoS Detection, Alert and Countermeasure

After binary scores for USIP, $score_{1,t}$ and NUDIP, $score_{2,t}$ are computed, these values are used by the DDoS attack detection module. As described in Algorithm 4.4 shown in Figure 4.5, for each traffic sample, if $score_{1,t}\ AND\ score_{2,t}$ is equal to 0, then there is no anomaly; Otherwise, the sample is abnormal and there is a DDoS attack to the system. Then the controller raises an attack alarm. The countermeasure module uses the hash table H_D to identify the destination IP address with the maximum occurrence, so called $DestIP_{max}$. This IP and the corresponding switch ID are added to a discard list. Then controller updates the flow table(s) of the corresponding switch by setting the drop action for each flow entry with $DestIP_{max}$ as the destination IP address. Although the defense mechanism discards the flow from the DDoS attacker and alleviates the burden on both controller and the switch, it also filters out all packets to the victim's IP address routed through that switch. Therefore, the controller should update the action of the flow entries as soon as the attack is over.

Although the countermeasure module takes the discard action for particular flow rules during the attack, it does not delete those flows from the flow table. Therefore, the statistic features remain unchanged. Since the detection module continues to monitor the switch, if $score_{1,t}\ AND\ score_{2,t}$ returns from 1 to 0, the attack alarm is cleared. The $DestIP_{max}$ and the corresponding switch ID are removed from the discard list and the action section of the flow entries returns to the normal state. Furthermore, there would be always some normal samples that are labeled as attack incorrectly. Because the detection module continuously monitors both USIP and NUDIP features, the state of the network will return to its normal in a few intervals.

4.2. Experimental Results

In this section, we evaluate the performance of the proposed detection and countermeasure scheme via simulation. We present our experimental results with respect to detection and countermeasure performances.

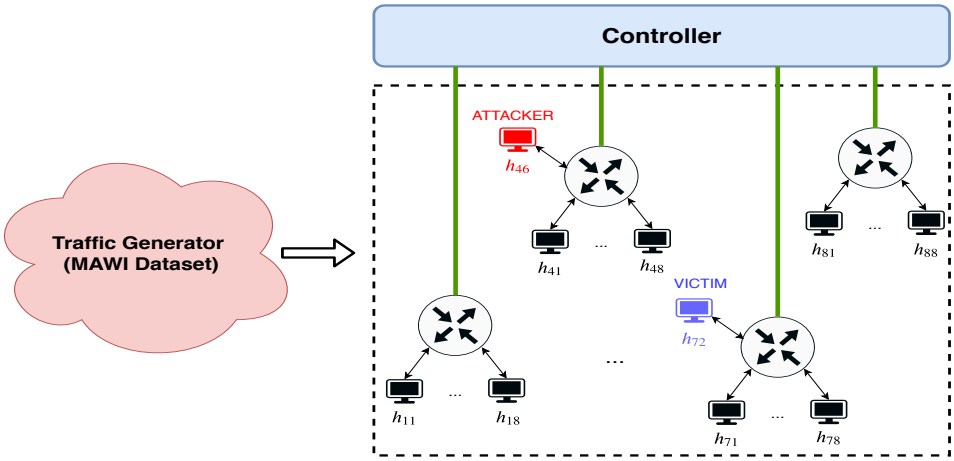


Figure 4.6. Experimental topology for Scheme A.

4.2.1. Dataset

In order to feed the network with real data, we use real network traffic dataset from MAWI Working Group Traffic Archive [155]. Since 2002, MAWILab has been collecting traffic measurement analysis on the Internet. We use this dataset to generate the normal traffic in our simulation network. We use the Jan 12, 2012 part of the MAWI dataset with the traffic data size of 1.1 GB. The MAWI traffic is regenerated using TcpReplay tool [156] and fed to the network. The IP addresses in MAWI dataset are also refashioned according to the host IPs in the experimental network. Therefore, each host in the network contributes to the traffic. In total, almost twelve hours normal and background traffic is regenerated by using TcpReplay. 255-minute (more than four hours) length attack traffic is injected into the normal traffic. The attack starts after 320 minutes that the normal traffic is started.

4.2.2. Simulation Environment

In order to simulate the network, Mininet emulation environment [44] is used to simulate the network. Due to its design and capabilities, it is appropriate for experimenting with SDN. POX controller [157] is run on a PC with 8 GB RAM and Intel Core i7 processor as the SDN controller. The network topology for the experiment is as shown in Figure 4.6. There are eight OF enabled switches, namely $S1, S2, \ldots, S8$,

in the network. Each switch Sk is connected to eight hosts, h_{kj}. We assume that, one of the host of $S4$, h_{46}, is infected by a malicious user who is generating attack on a victim which is the host, h_{72} connected to the switch, $S7$. We assume that the attacker uses different compromised hosts on the Internet and then tries to enter to the network through $S4$ to attack the victim [10–12].

4.2.3. DDoS Attack Detection Performance

In this section, we discuss the DDoS attack detection performance of the proposed scheme, by applying the method on the simulation environment. The controller organizes the network traffic passing through switches. At that point, the *Feature Extraction* module collects statistics from flow tables for each switch in every minute and extracts their features. Then, the USIP feature is processed by the first anomaly detection algorithm (Figure 4.3), which employs a time-series approach based on ARIMA and chaos theory to assign an anomaly score to the sample. On the other hand, the NUDIP feature is processed by another anomaly detection algorithm (Figure 4.4) which assigns another anomaly score to the same sample by applying the proposed adaptive thresholding method. Finally, our detection algorithm (Figure 4.5) decides whether the sample is an attack instance or a normal traffic instance. The time interval $t = 1$ minute is selected empirically and by considering the infrastructure of the experimental setup network.

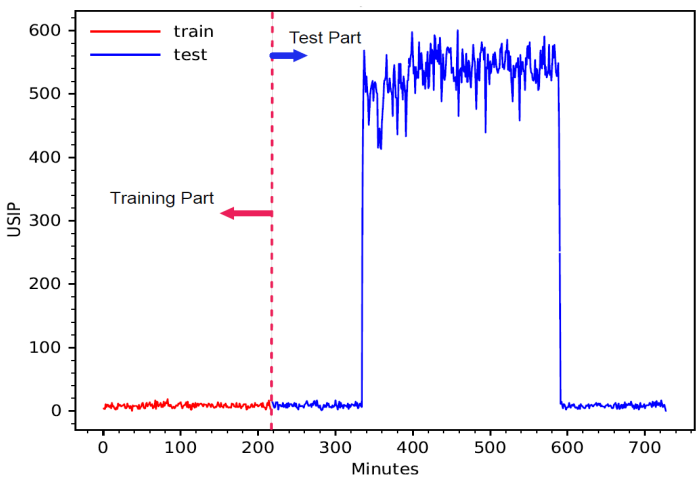


Figure 4.7. Number of unique source IPs.

Our scenario is illustrated in Figure 4.6. The attack is originated from the switch, $S4$; therefore, the following result are based on the analysis applied on the data extracted from $S4$. Totally, we obtain 729 traffic samples in our simulations. First 218 points are used in initializing the normal behavior of the network and the remaining 511 samples are left for test purposes. Training and test phases of USIP are shown in Figure 4.7. The trained model is employed to predict the upcoming value for x_t. Figure 4.8 displays the original and the predicted value for the USIP.

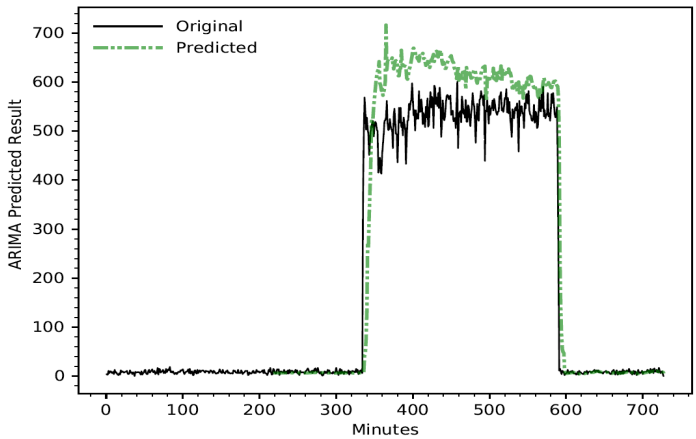


Figure 4.8. Original USIP and the predicted output by using ARIMA model.

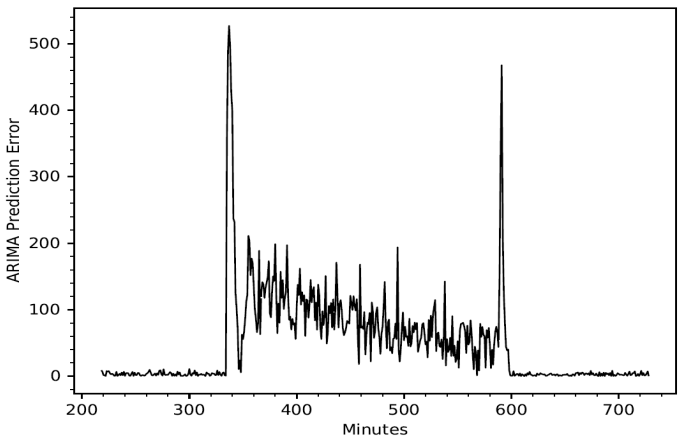


Figure 4.9. ARIMA prediction error.

Because the ARIMA model is trained based on the normal traffic data, the estimated value diverges from the actual value during the attack. Thus, the prediction error which is shown in Figure 4.9, is different from the normal condition. The chaotic behavior of the error is used to differ attack samples from normal traffic. Therefore,

the Lyapunov exponent of the error is estimated. The positive value of the Lyapunov value may correspond to attack samples. In order to convert the negative and positive Lyapunov values to scores, we map negative and positive values as 0 and 1, respectively. The Lyapunov values and the corresponding scores are depicted in the Figure 4.10. The actual DDoS attack instances are labeled by the red small triangles in the figure.

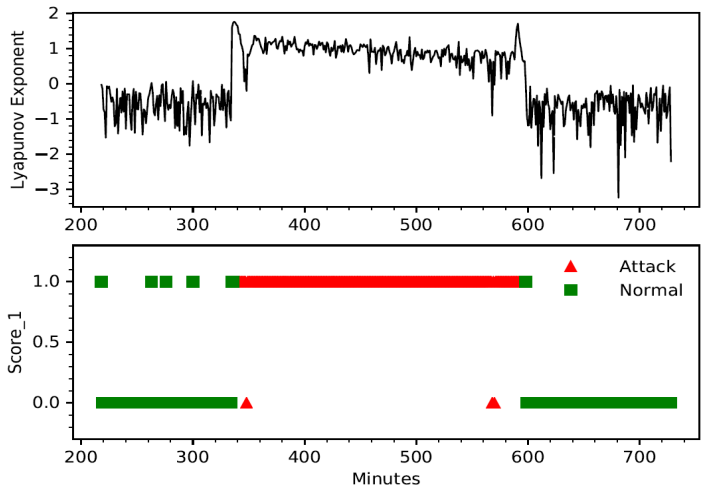


Figure 4.10. Local Lyapunov exponent for ARIMA prediction error and corresponding anomaly score.

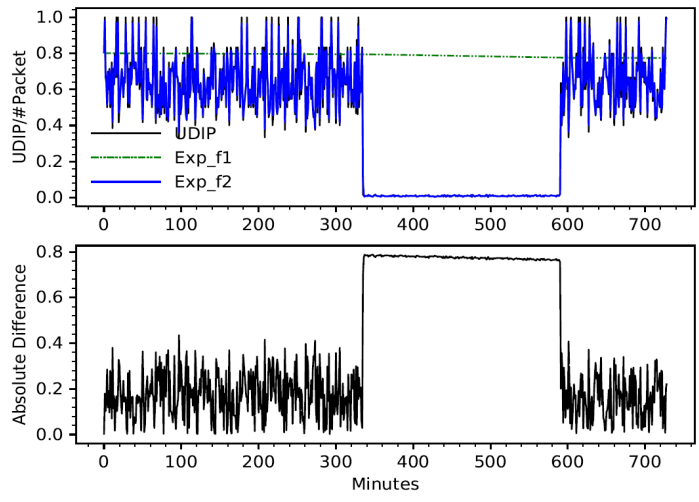


Figure 4.11. NUDIP time-series and the outputs of filters.

Figure 4.11 shows the NUDIP time-series, the output of two exponential filters, EXP_f1 and EXP_f2, and their difference. By applying rolling median with the window size of $w = 10$ on the difference, $\mathcal{M}$ is generated and used as the new feature to

distinguish between normal and attack traffic. The threshold value, $4 \times \sigma_m$, is empirically selected from the data and by considering Chebyshev's theorem which describes the minimum proportion of the measurements that must lie within one, two, or more standard deviations of the mean [158]. Figure 4.12 displays the median of difference and the corresponding score for the test part of the NUDIP. During the attack, the median is higher than the normal one. The detection algorithm decides whether the sample is anomaly or not. The final score of the detection algorithm is shown in Figure 4.13.

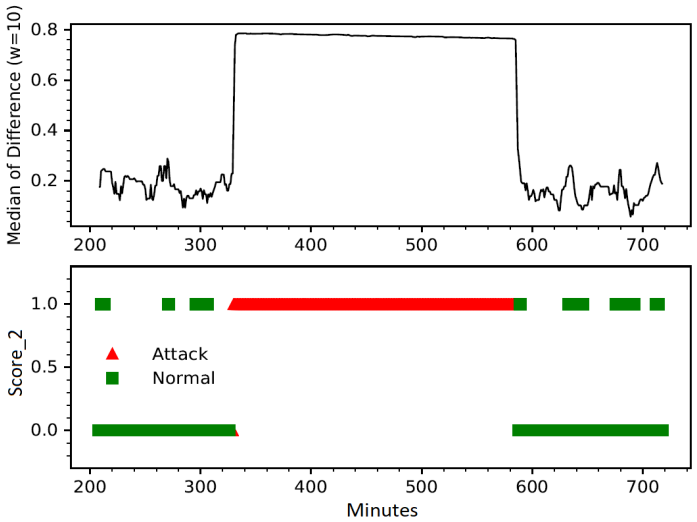


Figure 4.12. Median of difference and the score$_2$.

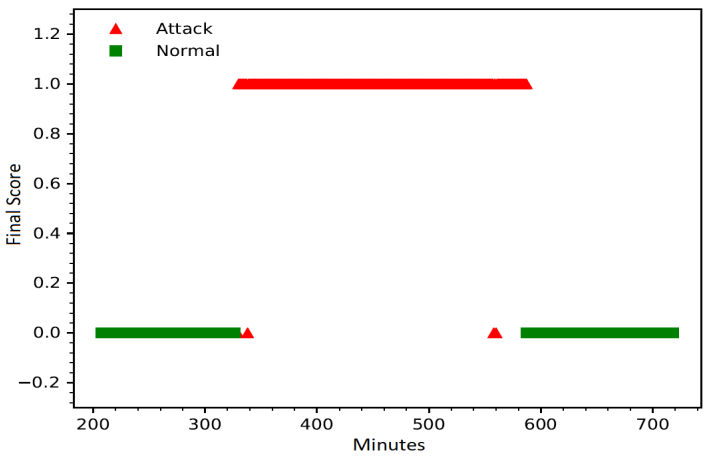


Figure 4.13. Final anomaly binary score of Scheme A.

The proposed algorithm is applied on each switch Si in the network and corresponding scores, $score_1$ and $score_2$ are obtained. Figure 4.14 displays the final anomaly

score for all switches in the network, obtained by multiplying the corresponding $score_1$ and $score_2$. As shown in the Figure 4.13 and Figure 4.14, the controller immediately identifies two engaged switches in the DDoS attack scenario once the attack starts.

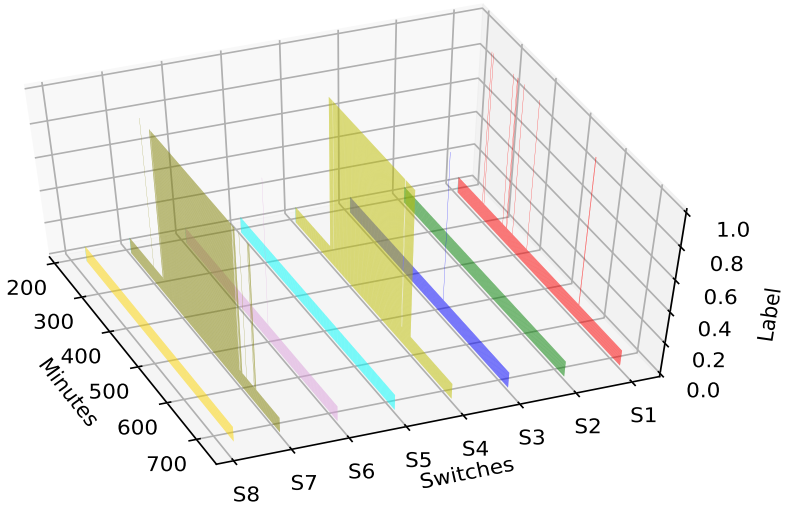


Figure 4.14. Final anomaly binary scores of Scheme A for all switches.

Table 4.2 shows the TPr, FPr, F1_score and ACC of the proposed method for each switch in the network. We just concentrate on the accuracy of $S4$ and $S7$. The detection algorithm performance on both switch $S4$ and $S7$ is the same and equals to 98.82%. Other switches are almost insensitive to the attack. Figure 4.15 displays the ROC curves for both switches $S4$ and $S7$. The AUC values for both switches are equal to 0.99.

Table 4.2. DDoS attack detection performance of Scheme A for all switches.

Switch	**TPr(%)**	**FPr(%)**	**F1_score**	**ACC(%)**
S1	1.92	1.60	—	—
S2	0	0	—	—
S3	0.38	0	—	—
S4	97.70	0	0.988	98.82
S5	0	0	—	—
S6	0.38	0	—	—
S7	98.46	0.8	0.988	98.82
S8	0	0	—	—

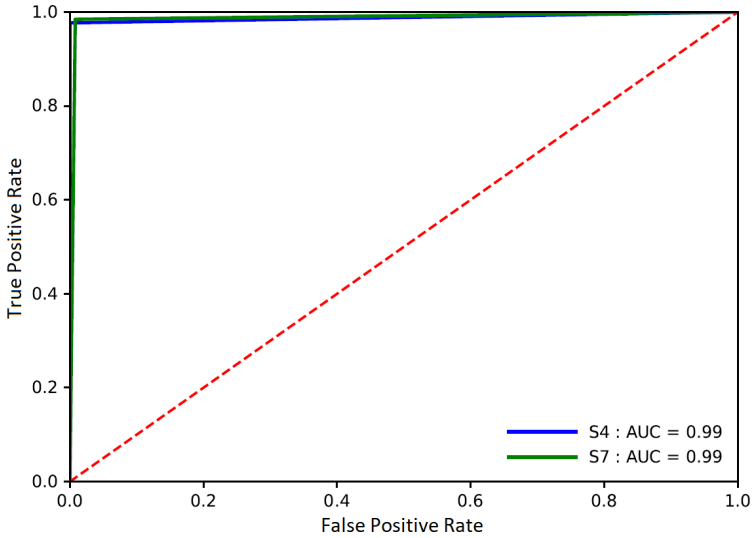


Figure 4.15. ROC of detection results for switches S4 and S7.

Because both thresholds are chosen dynamically, the detection performance is better compared to entropy-based detection algorithms which operates with a constant threshold. Furthermore, most of the referenced methods (see Chapter 2) have been carried out by topologies with just one switch, but in this work, the capability of DDoS detection and countermeasure for multiple switches is also discussed.

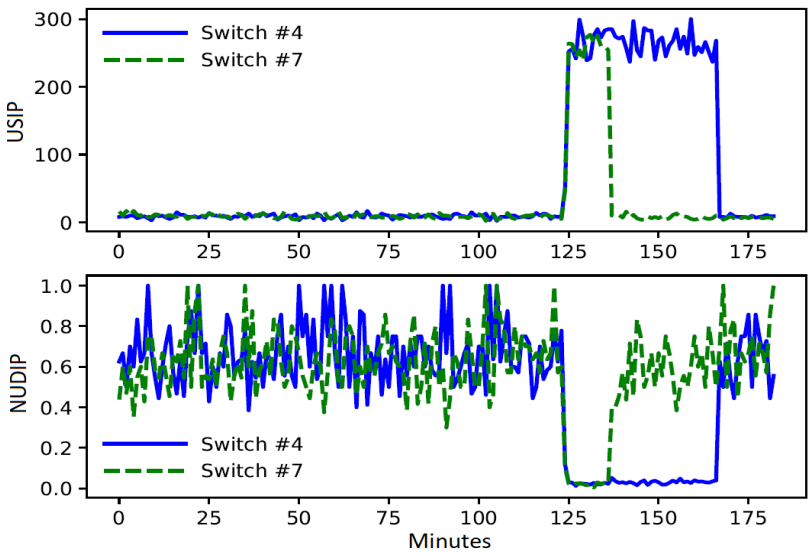


Figure 4.16. USIP and NUDIP for switches S4 and S7 during the countermeasure process.

