Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
Information Technology College
Information Networks Department

A DEVELOPED MODEL FOR DDOS DETECTION AND MITIGATION IN SDN ENVIRONMENT BASED ON ENTROPY AND MACHINE LEARNING

A Dissertation

Submitted to the Council of the College of Information Technology for Postgraduate Studies of the University of Babylon in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Information Technology / Information Networks

By

Mohammed Ibrahim Kareem Khaleel

Supervised by

Assist. Prof. Dr. Mahdi Nsaif Jasim Alwan

2022 A.D. **1444 A.H.**

بسم الله الرحمن الرحيم

﴿ وَٱللَّهُ يَدۡعُوٓاْ إِلَىٰ دَارِ ٱلسَّلَٰمِ وَيَهۡدِي مَن يَشَآءُ إِلَىٰ صِرَٰطٖ مُّسۡتَقِيمٖ﴾

صدق الله العظيم

سورة يونس 25

Supervisor Certification

I certify that the dissertation entitled (**A developed model for DDoS detection and mitigation in SDN environment based on entropy and machine learning**) was prepared under my supervision at the department of Information Networks/ College of Information Technology/ University of Babylon as partial fulfillment of the requirements of the degree of Doctor of Philosophy in Information Technology/Information Networks.

Signature:

Supervisor Name: **Assist. Prof. Dr. Mahdi Nsaif Jasim**

Date: / /2022

The Head of the Department Certification

In view of the available recommendations, I forward the dissertation entitled "**A developed model for DDoS detection and mitigation in SDN environment based on entropy and machine learning**" for debate by the examination committee.

Signature:

Prof. Dr. Saad Talib Al-Jebori

Head of Information Networks Department

Date: / /2022

Certification of the Examination Committee

We hereby certify that we have studied the dissertation entitled (**A developed model for DDoS detection and mitigation in SDN environment based on entropy and machine learning**) presented by the student (**Mohammed Ibrahim Kareem Khaleel**) and examined him, in its content and what is related to, and that, in our opinion, it is adequate with (**Excellent**) standing as a dissertation for the degree of Doctor of Philosophy in Information Technology /Information Networks.

Signature:
Name: **Dr. Ghassan H. Abdul-Majeed**
Title: **Professor**
Date: / / 2022
(**Chairman**)

Signature:
Name: **Dr. Ziyad Tariq Mustafa Al-Ta'i**
Title: **Professor**
Date: / / 2022
(**Member**)

Signature:
Name: **Dr. Wesam S. Bhaya**
Title: **Professor**
Date: / / 2022
(**Member**)

Signature:
Name: **Dr. Ashwaq T. Hashim**
Title: **Professor**
Date: / / 2022
(**Member**)

Signature:
Name: **Dr. Mehdi Ebady Manaa**
Title: **Assistant Professor**
Date: / / 2022
(**Member**)

Signature:
Name: **Dr. Mahdi Nsaif Jasim**
Title: **Assistant Professor**
Date: / / 2022
(**Member and Supervisor**)

Approved by the Dean of the College of Information Technology, University of Babylon.

Signature:
Name: **Dr. Hussein Attia Lafta**
Title: **Professor**
Date: / / 2022
(**Dean of Collage of Information Technology**)

Dedication

To the memory of my greatest father who gave me and teaches me a lot, whose death was the biggest loss to me.

Mohammed

Acknowledgement

All praise is to ALLAH Almighty, who enabled me to complete this task successfully, and our utmost respect to his last Prophet Mohammed and his holy family.

My deepest gratitude is to **my advisor**, for invaluable guidance, supervision, and untiring efforts during this work.

Sincere appreciation is due to **my mother**, **my wife**, **my sisters, and my brothers {Ali, Hussain, and Abbas}** for their patience, encouragement, and help during the work.

I would like to thank all **my dear teachers** in the College of Information Technology for their remarkable education and guidance during the study period.

I would like to thank all **my dear colleagues** in the Ph.D. course and to the whole staff of the college Information Technology.

Finally, I would like to thank all the helpful and lovely people who helped me directly or indirectly to complete this work.

Abstract

Software-Defined Networking (SDN) represents one of the most elegant technologies to make network management and configuration easy by increasing network programmability. SDN is characterized by its centralized management and control, so it becomes more attractive to Distributed Denial of Service (DDoS) attacks to affect the whole network. DDoS attacks are one of the most dangerous threats to networks, as they are easy to operate and difficult to detect. The attack exploits the SDN controller to be flooded by the incoming packets from the switches. There are many techniques to detect and mitigate the effects of DDoS attacks.

In this dissertation, a combination of entropy and machine learning (ML) models is proposed to overcome the weak points of both of them and consolidate the SDN network against DDoS attacks.

The proposed system consists of two levels of detection, the first is lightweight detection based on the entropy and the second is the deep inspections using the proposed ML techniques. The second level is the outcome of training the model using six ML algorithms (random forest (RF), rep tree (REPT), random tree (RT), decision stump (DS), decision tree (DS), and partial decision tree (PART)) in addition to a proposed semi-supervised model and then choosing the most suitable one for development. All the ML models were trained and tested using the well-known dataset "CICIDS2017" which is famous as the current benchmark dataset. The PART model shows the best performance is chosen.

To validate the performance of each of the detectors, four sets of testing groups are used (grouped based on feature selection). The feature set shows the best performance is chosen.

The accuracy results of 99.92% and 99.77% are achieved by using 6 and 11 features respectively on the CICIDS2017 dataset. The proposed model was also validated on the famous dataset "CICDDoS2019" which shows 99.99% and 99.73% accuracy for both UDP and SYN attacks respectively. Simulations are carried out in Mininet emulator with POX controller and open flow switches. The real-time proposed system has achieved a high detection rate of 98.06%. Additionally, a proposed efficient strategy for East/West communication to overcome a single point of failure in the SDN POX controller is used.

Declaration Associated with this Dissertation

Some of the works presented in this dissertation that has been published or accepted are listed below.

1. **Fast and accurate classifying model for denial-of-service attacks by using machine learning**

   Bulletin of Electrical Engineering and Informatics, 2022

2. **DDOS Attack Detection Using Lightweight Partial Decision Tree algorithm**

   Proceedings of the 2nd 2022 International Conference on Computer Science and Software Engineering, CSASE 2022

3. **The Current Trends of DDoS Detection in SDN Environment**

   Proceedings of 2021 2nd Information Technology to Enhance E-Learning and other Application Conference, IT-ELA 2021

4. **Entropy-based distributed denial of service attack detection in software-defined networking.**

   Indonesian Journal of Electrical Engineering and Computer Science,2022.

5. **Performance Evaluation of the POX Controller Under a DDoS Attack (Scopus Q3, IJEECS)**

6. **Machine Learning-Based DDoS Attack Detection in Software-Defined Networking (Springer Conference, NTICT2022)**

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Description
ACK	ACKnowledge
AMQP	The Advanced Message Queuing Protocol
API	Application Programming Interface
CICDDoS2019	Canadian Institute for Cybersecurity Distributed Denial of Service Attack 2019
CICIDS2017	Canadian Institute for Cybersecurity Intrusion Detection System 2017
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
Dest_IP	Destination IP
DoS	Denial of Service
DS	Decision Stump
FAR	False Alarm Rate
FN	False Negative
FP	False Positive
FPR	False Positive Rate
FSCA	Flat Distributed SDN Controllers Using AMQP
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IREP	Incremental Reduced Error Pruning
ITZ	Internet Topology Zoo
J48 or C4.5	Decision Tree
L3_learning	Layer Three
MAC	Media Access Control
MDL	Minimum Description Length
minobj	Minimum No of Object

ML	Machine Learning
NaN	Not a Number
NOX	Type of Controllers
OVS	Open Virtual Switch
Packet_In	Packet Incoming
PART	Projective Adaptive Resonance Theory
POD	Ping of Death
POX	Type of Controllers
QoS	Quality of Service
RAM	Random Access Memory
REPT	Reduced Error Pruning Tree
RF	Random Forest
RIPPER	Repeated Incremental Pruning to Produce Error Reduction
RT	Random Tree
SDN	Software-Defined Networking
SYN	SYNchronize
TCP	Transmission Control Protocol
Th or δ	Threshold
TLS	Transport Layer Security
TN	True Negative
TP	True Positive
UDP	User Datagram Protocol
Win_S	Window Size

Chapter One

General Introduction

1.1 Overview

Traditional network infrastructures have been unable to address certain requirements such as high bandwidth, accessibility, high connection speed, dynamic management, cloud computing, and virtualization. Thus, Software-Defined Networking (SDN) is a new networking technology that overcomes the constraints of traditional networks. Traditional networks' bottlenecks were their complicated design, the setup of individual network devices using language programming of the vendors, and a lack of a global perspective of the network. A global view of networks became feasible with the introduction of SDN, allowing for faster network setup and administration [1]. The SDN's main feature is the decoupling of the control plane and the data plane. The SDN architecture's network brain is centered on a logically centralized controller. The controller provides a network operating system. Under the SDN design, the network becomes programmable using high-level programming languages and is easily customized and maintained.

DDoS (Distributed Denial of Service) has grown in prominence as a dangerous cyber threat. DDoS aims to exhaust the victim's resources by preventing authorized users from accessing them, causing financial and reputational loss. SDN is a promising solution and the network's future, but it is vulnerable to DDoS assaults. DDoS assaults, as the name implies, are dispersed in nature and may be conducted globally by distributed botnets. The spreading nature of the assault, the attack's fluctuating time pattern, and the magnitude of the attack, DDoS is challenging to detect and remediate in part due to the use of spoofed IP addresses and the challenge of identifying traffic patterns [2].

The SDN controller is exposed to DDoS attacks through the communication line between the controller and the data plane. DDoS attacks direct a large amount of traffic to the OpenFlow switch on the data plane. If packets arriving at the OpenFlow switch do not match with the flow input in the

flow table (miss flow), packets are taken into the flow buffer. Then, they are transmitted to the controller with the Packet-In message to write a new rule. In this situation, the sources of the controller (memory, processor, bandwidth, etc.) remain incapable and the network becomes inoperative. In addition, the bandwidth of the communication line between the controller that is exposed to attack traffic and the OpenFlow switch is negatively affected. Therefore, network performance severely declines [3].

In traditional networks, a variety of strategies have been employed to mitigate the impact of DDoS attacks [4]. Packet analysis was resource-intensive in conventional networks; thus, sampling techniques were utilized to validate the packets. For traffic collection and analytics, Cisco's NetFlow flow monitoring technology and S-flow packet sampling technology were employed [5]. When a DDoS attempt is identified, flow rules may be dynamically introduced into the flow table due to the programmable nature of SDN.

Machine learning (ML)-based approaches can provide more dynamic, more efficient, and smarter solutions for SDN management, security, and optimization. In order to ensure network security, the detection of DDoS attacks is very important for taking the necessary measures in a timely manner. Processing SDN flow data with the ML-based DDoS attack detection systems integrated into SDN structures can lead to a self-determining network that is able to learn and act. Moreover, SDN having an integrated ML application may be a reference model in forming a secure structure in studies providing for the integration of 5G networks. Many protection systems in SDN employ statistical, ML, and deep learning approaches to identify and mitigate DDoS attacks [6].

The flow statistics can be provided through OpenFlow, which is a widely used southbound application programming interface (API) for communication between switches and controllers. The relevant characteristics may be derived from the flow statistics supplied by SDN switches and combined with ML

techniques for security analytics [7]. Many studies have used OpenFlow's flow capabilities to identify DDoS attacks in SDN [8].

1.2 The Related Works

Given that the SDN network is a modern network, it has got great attention from the researchers, especially the DDoS attacks detection. Tables (1.1 – 1.5) shows overall the related works and compare them with parts of a proposed system.

Table 1-1 : The entropy related works and the proposed method

No	Authors, Year	Objective of the paper	Key findings
1	Mousavi & St-Hilaire, 2015 [9]	Using Entropy to identify DDoS attacks in an SDN controller at an early stage. Create a simple, efficient algorithm to identify the assault.	SDN, DDoS attack, Entropy method to detect DDoS, Mininet(tool).
2	Wang et al., 2015 [10]	To design a flow statistics process in the switch to detect DDoS attacks based on entropy.	Entropy to detect DDoS, Open vSwitch.
3	da Silva et al., 2016 [11]	The proposed framework is called ATLANTIC and it combines the use of information theory to calculate deviations in the entropy of flow tables and a range of machine learning algorithms to classify traffic flows. As a result, ATLANTIC is a flexible framework capable of categorizing traffic anomalies and using the information collected to handle each traffic profile in a specific manner, e.g., blocking malicious flows.	The entropy of flow tables and a range of machine learning algorithms to classify traffic flows
4	Boite et al., 2017 [12]	Monitoring module based on state-employed on taking sample the traffic from SDN devices in the data plane. Statistical approach based on entropy is used to examine traffic in the network	Source and Destination Ports, Source and Destination IPs: Detection and Mitigation
5	Sanjeetha et al., 2018 [13]	To show DDoS attacks using Flow-entry tables of switches and ways to mitigate them.	Using Flow-entries as a vulnerability for DDoS attacks.
6	Koay et al., 2018 [14]	To propose a multi-classifier system to detect DDoS based on the proposed set of multiple entropy-based features and ML classifier to increase accuracy	Entropy, multiple classifier systems, and ML to classify traffic.

7	Kalkan et al., 2018 [15]	When congestion is discovered, the SDN switch provides packet information to the controller, and the controller's application calculates the joint entropy of pair profiles, and if it exceeds a certain threshold, a DDoS attack is detected.	Use both Source and Destination IPs for detection and mitigation ofDDoS attacks
8	Hong et al., 2019 [16]	The assembler collects packets supplied by switches on a regular basis and passes the results to the controller for processing. The controller is the unit in charge of determining whether or not a network is under attack. – The observer is instructed by the third unit, the executor, to install the base as ordered by the manager's unit.	Source and Destination for Ports and IPs: Detection and Mitigation
9	Liu et al., 2019	Based on the characteristics of SDN, a DDoS attack detection method combining generalized entropy and PSO-BP neural network is proposed. The traffic is pre-detected by the generalized entropy method deployed on the switch, and the detection result is divided into normal and abnormal. Locate the switch that issued the abnormal alarm. The controller uses the PSO-BP neural network to detect whether a DDoS attack occurs by further extracting the flow features of the abnormal switch	Entropy and PSO-BP
10	R. Li & Wu, 2020 [17]	The SDN controller periodically finds and collects information from SDN switches and then calculates the entropy value based on the IP destination addresses, where if this entropy is less than the threshold means an attack has been discovered.	Detection: Destination IP
11	Shen et al., 2020 [18]	The proposed TPDD, a two-phase DDoS detection system to detect DDoS attacks in SDN. In the first phase, utilize the characteristics of SDN to collect coarse-grained flow information from the core switches and locate the potential victim. Then monitor the edge switches located close to the potential victim to obtain finer-grained traffic information in the second phase. The collection method of each phase fully considers the impact on the bandwidth between the controller and switches. Without modifying the existing flow rules,	Two phases: Entropy and machine learning.

		the collection module can obtain sufficient information about traffic. By using entropy-based and machine learning-based methods, the detection module can effectively detect anomalies and identify whether the potential victim marked in the first phase is the target of attacks.	
12	Abou El Houda et al., 2020 [19]	WisdomSDN covers both detection and mitigation of illegitimate DNS requests and responses. WisdomSDN consists of: (1) a novel proactive and stateful scheme (PAS) to perform one-to-one mapping between DNS requests and DNS responses; it operates proactively by sending only legitimate responses, excluding amplified illegitimate DNS responses; (2) a machine learning DDoS detection module to detect, in real-time, illegitimate DNS requests. This module consists of (a) Flow statistics collection scheme (FSC) to gather the features of flows in an efficient and scalable way using sFlow protocol; (b) Entropy calculation scheme (ECS) to measure randomness of network traffic; and (c) Bayes Network based Filtering scheme (BNF) to classify, based on entropy values, illegitimate DNS requests; and (3) DNS Mitigation scheme (DM) to effectively mitigate illegitimate DNS requests	Entropy and BNF
13	AbdelAzim et al., 2021 [20]	The suggested hybrid algorithm is more resilient since it considers the type of current network traffic and can alter in real-time when that traffic's characteristics change.	The suggested framework can be utilized with various methods for evaluating trust because it is not dependent on the metric.
14	Mishra et al., 2021 [21]	Low-complexity mechanisms for DDoS assaults based on differences in entropy between DDoS attacks and regular traffic.	Mininet emulator with POX controller and open flow switches
15	Ujjan et al., 2021 [22]	Fast and efficient DDoS detection using entropy. Entropy calculations that combine Shannon and Renyi entropies assisted SDN controllers deal with high malicious traffic by accurately identifying distributed aspects of DDoS activity.	Shannon and Renyi entropy

16	Anil et al., 2022 [23]	The entropy approach, which determines the entropy variations at the destination IP address and the threshold, has been used to detect and mitigate DDoS attacks.	The attack can be identified as early as the first 250–500 packets of the randomly produced data, according to observations, indicating that early detection is achievable.
#	The Proposed	DDoS attack detection is based on entropy variations at the destination IP and 0.5 thresholds.	The proposed method is employed as indicator for DDoS attack detection. Requirements: POX Controller, Mininet with Scapy tool. The technique is also able to detect the start of a second attack (after 0.443 seconds)

Table 1-2: The features selection related works and the proposed method

No	Authors, Year	Objective Of the Paper	Feature selection techniques	Dataset
1	Filter & Filter, 2014 [24]	Principal Component Analysis, investigating Random Forests, Backward Feature Elimination, and Forward Feature Construction are the most widely used methods for dimensionality reduction. Other methods include removing data columns with excessive numbers of missing values, reducing highly correlated columns, removing low variance columns, and removing low variance columns.	1. A high number of missing values 2. Low variance 3. High correlation with other data columns 4. Principal Component Analysis 5. First cuts in random forest trees 6. Backward feature elimination 7. Forward feature construction	KDD data set
2	Lucky et al., 2020 [25]	Based on the quantity of features chosen, various DT models were built using the Low Variance filter approach and a variance threshold of 0.025.	Low Variance filter technique	CAIDA 2007, CIC 2017 and 2019
3	Maniriho et al., 2020 [26]	With increased performance, the proposed feature reduction method aims to reduce the number of characteristics needed to identify DoS assaults.	Filter-based feature selection algorithms: IGR, CR, and ReF.	CICIDS2017 and KDD Cup

4	Peneti & Hemalatha, 2021 [27]	The objective of this study is to use feature selection techniques to create an extremely effective intrusion-based system. The elimination of features increases IDS speed and lowers memory requirements.	Recursive Feature Elimination	CICIDS2017
5	Zaib et al., 2021 [28]	Reduce features using variance, correlation, and ¾ quartile method.	The forward selection wrapper	CSE-CIC IDS 2018 and NSL-KDD.
6	Ahsan et al., 2021 [29]	XGBoost importance, wrapper approach, and information gain ratio were used in conjunction with univariate test and Pearson coefficient test statistical analysis as well as various feature engineering processes to reduce feature dimensionality.	Univariate test and Pearson coefficient test along with XGBoost importance, wrapper technique, and information gain ratio	NSL-KDD and UNSW-NB15.
7	Kshirsagar & Kumar, 2022 [30]	The study proposes a DDoS attack detection methodology for quickly detecting refection and exploitation-based DDoS assaults.	information gain and correlation	CICDoS 2019 and KDD Cup 1999
#	The Proposed	The study proposed hybrid feature selection method based on information gain and low variance filter techniques	information gain and variance	CICIDS2017 and CICDDoS2019