4.2.4. Countermeasure Performance

In order to observe the effectiveness of the defense algorithm, we run another simulation on the same experiment. The countermeasure part of the detection module is activated after a while that the attack starts. The activation delay is intentionally added to see the effect of the DDoS attack on the USIP and NUDIP features. The attack is originated from the Switch, $S4$, and the victim resides in the Switch, $S7$. As the attack begins, the anomaly alarms for both mentioned switches are raised. The rules in the flow tables of both switches are updated to drop all packets with the destination IP addresses as the same as the most occurrence IP address in the corresponding hash tables.

Figure 4.16 shows the USIP and NUDIP features for both switches. As the attack starts, both features are changed accordingly. Because the attack is originated from $S4$, when the countermeasure is activated, the pattern of both features reverts to their normal condition in Switch $S7$. As a result, the anomaly alarm of the switch, $S7$ is cleared. The anomaly patterns in $S4$ does not change until the end of the attack. Once the attack is finished, the controller clears the alarm for the $S4$ and routing process of the switch is returned to its normal state.

The result shows that the proposed countermeasure scheme can be considered as an effective mechanism to mitigate DDoS attacks. Therefore, when an attack occurs, as soon as the involved switches are detected by the controller, the rules of the flow table entries are updated. As a result, the attack traffic that threatens involved switches in the network is isolated to protect the availability of the network.

5. A DDOS ATTACK DETECTION AND COUNTERMEASURE SCHEME BASED ON DWT AND AUTO-ENCODER NEURAL NETWORK FOR SDN.

In this chapter, we propose a DDoS attack detection and countermeasure scheme (Scheme B) based on Discrete Wavelet Transform (DWT) and auto-encoder neural network for SDNs. The contributions of this chapter are as follows:

(i) First, we propose a scheme that makes use of specific statistics such as the average hit rate ($avg_{hit-rate}$), Unique Source IP addresses (USIP), Total Number of Packets (totalPacket) and H_D from the flow table of SDN switches.

(ii) Since the characteristics of both USIP and totalPacket statistics change both in time and frequency domains during the DDoS attack, we utilize them to apply DWT for the extraction of the feature vector.

(iii) As an unsupervised ML-based technique, we propose to use auto-encoder neural network to obtain a dissimilarity score by applying Euclidean distance between the input and the output feature vectors. Once the attack is detected our scheme applies a countermeasure procedure to mitigate the DDoS attack on the victim server.

(iv) Finally, to evaluate the DDoS attack detection performance, we implement our scheme on an example SDN network via using GNS3 environment and Mininet emulator [43, 44] against DNS amplification, Network Time Protocol (NTP) and TCP SYN flood attacks for this particular example network.

5.1. Wavelet-Based DDoS Detection and Defense

In this section, we introduce the wavelet-based DDoS attack detection and defense scheme for SDN. Our proposed scheme is integrated into the controller of the SDN, where all the communications between the controller and the switches are carried out by the OpenFlow (OF) protocol.

Table 5.1. Definitions of symbols used in Chapter 5.

Symbol	**Definition**	**Symbol**	**Definition**
H_S,H_D	Hash tables for USIP and UDIP	$counter$	The hash table counter variable
t	The sampling time interval	r_t	The number of flow entries in the flow table at time t
k	Number of samples for the training phase	ι	Size of impulse responses of the filters in DWT
(x_t, z_t)	USIP and totalPacket instances at time interval t	$avg_{hit-rate_t}$	Average hit rate instances at time interval t
$\Upsilon = \{(x_1, y_1) \ldots, (x_k, y_k)\}$	The bi-variate time-series of USIP and totalPacket used in training phase	$Eigvec_{max}$	The eigenvector of the maximum eigenvalue of the covariance matrix of Υ
$Z = \{\zeta_1, \ldots, \zeta_w\}$	Time-series of product of $(x_t,\ z_t)$ and $Eigvec_{max}$	w	The sliding window size
$\mathcal{Z} = \{z_1 \ldots, z_w\}$	The uni-variate time-series of z_t used to apply DWT	l	The l^{th} sub-band of the DWT
μ_l	The mean value of l	σ_l^2	The variance value of l
skw_l	The skewness value of l	$kurt_l$	The kurtosis value of l
$V5_l, V25_l, V50_l, V75_l, V90_l$	The percentile values of l	H_l	The entropy value of l
f_t	The feature vector at time interval t	$\hat{f}_t$	The output of the auto-encoder at time interval t
n	The size of feature vector f_t	$Hit_i, 1 \leq i \leq k$	The time-series of $avg_{hit-rate_t}$ used in training phase
$th_{rate} = \mu_{rate} - \mathfrak{a} \times \sigma_{rate}$	The threshold value of $avg_{hit-rate}$, where μ_{rate} and σ_{rate} are the mean and standard deviation of Hit_T	$\mathcal{Z}_T = \{z_1 \ldots, z_k\}$	The uni-variate time-series of z_t used in training phase
F	A set of feature vectors used to train the auto-encoder	Euc_dist	Set of Euclidean distances between F and $\hat{F}$
$th_{dist} = \mu_d + b \times \sigma_d$	The threshold value of Euclidean distance, where μ_d and σ_d are the mean and standard deviation of Euc_dist	DIP_max	The destination IP address with the maximum occurrences

The overview of the scheme is illustrated in Figure 5.1 and the definitions of all symbols used in this scheme are given in Table 5.1. The scheme has three modules: (i) *Statistics Collection* module, (ii) *Feature Extraction* module and (iii) *DDoS Detection* module. First, the *Statistics Collection* module obtains statistics including USIP, totalPacket, H_D and the average hit rate ($avg_{hit-rate}$) from switches. Then, USIP and totalPacket are processed by the *Feature Extraction* module to extract discriminating features to form the feature vector, which is used by the *DDoS Detection* module to detect attack samples. Meanwhile, the $avg_{hit-rate}$ feature, which gives information about the average number of matched packets in the flow table during a specific time interval, is used for enabling the *DDoS Detection* module. Later, the hash table of UDIP, H_D is used for the countermeasure of the *DDoS Detection* module.

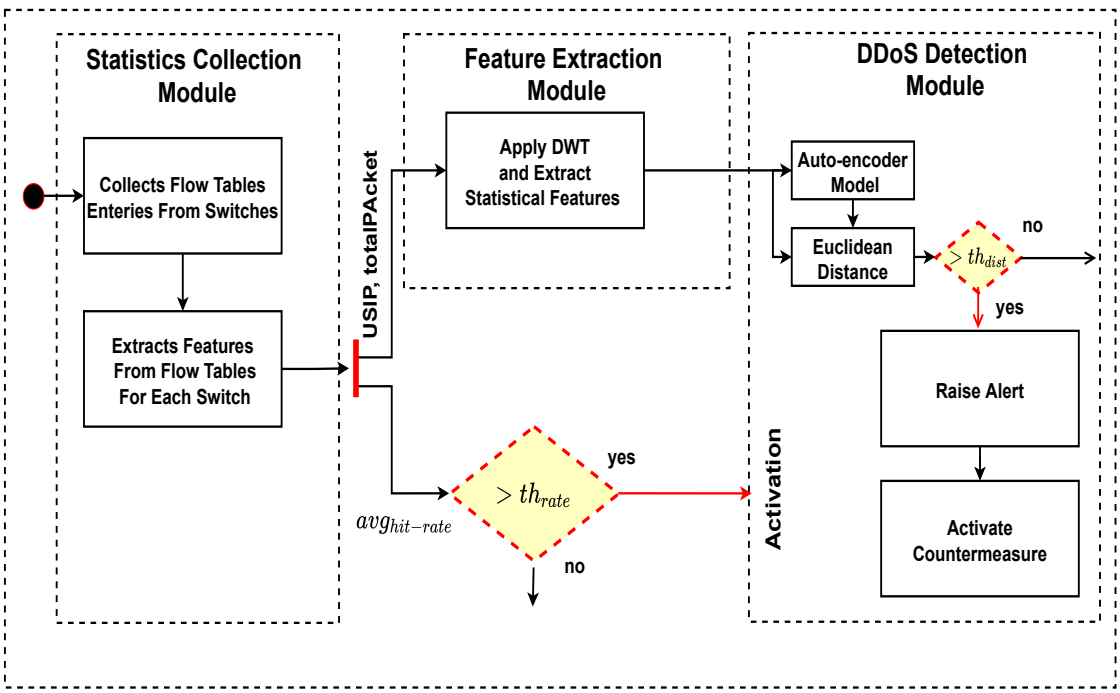


Figure 5.1. Proposed DDoS detection and defense scheme (Scheme B) for SDN.

5.1.1. Statistics Collection

As the name indicates, this module is used for collecting statistics (USIP, totalPacket, H_D and $avg_{hit-rate}$) from the flow tables of switches. For a time interval t, the

average hit rate statistic is defined as follows:

$$avg_{hit-rate_t} = \frac{flow\_count_t}{r_t \sum_{i=1}^{r_t} duration}, \tag{5.1}$$

where $flow\_count_t$ and r_t are the total number of flows and rows in the flow table at time t respectively. During the DDoS attack, since the USIP value of incoming packets increases, the number of matched flows in flow table decreases. These new flows should be recorded in the flow table as the new flow entries which yields a decrease in $avg_{hit-rate}$.

While the USIP and totalPacket statistics are used by the *Feature Extraction* module, the $avg_{hit-rate}$ and H_D feature is employed to activate the *DDoS Detection* module. As shown in Algorithm 5.1 illustrated in Figure 5.2, two hash tables namely, H_S and H_D are used to find USIP and UDIP for each specific time interval t, respectively. If a source IP address is already listed in H_S, the corresponding counter is incremented by one, otherwise it is added to the table and the counter is set to 1. The same procedure is applied for H_D and the destination IP addresses. Let x_t and z_t be the samples of USIP and totalPacket statistics at time interval t, respectively. The x_t value is obtained by counting the total number of non-empty elements in H_S and z_t is the total number of packets in the flow table at time t. Then, (x_t, z_t) is employed in *Feature Extraction* module. On the other hand, the H_D table is used during the countermeasure of *DDoS Detection* module to find the destination IP address with the maximum occurrence, which is highly critical to determine the victim node in the network. Although the $avg_{hit-rate}$ statistic extremely decreases during the DDoS attack [159], some other situations such as the flash crowd traffic can also yield to a low $avg_{hit-rate}$ value. Therefore, this statistic is just used for activating the *DDoS Detection* module when a suspicious condition (i.e., flash crowd traffic) occurs.

1:
2: EXTRACT_STATİSTİCS(t)
3: $H_D\langle Des_{IP}, counter\rangle \leftarrow 0$
4: $H_S\langle Src_{IP}, counter\rangle \leftarrow 0$
5: **for** $\forall(Src_{IP}, Des_{IP}) \in Flow\_T$ **do**
6: **if** $Src_{IP} \notin H_S$ **then**
7: $H_S\langle Src_{IP}, counter\rangle \leftarrow 1$
8: **else**
9: $H_S\langle Src_{IP}, counter\rangle \leftarrow counter + 1$
10: **end if**
11: **if** $Des_{IP} \notin H_D$ **then**
12: $H_D\langle Des_{IP}, counter\rangle \leftarrow 1$
13: **else**
14: $H_D\langle Des_{IP}, counter\rangle \leftarrow counter + 1$
15: **end if**
16: **end for**
17: $x_t \leftarrow$ Number of non-empty elements in H_S /* *USIP sample* */
18: $y_t \leftarrow$ Total number of packets /* *totalPacket sample* */
19: $r_t \leftarrow$ Number of flow entries /* *number of rows in the flow table* */
20: $dur_t \leftarrow$ Total duration /* *The sum of the duration in the flow entries* */
21: $flow\_count_t \leftarrow$ Total number of flows /* *Total counts of the flows in the flow tables at time t* */
22: $avg_{hit-rate_t} \leftarrow \frac{flow\_count_t}{r_t \times dur_t}$ /* *Average hit rate* */
23: **return** $H_D(Des_{IP}, occur),\ x_t,\ y_t,\ avg_{hit-rate_t}$
24:

Figure 5.2. Algorithm 5.1, extract statistic features (USIP, totalPacket and $avg_{hit-rate_t}$).

5.1.2. Feature Extraction

This module is used for extracting statistical features from the USIP and total-Packet statistics. In this section, we first define the training steps and then we introduce

the algorithm details of the *Feature Extraction* module which is given in Algorithm 5.2 illustrated in Figure 5.3. For the sake of the simplicity, Figure 5.3 does not include the training steps of the *Feature Extraction* module.

```
1:
2: /* From Training phase */
3: v ← load⟨Eigvec_max⟩
4: /* Initialization */
5: count ← 0
6: Z ← []
7: for each time interval t do
8:    Extract_Statistics(t)
9:    ζ_t ← (x_t, z_t) × v
10:   if count ≤ w then
11:      Z||ζ_t, where || stands for concatenation
12:      count + +
13:   else
14:      Z ← Z_2^w||ζ_t
15:      L ← DWT⟨Z⟩
16:      f_t ← []
17:      for ∀l ∈ L do
18:         f_t|| Statistical features of the level l
19:      end for
20:   end if
21: end for
22:
```

Figure 5.3. Algorithm 5.2, DWT-based feature extraction.

Definition 11. *Feature Extraction Module - Training Phase. Let k be the number of samples for (x_t, z_t), where $1 \leq t \leq k$ in the form of $\Upsilon_{k\times 2} = \{(x_1, z_1), \ldots, (x_k, z_k)\}$. Using the principal component analysis (PCA) [88], two eigenvectors and the corresponding eigenvalues of the covariance matrix, $\mathcal{C}_{2\times 2}$, of Υ are calculated. Then, the*

eigenvector of the largest eigenvalue, $Eigvec_{max} \in \mathbb{R}^{2\times1}$ is selected. The new time-series $\mathcal{Z}_T = \{z_{T_1}, \ldots, z_{T_k}\}$ is computed by using the following equation:

$$Z_{T_{k\times1}} = \Upsilon_{k\times2} Eigvec_{max_{2\times1}}. \tag{5.2}$$

Once $\mathcal{Z}_\mathcal{T}$ is obtained, it is divided into the equally-length subsets, $\{Z_{Ti}\}_{i=1}^{k-w+1}$, by applying a sliding window with the size of w and the step size of one. Then, DWT is applied on each subset to find the corresponding sub-band (level) coefficients sets. For each level, l, the statistical features including μ_l, σ_l^2, $V5_l$, $V25_l$, $V50_l$, $V75_l$, $V90_l$, skw_l, $kurt_l$ and H_l are estimated and then the result is concatenated with the results of other levels to form the feature vector, $f_i \in \mathbb{R}^{1\times n}$ where $n = number\ of\ levels(l) \times 10$. Finally, the feature set of $F \in \mathbb{R}^{(k-w+1)\times n}$ is built by f_i vectors in order to be used for the DDoS Detection module training phase.

Once the training phase has been completed and the corresponding parameter, $Eigvec_{max}$ has been obtained, the normal operating of the module begins. As described in Algorithm 5.2 (Figure 5.3), for each time interval t, the x_t and z_t values are obtained from the flow table of the switch. Then a new variable, ζ_t, is generated by multiplying (x_t, z_t) by $Eigvec_{max}$. Initially, w samples of ζ_t is used to generate a new series, $Z = \{\zeta_1, \ldots, \zeta_w\}$. Then, DWT is applied on Z. The feature sets from all levels are concatenated together to create the feature vector, f_t, which is then used by the *DDoS Detection* module. The process of feature extracting continues by adding each new ζ_t to the end of Z and discarding the first element.

5.1.3. DDoS Detection

This module is used for detecting the DDoS attack samples and to apply countermeasure against them. In order to activate this module the value of $avg_{hit-rate}$ is used. As soon as the module activated, the feature vector, f_t, obtained from the *Feature Extraction* module, is employed by the auto-encoder neural network to detect DDoS attack samples. If any attack sample is found, the countermeasure of the module is

activated. There are different parameters required to be found before using this module, which are obtained during the training phase. In this section, first the definition of the training phase is given and then the algorithm of detection, given in Algorithm 5.3 illustrated in Figure 5.4, is discussed.

```
1:
2: /* From Training phase */
3: Model ← load⟨auto_model⟩
4: th ← load⟨th⟩
5: th_d ← load⟨th_d⟩
6: for each time interval t do
7:    if log10(avg_hit−rate_t) ≤ th then
8:       f̂_t ← Model⟨f_t⟩
9:       Eu_dist_t ← d⟨f, f̂⟩ , where d⟨.⟩ stands for Euclidean distance
10:      if log10(Eu_dist_t) > th_d then
11:         DDoS attack is detected
12:         Countermeasure is activated
13:      else
14:         Normal traffic is detected
15:      end if
16:   end if
17: end for
18:
```

Figure 5.4. Algorithm 5.3, auto-encoder neural network-based DDoS detection.

Definition 12. *DDoS Detection Module - Training Phase. In order to activate the DDoS Detection module,* $log_{10}(avg_{hit-rate})$ *is compared with a pre-defined threshold value,* th_{rate}*. The* th_{rate} *value is estimated by obtaining k values of* $log_{10}(avg_{hit-rate_t})$ *and generating a series,* Hit_T*. Then, the statistical parameters of the empirical probability distribution of* Hit_T *including the mean value,* μ_{rate}*, and standard deviation,*

σ_{rate}, are obtained and th_{rate} is selected as follows:

$$th_{rate} = \mu_{rate} - \mathfrak{a}\sigma_{rate}.$$

The auto-encoder neural network model, auto_model, is obtained by using the feature set, F, obtained during the training phase of the Feature Extraction module. The detection is carried out by comparing the Euclidean distance between the input and the output of the auto-encoder with a pre-defined threshold, th_{dist}. Therefore, once the auto-encoder is trained, for each element in F, the corresponding output is estimated. The $log_{10}(.)$ value of the Euclidean distance between each element and its output is calculated and saved in a new set, Euc_dist. The mean value, μ_d and the standard deviation, σ_d, of the Euc_dist set are calculated and th_{dist} is selected as given in the following equation:

$$th_{dist} = \mu_d + b\sigma_d.$$

The *DDoS Detection* module operates as illustrated in Algorithm 5.3 (Figure 5.4). If $log_{10}(.)$ of the $avg_{hit-rate_t}$ value is less than th_{rate}, the *DDoS Detection* module is activated. Then, the auto-encoder gets f_t as the input and generates the output, $\hat{f}_t$. The Euclidean distance between f_t and $\hat{f}_t$ is obtained and $log_{10}(.)$ value is compared with th_{dist}. If the logarithmic distance is greater than th_{dist}, the DDoS attack alarm is activated. In order to identify whether the traffic sample belongs to the normal traffic or attack traffic, the change in the feature vector should be observed for a period of time (i.e., 41 sec. in our experiments for 1 second of sample rate). Therefore, there is a transient delay in the detection by the auto-encoder neural network.

When an attack instance is detected in a switch by the *DDoS Detection* module, the anomaly alarm is raised and the countermeasure is activated. The countermeasure uses the H_D table and extracts the unique destination IP address with the maximum occurrence, DIP_max. The switch ID and the corresponding DIP_max are added to the

discard list. In the flow table of the switch, the action section of all flows with the destination IP address as the same as DIP_max, are updated as 'drop' action. Therefore, the burden of the DDoS attack on the network is alleviated. If there are more than one switch through the pathway of the anomaly traffic to the victim, by applying the countermeasure method, the pattern of corresponding features in all switches, except the first one return to their normal state. While updating the victim's state from attack to the normal state, another transient delay occurs since the network traffic should be observed for a particular time period in order to get the correct feature vector values. For instance, it takes 39 seconds on average in our experiments. However, this delay does not have any significant effect on the normal process of the network, since the switch ID and the DIP_max are removed from the discard list and the action section of the flow entries are updated right after the attack is completed.

5.2. Experimental Results

This section presents the evaluation of the DDoS attack detection performance based on the simulations on an example attack scenario.

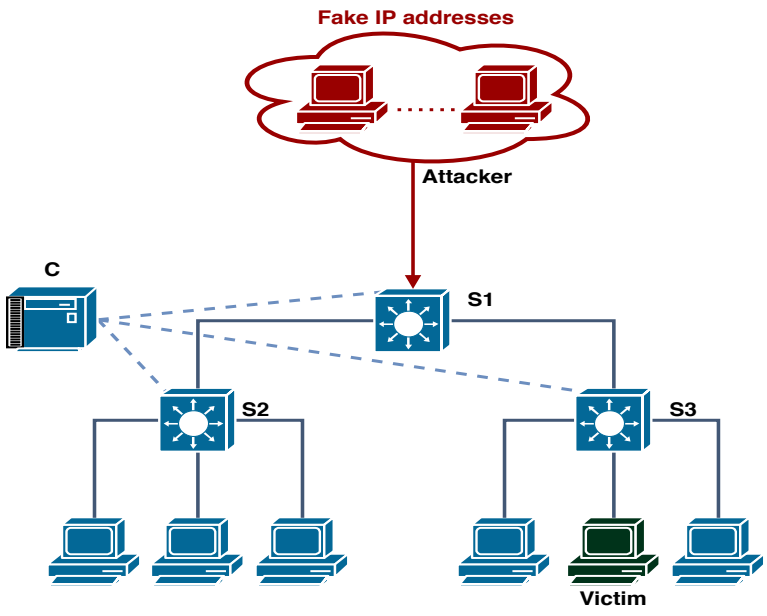


Figure 5.5. Experimental topology for Scheme B and Scheme C.