Table 1-3: The ML related works and the proposed method

No	Authors, Year	Technique name	Features	Results Discussion
1	Guozi et al., 2018 [31]	DDoS attacks and flash event detection based on flow characteristics in SDN	Byte on average, Entropy of source's average duration, Increasing the speed of flow	The suggested method derives five characteristics from the data provided by the flow table collecting module using flow feature extraction. KNN algorithm is used to identify the captured packet as normal or anomaly traffic.

			table entries by increasing the IP, entropy of the destination IP	
2	Li et al., 2018 [32]	Binary Bat algorithm Random Forest	The dynamic set of features	There are two primary processes at work in this project: By recording network traffic, an IDS may intelligently identify anomalies in the network. The Bat Algorithm is utilized to pick features, while Random Forest is employed as a classifier.
3	Deepa et al., 2019 [33]	KNN SVM Naïve Bayes	Time Difference	The systems proposed combined KNN-SOM, NV-SOM, and Support Vector Machine-SOM. Authors have found that the Support vector machine -SOM has good accuracy and a high detection rate.
4	Myint Oo et al., 2019 [34]	ASVM	Number of flow packets on average, the average number of bytes in a flow, Flow packets vary in size, as do flow bytes, Average length of time	Model capture sample traffic and send to ASVM classifier to determine attack or not.
5	Phan & Park, 2019 [35]	HIPF Support vector machine SOM	Flow, Total number of sources is active., Number of packets per flow on average	The authors deploy an ensemble of SVM, eHIPF, and SOM classifiers to defend the network. — The raw-data processing module gathers traffic, extracts characteristics, and provides the data to the classifier module. – Using an ensembled classifier, the data is classed as anomalous or normal. – The eHIPF filtering technique is used to filter traffic in order to reduce it.
6	Novaes et al., 2020 [36]	Long Short-Term Memory and Fuzzy Logic	bits/s, packets/s, source IP entropy, destination IP entropy, source Port entropy, destination Port Entropy	In SDN contexts, this paper proposes a solution for detecting and mitigating DDoS and Port scan attacks (LSTM-FUZZY). The three phases of the LSTM-FUZZY system described in this study are characterization, anomaly detection, and mitigation. The dataset used was CICDDoS 2019.

7	Gadallah et al., 2021 [37]	Naive Bayes, K-Nearest Neighbor, Decision Tree, and Random Forest are also utilized and compared with the SVM	The necessary statistics are used as features	The suggested approach starts by sending regular and attack traffic flow packets across the network. When packets arrive to the controller, the headers are extracted and appropriate flow calculations are performed to get the required characteristics. The characteristics are utilized to construct a dataset for the linear support vector machine classifier. The kernel radial basis function is used to train the model using the classifier. Weaknesses: no list features, use source IP as feature.
8	Zhou et al., 2022 [38]	Six classification methods are applied: Decision Tree, Deep Learning, K Nearest Neighbor, Logistic Regression, Random Forest, and Support Vector Machine.	entropy rate of IP source flow, entropy rate of flow, entropy of packet size, entropy rate of packet size, and number of ICMP destination unreachable packet	The researchers suggest five additional features: packet size entropy, packet size entropy rate, IP source flow entropy, flow entropy, and the number of ICMP destination unreachable packets. These characteristics can be used to detect DDoS assaults.
#	The Proposed	Modified PART Algorithm	6 useful features	1- Use new features in SDN 2- Extracted features in SDN (An applied IP flows collected from the SDN POX controller through emulation on Mininet. On the other hand, in the second scenario, the CICIDS2017 dataset was applied). 3- Source IP not used as feature to avoid spoofed IP 4- Both Training and Evaluation are offline setup. 5- Realtime proposed PART with less testing time and high accuracy rate. 6- Two well-known datasets are used (CICIDS2017 and CICDDoS2019)

Table 1-4: The Semi-supervised learning related works and the proposed method

No	Authors, Year	Technique name	Results Discussion
1	Lysenko et al., 2018 [39]	fuzzy c-means clustering.	DDoS botnet detection technique based on the use of the semi-supervised fuzzy c-means clustering. The research is based on network traffic characteristics retrieved from the network that might indicate the presence of DDoS botnets in the network. According to the findings of the experiments, the detection rate is around 95%, with only 6% of false positives.
2	Idhammad et al., 2018 [40]	Entropy estimation, Co-clustering, Information Gain Ratio and Exra-Trees algorithm	The entropy estimator computes and analyses the entropy of network traffic data using a time-based sliding window. When the entropy reaches its limit, the co-clustering process divides the incoming network traffic into three clusters within the current time frame. The average entropy of the network header attributes between the current time frame subset and each of the created clusters is then used to determine an information gain ratio. Anomaly network traffic data clusters with a high information gain ratio are selected for preprocessing and classification using ensemble classifiers based on the Extra-Trees approach.
3	Al-Jarrah et al., 2018 [41]	Random Forest, Bagging, and AdaboostM1	This study proposes a semi-supervised multi-layered clustering model for the detection and mitigation of network intrusion. SMLC may gain detection performance comparable to supervised ML-based IDPS by learning from partially labeled data. The performance of SMLC is compared against that of a well-known semi-supervised model (tri-training) and supervised ensemble ML models, namely Random Forest, Bagging, and AdaboostM1, on two benchmark network-intrusion datasets, NSL and Kyoto 2006.
4	Gu et al., 2019 [42]	K-Means algorithm and Hybrid Feature Selection	This study presents a weighted k-means identification technique that is semi-supervised. We first propose a Hadoop-based hybrid feature selection method to identify the most potent feature sets, and then we provide an improved density-based initial cluster centers selection approach to address the issue of outliers and local optimum. The detection of threats is then performed using the Semi-supervised K-Means approach with Hybrid Feature Selection. The DARPA DDoS dataset, CAIDA "DDoS assault 2007" dataset, CICIDS "DDoS attack 2017" dataset, and real-world dataset are used in the verification experiment.
5	Viegas et al., 2020 [43]	Verification approach	In order to produce consistent classifications across time, even in the absence of model updates, the researchers present a new semi-supervised intrusion detection model that makes use of a verification technique. To update the underlying ML models without requiring human input, use this verification method with semi-supervised learning. Depending on the outcome of the pool of classifiers, the pool verifier uses the classifications accepted by the verifier to assess its reliability.

6	Khamaiseh et al., 2020 [44]	K-Nearest Neighbor and Artificial Neural Network	This study suggests FloodDetector, an effective framework for identifying known and unidentified flooding attacks in SDN. It is a controller-independent SDN application that recognizes both known and unidentified flooding attacks using two ML classifiers: K-Nearest Neighbor and Artificial Neural Network.
7	Aamir & Ali Zaidi, 2021 [45]	Agglomerative and K-means with feature extraction under Principal Component Analysis	This study presents a clustering-based method for distinguishing data representing network traffic flows, including both conventional and DDoS activity. Two distinct clustering methods cluster the unlabeled data, and a vote procedure determines the final classification of traffic flows. After labeling, the trained models for future classification are obtained using supervised ML techniques such as k-Nearest Neighbors, Support Vector Machine, and Random Forest .
8	Akula, 2021 [46]	The N-Gram line generation, feature selection algorithm, and SVM algorithm	This paper offers network traffic flow-based approach for mobile malware detection that assumes each HTTP flow as a document and analyzes HTTP flow requests using natural language processing string analysis. An effective malware detection model is created using the N-Gram line generation, feature selection method, and SVM algorithm.
9	Najafimehr et al., 2022 [47]	DBSCAN, SVM, and Random Forest	The approach proposed in this study combines supervised and unsupervised methods. Using multiple flow-based criteria, a clustering method first isolates the abnormal traffic from the usual data. A classification technique is then used to label the clusters using particular statistical parameters. The authors analyze the suggested strategy using a large data processing framework, training on the CICIDS2017 dataset and testing on a different set of assaults from the more recent CICDDoS2019.
10	Aouedi et al., 2022 [48]	A semi-supervised and federated-learning	A semi-supervised, federated-learning-based Intrusion Detection System is provided in this work. This model was trained using both labeled and unlabeled data in a semi-supervised manner. A semi-supervised FL that combines client-side unsupervised learning with server-side supervised learning. The untrained and supervised models are then automatically combined to produce a unified learning and classification solution for IDS.
#	The Proposed	K-Means, CANOPY	1- Training offline, 2- Produce 2 centroids (DDoS and Normal) 3- Testing and Evaluation

Table 1.5: The Features of past East/West approaches compared to the proposed - FSCA

No	Approaches	Controller Platform	Programming Language	Network Distribution	Coordination Strategy	Cross-Platform Coordination	Data Storage	East/West Protocol

1	HyperFlow [49]	NOX	C++	Flat	Broker-based P2P	*No*	WheelFS [50]	Publishing/subscription via distribution file system
2	ONIX [51]	ONIX	C++	Flat	Cluster	*No*	Distribution hash table	ZooKeeper API [52]
3	OpenDayLight [53]	OpenDayLight	Java	Flat	Cluster	*No*	In-memory MD-SAL DB [54]	Akka, Raft [55]
4	ONOS [56]	ONOS	Java	Flat	Cluster	*No*	Distributed data Structure	Raft
5	CIDC [57]	Floodlight	Java	Flat	Broker-based P2P	*Yes*	In-memory MD-SAL DB	Custom-event-driven protocol
6	IRIS [58]	Floodlight	Java	Hierarchical	Leader-Based	*No*	MongoDB	Custom protocol
7	Kando [59]	Kando	C++	Hierarchical	Leader-Based	*No*	In-memory	RTPS-DDS [60]
#	**The Proposed - FSCA**	**POX**	**Python**	**Flat**	**Server-based**	***Yes***	**Platform - dependent**	**Publishing/subscription via AMQP in client-server mode**

1.3 The Problem Statement

SDN is a new technology that makes networks flexible, centralized, and programmable. The SDN architecture has a control plane and a data plane separated in different components. Through the centralized control plane, system administrators can take charge of the whole network (controller). These features of SDN can be used in the construction of intelligent and autonomous networks. Also, since the installation of SDN, the costs of running large data centers have gone down a lot.

But because the SDN controller is centralized and controlling large networks, it becomes attractive target for attackers. DDoS attacks is one of the most dangerous types of network attacks, it's simple to implement and difficult to detect. With the Internet's huge growth, there are a lot of hosts that can be attacked. Most DDoS attacks are caused by software installed on vulnerable hosts without the users' knowledge.

Consequently, this dissertation proposes two layers detection approach of DDoS attacks. It is an integration between the entropy - ML based methods. The merit of the proposed approach is as fast as entropy based and as accurate as ML based. The outcome of this proposal is protecting the SDN from the fraud of DDoS flooding which leads to whole network failure.

1.4 The Dissertation Objectives

The objectives of this dissertation can be described as follows:

a. Proposing an analytical study to reduce the number of features needed to determine the DDoS attacks in an SDN environment.

b. Proposing a hybrid model of both entropy and ML techniques to DDoS detection.

c. Developing a new ML approach to outperform the competitive approaches.

d. Proposing a new approach to mitigate the effects of DDoS attacks in case it happened.

1.5 The Scope of the research

The proposed system's scope can be applied in large organizations that need to protect their SDN networks from DDoS attacks by using the entropy and ML.

1.6 The Dissertation Contributions

The contribution of this dissertation can be summarized as follows:

a. Modification of original L3 learning switch code in network POX controller to operate the proposed system's algorithms automatically.
b. The DDoS attacks were early detected using entropy of destination IP.
c. The proposed approach is hybrid of both entropy and ML based DDoS attacks detection and that makes it gain the merits of both in SDN environment.
d. Developed a hybrid features selection approach based on both information gain and low variance filter techniques.

e. Determination of the fastest and more accurate algorithm amongst six ML algorithms using only six features to achieve high accuracy when it is evaluated on the CICIDS2017 dataset.
f. Modifying the PART algorithm to give high accuracy and less tree size. The PART algorithm is modified in such way that it uses minimum description length (MDL) to prune the rules set to avoid overfitting.
g. Developed a semi-supervised learning model to increase classification accuracy.
h. Developed efficient EAST/WEST mitigation recovery procedure for POX controllers.
i. Compared, analyse and evaluate the proposed detection and mitigation techniques with the approaches found in the literature.

1.7 Limitations

a. Dynamic window size.
b. Low-rate DDoS detection.
c. Block port of open vSwitch (in Mininet) when DDoS attack detection.

1.8 The Outlines

This dissertation contains five chapters organized as follows:

Chapter Two: The Theoretical Background

This chapter reviews both of DDoS attacks and SDN architecture, the challenges faced by SDN and its fundamental characteristics. In addition, the OpenFlow protocol specification and parameters that were used in the SDN were used. **The supervised and semi supervised algorithms** used to detect DDoS attacks and in addition, the **Entropy** will be presented in the proposed system. In the last part of the chapter, the evaluation of the ML algorithms will be explained.

Chapter Three: The Proposed System Design

In this chapter, the proposed system details have been mentioned, for include, the proposed system and the algorithms that are allocated to detect the DDoS attacks using the Entropy and ML techniques.

Chapter Four: The Results Discussion and Analysis

The results of proposed system for the DDoS attacks detection are discussed. The evaluation results are presented; these results are depended on accuracy, precision, recall, F-Measure, test time and tree size.

Chapter Five: The Conclusions and Suggested Future Works.

The conclusion was mentioned to DDoS attacks detection in the SDN network and also suggested future works to develop the proposed system was mentioned.

Chapter Two

The Theoretical Background

2.1 Software-Defined Networking

The SDN design decouples the data planes and control, thus all network control logic is relocated from network devices to a logically centralized software-based entity the controller. The SDN controller understands the whole network. It is software that resides in the control plane that instructs forwarding behavior to all nodes scattered throughout the data plane via centralized control and network management capabilities [61]. On the data plane side, the nodes are switches that manage packet routing based on a schedule built by the network controller. Before any redirection choice is visualized, this process executes. According to the SDN's core characteristic, the network controller is constantly aware of the network's status, and all traffic flows are relayed to the controller at least once during the network's lifespan to decide on redirection behavior [62]. SDN, like any new technology, has advantages and problems. The approach to network administration and setup in an SDN environment is different. This infrastructure plays a significant role in network security. This effect can be divided into two cases. In the first case, it can be called "security via SDN", since in this case the centralized logical visibility and programmability of the SDN make it easy to implement and enforce network-wide security policies from the central controller's application. The second aspect of SDN security, which is referred to as “SDN security”, is how to protect or secure the SDN platform itself, as the network architecture is subject to a variety of attacks, so this dissertation focuses on it, specifically reviewing the works that used entropy and ML mechanisms [6]. SDN's centralized network design makes it attractive to a variety of threats, it also motivates the way for the development and implementation of more efficient detection systems [63]. DDoS has been a major attack in the past, and it is always evolving with new methods of attacking the system. As the Internet expands, so does the attack system, rendering more and several hosts exposed to DDoS / DoS attacks. In SDN, various aspects have been employed to

identify threats. High traffic rate and divergence from the typical traffic flooding in the flow are two crucial criteria for detecting a DDoS attack. The majority of SDN IDS are based on these two characteristics [64]. In the forwarding plane, SDN can make decisions about how packets should flow through the network. A controller sends packet handling rules to the switches. The controller is a software application that runs on a remote server. For packet handling, the switches seek guidance from the controller. The controller's southbound interface connects switches and the controller. The OpenFlow protocol facilitates this communication [65]. Similarly, applications can communicate with the controller via the northbound interface. Figure (2.1) depicts the SDN architecture.

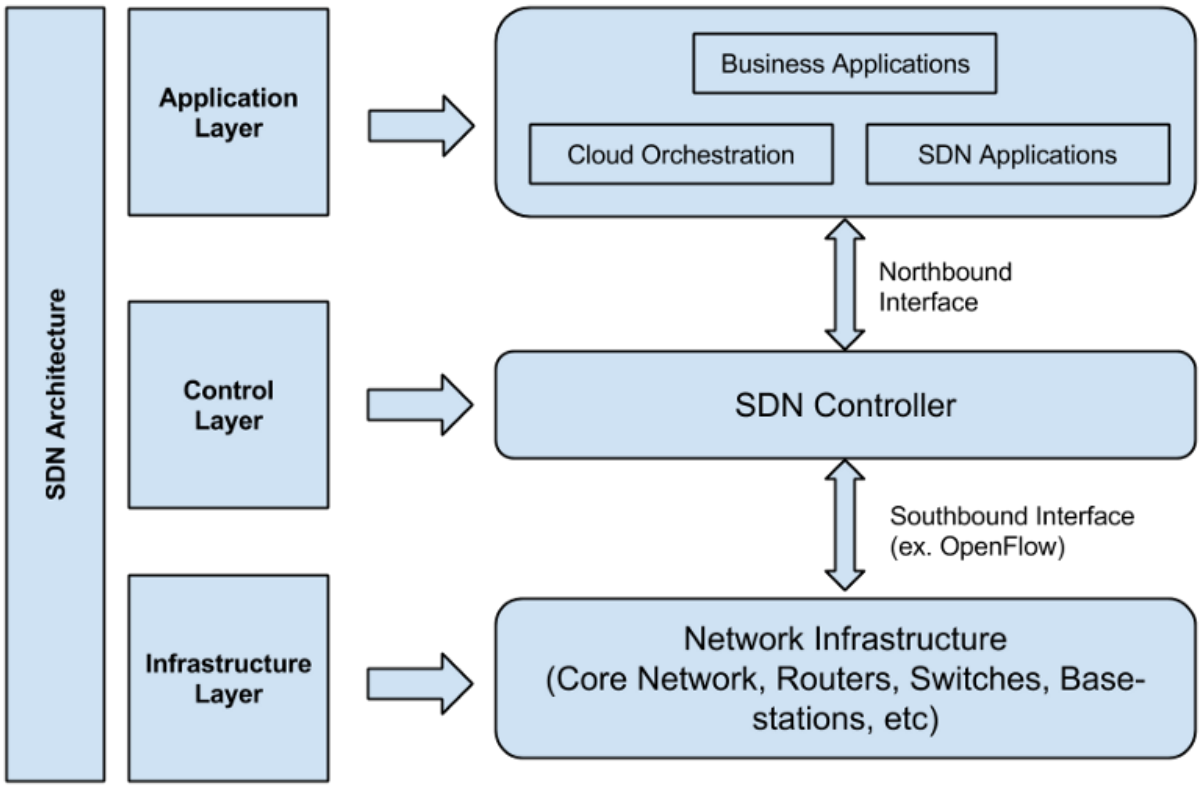


Figure 2-1: The SDN Architecture [66]

The decoupling of the data plane and control plane in SDN is depicted in Figure (2.1). The control plane's job is to decide where to direct the flow of traffic. One or more SDN controllers make up the control plane. The controller is nothing more than a piece of software. The centrally located controller keeps an overall

view of the network in the control plane. OpenFlow switches are just one example of forwarding devices that can be controlled by a single control plane. It can remotely set up all of the data plane's devices and determines the forwarding rules for each one. According to these guidelines, network devices in the data plane forward traffic.