5.2.1. Dataset and Simulation

Figure 5.5 shows the experimental setup used in this work. All the simulations are carried out by using GNS3 environment [43]. The SDN topology is implemented using Mininet environment [44] that consists of a Ryu controller [160] and three OF L4-switches ($S1$, $S2$ and $S3$) which communicate with the controller through OF 1.3 protocol [161]. The real traffic trace data from MAWI dataset [162] is fed to the network using TcpReplay tool [156] as the normal and background traffic. An Ubuntu docker is used to carry out the DDoS attack to the network. The attack traffic source IP addresses are spoofed and randomized; therefore, from the victim side, the attack traffic is seemed to come from different sources. As illustrated in the figure, the attack traffic goes into the network from $S1$ and reaches to the victim device through $S3$. In order to evaluate the performance of the proposed scheme, the following attacks are simulated in this work:

- DNS Amplification Attack: The protocol, destination port and attack packet size are selected as DNS, 53 and 60 bytes respectively. Other parameters are randomly chosen.
- NTP Attack: The protocol, destination port and attack packet size are selected as NTP, 123 and 90 bytes respectively. Other parameters are randomly chosen.
- TCP SYN Flood Attack: The protocol is selected as TCP and TCP flag is set to SYN flag. Other attributes have been randomly chosen.

5.2.2. DDoS Attack Detection Performance

For each second of the traffic data, the *Statistics Collection* module uses Algorithm 5.1 (Figure 5.2) to extract the parameters including USIP, totalPacket, $avg_{hit-rate}$ and the H_D table, from each switch. While the USIP and totalPacket values are processed in the *Feature Extraction* module, the $avg_{hit-rate}$ value is used by the *DDoS Detection* module. First, the USIP and totalPacket parameters are processed by the *Feature Extraction* module to build the feature vector using DWT which is given in

Algorithm 5.2 (Figure 5.3). Then, if $avg_{hit-rate}$ is less than th_{rate}, it indicates that a potential anomaly exists in the network traffic; therefore, for further inspection and to detect DDoS samples, the *DDoS Detection* module is activated and starts to employ Algorithm 5.3 (Figure 5.4). During this detection process, the auto-encoder neural network gets the feature vector as the input and tries to rebuild it as the output. The Euclidean distance between the input and the output is compared with th_{dist} to classify the sample as normal or DDoS attack. As illustrated in the example attack scenario in Figure 5.5, the attack traffic is generated from the outside of the network and reaches to the victim through switches $S1$ and $S3$. Therefore, the following results are based on the analysis applied on these switches. As we described in Section 5.2.1, in this work, we present our experiments for DNS amplification attack, NTP attack and TCP SYN attack to evaluate the performance of the proposed scheme. In order to detect these attacks, the eigenvector ($Eigvec_{max}$), the threshold (th_{rate}) and the threshold (th_{dist}) are obtained in advance during the training phase from the normal traffic data.

During the training phase, in total, $k = 18000$ samples of normal data are employed. The th_{rate} value is obtained by estimating the statistical parameters of empirical probability distribution of the logarithm of the $avg_{hit-rate}$ training samples. Table 5.2 summarizes statistical parameters for $S1$ and $S3$. Both distributions are heavy-tailed and right skewed. As a result, the concentration of the logarithm of the $avg_{hit-rate}$ values are mostly shifted to the right; therefore, considering one standard deviation (σ_{rate}) from the mean (μ_{rate}) to the left would be enough to select the threshold, th_{rate}, for both switches. In our experiment, the th values for $S1$ and $S3$ are empirically chosen as -2.52 and -2.11, respectively.

Table 5.2. Statistical parameters of empirical pdf of $log_{10}(avg_{hit-rate})$.

Switch	$\mu_{\mathbf{rate}}$	$\sigma_{\mathbf{rate}}$	**Skewness**	**Kurtosis**
S1	-2.32	0.197	2.07	7.49
S3	-1.93	0.18	1.93	8.01

USIP (x_t) and totalPacket (z_t) values are extracted in the *Statistics Collection* module and these values are passed to the *Feature Extraction* module. In total,

$k = 18000$ samples are used to generate a time-series, Υ. Using the PCA method, the eigenvector, $Eigvec_{max}$, corresponding to the largest eigenvalue of the covariance matrix of Υ is calculated and the series, Υ, is transformed to a uni-variate time-series, $\mathcal{Z}_T$, by multiplying each element of it by $Eigvec_{max}$. A sliding window with the size of $w = 64$ and step size of 1, is applied on $\mathcal{Z}_T$. For each window, $Z_{Tii=1}^{17937}$, DWT with Haar basis function [163] is applied to generate the sub-band vector set including four elements as, $L_i = \{l_1^{1\times32}, l_2^{1\times16}, l_3^{1\times8}, l_4^{1\times8}\}$. The feature vector, $f_i \in \mathbb{R}^{1\times40}$ is obtained by estimating and concatenating statistical parameters of each element in L_i. The feature vector set, $F \in \mathbb{R}^{17937\times40}$, is used for training the auto-encoder neural network. Figure 5.6 shows the auto-encoder neural network used in this experiment. The number of layers and the number of neurons in each layer are chosen empirically. The input and the output layers have 40 neurons. The network reduces the dimension of the input to 3 and then tries to reconstruct the input.

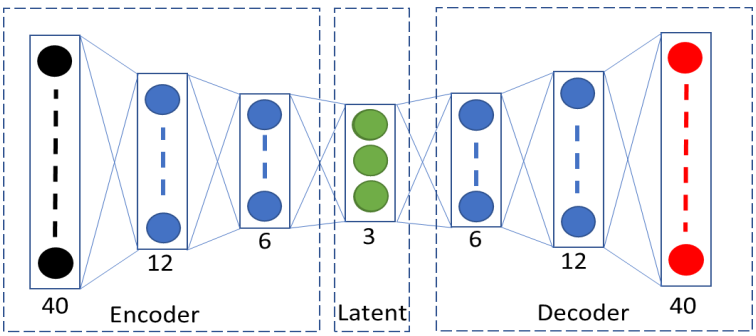


Figure 5.6. Auto-encoder neural network used in experimental setup of Scheme B.

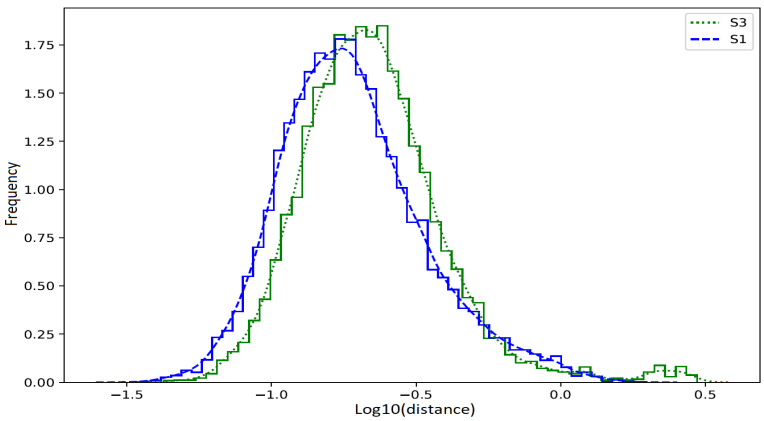
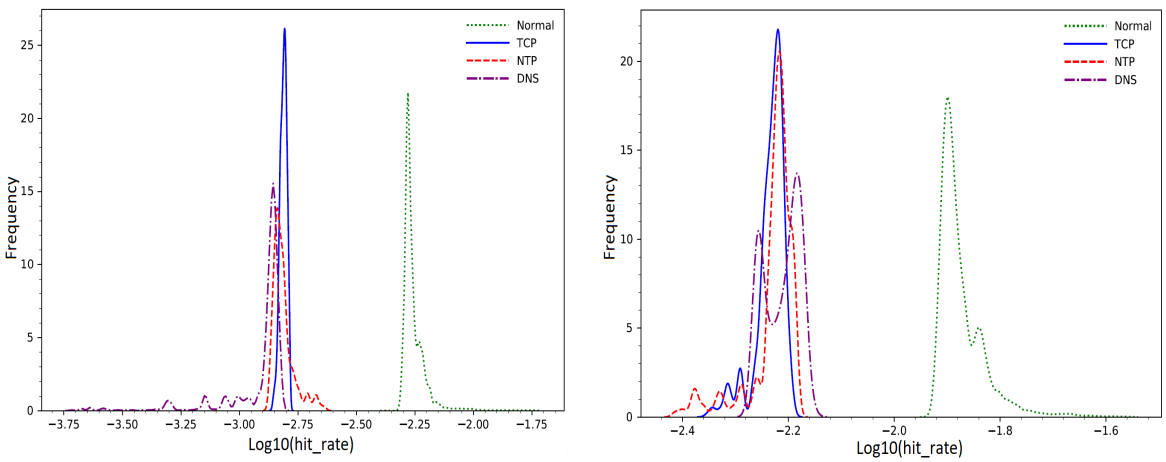


Figure 5.7. Probability distributions of logarithmic Euclidean distances between input and output of the auto-encoder for $S1$ and $S3$.

Once the parameters of the auto-encoder estimated, the Euclidean distance between the input and the output is obtained. Figure 5.7 displays the empirical probability distribution of the logarithmic Euclidean distance between the input and the output of the auto-encoder. The second threshold, th_{dist}, is obtained by studying the statistical parameters of the distributions. The value of th_{dist} is the critical logarithmic distance between the input and the output of the auto-encoder neural network that separates the attack from the normal traffic. The mean and the standard deviation of the distributions are used to determine the corresponding th_{dist} for each switch. Table 5.3 summarizes the th_{dist} values for different distances of multiple values of standard deviation (σ_d) from the mean.

Table 5.3. Threshold, th_{dist}, for different multiple of standard deviation for $S1$ and $S3$.

Switch	$\sigma_\mathbf{d}$	$\mathbf{2} \times \sigma_\mathbf{d}$	$\mathbf{3} \times \sigma_\mathbf{d}$	$\mathbf{4} \times \sigma_\mathbf{d}$	$\mathbf{5} \times \sigma_\mathbf{d}$
S1	-0.47	-0.22	0.03	0.28	0.53
S3	-0.29	0.04	0.37	0.7	1.03

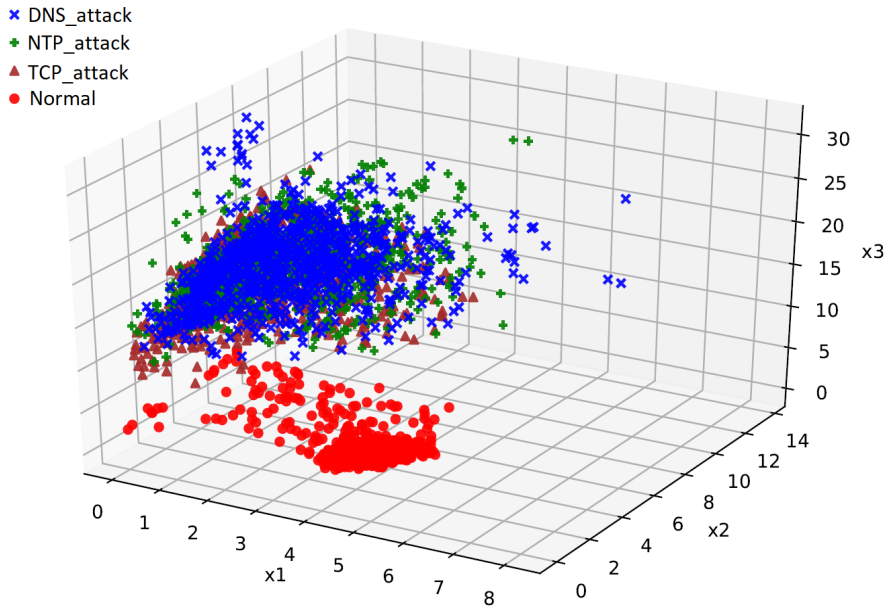


(a) Probability distribution for S1.

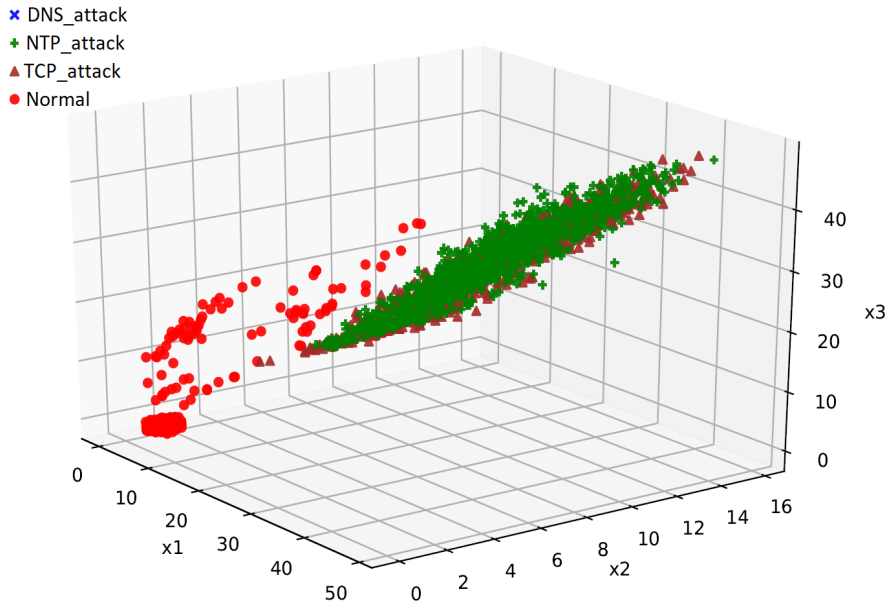
(b) Probability distribution for S3.

Figure 5.8. Probability distribution of $log_{10}(avg_{hit-rate})$ for normal traffic and attack traffics.

Figure 5.8 shows the empirical probability distribution of the logarithm of the $avg_{hit-rate}$ values for normal traffic and three types of attack traffics, where th_{rate} for both switches (-2.52 for $S1$ and -2.11 for $S3$) are enough to discriminate anomaly from normal condition.



(a) Switch S1.



(b) Switch S3.

Figure 5.9. Auto-encoder latent layer output for $S1$ and $S3$ obtained by using features from DWT.

Figure 5.9 displays the output of the latent layer for both $S1$ and $S3$, where the normal samples are almost concentrated in different location compared to the attack samples. Figure 5.10 displays the boxplot chart of the logarithmic Euclidean distance for different attack types and the normal samples of $S1$ and $S3$.

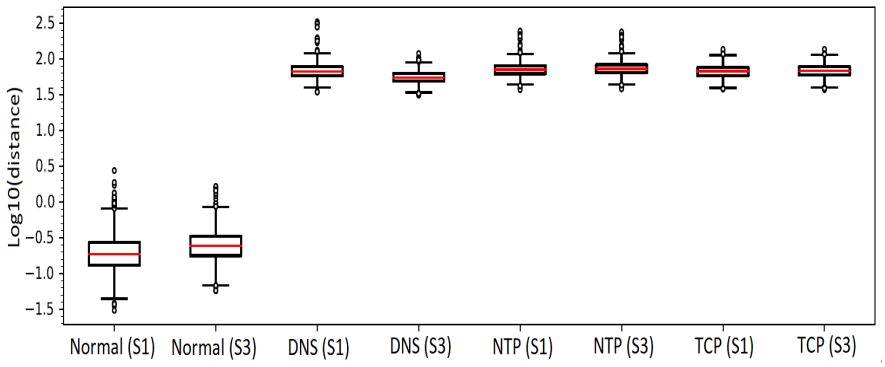


Figure 5.10. Boxplot chart of the logarithmic Euclidean distance between the input and the output of the auto-encoder for attacks and the normal traffic for $S1$ and $S3$ using DWT features.

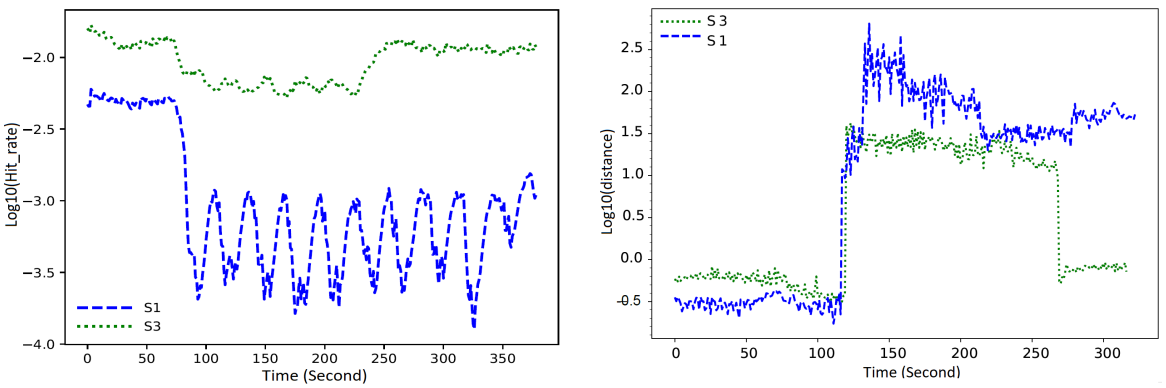
By considering and using different th_d values in Table 5.3, TPr, FPr, ACC and F1_score metrics are calculated as given in Table 5.4. Since it yields the best performance result for both $S1$ and $S3$, we prefer to use $5 \times \sigma_d$ as the threshold value for $S1$ and $S3$, which are 0.53 and 1.03, as given in Table 5.3, respectively.

Table 5.4. DDoS attack detection performance for $S1$ and $S3$ using DWT and auto-encoder neural network.

Switch	**Metric**	σ_d	$2 \times \sigma_d$	$3 \times \sigma_d$	$4 \times \sigma_d$	$5 \times \sigma_d$
S1	TPr(%)	100	100	100	100	100
	FPr(%)	16.24	5.59	0.65	0.05	0
	ACC(%)	91.9	97.2	99.7	99.9	100
	F1_score	0.92	0.98	0.99	099	1
S3	TPr(%)	100	100	100	100	100
	FPr(%)	5.7	1.68	0.59	0	0
	ACC(%)	97.1	99.2	99.7	100	100
	F1_score	0.98	0.99	0.99	1	1

All discussed results so far are based on the assumption that each window consists just attack or just normal data. There are two transient delays during the attack detection. The first delay is the transition from normal traffic to attack traffic. From the empirical results, for window size, $w = 64$ and sample rate of 1 second, it takes 41 seconds in average that the feature characteristics change from normal to attack. The second delay happens during the transition from attack to normal state. In average,

it takes around 39 seconds that the feature characteristics back to the normal state. Although entropy-based methods are widely used in literature for DDoS attack detection, the mapping of two distributions to the same entropy values can be employed by the attackers to generate attack samples similar to the normal traffic regarding to the distribution. The fundamental difference between the attack and normal traffic in frequency domain [45], makes it difficult for the attacker to generate normal-like packages with respect to the frequency domain. Moreover, the use of frequency-based discriminating features as the input for the ML-based methods, result in the better performance.



(a) Logarithmic $avg_{hit-rate}$.

(b) Logarithmic Euclidean distance.

Figure 5.11. Logarithmic $avg_{hit-rate}$ and logarithmic Euclidean distance during the countermeasure experiment.

Once an attack sample is detected in a switch by the *DDoS Detection* module, the module raises the threat alarm and the countermeasure is activated. Using the hash table H_D, the most occurrence destination IP address, DIP_max, is obtained and the action parts of all flow entries with the destination IP addresses of DIP_max, are set to drop action. Using the same experimental setup, Figure 5.5, we run another experiment to evaluate the countermeasure. The countermeasure is activated after a while intentionally, to see the effect of DDoS attack on the statistics. Figure 5.11 illustrates the log_{10}(Euclidean distance) and $log_{10}(avg_{hit-rate})$ for $S1$ and $S3$. As the attack starts, the $log_{10}(avg_{hit-rate})$ value decreases in both switches. It takes 37 and 39 seconds to see the change in log_{10}(Euclidean distance) parameters in $S1$ and $S3$, respectively. When the countermeasure is activated, while $log_{10}(avg_{hit-rate})$ in switch

$S3$ returns to its normal state, it takes 37 seconds for log_{10}(Euclidean distance) to return to its normal state. $log_{10}(avg_{hit-rate})$ and log_{10}(Euclidean distance) for switch $S1$ remain abnormal till the end of the attack.

5.2.3. DFT Vs. DWT

In order to compare the performance of DFT with the performance of DWT with respect to detection accuracy, we replace the DWT with DFT in Figure 5.1, run the scheme and obtain the result. Contrary to DWT, which provides different sub-band vectors, DFT provides just one vector with the same size of the input, $w = 64$. The absolute value of the obtained vector is calculated and half of the data is discarded due to the symmetric characteristic of the absolute value of the DFT output. Then the feature vector, $f_i \in \mathbb{R}^{1\times 10}$, is obtained. The feature vector set, $F \in \mathbb{R}^{17937\times 10}$, is used for training the auto-encoder neural network. Because the feature vector size is 10, we eliminate one layer with the size of 12 in both encoder and decoder. Therefore, the new auto-encoder neural network has the input and output size of 10 with the hidden layer-size sequence of $\{6, 3, 6, 3\}$.