2.1.1 The Benefits of SDN

The decoupled nature of the control plane and data plane makes SDN technology programmable. The controller is a logical entity that gives the global view of the network. It can communicate with both the SDN applications and the hardware network devices about the statistics and events happening [67]. SDN facilitates automated load balancing and it can scale network resources dynamically. The open standard implementation simplifies the network design and operations. It is ideal for today's high-bandwidth applications.

2.1.2 OpenFlow Protocol

The OpenFlow protocol [68] has represented the backbone of SDN. The SDN switches have managed by OpenFlow, the controller can process all flows of network packets by OpenFlow. The tables which exist in the switches explain the ingress and egress paths of incoming packets. OpenFlow has made these tables reachable by controller software. Then, the OpenFlow switch has received flow table entries or flow entries from the controller by the secure channel. Figure 2-2) illustrates a simple SDN network.

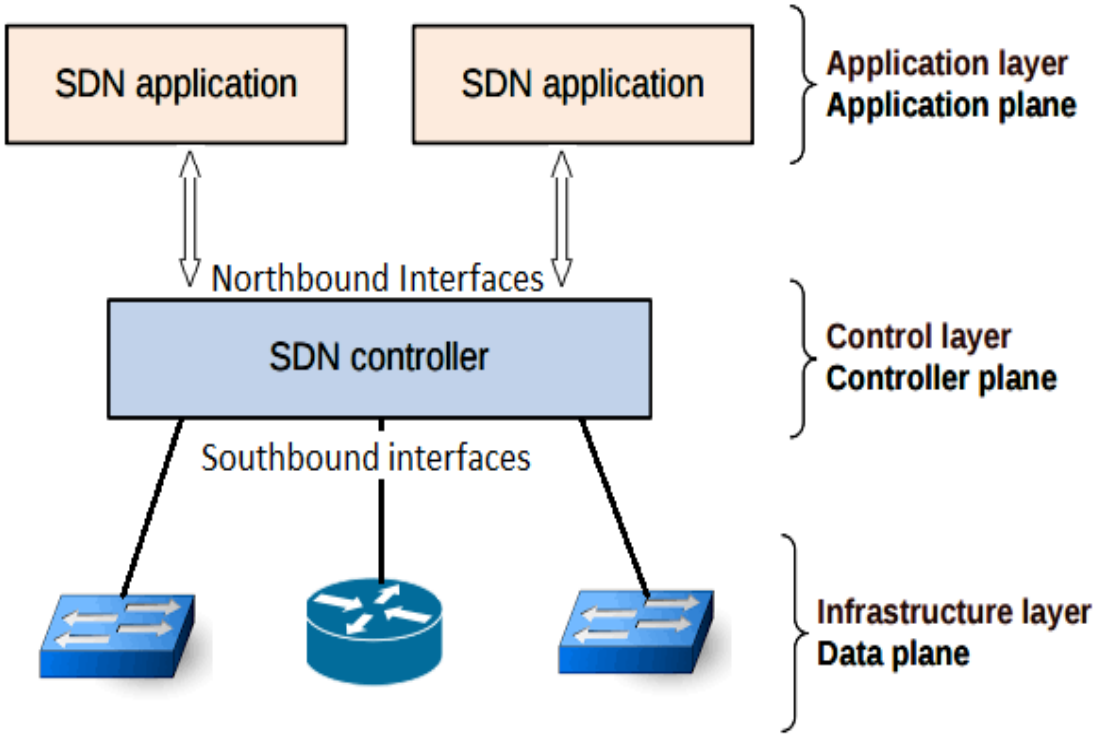


Figure 2-2: The simple SDN network [69]

A. OpenFlow Specification

The OpenFlow specification has encouraged vendors to execute OpenFlow on their devices. The organization which encourages acceptance of SDN and works with many vendors is called Open Networking Foundation, it has various groups to develop OpenFlow specifications. Figure (2.3) explains the architecture of an OpenFlow [70]. Searching, matching, and forwarding according to action determines by the controller software, all rely on the specification of OpenFlow. Version 1.0 has been utilized, which is used for the proposed system.

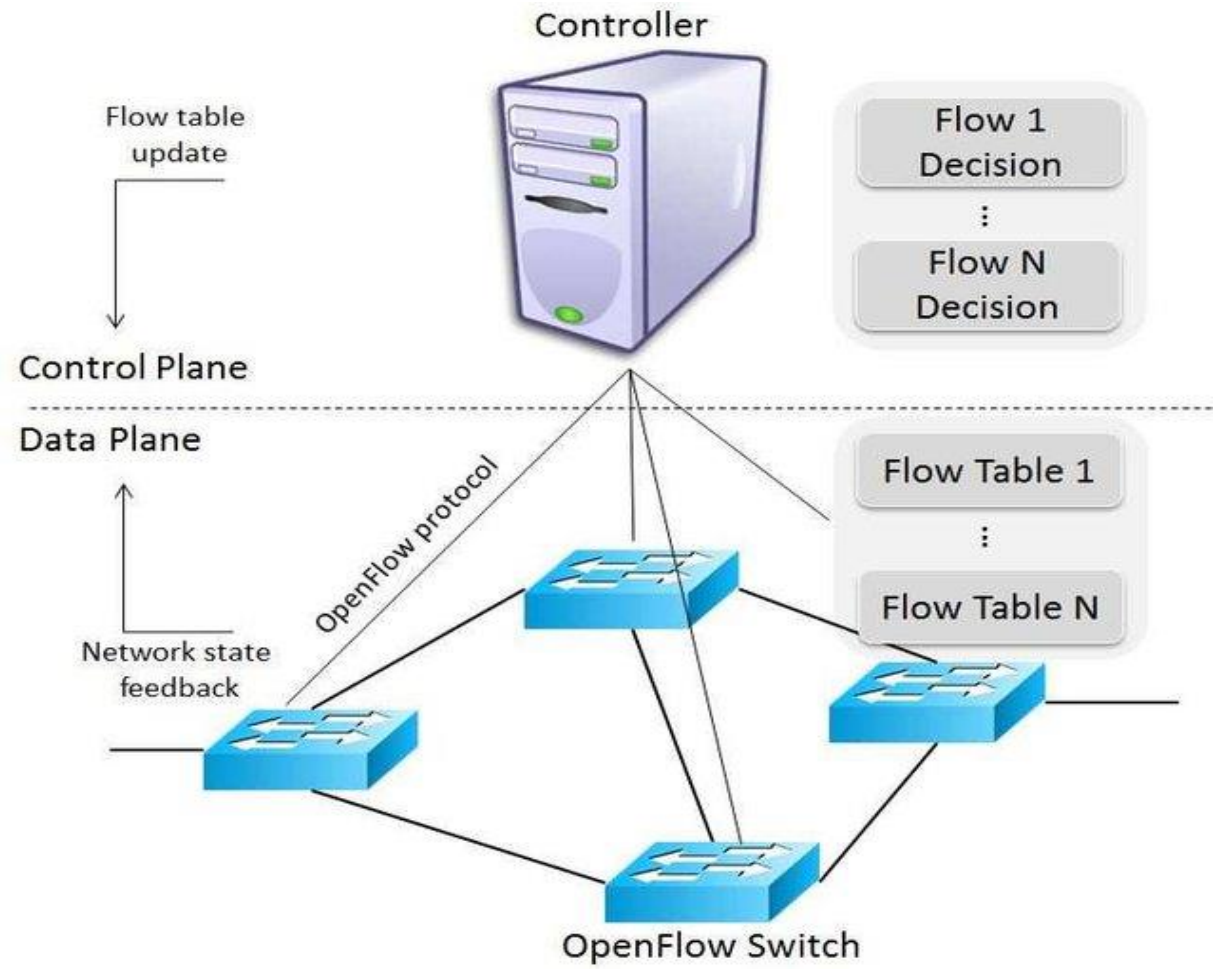


Figure 2-3: The architecture of an OpenFlow

The main process of an OpenFlow is as follows:

1. The controller software puts flow entries in the switch.
2. Incoming packets have been checked by the switch and detected matching flow, then the SDN switch executes the related action.
3. If matching is not found in the switch's flow table, the SDN switch passes the packet to the controller to learn the switch how to treat the packet.
4. After that, the switch with new flow entries will be updated by the controller, then the switch can process the packet locally.

B. Secure Channel

The data plane and logic control plane have been connected by a secure channel. When the switch has activated in the network, the switch looks for the controller to connect to it. Then, the switch builds a secure channel through which

it exchanges information with the controller. The secure channel may be connected to the controller physically or virtually (by another switch remotely), this connection is a TCP connection [71]. To protect mutual data between the controller and switch from penetration, data must be encrypted. In general, transport layer security (TLS) has been used in encryption. If the connection has failed, the SDN switch will search for another controller to connect with it. If there only one controller has been found, it enters in "fail secure mode" then, every unknown packet has been dropped. When the OpenFlow [72] switch processed the unknown incoming packet without sending it to the controller, this type is named Hybrid Switch. In case of a controller failure and the switch depends on his intelligence, it means that the switch has not followed an OpenFlow protocol, which means the network has lost SDN architecture. Figure (2.4) illustrates the secure channel in components of OpenFlow.

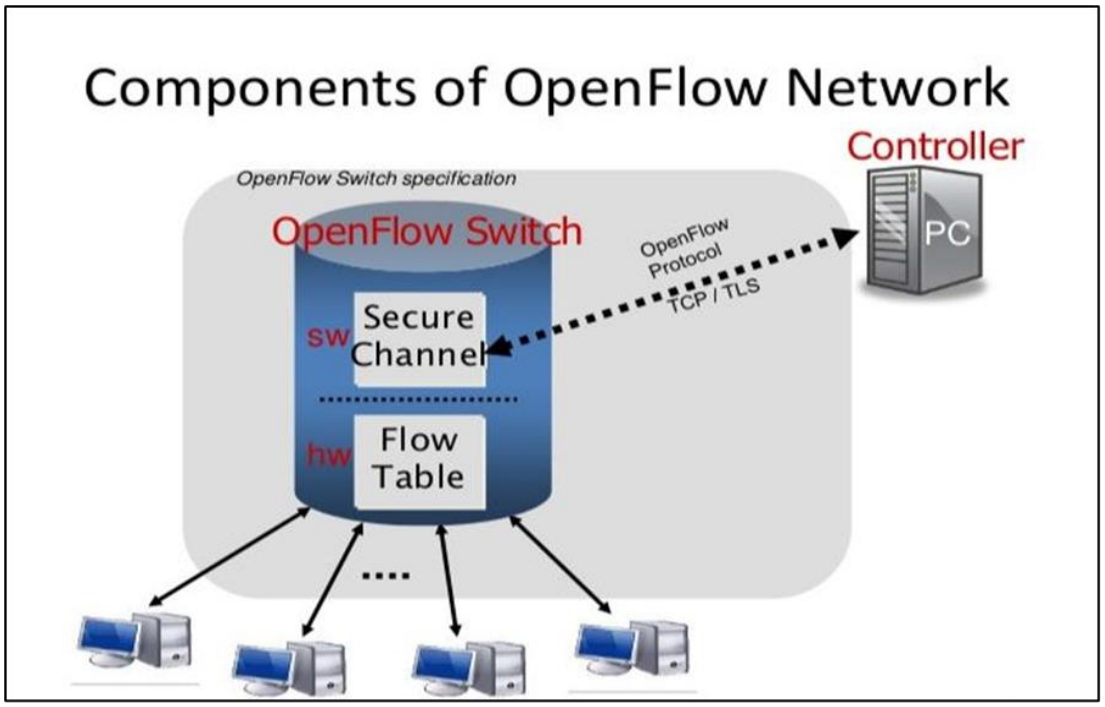


Figure 2-4: The components of OpenFlow [73]

C. Flow Table

The flow table an existing in the switch, it includes flow entries. The fields in OpenFlow version 1.0 as shown in figure (2.5).

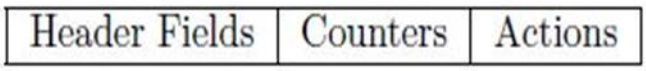


Figure 2-5: The fields of flow table in OpenFlow version 1.0

The fields in a flow table of OpenFlow version 1.3 will increase as shown in figure (2.6). These fields will be explained.

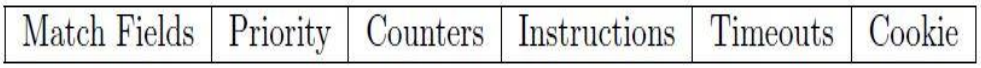


Figure 2-6: The fields of flow table in openflow version 1.3

1. **Match Fields**: match fields of the flow table in OpenFlow version 1.3 are the same header fields of the flow table in OpenFlow version 1.0, these fields were extracted from the packet. These fields are necessary for the lookup process inside the flow table, like Src_mac address, Des_IP address, and so on. Figure (2.7) explains match fields of the flow table. The solution which will be introduced in the proposed system relies on the IPv4 destination, which exists in this table.

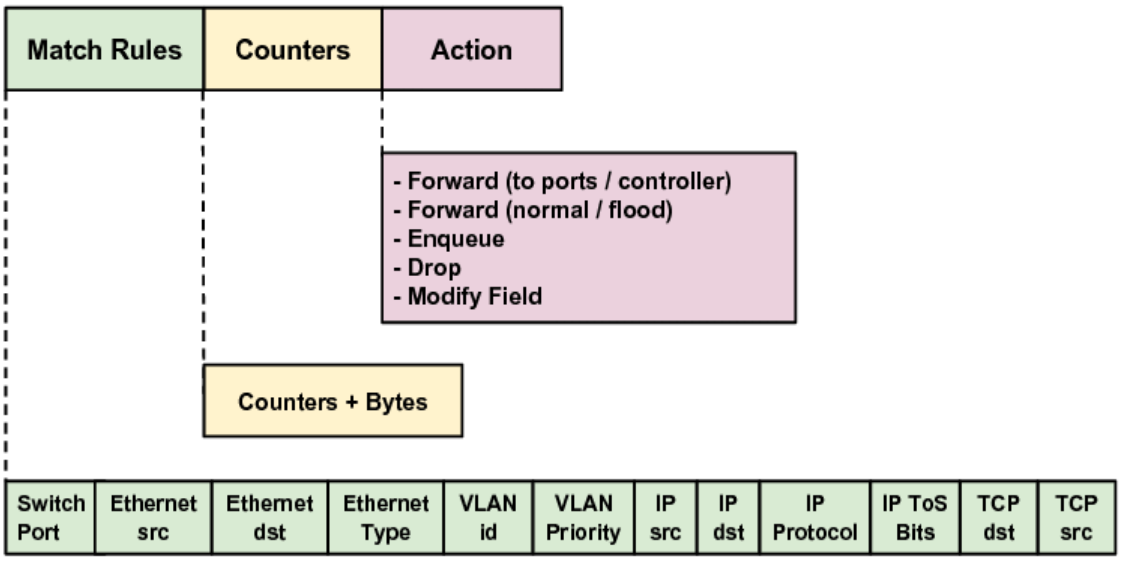


Figure 2-7: The match fields of the flow table [74]

2. **Priority**: the flow rule in SDN means flow entry in the switch's flow table. When the SDN switch searches in all flow rules to match an incoming packet of traffic flow, if there are relevant flow rules, the rule with the highest priority has selected.
3. **Counters**: when any incoming packet has matched the header fields of flow entries in the switch's flow table, the counter will be increased.
4. **Instructions**: this means actions to be taken such as forwarding to a specific port or dropping packets or forwarding to the controller if matching not occurs in the flow table.
5. **Timeout**: flow entries have an idle and hard timeout. Hard timeout means timeout after which the entry is removed from the device. Idle timeout means timeout in which if no packets are hitting to specific entry for the duration, the entry will be removed from the device.
6. **Cookie**: data value has placed by the controller, it has used to filter flow statistics, and flow modification.

D. The OpenFlow Messages

Three kinds of messages [70], [72] have been supported by an OpenFlow.

1. **Controller to Switch Messages:** These messages are called **PacketOut** messages. Which have been created from the controller, and used to configure and manage the SDN switches. This means messages will manage the flow table, such as adding flow entries, or modifying or deleting specific flow entries.
2. **Asynchronous Messages:** These messages are called **Packet_In** messages. Which have been sent from the SDN switches, and been have used to inform the controller about entry must remove, or there have new packets must adding, or inform SDN the controller there a problem has been found in ports.

Symmetric Messages: These messages are called **Hello** messages. Which have been sent from both SDN switches and controller, and have been used to detect any problems between controller and switches. During connection setup, these messages have used for negotiation about the version. When the connection has been started, the controller and SDN switch must instantly transmit a Hello message with putting version field number, it will be agreed upon depending on the highest version supported by the OpenFlow switch. If the version of negotiation has failed, an error message will appear with Hello Failed, this means the version is incompatible.

2.1.3 The SDN controllers

The controller is considered the core of an SDN network. The controller uses protocols like OpenFlow to communicate with networking devices. There are different types of controllers like POX, Ryu, OpenDayLight, Beacon, etc. The various SDN controllers are shown in Figure (2.8).

	POX	Ryu	Trema	Floodlight	Open Day Light
Language Support	Python	Python	C Ruby	Java	Java
OpenFlow Support	v1.0	v1.0 v1.2 v1.3	v1.0	v1.0	v1.0
OpenSource	Yes	Yes	Yes	Yes	Yes
GUI	Yes	Yes	No	Web GUI	Yes
REST API	No	Yes	No	Yes	Yes
Platform Support	Linux Mac Windows	Linux	Linux	Linux	Linux Mac Windows

Figure 2-8: The types of SDN controllers [5]

POX is inherited from the NOX controller [75]. It is an open-source development platform, used to create an SDN controller using the python

programming language. POX controller provides an efficient way to handle OpenFlow devices. Using the POX controller, you can run different applications like a hub, switch, load balancer, and firewall. It is a great tool for SDN research works. The proposed algorithm in this research is implemented in the pox controller.

A. The Distributed SDN Controllers

The current section of the dissertation introduces of main parts of the single controller and the distributed controllers.

One of the most critical research concerns for distributed control systems is the proper management of controllers, which includes the allocation of sufficient controllers to suitable network locations [76]. In recent years, it has become increasingly clear that a novel approach to networking is required to address the diverse problems that are associated with existing networks. Simplifying networking operations, improving network administration, and fostering innovation and flexibility are the primary goals of the SDN method, which was developed as an alternative to traditional approaches to network design.