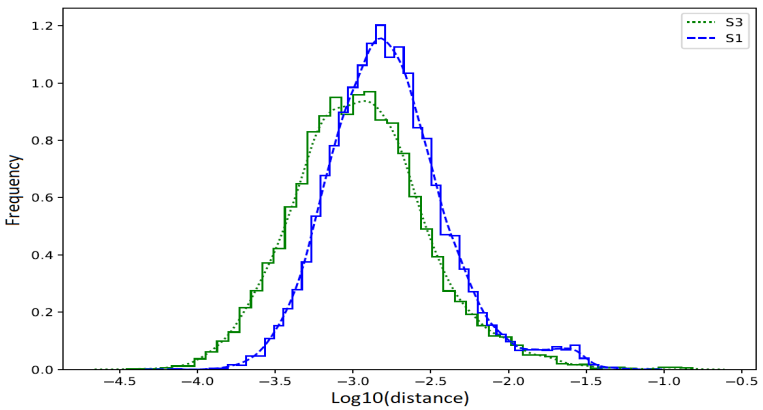


Figure 5.12. Probability distributions of logarithmic Euclidean distances between input and output of the auto-encoder for $S1$ and $S3$ using DFT features.

Figure 5.12 displays the empirical probability distribution of the logarithmic Euclidean distance between the input and the output of the auto-encoder during the training phase and Table 5.5 summarizes the th_{dist} values for different multiple values of standard deviation, σ_d, for both S_1 and S_3 .

Table 5.5. Threshold, th_{dist}, for different multiple of standard deviation for $S1$ and $S3$, using DFT and auto-encoder neural network.

Switch	$\sigma_{\mathbf{d}}$	$\mathbf{2} \times \sigma_{\mathbf{d}}$	$\mathbf{3} \times \sigma_{\mathbf{d}}$	$\mathbf{4} \times \sigma_{\mathbf{d}}$	$\mathbf{5} \times \sigma_{\mathbf{d}}$
S1	-2.39	-2.01	-1.63	-1.25	-0.86
S3	-2.54	-2.08	-1.62	-1.16	-0.71

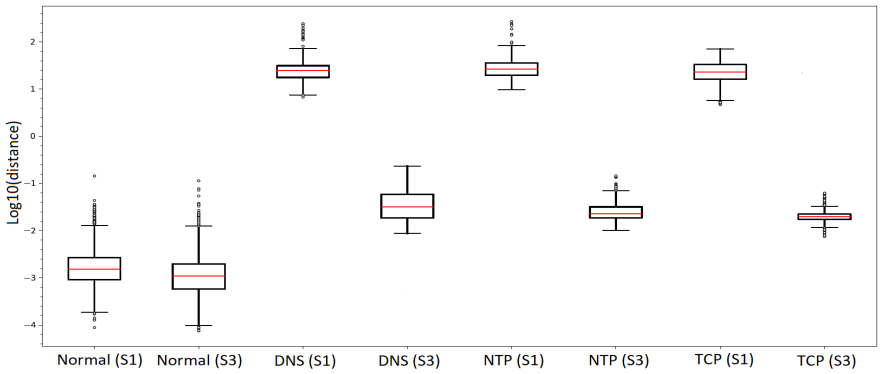


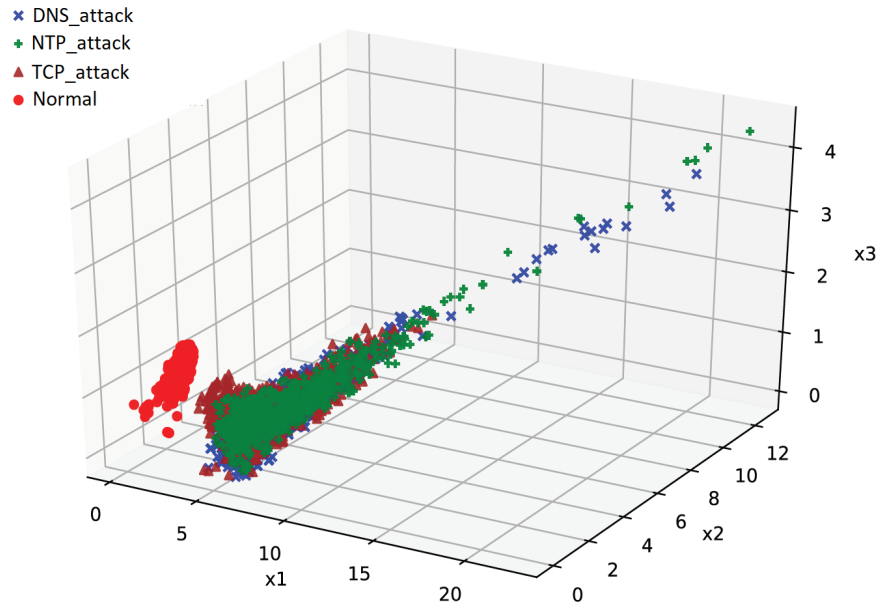
Figure 5.13. Boxplot chart of the logarithmic Euclidean distance between input and output of the auto-encoder for all traffic types for $S1$ and $S3$ using DFT features.

Once the auto-encoder neural network is trained, test data is introduced to the network which consists of attack and normal samples. Figure 5.13 displays the boxplot chart of the logarithmic Euclidean distance for different attack types and the normal samples of $S1$ and $S3$. The detection rates for different th_{dist} values are summarized in Table 5.6. Comparing the results between DFT and DWT approaches, the DWT approach outperforms.

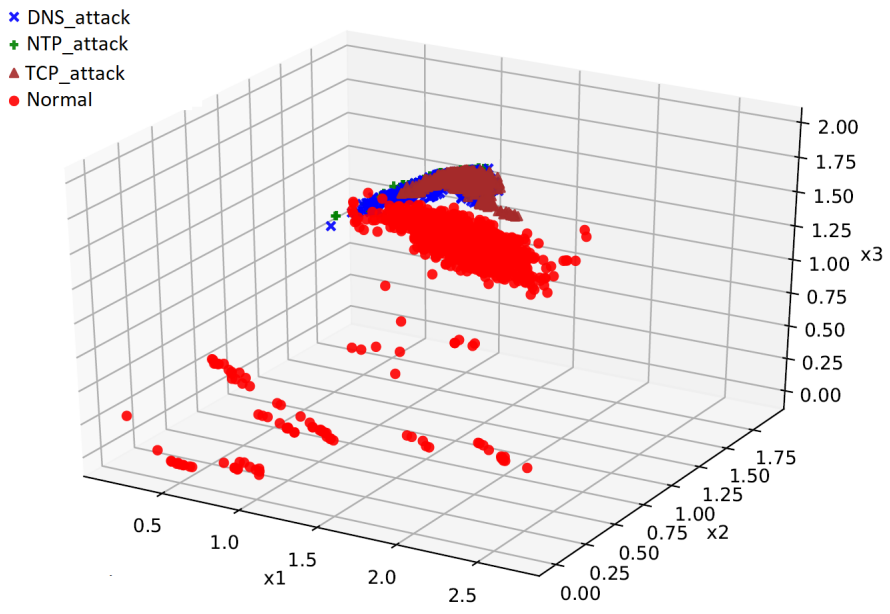
Table 5.6. DDoS attack detection performance for S1 and S3 using DFT and auto-encoder neural network.

Switch	**Metric**	$\sigma_{\mathbf{d}}$	$\mathbf{2} \times \sigma_{\mathbf{d}}$	$\mathbf{3} \times \sigma_{\mathbf{d}}$	$\mathbf{4} \times \sigma_{\mathbf{d}}$	$\mathbf{5} \times \sigma_{\mathbf{d}}$
S1	TPr(%)	100	100	100	100	100
	FPr(%)	59.1	20.3	6.2	2.3	0.1
	ACC(%)	94.7	98.1	99.5	99.9	99.9
	F1_score	0.95	0.98	0.99	0.99	0.99
S3	TPr(%)	100	99.8	45.1	7.94	0.04
	FPr(%)	14.24	3.86	0.7	0.16	0
	ACC(%)	95.3	98.1	81.63	75.9	66.3
	F1_score	0.95	0.98	0.82	0.74	0.62

Figure 5.14 displays the output of the latent layer for both $S1$ and $S3$. The output of the latent layer for the normal samples in $S3$ is widely distributed which yield to the distance set with the wide range of values.



(a) Switch S1.



(b) Switch S3.

Figure 5.14. Auto-encoder latent layer output for S1 and S3 obtained by using features from DFT.

6. A DDOS ATTACK DETECTION AND COUNTERMEASURE SCHEME BASED ON CWT AND CNN FOR SDN.

In this chapter, we propose a DDoS attack detection and countermeasure scheme (Scheme C) based on Continuous Wavelet Transform (CWT) and Convolutional Neural Network (CNN) for SDNs. The contributions of this chapter are as follows:

(i) CWT is applied on both Unique Source IP addresses (USIP) and Normalized Unique Destination IP addresses with respect to total number of packets (NUDIP) to obtain two two-dimensional features. Because of the CWT characteristic, the elements in each obtained future are correlated.

(ii) With respect to the correlation among the elements of the obtained features, the CNN architecture is employed to classify traffic samples as attack or normal. The DDoS attack detection is activated when the median of USIP statistic is larger than a predefined value which is obtained during the training phase of the scheme. Therefore, the computational burden imposed by the CNN classifier is reduced. Once an attack sample is detected, the hash table of the number of unique destination IP addresses (H_D) obtained from the flow table of SDN switches is used to find the potential victim IP address and then all flows forwarded to that IP are dropped temporarily.

6.1. DDoS Attack Detection and Defense Scheme

In this section, the DDoS attack detection and defense scheme based on CWT and CNN is introduced. The proposed scheme is integrated into the controller of the SDN. The OpenFlow (OF) protocol is responsible for handling all communications between controller and the switches.

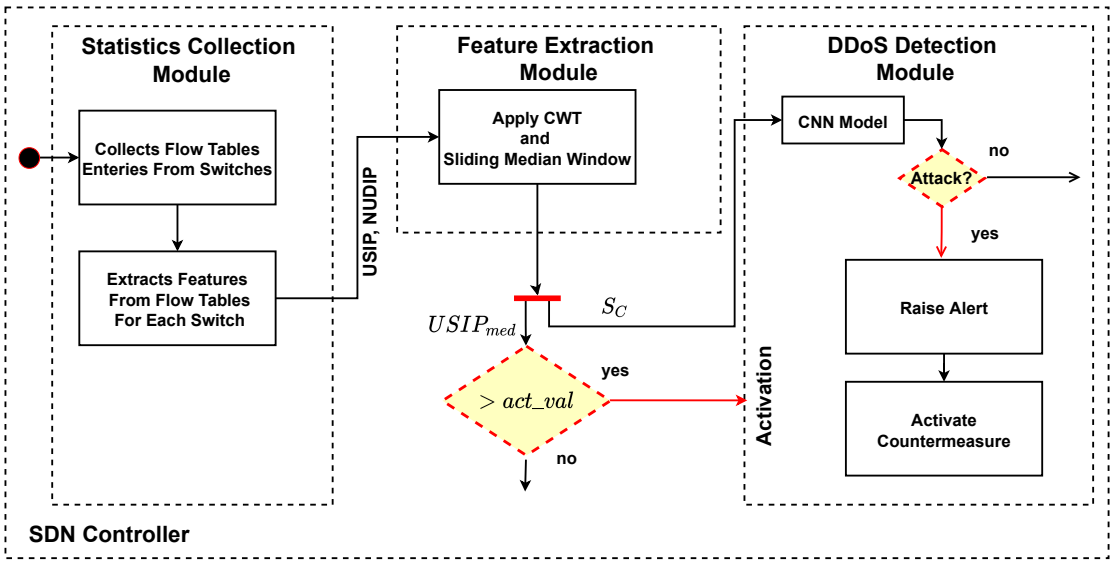


Figure 6.1. Proposed DDoS detection and defense scheme (Scheme C) for SDN.

Table 6.1. Definitions of symbols used in Chapter 6.

Symbol	Definition	Symbol	Definition
H_S,H_D	Hash tables for USIP and UDIP	*counter*	The hash table counter variable
t	The sampling time interval	r_t	The number of flow entries in the flow table at time t
τ, S	The transition and scale parameters of CWT	$\Phi(t)$	The mother wavelet
(x_t, z_t), $\mathcal{X}$, $\mathcal{Z}$	The USIP and totalPacket instances at time interval t and corresponding time-series	w	The sliding window size
$S_{c_{qt}}$	The two-dimensional output of CWT at time interval t where $q \in \{x, y\}$	$USIP_{med_t}$	The median of $x_{N-w}, \ldots, x_w \in \mathcal{X}$ at time interval t
act_val	The $USIP_{med}$ value that activates the *Detection* module		

Figure 6.1 illustrates the overall view of the proposed scheme. The definitions of all symbols used in this scheme are given in Table 6.1. The scheme consists of three modules: (i) *Statistics Collection* module, (ii) *Feature Extraction* module and

(iii) *DDoS Detection* module. As the first step, different statistics including USIP, NUDIP and H_D are obtained from the flow tables of the OF switches by the *Statistics Collection* module. Then, the *Feature Extraction* module applies CWT on two time-series generated from the USIP and NUDIP statistics. Moreover, The moving median of the time-series obtained from USIP is also calculated by the module which is used to activate the *DDoS Detection* module. The outputs of the CWT are used by the *DDoS Detection* module to classify the samples as DDoS attack or normal. The H_D table is used during the countermeasure when DDoS samples are detected.

6.1.1. Statistics Collection

In order to detect DDoS attack traffic, in the first step, the scheme obtains different statistics from the flow tables of the OF switches. The *Statistics Collection* module is employed for this purpose. The module uses the infrastructure of the OF protocol to interact with the switches and then, collects statistics including USIP, NUDIP and H_D.

Both USIP and NUDIP statistics are used by the *Feature Extraction* module, to extract discriminating features. On the other hand, the H_D table is used during the countermeasure against ongoing DDoS attack by the *DDoS Detection* module. The *Statistic Collection* module follows Algorithm 4.1 shown in Figure 4.2 to collect the aforementioned statistics. For each time interval t two hash tables namely, H_S and H_D are initialized to find USIP and NUDIP, respectively. Each source IP address in the flow table of the OF switch is mapped to a specific cell of the H_S table by using a hash function. There is also a counter for each cell of the H_S table. If a source IP addresses already listed in the hash table, the corresponding counter is incremented by one; on the other hand, the address is added to the table and the counter is set to 1. The same process is applied for the destination IP address and the H_D table. Let x_t and y_t be the samples of the USIP and NUDIP statistics at time interval t respectively. The x_t value is obtained by counting the total number of non-zero cells of the H_S table. The y_t statistic is acquired by counting the total number of non-zero cells of the H_D table

divided by the total number of packets in the flow table. Then, (x_t, y_t) is employed in *Feature Extraction* module.

1:
2: /* *Initialization* */
3: $count \leftarrow 0$
4: $\mathcal{X} \leftarrow []$
5: $\mathcal{Y} \leftarrow []$
6: **for** each time interval t **do**
7: $Extract\_Statistics(t)$
8: **if** $count \leq w$ **then**
9: $\mathcal{X}||x_t$, where $||$ stands for concatenation
10: $\mathcal{Y}||y_t$
11: $count++$
12: **else**
13: $\mathcal{X} \leftarrow \mathcal{X}_2^w||x_t$
14: $\mathcal{Y} \leftarrow \mathcal{Y}_2^w||y_t$
15: $Sc_{x_t} \leftarrow CWT\langle\mathcal{X}\rangle$, where $Sc_{x_t} \in \mathbb{R}^{w\times w}$
16: $Sc_{y_t} \leftarrow CWT\langle\mathcal{Y}\rangle$, where $Sc_{y_t} \in \mathbb{R}^{w\times w}$
17: $Sc_t \leftarrow [Sc_{x_t}, Sc_{y_t}]$, where $Sc_t \in \mathbb{R}^{w\times w\times 2}$
18: $USIP_{med_t} \leftarrow median\langle\mathcal{X}\rangle$
19: **end if**
20: **end for**
21:

Figure 6.2. Algorithm 6.2, CWT-based feature extraction.

6.1.2. Feature Extraction

The USIP and NUDIP statistics collected by the *statistic collection* module is processed in this module to extract the discriminating features. As given in Algorithm 6.2 illustrated in Figure 6.2, for each time interval t, x_t and y_t samples are appended

to corresponding time-series $\mathcal{X}$ and $\mathcal{Y}$ respectively. At each time interval t, CWT is applied on the w samples of the recent values of $x_{N-w}, \ldots, x_w \in \mathcal{X}$ and $y_{N-w}, \ldots, y_w \in \mathcal{Y}$ as shown in Figure 6.3. As a result, corresponding two-dimensional variable, $Sc_{q_t} \in \mathbb{R}^{w \times w}$ where $q \in \{x, y\}$ is obtained. The correlation among the elements of Sc_{q_t} and its similarity to the image variables, makes this feature as an appropriate input for the CNN classifier. Sc_{x_t} and Sc_{y_t} can be considered as two channels of an image, $Sc_t \in \mathbb{R}^{w \times w \times 2}$. The Sc_t is the output of the *Feature Extraction* module employed by the *DDoS Detection* module to detect anomalies caused by DDoS attack. Furthermore, the median of $x_{N-w}, \ldots, x_w \in \mathcal{X}$, $USIP_{med}$, is also calculated in this module. The $USIP_{med}$ value is used to activate the *DDoS Detection* module.

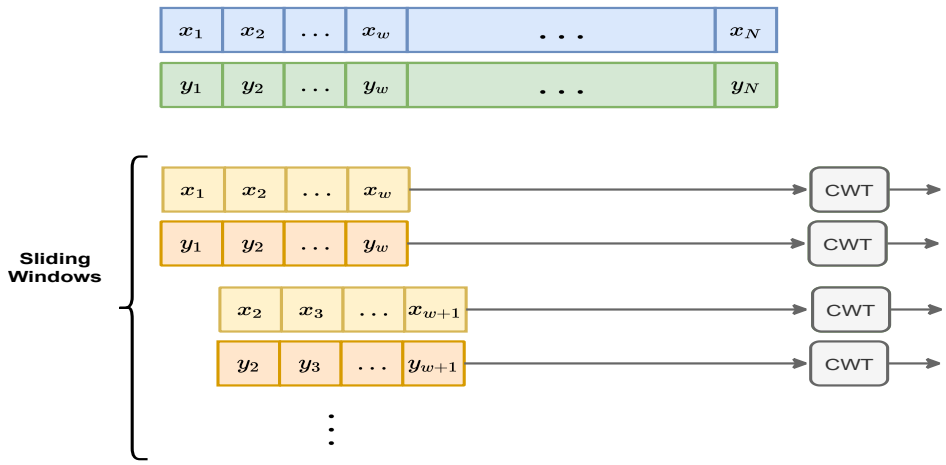


Figure 6.3. CWT feature extraction from $\mathcal{X}$ and $\mathcal{Y}$.

6.1.3. DDoS Detection

The DDoS detection and countermeasure are carried out in this module. The CNN architecture is employed as the main classifier for this purpose. The module is activated when the $USIP_{med_t}$ value is larger than a pre-defined value, act_val. The act_val value is obtained during the training phase by finding the discriminating value to separate the median of normal traffic USIP from median of USIP of DDoS attack traffic. The aim of using act_val is to alleviate the computational complexity of the scheme. The scheme uses the *DDoS Detection* module during abnormal conditions caused by the increase in $USIP_{med}$. For each time interval t, the Sc_t feature of the traffic sample

which is extracted by *Feature Extraction* module, is introduced as the input to the CNN classifier. The classifier marks the sample as attack or normal traffic based on the input. The DDoS detection procedure is described in Algorithm 6.3 illustrated in Figure 6.4. The CNN architecture should be trained with labeled samples from both attack and normal traffic in advance to discriminate attack and normal samples. As the attack begins, pattern in the features change with some delay; therefore, the *DDoS Detection* module rises the detection alarm after the aforementioned delay time (i.e., 41 sec. in our experiments for 1 second of sample rate). Although the change in the pattern of the feature from attack to normal situation also has some delay, due to the rapid change in $USIP_{med}$, the *DDoS Detection* module clear the DDoS attack alarm as soon as the attack ends. Therefore, the use of $USIP_{med}$ not only decreases the computational cost but also accelerates the recovering of the network after DDoS attack ends.

```
1:
2: /* From Training phase */
3: Model ← load⟨CNN_model⟩
4: act_val ← load⟨act_val⟩
5: for each time interval t do
6:    if USIP_med_t > act_val then
7:       label_t ← Model⟨Sc_t⟩
8:       if  label_t = Attack then
9:          DDoS attack is detected
10:         Countermeasure is activated
11:      else
12:         Normal traffic is detected
13:      end if
14:   end if
15: end for
16:
```

Figure 6.4. Algorithm 6.3, CNN-based DDoS detection.

As soon as a sample in an OF switch is classified as the attack sample, the anomaly alarm is raised and the countermeasure is triggered. The destination IP address with the maximum occurrence is extracted from the H_D table. This destination IP address and the corresponding OF switch ID are recorded in a discard list. Then, in the flow table of the OF switch the action part of all flows with the destination IP addresses as the same as the obtained destination IP address are updated as drop action. Therefore, all flows routed toward the victim through that switch are discarded and the traffic load over the victim is alleviated. Although the method eliminates the destructive effect of the ongoing DDoS attack on the victim, it also discards all possible benign traffics toward the victim which are routed through that switch. Hence, the countermeasure needs to be deactivated as soon as the attack is completed. Because the countermeasure just affects the action part of the flows, the pattern of the statistics and as the consequence the pattern of the features are not changed. As a result, the scheme can identify the completion of the attack scenario by continuing the DDoS detection process. Once the attack is over, the action part of all flows is updated to their normal state and the OF switch and the destination IP address are removed from the discard list. It is also possible that the DDoS attack flows pass through multiple OF switches to reach the victim. Because the scheme applies the detection and defense process on each switch independently, the attack detection and countermeasure are carried out on each switch on the pathway toward the victim.

6.2. Experimental Results

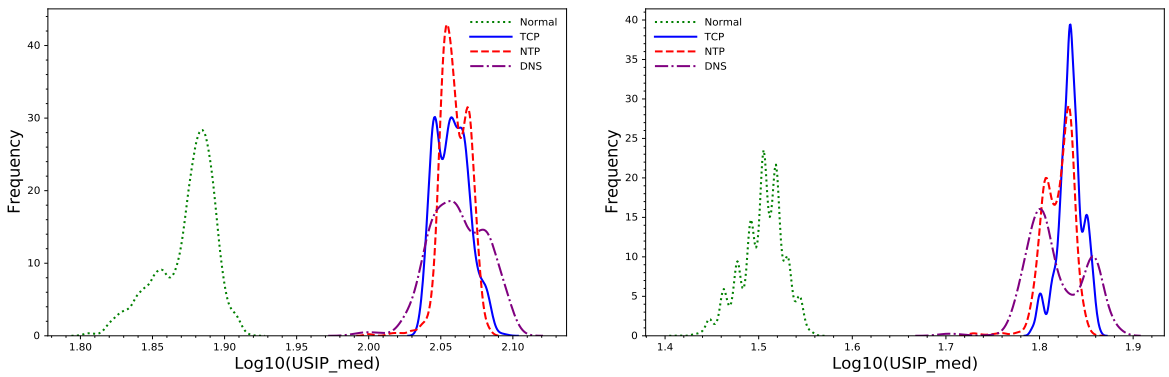
In this section, we evaluate the performance of the proposed scheme through simulation. The same dataset and the same network topology depicted in Figure 5.5 as employed in Chapter 5, are used in this chapter.

6.2.1. Detection Performance

Using Algorithm 4.1 (Figure 4.2), the *Statistics Collection* module extracts USIP (x_t), NUDIP (y_t) and H_D parameters for each $t = 1$ second from each OF switch. The

x_t and y_t parameters are employed by the *Feature Extraction* module to extract the discriminating features by following Algorithm 6.2 (Figure 6.2). The module applies CWT to obtain two two-dimensional variables, Sc_{x_t} and Sc_{y_t}, from the aforementioned parameters as the feature. Two variables are treated as two channels of an image, Sc_t and passed as the input to the CNN classifier of the *DDoS Detection* module. Using Sc_t, the module identifies DDoS attack samples by using Algorithm 6.3 (Figure 6.4).

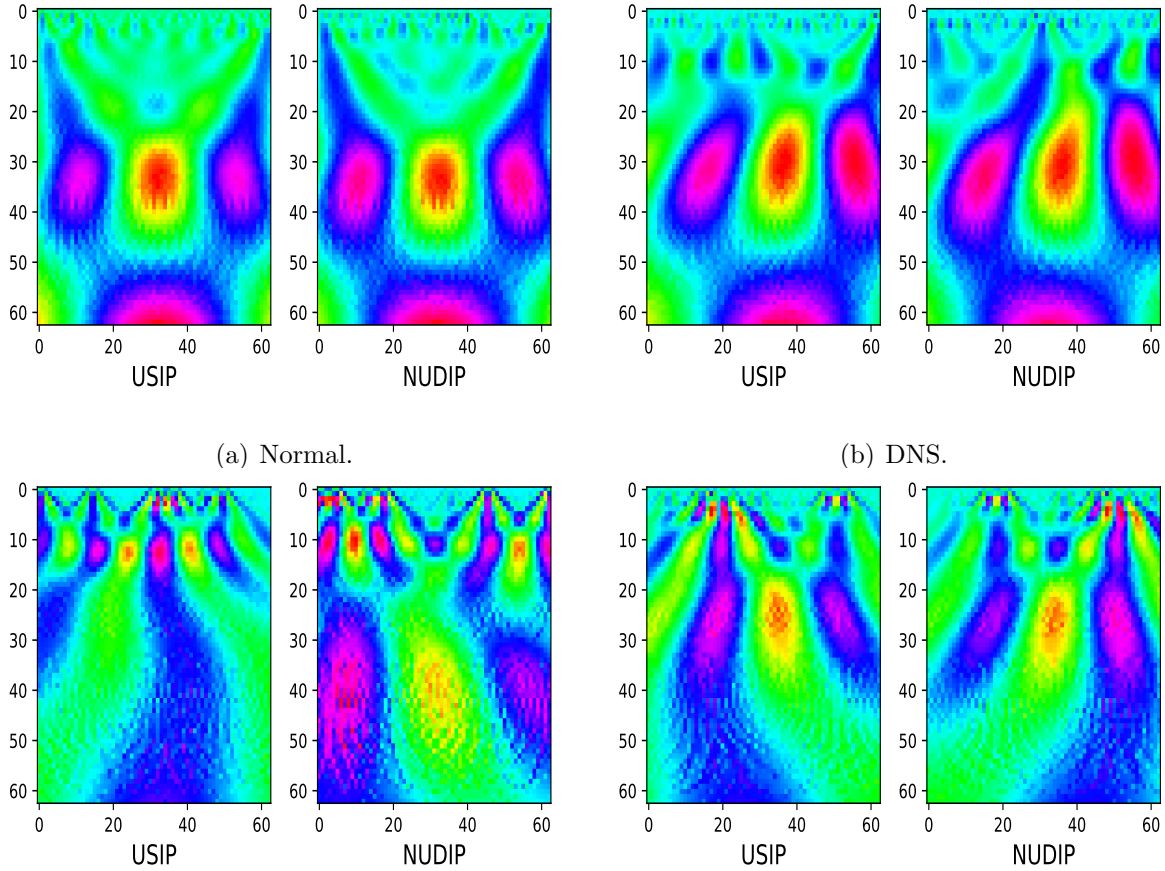
Before evaluating the performance of the proposed scheme, act_val, the $USIP_{med}$ value to activate the *DDoS Detection* module, and the parameters of the CNN classifier are estimated during the training phase. The training dataset consists of 27000 labeled samples per each attack and normal traffic. In the training phase, for each traffic types (normal and attack), x_t and y_t are collected in the *Statistics Collection* module and these values are passed to the *Feature Extraction* module. Two time-series of $\mathcal{X}$ and $\mathcal{Y}$ are generated from x_t and y_t, respectively and then, a sliding window with the size of $w = 64$ and step size of 1 is applied on each time-series separately. As a result, $\mathcal{X}$ and $\mathcal{Y}$ are divided into the sub-windows and two sets of $X_{i=1}^{27000}$ and $Y_{i=1}^{27000}$ are obtained.



(a) S1. (b) S3.

Figure 6.5. Empirical distribution of $USIP_{med}$ of normal and attack samples for both $S1$ and $S3$ (Training phase).

In order to find the act_val value, the module find the median for each element in $X_{i=1}^{27000}$. Figure 6.5 shows the empirical logarithmic distribution of $USIP_{med}$ of normal and attack samples for both $S1$ and $S3$. The act_val value can be chosen as 1.95 and 1.6 for $S1$ and $S3$, respectively in this experiment.



(a) Normal.

(b) DNS.

(c) NTP.

(d) TCP.

Figure 6.6. Scalograms of USIP and NUDIP for different traffic types obtained from switch $S1$.

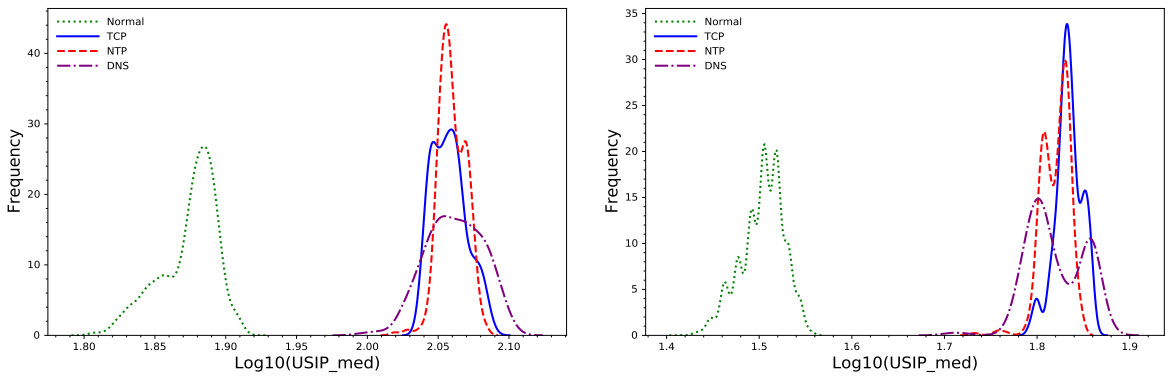
Although the distribution of $USIP_{med}$ of normal samples for both switches are located far enough from attack ones, it could have the same result for other anomalous situations rather than DDoS attack such as flash crowd events. Therefore, the module applies CWT on each sub-window and two sets of $Sc_x \in \mathbb{R}^{64\times64\times27000}$ and $Sc_y \in \mathbb{R}^{64\times64\times27000}$ are obtained. The most suitable mother wavelet for a specific time-series should match the variation of the time-series [108]. However, we cannot employ the aforementioned technique in this study since this selection technique is suitable when operating on stationary time-series and the network traffic in general has non-stationary characteristics. Therefore, a set of various wavelets including Morlet, complex Morlet (C-Morlet), Gaussian derivative (Guass), complex Gaussian derivative (C-Guass), shannon, Mexican Hat (M-Hat) and frequency B-spline (F-B-spline) are utilized in this study. Figure 6.6 displays of examples of the CWT result for differ-

ent traffic types obtained from the $S1$ samples. As the figure shows, each traffic has its own unique pattern; therefore, these features seem appropriate for discriminating attack from normal.

Table 6.2 summarizes the layers and the numbers of associated parameters of the CNN classifier used for evaluating the performance of the scheme. Number of layers, number of filters and the size of filters are selected empirically. Each kernel in the network is a 5×5 filter. In total, the CNN classifier has 10,871,898 parameters that are estimated during the training phase. Except the last layer which has Softmax activation function, the rest of activation functions are ReLU.

Table 6.2. Layers and parameters of CNN architecture.

Layer	**Output shape**	**# of Parameters**
Conv_1	$60 \times 60 \times 32$	1632
MaxPooling	$30 \times 30 \times 32$	0
Conv_2	$26 \times 26 \times 64$	51264
MaxPooling	$13 \times 13 \times 64$	0
Flatten	10816	0
Fully connected	1000	10817000
Fully connected	2	2002



(a) S1. (b) S3.

Figure 6.7. Empirical distribution of $USIP_{med}$ of normal and attack samples for both $S1$ and $S3$ (Test phase).

Once the training phase is completed, the scheme starts working to detect potential anomalies caused by DDoS attack samples. Each switch in the network is monitored

by the scheme separately and independently. The empirical logarithmic distributions of $USIP_{med}$ for both switches during the test are shown in Figure 6.7. As the figure shows, the selected act_val values, 1.95 and 1.6 for $S1$ and $S3$, respectively, are well defined. In order to evaluate the performance of the detection of the scheme, the *DDoS Detection* module is kept active during the test. Table 6.3 summarizes the accuracy results for wavelet functions employed in this study. The complex Morlet wavelet outperforms for both $S1$ and $S3$. The detection performance result of the scheme based on Morlet wavelet is summarized in Table 6.4.

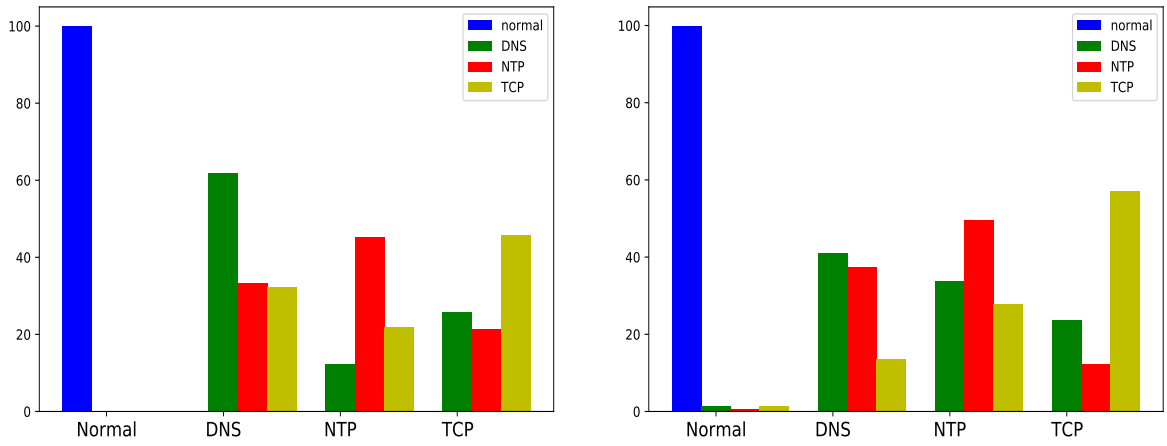
Table 6.3. Accuracy results for various continuous mother wavelets.

Wavelet	**ACC(%)-S1**	**ACC(%)-S3**	**Wavelet**	**ACC(%)-S1**	**ACC(%)-S3**
C-Gauss1	97.14	98.62	**Gauss4**	79.36	74.78
C-Gauss2	94.97	99.93	**Gauss5**	95.75	99.30
C-Gauss3	75.21	74.62	**Gauss6**	91.56	74.48
C-Gauss4	74.52	74.78	**Gauss7**	74.35	74.77
C-Gauss5	98.65	99.88	**Gauss8**	66.89	74.72
C-Gauss6	97.35	99.87	**Morlet**	99.57	97.77
C-Gauss7	94.75	99.86	**C-Morlet**	99.96	99.94
C-Gauss8	95.79	99.77	**F-B-Spline**	99.24	99.84
Gauss1	83.61	74.78	**M-Hat**	90.80	99.88
Gauss2	92.85	99.82	**Shannon**	98.2	99.79
Gauss3	95.25	74.78			

Table 6.4. DDoS attack detection performance for S1 and S3 using CWT and CNN.

Switch	**TPr (%)**	**FPr (%)**	**PEr (%)**	**ACC (%)**	**F1_score**
S1	99.88	0.0	100	99.96	0.99
S3	99.76	0.02	99.93	99.94	0.99

The employed CNN classifier is a binary classifier which classifies the input samples as attack or normal. In order to evaluate the performance of the proposed scheme in separating attack types, we modify the last layer of the CNN classifier to categorize the samples in four classes of normal, DNS attack, TCP-SYN attack and NTP attack. Figure 6.8 shows the barplot of the percentage of classification of each class versus the other classes. Although the classification rate is not high compared to the first experiment of binary classification, for each attack type, the percentage of the correct assigned label is the highest ones.



(a) S1. (b) S3.

Figure 6.8. Separation of four traffic types (Normal, DNS, NTP and TCP) for $S1$ and $S3$ using CWT and CNN.

All results discussed so far, are based on the assumption that each window which CWT is applied belongs to attack or normal. When attack starts, the window that CWT is applied, is filled gradually with attack samples. Therefore, there is a transient delay in changing pattern from normal to attack pattern. For window size, $w = 64$ and sample rate of 1 second, it takes 45 seconds in average that the feature characteristics change from normal to attack. Although during the change in pattern from attack to normal, there is another transient delay, 37 seconds, the use of $USIP_{med}$ eliminates the delay of returning the network state to the normal.

7. Comparative Performance Evaluation for the Proposed DDoS Detection Schemes

In this chapter, we compare the performances of proposed DDoS Detection and countermeasure schemes for SDN, namely Scheme A [48], Scheme B [50] and Scheme C [49]. First, we compare these schemes with respect to the DDoS attack detection performance. Later, we compare them in terms of the computational complexity cost analysis.

7.1. DDoS Attack Detection Performance

In order to have the same testbed environment for the comparison of the detection performance, we use the topology illustrated in Figure 5.5 (in Section 5.2). Since the topology used in the simulations of Scheme A is different for the simulations of Scheme B and Scheme C, we have re-evaluated the performance of the Scheme A using the topology in Figure 5.5 and we have tested the performance of Scheme A for DNS, NTP and TCP SYN attacks. New simulation results of Scheme A are given in Table 7.1 on switches $S1$ and $S3$.

Table 7.1. Detection rate performance for switches $S1$ and $S3$ for Scheme A.

Switch	**Attack type**	**TPr (%)**	**FPr (%)**	**ACC (%)**	F1_score
$S1$	**DNS**	98.8	9.5	95.7	0.95
	NTP	98.5	6.2	96.8	0.98
	TCP	98.3	6.2	96.6	0.97
$S3$	**DNS**	94	6.2	93.9	0.94
	NTP	93.7	5.2	94.1	0.94
	TCP	95.7	4.3	95.7	0.96

Simulation results show that Scheme A gives better detection performance in $S1$ when compared with $S3$. The reason is that $S1$ is the main switch on this simulation setup and it has a higher traffic volume than $S3$. Nonetheless, it has been showed that the detection performance of Scheme A decreased from 98.8 to 96.6 for TCP SYN attack, which was the only attack that we used for the simulations of Scheme A in

Section 4. We give more details on this decrease in the following paragraph while comparing the detection performance of all schemes.

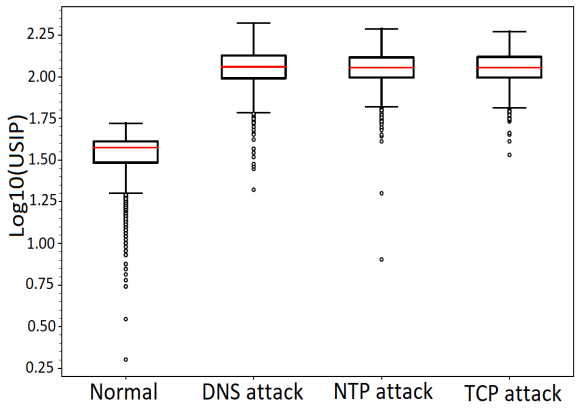
The overall detection performance of all schemes is given in Table 7.2. According to these results, Scheme B outperforms other schemes in terms of the detection performance for all TPr, FPr and ACC metrics. In $S1$, the Scheme A provides better TPr result than Scheme C. However, Scheme C has better detection accuracy than Scheme A. As a result, the detection performance of Scheme A decreases when it is used for larger networks.

Table 7.2. DDoS detection performance comparison between three proposed schemes for SDN.

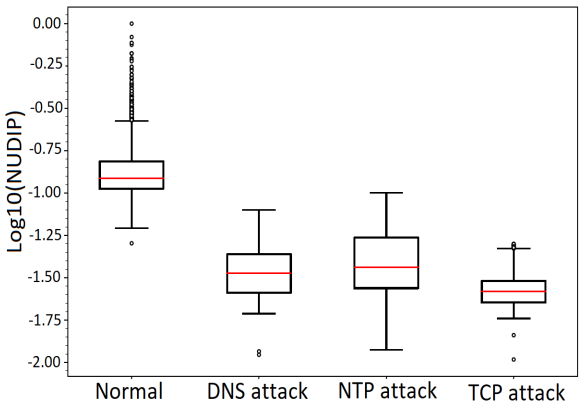
Switch	**Scheme**	**TPr (%)**	**FPr (%)**	**ACC (%)**	F1_score
$S1$	**Scheme A**	98.57	7.30	96.37	0.97
	Scheme B	100	0.05	99.9	0.99
	Scheme C	93.57	1.56	97.57	0.98
$S3$	**Scheme A**	94.48	5.24	94.58	.95
	Scheme B	100	0	100	1
	Scheme C	96.16	0.07	99.25	0.99

Following our discussion, to justify this result of Scheme A, we present the boxplot charts of USIP and NUDIP statistics of $S1$ and $S3$ in Figure 7.1. Although USIP and NUDIP of normal traffic have different distributions, they overlap with the statistics of DDoS attacks. Additionally, USIP and NUDIP have the similar patterns for all types of attacks. Therefore, using the USIP and NUDIP values in time-domain as the feature to discriminate attack from normal samples would yield high false alarm errors. One solution could be to modify Scheme A and to transform USIP and NUDIP from time-domain into frequency-domain. However, this modification has a significant overhead in terms of finding the optimal threshold value, re-selection of suitable features and the training of the ARIMA model. Additionally, we already designed Scheme B and Scheme C to operate on frequency domain. Therefore, we left the modification of Scheme A as a future work. To conclude, according to our detection performance comparison, we can say that if both attack and normal traffic share the similar patterns in the time-domain, they may have different patterns in the frequency-domain or conversely. Therefore, a

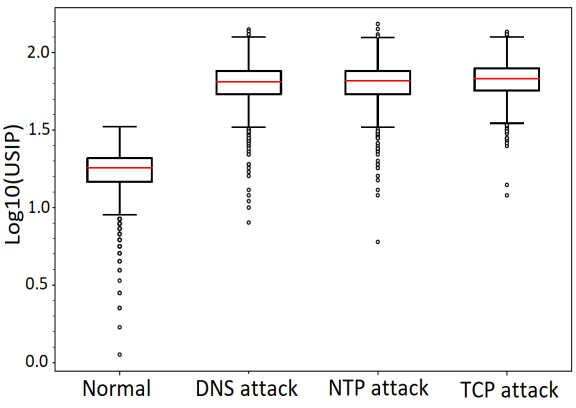
DDoS detection scheme that uses time-domain and frequency domain interchangeable can provide a solid detection mechanism different networks with different topologies.