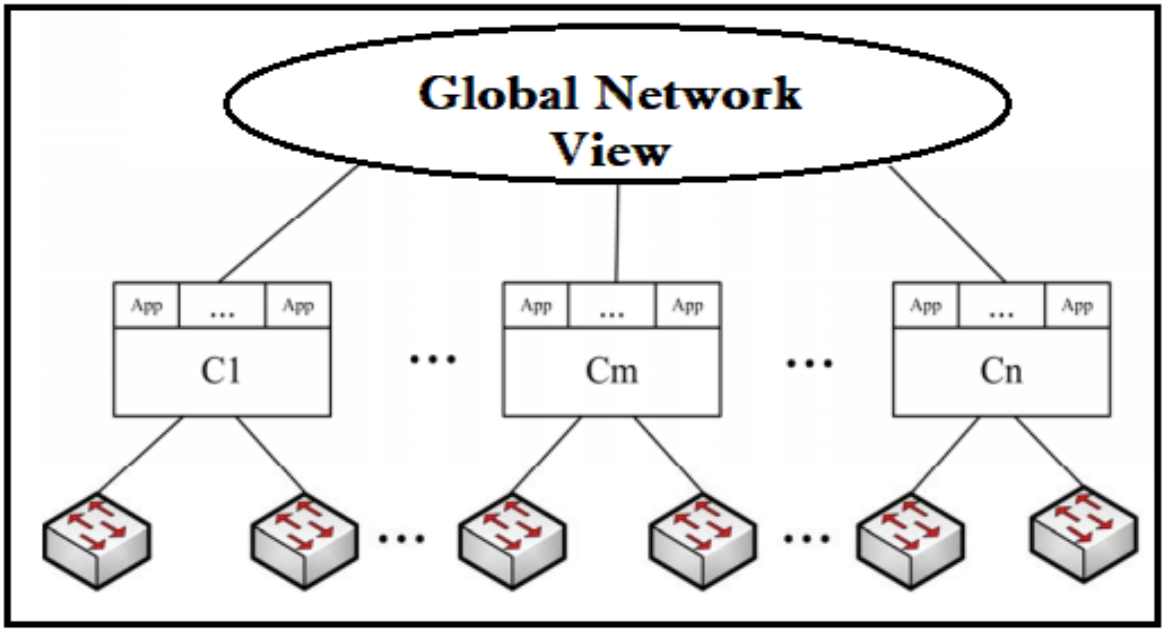


Figure 2-9: The logically centralized controller [77]

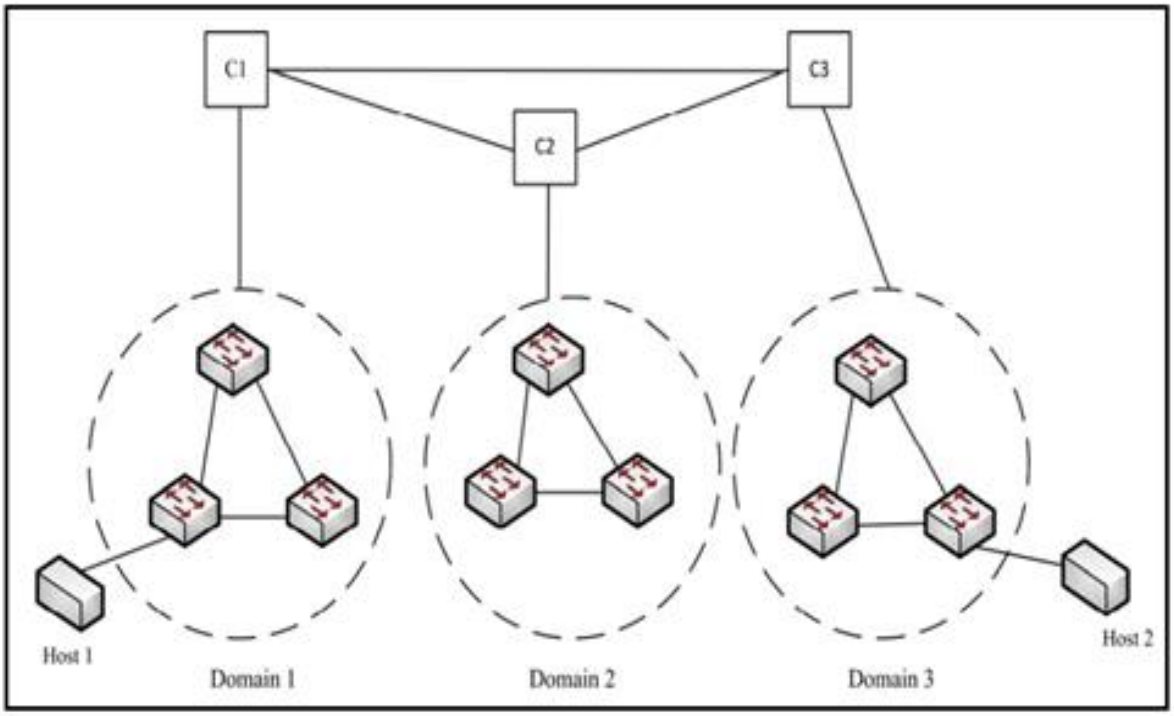


Figure 2-10: The flat distributed SDN controller [77]

Top-down software centralized network control is used in SDN architecture [78]. SDN controllers have SDN applications that are programmed by a network programmer. The centralized SDN controller includes more effective network management and a faster response to dynamic occurrences. There are two implementation approaches for achieving software centralization in SDN: single and distributed SDN controllers (see Figures 2.11).

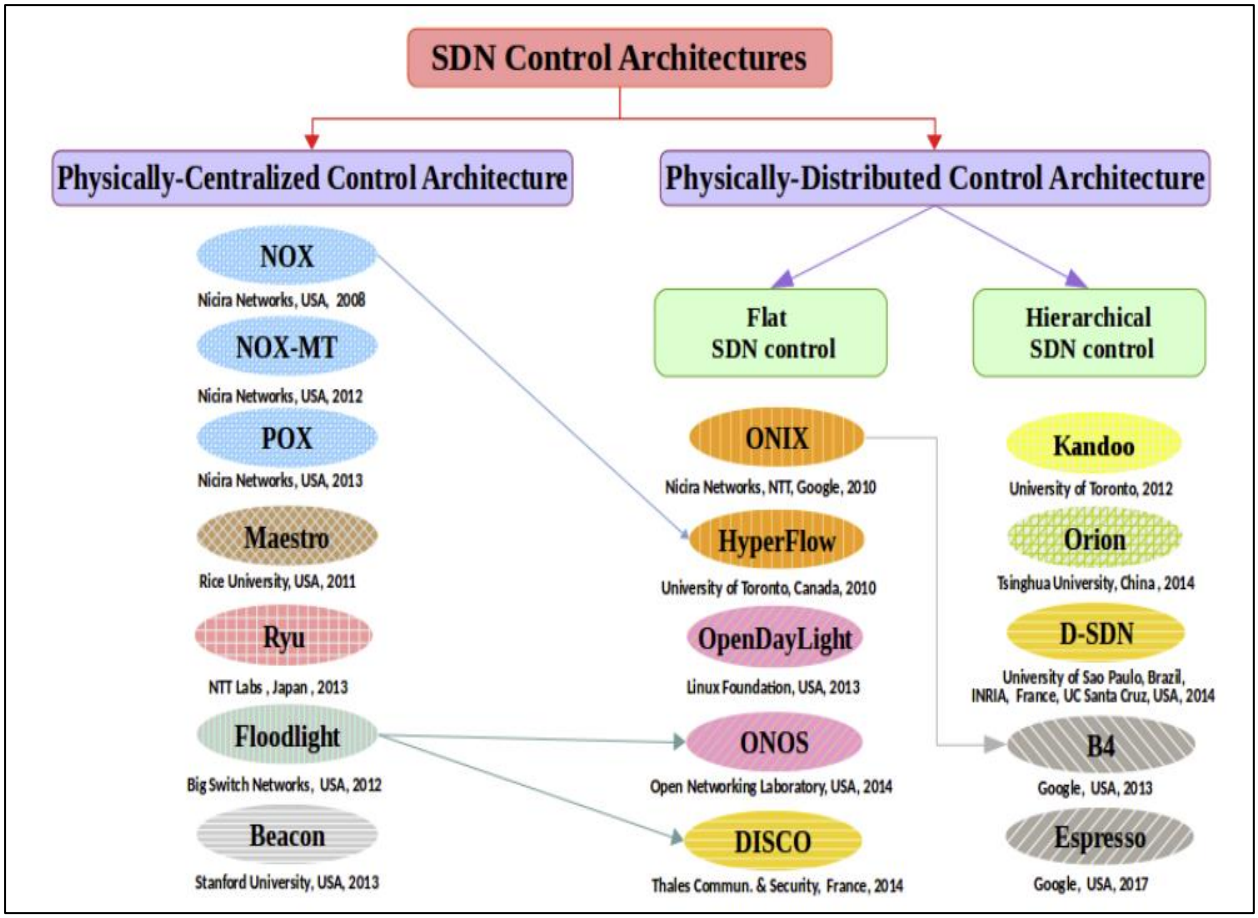


Figure 2-11: The SDN control plane designs are classified physically [79].

1. The Single SDN Controller

There is only one SDN controller per machine in the SDN network. Many early SDN adopters use this variation. The scalability and robustness of this variant are both seriously flawed, which is a shame (single point of failure problem).

If an SDN controller goes down, the SDN switches on the data plane will be unable to process new packets, which would ultimately destroy the entire network. The switch can be configured as an OpenFlow hybrid, which will allow researchers to rectify this issue [78]. When the switch fails to connect to the SDN controller, it will transition from an OpenFlow switch to a conventional switch (i.e., non-SDN switch mode).

2. The Distributed SDN Controller

The distributed SDN controller provides a solution to the issues caused by the single SDN controller. The distributed SDN controller paradigm has been acknowledged for some time as a potential solution for single controller issues. The primary concept is to construct a large number of controllers that can evenly distribute network demand. Moreover, if one controller fails, another can take its place. In recent years, physically distributed control plane architectures have emerged as a possible answer to the issues that are caused by centralized SDN design, such as limited scalability, a single point of failure (see Figure 2.9), performance limitations, etc. As a direct consequence of this, several new designs for SDN control planes have been recently published in publications that are subject to peer review. Based on the physical design of SDN controllers, we discriminate between two basic classes of distributed SDN control architectures: flat SDN control architectures and hierarchical SDN control architectures (see Figure 2.10).