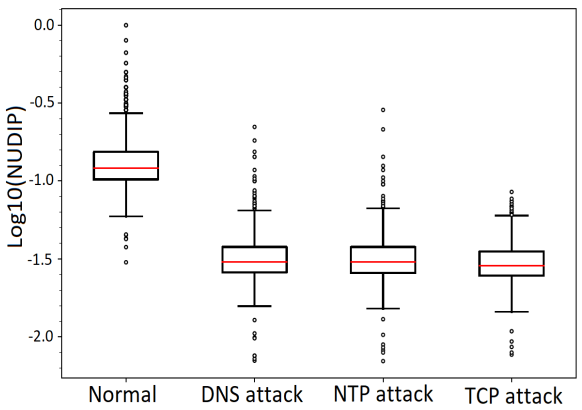
(a) USIP (S1).



(b) NUDIP (S1).

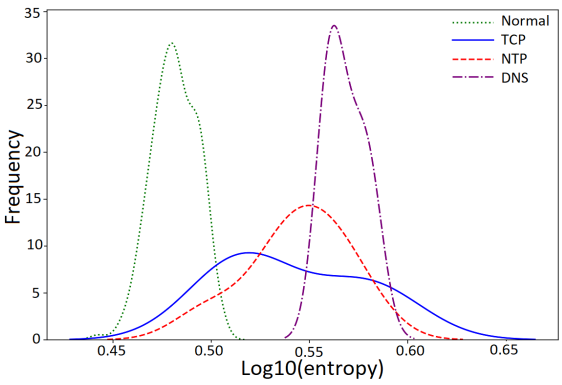


(c) USIP (S3).

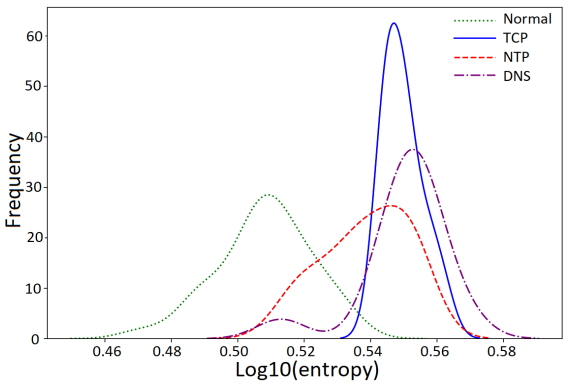


(d) NUDIP (S3).

Figure 7.1. Boxplot chart of USIP and NUDIP for the network traffic.



(a) S1.



(b) S3.

Figure 7.2. Empirical distribution of entropy of USIP for normal and attack samples for S1 and S3.

Finally, we also compare the proposed schemes with the entropy-based approaches, which are the most popular approaches in literature [120–122]. In order to compare the performance, the entropy of USIP time-series is obtained for $S1$ and $S3$ switches. The empirical probability distribution of entropy values for attack and normal traffic are shown in Figure 7.2. Although the entropy of DNS attack is separated from normal traffic in $S1$, it is overlapped with normal traffic in S3. Table 7.3 summarizes the DDoS attack detection performance for entropy-based approach. Because the entropy of attack and normal traffic overlap with each other, the DDoS attack detection performance of entropy-based approach is less than our proposed schemes.

Table 7.3. DDoS attack detection performance using the entropy of USIP.

Switch	**TPr (%)**	**FPr (%)**	**ACC (%)**	**F1_score**
S1	69.74	6.75	81.5	0.79
S3	71.81	11.92	79.94	0.78

7.2. Computational Complexity Cost Analysis

This section compares the performance of proposed schemes for DDoS attack detection for SDN with respect to the computational complexity cost analysis. First, we give the detailed computational complexity cost analysis of each scheme. Then, we compare these schemes with respect to their computation complexity costs.

7.2.1. Analysis of Scheme A

As defined in Section 4.1, Scheme A consists of the algorithms below:

- Algorithm 4.1 (Figure 4.2). *Extract Statistic Features (USIP and NUDIP):* The algorithm is used to extract statistics including USIP (x_t) and NUDIP (y_t) from the flow table of an OpenFlow switch with r entries for each time interval t.
- Algorithm 4.2 (Figure 4.3). *Anomaly Score Computation for Unique Source IP Addresses:* The algorithm is used to generate ARIMA model (training phase) for USIP and then use the model to obtain a binary anomaly score ($score_{1,t}$) assigned to x_t.

- Algorithm 4.3 (Figure 4.4). *Anomaly Score Computation for Normalized Unique Destination IP:* The algorithm used NUDIP to generate a dynamic thresholding model (training phase) and then applies the dynamic threshold threshold to assign binary anomaly score ($score_{2,t}$) to y_t.
- Algorithm 4 (Figure 4.5). *DDoS Detection:* The algorithm assigns DDoS or normal labels to each traffic samples by multiplying $score_{1,t}$ and $score_{2,t}$.

During the computational complexity cost analysis of Scheme A, we first analyze the computational complexity costs of each algorithm then we give the total computational complexity cost of the scheme in terms of finding binary anomaly scores, based on ARIMA modeling and dynamic thresholding method, and DDoS detection. The detailed analysis is as follows:

- Algorithm 4.1 (Figure 4.2):
  - Training phase: Algorithm 4.1 extracts USIP and NUDIP using hash table indexing. Considering the computational complexity cost of hash table indexing as $O(1)$ and r entries in the flow table, the computational complexity cost of Algorithm 4.1 is $O(r)$.
  - Detection phase: Algorithm 4.1 is employed during the detection phase with the same computational complexity cost of $O(r)$.
- Algorithm 4.2 (Figure 4.3):
  - Training phase: the training phase of ARIMA contains generating $\mathcal{X}$ time-series from k samples of $x_t \in \mathcal{X}$ and ARIMA modeling. While the complexity of $\mathcal{X}$ is $O(rk)$, the complexity of ARIMA modeling which deals with the identification of its parameters and model estimation is $O(k^2)$ [164]. Therefore, the computational complexity cost of the training phase of Algorithm 4.2 is $O(k^2 + rk)$.
  - Detection phase: the computational complexity cost of the detection is $O(r)$.
- Algorithm 4.3 (Figure 4.4):
  - Training phase: time-series $\mathcal{Y}$, $\mathcal{D}_f$, $\mathcal{M}$ and min_dist are employed in the training phase of the adaptive thresholding method in Algorithm 4.3. Gen-

erating $\mathcal{Y}$ from k samples of $y_t \in \mathcal{Y}$ has the computational complexity cost of $O(rk)$. The $\mathcal{D}_f$, which is the time-series of the difference between two exponential filters (EXP_f1 and EXP_f2), has the complexity of $O(k)$. The calculation of $\mathcal{M}$ and min_dist require the use of quicksort [165]. Therefore, we just consider the average computational complexity cost of quicksort, which is $O(wlogw)$ The computational complexity cost of $\mathcal{M}$ and min_dist are $O((k-w) \times wlogw)$ and $O((k-w) \times (k-w)log(k-w))$, respectively, for a rolling median window with the size of w. Moreover, both the mean value and standard deviation computation parts have the computational complexity cost of $O(k-w)$. Therefore, the training computational complexity cost of Algorithm 4.4.3 is $O(rk + k + (k-w) + (k-w) \times wlogw + (k-w) \times (k-w)log(k-w))$ which is simplified to $O(rk + k^2logk)$ by assuming $w << k$.

  - Detection phase: the computation of $score_{2,t}$ during the detection phase includes the median, minimum, mean and standard deviation calculations. Therefore, the computational complexity cost is $O(wlogw + (k-w)log((k-w)))$ which is simplified to $O(klogk)$.

- Algorithm 4.4 (Figure 4.5): This algorithm is just used during the detection phase with the computational complexity cost of $O(1)$.
- Finally, the total computational complexity cost of Scheme A is $O(k^2 + rk + k^2logk) \approx O(rk + k^2logk)$ and $O(r + klogk)$ for training and detection phases, respectively.

7.2.2. Analysis of Scheme B

As defined in Section 5.1, Scheme B consists of the algorithms below:

- Algorithm 5.1 (Figure 5.2). *Extract Statistic Features (USIP, totalPacket and $avg_{hit-rate_t}$):* The algorithm is used to extract statistics including USIP (x_t), totalPacket (z_t) and average hit rate ($avg_{hit-rate_t}$) from the flow table of an OpenFlow switch with r entries for each time interval t.
- Algorithm 5.2 (Figure 5.3). *DWT-based Feature Extraction:* The algorithm is

used to obtain statistical features from Discrete Wavelet Transform (DWT) of the uni-variate time-series (Z) generated from USIP and totalPacket time-series. The algorithm includes training and detection phases. In training phase, the vector ($Eigvec_{max}$) for dimension reduction is obtained. During the detection phase, statistical features are extracted.

- Algorithm 5.3 (Figure 5.4). *Auto-encoder Neural Network-based DDoS Detection:* The algorithm checks the value of $avg_{hit-rate_t}$ to activate *DDoS Detection* module. An auto-encoder model obtained during training phase from normal traffic samples is employed to receive statistical feature vector as the input and replicates it as the output. The distance between input and output is used to find DDoS samples.

During the computational cost analysis of Scheme B, the computational costs of each algorithm is given then we give the total computational cost of the scheme in terms of extracting DWT-based statistical features and anomaly detection based on auto-encoder neural network. The detailed analysis is as follows:

- Algorithm 5.1 (Figure 5.2):
  - Training phase: Algorithm 5.1 uses hash indexing to extract x_t, z_t and $avg_{hit-rate_t}$ from the flow table of a switch with r entries. Therefore the computational complexity cost of Figure 5.2 is $O(r)$.
  - Detection phase: detection phase has the same computational complexity cost as training phase as $O(r)$.
- Algorithm 5.2 (Figure 5.3)
  - Training phase: Algorithm 5.2 extracts statistical features from DWT applied on Z. Let w be the size of the window, then the computational complexity cost of generating Z and DWT are $O(rw)$ and $O(w)$, respectively. Since the computational complexity cost of the mean, variance, kurtosis, skewness and entropy is $O(max\_level \times \frac{w}{2})$, the computational complexity cost of calculating percentiles is $O(max\_level \times \frac{w^2}{4}) \approx O(max\_level \times w^2)$ where max_level is the maximum number of sub-bands in DWT. The per-

centiles are calculated using quicksort method [165] with the average computational complexity cost of $O(wlogw)$. Therefore, the total complexity is $O(wr + w^2 + wlogw) \approx O(wr)$, by assuming $r >> w$ and $w >> max\_level$. Moreover, Algorithm 5.2 applies PCA on $\Upsilon = \{(x_1, z_1), \ldots (x_k, z_k)\}$ to find $Eigvec_{max}$. The total computational cost of generating Υ and PCA are $O(rk)$ and $O(k^3)$, respectively. Once $Eigvec_{max}$ is obtained, its product with Υ is used to generate $\mathcal{Z}_T$. $\mathcal{Z}_T$ is divided into $(k - w + 1)$ subsets with the size of w, then DWT is applied on each subset and corresponding statistical features are obtained which results in computational cost complexity of $O((k - w + 1) \times wr) \approx O(kwr)$. As a result, Algorithm 5.2 has the computational cost complexity of $O(rk + k^3 + kwr) \approx O(k^3 + kwr)$ for the training phase.

  - Detection phase: during detection phase, Algorithm 5.2 applies all procedures mentioned in the training phase except PCA procedure. Therefore, Algorithm 5.2 has $O(wr)$ computational complexity cost.

- Algorithm 5.3 (Figure 5.4):
  - Training phase: during the training phase, $(k-w+1)$ feature vectors are used to train the auto-encoder neural network. The auto-encoder neural network training is based on back-propagation method; therfore, its computational cost complexity is $O(e \times (\text{k-w+1}) \times n^4) \approx O(e \times k \times n^4)$ where e and n are the number of epochs and size of feature vector, respectively.
  - Detection phase: Algorithm 5.3 uses pre-trained auto-encoder neural network (Figure 5.6) to replicate the feature vector by using feed forward procedure. The computational cost complexity of the feed forward procedure depends on the matrix multiplication. As a result the computational complexity cost is $O(n^3)$.
- The overall computational cost complexity of Scheme B is $O(k^3+kwr+e\times k\times n^4)$ and $O(wr + n^3)$ for training and detection phases, respectively.

7.2.3. Analysis of Scheme C

As defined in Section 6.1, Scheme B consists of Algorithm 4.1 (Figure 4.2) and two algorithms below:

- Algorithm 6.2 (Figure 6.2). *CWT-based Feature Extraction:* The algorithm is used to apply Continuous Wavelet Transform (CWT) on USIP ($\mathcal{X}$) and NUDIP ($\mathcal{Y}$) time-series and obtain two 2D features,Sc_{x_t} and Sc_{y_t}, respectively for each time interval t. Moreover, the median of $\mathcal{X}$ ($USIP_{med_t}$) is calculated.
- Algorithm 6.3 (Figure 6.4). *CNN-based DDoS Detection:* The algorithm checks the value of $USIP_{med_t}$ to activate *DDoS Detection* module. A CNN model obtained during training phase is employed to classify traffic sample as attack or normal traffic by using Sc_{x_t} and Sc_{y_t}.

During the computational cost analysis of Scheme C, the computational costs of each algorithm is given then we give the total computational cost of the scheme in terms of obtaining CWT-based features and DDoS detection based on CNN architecture. The detailed analysis is as follows:

- Scheme C adopts Algorithm 4.1 (Figure 4.2) to extract statistics; therefore, the computational complexity cost is $O(r)$ for training and detection phases.
- Algorithm 6.2 (Figure 6.2):
  - Training phase: Algorithm 6.2 extracts CWT features and the median of the USIP time-series, $\mathcal{X}$, where $|\mathcal{X}| = w$. Therefore, the performing of CWT has $O(w \log w)$ computational complexity cost. Additionally, as defined in [165], the median calculation has the average computational complexity cost of $O(wlogw)$. Thus, the total computational complexity cost of Figure 6.2 is $O(w^2 + w \log w + wr) \approx O(wr)$, where $r >> w$. During the training $(k - w + 1)$ samples with the size of w are employed and the corresponding CWT features are estimated. Therefore, computational complexity cost of Figure 6.2 is $O((k - w + 1) \times wr) \approx O(kwr)$ during the training phase.

  - Detection phase: Algorithm 6.2 applies the same procedure as the training phase, except that in detection phase instead of k samples we have one sample; therefore, the computational complexity cost is $O(wr)$.

- Algorithm 6.3 (Figure 6.4)
  - Training phase: the detection in Algorithm 6.3 employs a dedicated CNN architecture (given in Table 4), where the convolution layer computations, particularly the computation of the first convolution layer (Conv_1) is the most expensive operation in terms of the computational complexity cost. Therefore, for Algorithm 6.3, we only consider the computations in the first convolution layer while analyzing the computational complexity cost. Let $Sc_t \in \mathbb{R}^{w \times w \times 2}$ be the input obtained from the Algorithm 6.2, for a 2D kernel with the size of $\kappa \times \kappa$, the convolution has the computational complexity cost of $O(w^2\kappa^2)$ [166]. Moreover, by taking into the account l_w and l_k as the general number of input channels and number of kernels, respectively, the total computational complexity cost of convolution is $O(l_w l_k w^2 \kappa^2)$. Since κ, l_k and l_w have relatively small values and they are constant for a given CNN, we can approximate the computational complexity cost as $O(w^2)$. Although there are other layers such as MaxPooling layers, activation functions, the second convolution layer (Conv_2) and fully-connected layers (consist of only vector-matrix multiplications), the upper bound for the computational complexity cost of CNN is $\approx O(w^2)$.
  - Detection phase: the most time consuming procedure in the proposed scheme is the training phase of the CNN because of using extra back propagation procedure. Hence, the execution of the model with back propagation is order of $O(w^3)$ and the total cost is $O(e \times (k - w + 1) \times w^3) \approx O(e \times k \times w^3)$, where e is the number of epochs
- The overall computational cost complexity of Scheme C is $O(kwr + e \times k \times w^3)$ and $O(wr + w^2)$ for training and detection phases, respectively.

7.3. Computational Complexity Cost Comparison

We compare the computational complexity costs of schemes as illustrated in Table 7.4 by considering the following assumptions below during the computational complexity cost calculations:

(i) $r >> w$, where r and w are the number of flow entries in the flow table and the window size, respectively.

(ii) $k >> w$, where k is the number of training samples.

(iii) $w >> max\_level$, where max_level is the maximum number of sub-bands in DWT.

(iv) e is an arbitrary constant that shows the number of iterations to be used while training the NN model.

(v) Please note that the training phase is executed only once while deploying the scheme to the controller.

(vi) The number of training samples are much greater than the number of flow entries ($k >> r$). Each flow entry in the flow table of a switch has an idle timeout and or a hard timeout associated with it according to the OpenFlow switch definition in [115]. If no packet is matched by the flow entry for that specific idle timeout, the flow is removed from the flow table. On the other hand, the switch should remove the flow entry after the time specified by hard timeout, regardless of how many packets it is matched. Considering idle timeout and hard timeout, we can conclude that flow tables of switches have limited number of flow entries (r) for each time interval t.

(vii) $n = 40$, where n is the size of feature vectors used by the auto-encoder neural network as described in Section 5.2.2.

By employing the aforementioned assumptions, the computational complexity cost of schemes can be simplified as follows:

- Considering the assumption (vi) the computational complexity cost of Scheme A can be simplified as $O(k^2 logk)$ and $O(klogk)$ for training and detection phases, respectively.

Table 7.4. Computational complexity cost comparison between three proposed schemes for SDN.

Scheme	**Training Cost**	**Detection Cost**	**Simplified Training Cost**	**Simplified Detection Cost**
Scheme A	$O(rk + k^2logk)$	$O(r + klogk)$	$O(k^2logk)$	$O(klogk)$
Scheme B	$O(k^3 + kwr + e \times k \times n^4)$	$O(wr + n^3)$	$O(k^3)$	$O(r)$
Scheme C	$O(kwr + e \times k \times w^3)$	$O(wr)$	$O(kr)$	$O(r)$

- Considering the assumptions (i, ii, iii, iv, vi and vii), the computational complexity cost of Scheme B can be simplified as $O(k^3)$ and $O(r)$ for training and detection phases, respectively.
- Considering the assumptions (i, ii, iii, iv, vi and vii), the computational complexity cost of Scheme C can be simplified as $O(kr)$ and $O(r)$ for training and detection phases, respectively.

According to training phase comparison in Table 7.4, Scheme C has the best computational complexity cost performance followed by Scheme A and then Scheme B. Although the computational cost complexity of Scheme C during the training phase depends on various parameters as shown in Table 7.4, by applying the aforementioned assumptions and considering the CNN architecture that have been used (Table 6.2) in this study, the computational complexity cost of this scheme can be simplified to $O(kr)$. In Scheme A, the overall computational complexity cost of the training phase depends on median-based approach which uses quicksort algorithm [165] with the average computational complexity cost of $O(klogk)$. Because this median-based approach needs to be repeated for k times during the training phase, the final computational complexity cost of Scheme A is $O(k^2logk)$. Taken into account the aforementioned assumptions, the computational complexity cost of Scheme B mainly depends on PCA with computational complexity cost of $O(k^3)$, which causes Scheme B to have the worst computational complexity cost.

For the detection phase, the best computational complexity cost performance belongs to Scheme B and Scheme C, and Scheme A has the worst computational com-

plexity cost performance. As we already described for the training phase, in Algorithm 4.3 (Figure 4.4), the detection phase Scheme A also needs to apply median-based dynamic threshold method in order to update the associated threshold value (th_{med}) which depends on k latest values of feature samples with $O(klogk)$ computational complexity cost performance. On the other hand, Scheme B and Scheme C use pre-trained neural network models during the detection process which just need a feature sample to be classified. Since most of the operations of either auto-encoder or convolutional neural network requires matrix multiplication as the most computationally intensive operation, Scheme B an Scheme C has the best performance in terms of the computational complexity cost.

8. CONCLUSION

In Distributed Denial of Service (DDoS) attacks, the main purpose is the degradation of the online service availability. In general, attackers use distributed and connected devices to the Internet and masquerade source IP addresses of those devices to flood the targeted network or server(s). Previously, DDoS attacks were a significant threat for the traditional networks. With the emerging use of Software Defined Networks (SDNs), it has been shown in the recent studies that such attacks are also very important threat for SDNs. For this respect, In this thesis, we have proposed DDoS detection methods for traditional networks SDNs.

8.1. Contributions of the Thesis

Regarding the DDoS attack detection for traditional network managements, we have proposed three methods which the summary of each methods is described as follows:

- First, we have proposed a DDoS detection method based on frequency domain analysis and naïve Bayes classifier. The number of packets for each time interval $t = 1$ ms have been used as the time-series. DFT and DWT have been applied on the time-series to extract discriminating features. By analyzing the energy distribution of DFT and DWT for both attack and normal traffic, the number of features has been chosen manually and then being used by the naïve Bayes classifier. Using DFT features has resulted in higher accuracy compared to the features obtained from DWT; however, the combination of two feature sets has increased the accuracy of detection (95.93%). Additionally, a simple thresholding method has been also implemented which again the DFT has outperformed with respect to accuracy (85.32%). Although the accuracy of thresholding method is less than that of naïve Bayes method, it could be used as the preliminary step of DDoS attack detection in an IDS.

- Next, we have investigated the use of statistical measures including periodicity, kurtosis, skewness and self-similarity for DDoS attack detection. The aforementioned measures have been extracted from the time-series obtained by sampling the number of arrival packets for each time interval $t = 1$ ms. The empirical probability distribution of each measure has been obtained and a threshold has been chosen to discriminate attack samples from normal ones. The performance of all four measures have been compared with respect to the accuracy in detection. The skewness measure outperforms other parameters in separating DDoS attack from normal traffic (98.33%). Kurtosis and Hurst exponent measures provide moderate performance for the detection DDoS attacks with the DDoS attack detection of (95.83%) and (91.66%), respectively.
- Finally, we have proposed an anomaly-based DDoS detection by using sparse coding and frequency domain analysis. A time-series has been generated by counting the number of arrival packets for each time interval $t = 1$ ms. The obtained time-series has been further divided into 128-length widows which each new window has shared first 64 samples with the previous one. The absolute value of the DFT of windows have been employed as the dataset for this work. While normal data has been divided into three parts of training, evaluation and test, the attack data has been kept for the test step. An over-complete dictionary has been generated by applying K-SVD algorithm on the training part of the normal dataset. For each dataset, five sparse coefficients have been estimated by using the obtained dictionary and BMP algorithm. The sparse coefficients of the training part of the normal data have been given as the input to the SOM algorithm to generate a SOM lattice. The minimum Euclidean distance between sparse coefficients of each evaluation samples and the neurons of the SOM lattice has been calculated and its empirical probability distribution has been used as the normal behavior. By comparing the minimum Euclidean distance between the coefficients of each samples in the test part and SOM lattice, the performance of the proposed method has been analyzed. Our experimental results have shown that the proposed algorithm provides 99.11% accuracy for DDoS attack detection
- Finally, we have compared the performance of three proposed schemes in terms

of DDoS attack detection accuracy. According to the results, Scheme 3 has the best performance followed by Scheme 2 and Scheme 1.

For SDN architectures, we have proposed three DDoS detection and defense schemes integrated into the controller of the SDN. Each scheme has monitored each OpenFlow switch individually to find any anomalies caused by DDoS attacks. The summary of each scheme is given as follows:

- In the first scheme, we have proposed a DDoS attack detection and countermeasure scheme based on time-series analysis for SDN. For this purpose, four modules including *Feature Extraction*, *Anomaly Detection for USIP*, *Anomaly Detection for NUDIP* and *DDoS Detection*, have been added to the SDN controller. In this work, our scheme has monitored the network traffic features to distinguish DDoS attack instances. For this purpose, we have considered both the number of unique source IP addresses and the normalized unique destination IP addresses to generate two anomaly scores. In anomaly score Computation for USIP, we have employed the ARIMA forecasting error and chaos theory. For anomaly score Computation for NUDIP, we have adopted the dynamic threshold method. Then, the results obtained from these two features have been used for labeling each traffic sample as normal or DDoS traffic. Moreover, if any DDoS attack instances has been detected in a switch, the DDoS attack alarm has been raised for that switch and the countermeasure module has updated the flow table of the switch accordingly to prevent the DDoS attack. The experimental results have shown that, the proposed scheme has achieved (98.82%) accuracy for DDoS attack detection.
- In the second scheme, we have proposed a DDoS attack detection and countermeasure scheme based on DWT and auto-encoder neural network for SDN. For this purpose, the proposed scheme has used the *statistics collection*, *feature extraction* and *DDoS Detection* modules. USIP and total number of packets statistics has been extracted from the flow table and then PCA has been applied to obtain a time-series. DWT has been applied on the time-series and for each

DWT sub-band various statistics including mean, variance, kurtosis, percentiles, entropy and Hurst parameters have been obtained and concatenated to generate a feature vector. Feature vector has been used as the input by the auto-encoder neural network and the network tries to replicate it as the output. The Euclidean distance between the input and the output of the auto-encoder neural network has been calculated and used to classify network traffic as DDoS or normal traffic. The proposed scheme has provided at least (99.9%) accuracy for DDoS attack detection.

- In the third scheme, we have proposed a DDoS attack detection and counter-measure scheme for SDN based on CWT and CNN classifier. The scheme has comprised three modules which are *statistics collection*, *feature extraction* and *attack detection* modules. On each w samples of two time-series obtained from USIP and NUDIP, CWT has been applied and two 2D variables with the size of $w \times w$ have been obtained. Then those outputs have been used by a CNN architecture to classify network traffic as DDoS or normal traffic. The experimental results have shown that the proposed scheme has achieved at least (97.57%) accuracy for DDoS attack detection.
- Finally, the detection performance of each schemes against DNS amplification, Network Time Protocol and TCP SYN flood attacks has been evaluated via simulation using GNS3 environment and Mininet emulator [43, 44]. According to the simulation results, Scheme B has outperformed. Moreover, the computational complexity cost of each scheme has been evaluated and compared. According to the results, for training phase, Scheme C and Scheme B have the lowest and highest computational cost, respectively. For detection phase, Scheme B and Scheme C share the same computational complexity cost and Scheme A has the highest computational complexity cost.

8.2. Future Research Directions

In order to improve the studies presented in this thesis, we will consider following topics as the future research directions:

- The first research direction is to test the detection performance of proposed schemes in this thesis with the new DDoS attack types such as the low-rate DDoS attacks. Those attacks use a low amount of traffic to overwhelm the victim in contrast to the conventional DDoS attack techniques that overwhelm the network by sending a large amount of fake traffic. Low-rate attacks often open a low number of connections on the target victim and leave connections open for a long period of time. Hence, the proposed schemes in this thesis can be extended for the detection of low-rate DDoS attacks.
- 5G technology and beyond provide users with ultra low latency, massive capacity, and more reliable connection. Therefore, the number of connected devices with different Quality of Service (QoS) requirements will continuously increase with the deployment of 5G networks. To deal with those heterogeneous services, slicing concept has been put forward and implemented in 5G technology and beyond. Although the security-related requirements for the slicing have been discussed and defined in security standards for 5G technology, the 5G core infrastructure is still vulnerable to cybersecurity threats such as DDoS attack. Due to the important role of this concept in 5G technology network, another possible future research direction can be to investigate the possible vulnerabilities of slicing in terms of DDoS attacks.
- Massive Machine Type Communications (mMTC) service provides connection to a large number of IoT devices that each IoT device transmits small amounts of traffic. The lack of security and privacy standardization in 5G systems puts those IoT devices at a great risk which can be compromised by the DDoS attackers. Consequently, the high volume of malicious traffic generated from those IoT devices can be used for attacking the online services. To prevent the potential damage caused by such kind of attacks, we need the detection and countermeasure schemes that operate on connections with small delay values. Investigation of the DDoS attack detection for 5G-enabled IoT device networks can also be considered as another future research direction.

REFERENCES

1. Rai, S., K. Sharma and D. Dhakal, "A Survey on Detection and Mitigation of Distributed Denial-of-Service Attack in Named Data Networking", *Advances in Communication, Cloud, and Big Data*, pp. 163–171, Springer, 2019.

2. Yang, Y., L. Wu, G. Yin, L. Li and H. Zhao, "A Survey on Security and Privacy Issues in Internet-of-Things", *IEEE Internet of Things Journal*, Vol. 4, No. 5, pp. 1250–1258, 2017.

3. Vormayr, G., T. Zseby and J. Fabini, "Botnet Communication Patterns", *IEEE Communications Surveys & Tutorials*, Vol. 19, No. 4, pp. 2768–2796, 2017.

4. Hodo, E., X. Bellekens, A. Hamilton, C. Tachtatzis and R. Atkinson, "Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey", *arXiv preprint arXiv:1701.02145*, 2017.

5. Depren, O., M. Topallar, E. Anarim and M. K. Ciliz, "An Intelligent Intrusion Detection System (IDS) for Anomaly and Misuse Detection in Computer Networks", *Expert systems with Applications*, Vol. 29, No. 4, pp. 713–722, 2005.

6. Ma, X. and Y. Chen, "DDoS Detection Method Based on Chaos Analysis of Network Traffic Entropy", *IEEE Communications Letters*, Vol. 18, No. 1, pp. 114–117, 2014.

7. Qin, X., T. Xu and C. Wang, "DDoS Attack Detection Using Flow Entropy and Clustering Technique", *Computational Intelligence and Security (CIS), 2015 11th International Conference on*, pp. 412–415, 2015.

8. Ubale, T. and A. K. Jain, "Survey on DDoS Attack Techniques and Solutions in Software-Defined Network", *Handbook of Computer Networks and Cyber Security*, pp. 389–419, Springer, 2020.

9. Gupta, S. and B. B. Gupta, "Detection, Avoidance, and Attack Pattern Mechanisms in Modern Web Application Vulnerabilities: Present and Future Challenges", *International Journal of Cloud Applications and Computing (IJCAC)*, Vol. 7, No. 3, pp. 1–43, 2017.

10. Mousavi, S. M. and M. St-Hilaire, "Early Detection of DDoS Attacks Against SDN Controllers", *2015 International Conference on Computing, Networking and Communications (ICNC)*, pp. 77–81, 2015.

11. Sahoo, K. S., D. Puthal, M. Tiwary, J. J. Rodrigues, B. Sahoo and R. Dash, "An Early Detection of Low Rate DDoS Attack to SDN Based Data Center Networks Using Information Distance Metrics", *Future Generation Computer Systems*, Vol. 89, pp. 685–697, 2018.

12. Kalkan, K., L. Altay, G. Gür and F. Alagöz, "JESS: Joint Entropy-Based DDoS Defense Scheme in SDN", *IEEE Journal on Selected Areas in Communications*, Vol. 36, No. 10, pp. 2358–2372, 2018.

13. Ahalawat, A., S. S. Dash, A. Panda and K. S. Babu, "Entropy Based DDoS Detection and Mitigation in OpenFlow Enabled SDN", *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, pp. 1–5, 2019.

14. AbdelAzim, N. M., S. F. Fahmy, M. A. Sobh and A. M. B. Eldin, "A Hybrid Entropy-Based DoS Attacks Detection System for Software Defined Networks (SDN): A Proposed Trust Mechanism", *Egyptian Informatics Journal*, 2020.

15. Galeano-Brajones, J., J. Carmona-Murillo, J. F. Valenzuela-Valdés and F. Luna-Valero, "Detection and Mitigation of DoS and DDoS Attacks in IoT-Based Stateful SDN: An Experimental Approach", *Sensors*, Vol. 20, No. 3, p. 816, 2020.

16. Da Silva, A. S., J. A. Wickboldt, L. Z. Granville and A. Schaeffer-Filho, "AT-

LANTIC: A Framework for Anomaly Traffic Detection, Classification, and Mitigation in SDN", *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, pp. 27–35, 2016.

17. Niyaz, Q., W. Sun and A. Y. Javaid, "A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN)", *CoRR*, Vol. abs/1611.07400, 2016.

18. Phan, T. V., T. Van Toan, D. Van Tuyen, T. T. Huong and N. H. Thanh, "Openflowsia: An Optimized Protection Scheme for Software-Defined Networks From Flooding Attacks", *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*, pp. 13–18, 2016.

19. Meti, N., D. Narayan and V. Baligar, "Detection of Distributed Denial of Service Attacks Using Machine Learning Algorithms in Software Defined Networks", *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1366–1371, 2017.

20. Nam, T. M., P. H. Phong, T. D. Khoa, T. T. Huong, P. N. Nam, N. H. Thanh, L. X. Thang, P. A. Tuan, V. D. Loi *et al.*, "Self-Organizing Map-Based Approaches in DDoS Flooding Detection Using SDN", *2018 International Conference on Information Networking (ICOIN)*, pp. 249–254, 2018.

21. Cui, J., M. Wang, Y. Luo and H. Zhong, "DDoS Detection and Defense Mechanism Based on Cognitive-Inspired Computing in SDN", *Future Generation Computer Systems*, Vol. 97, pp. 275–283, 2019.

22. Wang, Y., T. Hu, G. Tang, J. Xie and J. Lu, "SGS: Safe-Guard Scheme for Protecting Control Plane Against DDoS Attacks in Software-Defined Networking", *IEEE Access*, Vol. 7, pp. 34699–34710, 2019.

23. Li, C., Y. Wu, X. Yuan, Z. Sun, W. Wang, X. Li and L. Gong, "Detection and

Defense of DDoS Attack Based on Deep Learning in OpenFlow-Based SDN", *International Journal of Communication Systems*, Vol. 31, No. 5, p. e3497, 2018.

24. Rasool, R. U., U. Ashraf, K. Ahmed, H. Wang, W. Rafique and Z. Anwar, "Cyberpulse: A Machine Learning Based Link Flooding Attack Mitigation System for Software Defined Networks", *IEEE Access*, Vol. 7, pp. 34885–34899, 2019.

25. Haider, S., A. Akhunzada, I. Mustafa, T. B. Patel, A. Fernandez, K.-K. R. Choo and J. Iqbal, "A Deep CNN Ensemble Framework for Efficient DDoS Attack Detection in Software Defined Networks", *IEEE Access*, Vol. 8, pp. 53972–53983, 2020.

26. Zhao, R., R. Yan, Z. Chen, K. Mao, P. Wang and R. X. Gao, "Deep Learning and Its Applications to Machine Health Monitoring", *Mechanical Systems and Signal Processing*, Vol. 115, pp. 213–237, 2019.

27. Bawany, N. Z., J. A. Shamsi and K. Salah, "DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions", *Arabian Journal for Science and Engineering*, Vol. 42, No. 2, pp. 425–441, 2017.

28. Wanda, P., "A Survey of Intrusion Detection System", *International Journal of Informatics and Computation*, Vol. 1, No. 1, pp. 1–10, 2020.

29. Tan, P.-N., *Introduction to Data Mining*, Pearson Education India, 2018.

30. Fouladi, R. F., T. Seifpoor and E. Anarim, "Frequency Characteristics of DoS and DDoS Attacks", *Signal Processing and Communications Applications Conference (SIU), 2013 21st*, pp. 1–4, 2013.

31. Antoniou, A., *Digital Signal Processing*, McGraw-Hill, 2016.

32. Sundararajan, D., *Discrete Wavelet Transform: A Signal Processing Approach*, John Wiley & Sons, 2016.

33. Rish, I., "An Empirical Study of The Naive Bayes Classifier", *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, Vol. 3, pp. 41–46, IBM New York, 2001.

34. Van Trees, H. L., *Detection, Estimation, and Modulation Theory, Part I: Detection, Estimation, and Linear Modulation Theory*, John Wiley & Sons, 2004.

35. Shanmugam, R. and R. Chattamvelli, *Statistics for Scientists and Engineers*, John Wiley & Sons, Incorporated, 2015.

36. Qian, B. and K. Rasheed, "Hurst Exponent and Financial Market Predictability", *IASTED Conference on Financial Engineering and Applications*, pp. 203–209, Proceedings of the IASTED International Conference Cambridge, MA, 2004.

37. Olshausen, B. A. and D. J. Field, "Sparse Coding with An Overcomplete Basis Set: A Strategy Employed by V1?", *Vision Research*, Vol. 37, No. 23, pp. 3311–3325, 1997.

38. Kohonen, T., "The Self-Organizing Map", *Proceedings of The IEEE*, Vol. 78, No. 9, pp. 1464–1480, 1990.

39. Hyndman, R. J. and G. Athanasopoulos, *Forecasting: Principles and Practice*, OTexts, 2018.

40. Yu, J., X. Zheng and S. Wang, "A Deep Autoencoder Feature Learning Method for Process Pattern Recognition", *Journal of Process Control*, Vol. 79, pp. 1–15, 2019.

41. Daubechies, I., "The Wavelet Transform, Time-Frequency Localization and Signal Analysis", *IEEE Transactions on Information Theory*, Vol. 36, No. 5, pp. 961–1005, 1990.

42. Alpaydin, E., *Introduction to Machine Learning*, MIT Press, 2020.

43. Welsh, C., *GNS3 Network Simulation Guide*, Packt Publ., 2013.

44. Yan, J. and D. Jin, "VT-Mininet: Virtual-Time-Enabled Mininet for Scalable and Accurate Software-Define Network Emulation", *Proceedings of The 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, pp. 27:1–27:7, 2015.

45. Fouladi, R. F., C. E. Kayatas and E. Anarim, "Frequency Based DDoS Attack Detection Approach Using Naive Bayes Classification", *2016 39th International Conference on Telecommunications and Signal Processing (TSP)*, pp. 104–107, 2016.

46. Fouladi, R. F., C. E. Kayatas and E. Anarim, "Statistical Measures: Promising Features for Time Series Based DDoS Attack Detection", *Multidisciplinary Digital Publishing Institute Proceedings*, Vol. 2, p. 96, 2018.

47. Fouladi, R. F., O. Ermiş and E. Anarim, "Anomaly-Based DDoS Attack Detection by Using Sparse Coding and Frequency Domain", *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1–6, 2019.

48. Fouladi, R. F., O. Ermiş and E. Anarim, "A DDoS Attack Detection and Defense Scheme Using Time-Series Analysis for SDN", *Journal of Information Security and Applications*, Vol. 54, p. 102587, 2020.

49. Fouladi, R. F., O. Ermiş and E. Anarim, "A Novel Approach for Distributed Denial of Service Defense Using Continuous Wavelet Transform and Convolutional Neural Network for Software-Defined Network", *Submitted to Elsevier Computers & Security Journal (Minor Revision)*, 2021.

50. Fouladi, R. F., O. Ermiş and E. Anarim, "A DDoS Attack Detection and Countermeasure Scheme Based on DWT and Auto-Encoder Neural Network for SDN",

Submitted to Elsevier Computer Networks Journal (Under Review), 2021.

51. Ghali, A. A., R. Ahmad and H. S. A. Alhussian, "Comparative Analysis of DoS and DDoS Attacks in Internet of Things Environment", *Computer Science Online Conference*, pp. 183–194, Springer, 2020.

52. Chauhan, K. and V. Prasad, "Distributed Denial of Service (DDoS) Attack Techniques and Prevention on Cloud Environment", *International Journal of Innovations & Advancement in Computer Science*, Vol. 4, pp. 210–215, 2015.

53. Vishwakarma, R. and A. K. Jain, "A Survey of DDoS Attacking Techniques and Defence Mechanisms in The IoT Network", *Telecommunication Systems*, Vol. 73, No. 1, pp. 3–25, 2020.

54. Prasad, K., A. Reddy and K. Rao, "Discriminating DDoS Attack Traffic From Flash Crowds on Internet Threat Monitors (ITM) Using Entropy Variations", *African Journal of Computing & ICT*, Vol. 6, No. 2, 2013.

55. Wu, Y.-C., H.-R. Tseng, W. Yang and R.-H. Jan, "DDoS Detection and Traceback with Decision Tree and Grey Relational Analysis", *International Journal of Ad Hoc and Ubiquitous Computing*, Vol. 7, No. 2, pp. 121–136, 2011.

56. Ramanauskaite, S. and A. Cenys, "Taxonomy of DoS Attacks and Their Countermeasures", *Central European Journal of Computer Science*, Vol. 1, No. 3, pp. 355–366, 2011.

57. Specht, S. and R. Lee, "Taxonomies of Distributed Denial of Service Networks, Attacks, Tools and Countermeasures", *CEL2003-03, Princeton University, Princeton, NJ, USA*, 2003.

58. Van Loon, R. and J. Lo, "An IRC Tutorial", *Internet Relay Chat (IRC) Help*, 1997.

59. Maheshwari, R., P. Scholar and C. R. Krishna, "Mitigation of DDoS Attacks Using Probability Based Distributed Hop Count Filtering and Round Trip Time", *International Journal of Engineering*, Vol. 2, No. 7, 2013.

60. G, D. K., C. G. Rao, M. W. Ahmed and S. G, "Multilayered Probabilistic Model for Distributed DoS Attacks", *2013 13th International Conference on Computational Science and Its Applications*, pp. 99–104, 2013.

61. Mirkovic, J. and P. Reiher, "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms", *ACM SIGCOMM Computer Communication Review*, Vol. 34, No. 2, pp. 39–53, 2004.

62. Singh, N., A. Dumka and R. Sharma, "Comparative Analysis of Various Techniques of DDoS Attacks for Detection & Prevention and Their Impact in MANET", *Performance Management of Integrated Systems and Its Applications in Software Engineering*, pp. 151–162, Springer, 2020.

63. Odusami, M., S. Misra, O. Abayomi-Alli, A. Abayomi-Alli and L. Fernandez-Sanz, "A Survey and Meta-Analysis of Application-Layer Distributed Denial-of-Service Attack", *International Journal of Communication Systems*, Vol. 33, No. 18, p. e4603, 2020.

64. Shinde, P. and S. Guntupalli, "Early DoS Attack Detection Using Smoothened Time-Series and Wavelet Analysis", *Third International Symposium on Information Assurance and Security*, pp. 215–220, 2007.

65. Hamilton, J. D., *Time Series Analysis*, Princeton University Press, 2020.

66. Cowpertwait, P. S. and A. V. Metcalfe, *Introductory Time Series with R*, Springer Science & Business Media, 2009.

67. Bornmann, L., "How to Analyse Percentile Impact Data Meaningfully in Bibliometrics: The Statistical Analysis of Distributions, Percentile Rank Classes and

Top-Cited Papers", *Journal of The Association for Information Science & Technology*, Vol. 64, No. 3, pp. 587–595, 2013.

68. Wu, Q. and Z. Shao, "Network Anomaly Detection Using Time Series Analysis", *Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services-(ICAS-ICNS-2005.69)*, pp. 42–42, 2005.

69. Brandes, O., J. Farley, M. Hinich and U. Zackrisson, "The Time Domain and The Frequency Domain in Time Series Analysis", *The Swedish Journal of Economics*, pp. 25–42, 1968.

70. Hyndman, R., A. B. Koehler, J. K. Ord and R. D. Snyder, *Forecasting with Exponential Smoothing: The State Space Approach*, Springer Science & Business Media, 2008.

71. Perraudin, N. and P. Vandergheynst, "Stationary Signal Processing on Graphs", *IEEE Transactions on Signal Processing*, Vol. 65, No. 13, pp. 3462–3477, 2017.

72. Rosenstein, M. T., J. J. Collins and C. J. De Luca, "A Practical Method for Calculating Largest Lyapunov Exponents From Small Data Sets", *Physica D: Nonlinear Phenomena*, Vol. 65, No. 1-2, pp. 117–134, 1993.

73. Nezhad, S. M. T., M. Nazari and E. A. Gharavol, "A Novel DoS and DDoS Attacks Detection Algorithm Using ARIMA Time Series Model and Chaotic System in Computer Networks.", *IEEE Communications Letters*, Vol. 20, No. 4, pp. 700–703, 2016.

74. Bracewell, R. N. and R. N. Bracewell, *The Fourier Transform and Its Applications*, Vol. 31999, McGraw-Hill New York, 1986.

75. Sandifer, C. E., *The Early Mathematics of Leonhard Euler*, Vol. 98, American Mathematical Society, 2020.

76. Mateo, C. and J. A. Talavera, "Short-Time Fourier Transform with The Window Size Fixed in The Frequency Domain (STFT-FD): Implementation", *SoftwareX*, Vol. 8, pp. 5–8, 2018.

77. Harti, A., "Discrete Multi-Resolution Analysis and Generalized Wavelets", *Applied Numerical Mathematics*, Vol. 12, No. 1-3, pp. 153–192, 1993.

78. Papandreou-Suppappola, A., *Applications in Time-Frequency Signal Processing*, CRC Press, 2018.

79. Vetterli, M., "Subband Coding", *Subband Image Coding*, Vol. 115, p. 43, 2013.

80. Oppenheim, A. V., *Discrete-Time Signal Processing*, Pearson Education India, 1999.

81. Kohonen, T., "Essentials of The Self-Organizing Map", *Neural Networks*, Vol. 37, pp. 52–65, 2013.

82. Ruder, S., "An Overview of Gradient Descent Optimization Algorithms", *CoRR*, Vol. ABS/1609.04747, 2016.

83. Alloghani, M., D. Al-Jumeily, J. Mustafina, A. Hussain and A. J. Aljaaf, "A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science", *Supervised and Unsupervised Learning for Data Science*, pp. 3–21, Springer, 2020.

84. Bisong, E., "Autoencoders", *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, pp. 475–482, Springer, 2019.

85. Bengio, Y., *Learning Deep Architectures for AI*, Now Publishers Inc, 2009.

86. Agarap, A. F., "Deep Learning Using Rectified Linear Units (Relu)", *CoRR*, Vol. ABS/1803.08375, 2018.

87. Murray, N. and F. Perronnin, "Generalized Max Pooling", *Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2473–2480, 2014.

88. Bisong, E., "Principal Component Analysis (PCA)", *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, pp. 319–324, Springer, 2019.

89. Nevin, J. A., "Signal Detection Theory and Operant Behavior: A Review of David M. Green and John A. Swets' Signal Detection Theory and Psychophysics.", *Journal of The Experimental Analysis of Behavior*, Vol. 12, No. 3, pp. 475–480, 1969.

90. Yaacob, A. H., I. K. Tan, S. F. Chien and H. K. Tan, "ARIMA Based Network Anomaly Detection", *2010 Second International Conference on Communication Software and Networks*, pp. 205–209, 2010.

91. Nezhad, S. M. T., M. Nazari and E. A. Gharavol, "A Novel DoS and DDoS Attacks Detection Algorithm Using ARIMA Time Series Model and Chaotic System in Computer Networks", *IEEE Communications Letters*, Vol. 20, No. 4, pp. 700–703, 2016.

92. Ni, T., X. Gu, H. Wang and Y. Li, "Real-Time Detection of Application-Layer DDoS Attack Using Time Series Analysis", *Journal of Control Science and Engineering*, Vol. 2013, p. 4, 2013.

93. Sangkatsanee, P., N. Wattanapongsakorn and C. Charnsripinyo, "Practical Real-Time Intrusion Detection Using Machine Learning Approaches", *Computer Communications*, Vol. 34, No. 18, pp. 2227–2235, 2011.

94. Sofi, I., A. Mahajan and V. Mansotra, "Machine Learning Techniques Used for The Detection and Analysis of Modern Types of DDoS Attacks", *learning*, Vol. 4,

No. 6, pp. 1085–1093, 2017.

95. Kumar, V. and H. Sharma, "Detection and Analysis of DDoS Attack at Application Layer Using Naïve Bayes Classifier", *Journal of Computer Engineering & Technology*, Vol. 9, No. 3, pp. 208–217, 2018.

96. Alkasassbeh, M., G. Al-Naymat, A. Hassanat and M. Almseidin, "Detecting Distributed Denial of Service Attacks Using Data Mining Techniques", *International Journal of Advanced Computer Science and Applications*, Vol. 7, No. 1, pp. 436–445, 2016.

97. Pan, W. and W. Li, "A Hybrid Neural Network Approach to The Classification of Novel Attacks for Intrusion Detection", *International Symposium on Parallel and Distributed Processing and Applications*, pp. 564–575, Springer, 2005.

98. Norouzian, M. R. and S. Merati, "Classifying Attacks in A Network Intrusion Detection System Based on Artificial Neural Networks", *13th International Conference on Advanced Communication Technology (ICACT2011)*, pp. 868–873, 2011.

99. Sastri, T., "An Adaptive Autoregressive Model", *Computers & Industrial Engineering*, Vol. 9, No. 1, pp. 9–27, 1985.

100. Scholkopf, B. and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT press, 2001.

101. Safavian, S. R. and D. Landgrebe, "A Survey of Decision Tree Classifier Methodology", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21, No. 3, pp. 660–674, 1991.

102. Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten, "The WEKA Data Mining Software: An Update", *ACM SIGKDD Explorations Newsletter*, Vol. 11, No. 1, pp. 10–18, 2009.

103. Han, J., J. Pei and M. Kamber, *Data Mining: Concepts and Techniques*, Elsevier, 2011.

104. Wen, T. and Z. Zhang, "Effective and Extensible Feature Extraction Method Using Genetic Algorithm-Based Frequency-Domain Feature Search for Epileptic EEG Multiclassification", *Medicine*, Vol. 96, No. 19, 2017.

105. Wu, Z., M. Yue, D. Li and K. Xie, "SEDP-Based Detection of Low-Rate DoS Attacks", *International Journal of Communication Systems*, Vol. 28, No. 11, pp. 1772–1788, 2015.

106. Wen, K., J. Yang, F. Cheng, C. Li, Z. Wang and H. Yin, "Two-Stage Detection Algorithm for RoQ Attack Based on Localized Periodicity Analysis of Traffic Anomaly", *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–6, 2014.

107. Cheng, C.-M., H. Kung and K.-S. Tan, "Use of Spectral Analysis in Defense Against DoS Attacks", *Global Telecommunications Conference, 2002. GLOBECOM '02*, Vol. 3, pp. 2143–2148, 2002.

108. Barford, P., J. Kline, D. Plonka and A. Ron, "A Signal Analysis of Network Traffic Anomalies", *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, pp. 71–82, ACM, 2002.

109. Hussain, A., J. Heidemann and C. Papadopoulos, "A Framework for Classifying Denial of Service Attacks", *Proceedings of The 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 99–110, ACM, 2003.

110. Zhang, Y., L. Cui, W. Wang and Y. Zhang, "A Survey on Software Defined Networking with Multiple Controllers", *Journal of Network and Computer Applications*, Vol. 103, pp. 101–118, 2018.

111. Ramjee, R., F. Ansari, M. Havemann, T. Lakshman, T. Nandagopal, K. Sabnani and T. Woo, "Separating Control Software From Routers", *2006 1st International Conference on Communication Systems Software & Middleware*, pp. 1–10, 2006.

112. Benzekki, K., A. El Fergougui and A. Elbelrhiti Elalaoui, "Software-Defined Networking (SDN): A Survey", *Security and Communication Networks*, Vol. 9, No. 18, pp. 5803–5833, 2016.

113. Sahay, R., W. Meng and C. D. Jensen, "The Application of Software Defined Networking on Securing Computer Networks: A Survey", *Journal of Network and Computer Applications*, Vol. 131, pp. 89–108, 2019.

114. Chica, J. C. C., J. C. Imbachi and J. F. Botero, "Security in SDN: A Comprehensive Survey", *Journal of Network and Computer Applications*, p. 102595, 2020.

115. McKeown, N., T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks", *ACM SIGCOMM Computer Communication Review*, Vol. 38, No. 2, pp. 69–74, 2008.

116. Osinski, T. and A. Dandoush, "XMPP As A Scalable Multi-Tenants Isolation Solution for ONOS-Based Software-Defined Cloud Networks", *2018 14th International Conference on Network and Service Management (CNSM)*, pp. 300–303, 2018.

117. Lee, W. and D. Xiang, "Information-Theoretic Measures for Anomaly Detection", *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*, pp. 130–143, 2000.

118. Bereziński, P., M. Szpyrka, B. Jasiul and M. Mazur, "Network Anomaly Detection Using Parameterized Entropy", *IFIP International Conference on Computer*

Information Systems and Industrial Management, pp. 465–478, Springer, 2015.

119. Yu, S., W. Zhou, R. Doss and W. Jia, "Traceback of DDoS Attacks Using Entropy Variations", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 22, No. 3, pp. 412–425, 2010.

120. Gu, Y., A. McCallum and D. Towsley, "Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation", *Proceedings of The 5th ACM SIGCOMM Conference on Internet Measurement*, pp. 32–32, USENIX Association, 2005.

121. Mehdi, S. A., J. Khalid and S. A. Khayam, "Revisiting Traffic Anomaly Detection Using Software Defined Networking", *International Workshop on Recent Advances in Intrusion Detection*, pp. 161–180, Springer, 2011.

122. Giotis, K., C. Argyropoulos, G. Androulidakis, D. Kalogeras and V. Maglaris, "Combining OpenFlow and SFlow for An Effective and Scalable Anomaly Setection and Mitigation Mechanism on SDN Environments", *Computer Networks*, Vol. 62, pp. 122–136, 2014.

123. Wang, M., H. Zhou, J. Chen and H. Zhang, "An Entropy Based Anomaly Traffic Detection Approach in SDN", *Telecommunications Science*, Vol. 31, No. 9, p. 83, 2015.

124. Bozdogan, H., "Akaike's Information Criterion and Recent Developments in Information Complexity", *Journal of Mathematical Psychology*, Vol. 44, No. 1, pp. 62–91, 2000.

125. Van Erven, T. and P. Harremos, "Rényi Divergence and Kullback-Leibler Divergence", *IEEE Transactions on Information Theory*, Vol. 60, No. 7, pp. 3797–3820, 2014.

126. Braga, R., E. De Souza Mota and A. Passito, "Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow.", *LCN*, Vol. 10, pp. 408–415, 2010.

127. Polat, H., O. Polat and A. Cetin, "Detecting DDoS Attacks in Software-Defined Networks Through Feature Selection Methods and Machine Learning Models", *Sustainability*, Vol. 12, No. 3, p. 1035, 2020.

128. Bhatia, N. *et al.*, "Survey of Nearest Neighbor Techniques", *CoRR*, Vol. ABS/1007.0085, 2010.

129. Abiodun, O. I., A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed and H. Arshad, "State-of-The-Art in Artificial Neural Network Applications: A Survey", *Heliyon*, Vol. 4, No. 11, p. e00938, 2018.

130. Umadevi, S. and K. J. Marseline, "A Survey on Data Mining Classification Algorithms", *2017 International Conference on Signal Processing and Communication (ICSPC)*, pp. 264–268, 2017.

131. Sundermeyer, M., R. Schlüter and H. Ney, "LSTM Neural Networks for Language Modeling", *Thirteenth Annual Conference of The International Speech Communication Association*, 2012.

132. Yin, W., K. Kann, M. Yu and H. Schütze, "Comparative Study of CNN and RNN for Natural Language Processing", *CoRR*, Vol. abs/1702.01923, 2017.

133. Kaelbling, L. P., M. L. Littman and A. W. Moore, "Reinforcement Learning: A Survey", *Journal of Artificial Intelligence Research*, Vol. 4, pp. 237–285, 1996.

134. Santanna, J. J., R. van Rijswijk-Deij, R. Hofstede, A. Sperotto, M. Wierbosch, L. Zambenedetti Granville and A. Pras, "Booters: An Analysis of DDoS As A Service Attacks", *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 243–251, 2015.

135. Erhan, D. and E. Anarım, "Boğaziçi University Distributed Denial of Service Dataset", *Data in Brief*, Vol. 32, p. 106187, 2020.

136. Leang, K. K., "Matlab Tricks and Tips [Focus on Education]", *IEEE Control Systems Magazine*, Vol. 33, No. 4, pp. 39–40, 2013.

137. Stanković, R. S. and B. J. Falkowski, "The Haar Wavelet Transform: Its Status and Achievements", *Computers & Electrical Engineering*, Vol. 29, No. 1, pp. 25–44, 2003.

138. Chen, Y. and K. Hwang, "Spectral Analysis of TCP Flows for Defense Against Reduction-of-Quality Attacks", *Communications, 2007. ICC'07. IEEE International Conference on Communications*, pp. 1203–1210, 2007.

139. Parthasarathy, S., S. Mehta and S. Srinivasan, "Robust Periodicity Detection Algorithms", *Proceedings of the 15th ACM international conference on Information and knowledge management*, pp. 874–875, 2006.

140. Shannon, C., E. Aben, K. Claffy, D. Andersen and N. Brownlee, *CAIDA UCSD Anonymized Internet Traces Dataset*, 2007, `http://www.caida.org/data/passive/passive_dataset.xml`, accessed in October 2012.

141. Aharon, M., M. Elad and A. Bruckstein, "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation", *IEEE Transactions on Signal Processing*, Vol. 54, No. 11, pp. 4311–4322, 2006.

142. Teknomo, K., "K-Means Clustering Tutorial", *Medicine*, Vol. 100, No. 4, p. 3, 2006.

143. Mallat, S. G. and Z. Zhang, "Matching Pursuits with Time-Frequency Dictionaries", *IEEE Transactions on Signal Processing*, Vol. 41, No. 12, pp. 3397–3415, 1993.

144. Aharon, M., M. Elad and A. Bruckstein, "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation", *IEEE Transactions*

on Signal Processing, Vol. 54, No. 11, pp. 4311–4322, 2006.

145. Khalid, S., T. Khalil and S. Nasreen, "A Survey of Feature Selection and Feature Extraction Techniques in Machine Learning", *2014 Science and Information Conference*, pp. 372–378, 2014.

146. Tian, J., M. H. Azarian and M. Pecht, "Anomaly Detection Using Self-Organizing Maps-Based K-Nearest Neighbor Algorithm", *Proceedings of The European Conference of The Prognostics and Health Management Society*, Citeseer, 2014.

147. Hanusz, Z., J. Tarasinska and W. Zielinski, "Shapiro-Wilk Test with Known Mean", *REVSTAT-Statistical Journal*, Vol. 14, No. 1, pp. 89–100, 2016.

148. Nussbaumer, H. J., "The Fast Fourier Transform", *Fast Fourier Transform and Convolution Algorithms*, pp. 80–111, Springer, 1981.

149. Baishya, R. C., N. Hoque and D. K. Bhattacharyya, "DDoS Attack Detection Using Unique Source IP Deviation.", *IJ Network Security*, Vol. 19, No. 6, pp. 929–939, 2017.

150. Maurer, W. D. and T. G. Lewis, "Hash Table Methods", *ACM Computing Surveys (CSUR)*, Vol. 7, No. 1, pp. 5–19, 1975.

151. Montgomery, D. C., C. L. Jennings and M. Kulahci, *Introduction to Time Series Analysis and Forecasting*, Wiley series in probability and statistics, 2015.

152. Legendre, P. and D. Borcard, "Box–Cox-Chord Transformations for Community Composition Data Prior to Beta Diversity Analysis", *Ecography*, Vol. 41, No. 11, pp. 1820–1824, 2018.

153. Jain, G. and B. Mallick, "A Study of Time Series Models ARIMA and ETS", *Available at SSRN: https://ssrn.com/abstract=2898968*, 2017.

154. Thottan, M. and C. Ji, "Anomaly Detection in IP Networks", *IEEE Transactions on Signal Processing*, Vol. 51, No. 8, pp. 2191–2204, 2003.

155. Cho, K. and R. Kaizaki, *MAWI Working Group Traffic Archive*, 2018, `https://mawi.wide.ad.jp/mawi`, accessed in September 2019.

156. Turner, A., *Tcpreplay*, 2011, `http://tcpreplay. synfin. net/trac/`, accessed in November 2019.

157. Kaur, S., J. Singh and N. S. Ghumman, "Network Programmability Using POX Controller", *International Conference on Communication, Computing & Systems (ICCCN'2014)*, pp. 134–138, 2014.

158. Amidan, B. G., T. A. Ferryman and S. K. Cooley, "Data Outlier Detection Using The Chebyshev Theorem", *2005 IEEE Aerospace Conference*, pp. 3814–3819, 2005.

159. Cui, J., J. He, Y. Xu and H. Zhong, "TDDAD: Time-based Detection and Defense Scheme Against DDoS Attack on SDN Controller", *Australasian Conference on Information Security and Privacy*, pp. 649–665, Springer, 2018.

160. Asadollahi, S., B. Goswami and M. Sameer, "Ryu Controller's Scalability Experiment on Software Defined Networks", *2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC)*, pp. 1–5, 2018.

161. Fernandes, E. L. and C. E. Rothenberg, "OpenFlow 1.3 Software Switch", *Salao de Ferramentas do XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos SBRC*, pp. 1021–1028, 2014.

162. Borgnat, P., G. Dewaele, K. Fukuda, P. Abry and K. Cho, "Seven Years and One Day: Sketching The Evolution of Internet Traffic", *INFOCOM 2009*, pp. 711–719, 2009.

163. Zhang, D., "Wavelet Transform", *Fundamentals of Image Data Mining*, pp. 35–44, Springer, 2019.

164. Pena, E. H., L. F. Carvalho, S. Barbon Jr, J. J. Rodrigues and M. L. Proença Jr, "Anomaly Detection Using the Correlational Paraconsistent Machine with Digital Signatures of Network Segment", *Information Sciences*, Vol. 420, pp. 313–328, 2017.

165. Yao, Y., "Using Nonlinear Difference Equations to Study Quicksort Algorithms", *Journal of Difference Equations and Applications*, Vol. 26, No. 2, pp. 275–294, 2020.

166. Mathieu, M., M. Henaff and Y. A. LeCun, "Fast Training of Convolutional Networks through FFTs", *CoRR*, Vol. ABS/1312.5851, 2014.