2.1.4 The SDN Challenges

In existing traditional networks, the data and control planes are combined into one node to provide service management to users, including scalability, and reliability.

SDN's principal challenge is to give a total of these abilities while separating data and control planes with fault tolerance capabilities. While separating the control and data planes offers many benefits, it also brings many challenges.

A. Security

Separation of the planes and aggregation of control plane functionality to a centralized system (e.g., OpenFlow controller) may be fundamental to future networks, but it introduces new security challenges. For instance, communication channels between isolated aircraft can be exploited to disguise one aircraft as an

attacker against the other. Due to its visibility, the control plane is more susceptible to security attacks, particularly DoS and DDoS attacks. In the event of a security breach, the SDN controller can become a single point of failure and bring the entire network to a halt. Visibility of network resources is of utmost importance in SDN, but these resources must not be visible to all or uninterested applications. With the gradual deployment of SDN technologies, the list of security challenges is anticipated to grow [63].

B. Scalability

The central control requires a high computing and processing power for the increased network size to avoid response time and performance problems [80]. The SDN controllers have shown that controllers have a limited ability flow rate /sec to handle communication devices at the data plane. One of the common popular controllers, NOX, can only handle 30K flow requests. At the same time, data centers and large network organizations require much fast flow rates, which can exceed 10 million flow /sec, depending on service requirements and size. Therefore, as the network size grows with the number of OpenFlow switches, the central control becomes the bottleneck. Furthermore, the switch in the data plane has a limitation on the hardware, for example, the limited TCAM size, the and weakness of TCAM size that leads to bottlenecks in the network and resource consumption [81]. SDN has a centralized controller, which enables it to improve management control. However, this centralized management characteristic can also become a security threat to the SDN so the SDN controller is the weak target for DDoS attackers. This is especially true in the nonexistence of a secure and robust controller. Within this context, favorable efforts are required to explore innovative ways to ensure the SDN controller's security. Likewise, the communication channels must be secured between the switches and the controller. Every attack on the communication channel between the SDN controller and the OpenFlow switch can cause network damage. The development

and implementation of effective security specifications for the controller and switch interface is too a concern for the SDN community [82].

C. Performance

It refers to the processing speed of the network node, taking into account both throughput and latency [83]. Programmability is provided by SDN's method of handling new packets. Nonetheless, it results in performance issues. The authors of [84] demonstrate that current controllers are incapable of managing a large number of flows on 10 Gbps links. Therefore, additional research is required to determine how to enhance performance and preserve programmability.

D. Interoperability

For full SDN deployment, there is a must to replace conventional network devices with other devices that support the SDN. For a closed environment of the network, such as data centers or campus networks, it is easier to conduct such a swap with less interference. However, in an open network that supports many vital companies and systems, this is difficult to replace these network devices right away [85].

E. Centralization

Network devices have become simple forwarders as a result of the drastic change brought about by SDN to the traditional network model, and the SDN controller is a critical software component. At the same time, updating network policies, and firewalls, and monitoring the entire network from a logically central location is much easier. The SDN controller is also regarded as a single point of failure that, in the event of a DDoS attack, can cause the entire network to go down. If one or more network devices fail or a connection between them fails in a traditional network, network traffic can take an alternate path to maintain flow continuity. Packets traveling on the downlink will be dropped in the SDN. The controller recognizes them and updates their flow entries. Furthermore, if the

switches lose contact with the controller, they lose their ability to decide on an incoming packet [81].

F. Reliability

The failure of the SDN controller may result in a single node failure, and, for this reason, it must check network topology and intelligently configure to prevent breakdown also increase availability. When any network device fails at traditional networks, packets can pass through alternative devices without interrupting the flow continuity. In devices, with SDN central controller environment, in the nonexistence of a backup controller, the breakdown of the SDN controller can lead to the down of the entire network. Also, the controller must have the capability to support a multipath solution in link failure. Controller redundancy can be utilized to solve this problem. At least one controller must be used in active standby. While one controller is down, the other must work, maintaining synchronization and coordination between them [85].

2.2 The Distributed Denial of Service Attacks

An online service or network is at risk of a DDoS attack in terms of security. This dissertation, address the study issue of DDoS attacks and offer several frameworks for its resolution. DDoS assaults try to reduce a service's availability by using up all of the network or computational resources available for traffic or computation/processing, which prevents authorized users from using the services of its victims [86].

2.2.1 How a DDoS Can be Launched

A DDoS attack can be started in many ways [87], but the most common way is for an attacker to send a stream of packets to a server that is being attacked. This uses up important resources and makes it hard for real users to get to these resources. Sending a few bad packets that cause the victim servers to freeze or restart is another common method. Another way to stop a service from working is to take over machines in the victim network and use up key resources. This

makes the network unavailable for internal or external service. There are many other ways to do these kinds of attacks [88], and they are hard to predict. Often, they are only found out after they have already happened. Figure (2.12) shows DDoS attack scenario.

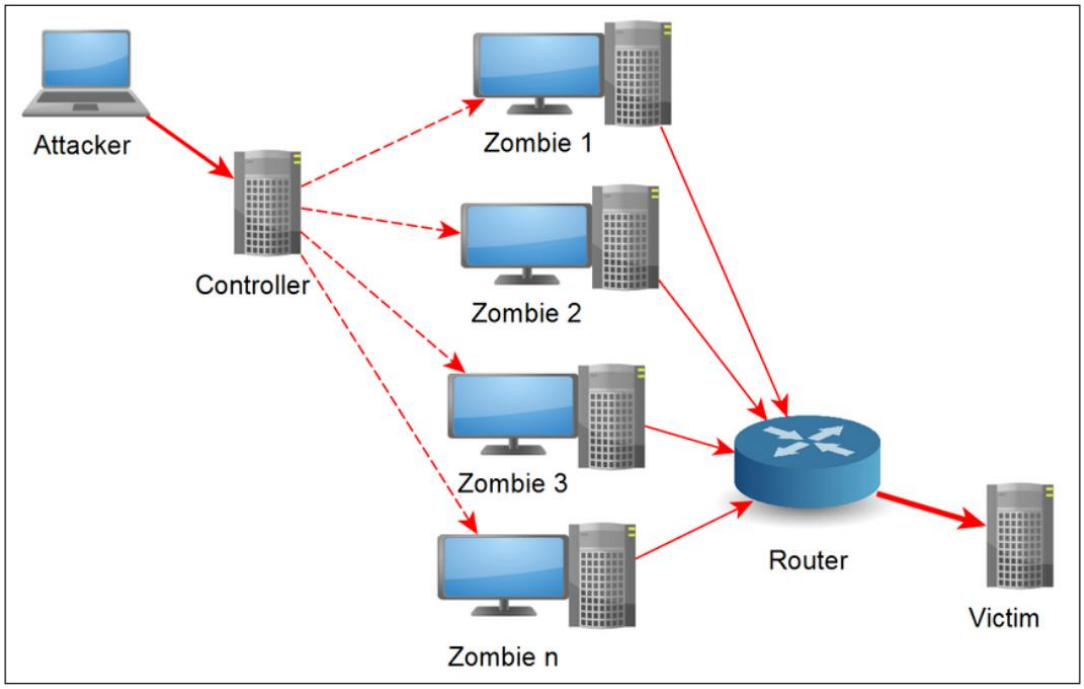


Figure 2-12: The attacks that use DDoS [89]

A DDoS attack is done in stages and involves four main things: the attacker, the controller, the zombies, and the victim. To start an attack, the attacker looks for machines with open ports that can be hacked from afar. Once the weakness is found, the attacker sends a piece of malicious code that, when run on the machine being attacked, copies itself and starts the attack. Malware can also be spread by making it look like a normal internet packet. For example, it can be attached to an email. All of this can be controlled from far away by the attacker. Except reflecting attacks, spoofing is used to make it harder to find and describe attacks so that agent machines don't get found [89]. In the proposed system, to generate attack packets will be used scapy tool [90], in addition the proposed algorithm (3.1).

2.2.2 The Types of DDoS Attacks

DDoS attacks [91] can be classified into Volume-based attacks, Protocol attacks, and Application layer attacks. In volume-based attacks, a victim's server is directly flooded with Internet packets to carry out the attack. UDP flooding, spoofed packet flooding, and ICMP flooding are methods for achieving this. Fragmented packets, Ping of Death, and SYN Flooding are all examples of Protocol-Based Attacks. Protocol attacks use up server resources on the Internet and burden it with more traffic. Attacks on the application layer, like GET/POST Floods, focus on particular security protocol flaws. Now each category's representative attack is described [91], [92], [93]:

A. UDP Flood Attack

In contrast to Transmission Control Protocol (TCP), User Datagram Protocol (UDP) does not require a handshake before sending a packet to the target server. The attacker overwhelms the target server's network resources by sending a lot of traffic using this protocol property. UDP floods are used frequently for larger bandwidth DDoS attacks because they are connectionless and it is easy to generate protocol 17 (UDP) messages from many different scripting and compiled languages.

B. SYN Flood (TCP/SYN)

The TCP protocol requires a three-way handshake between the server and client to establish a connection (SYN, ACK). The most common way that a client can launch an SYN attack is by sending the server an incorrect ACK message, which includes a forged IP address. The server responds to the wrong IP address Synchronize (SYN) message and waits for a response from the client before continuing. There is a pause in service while the connection waits for a valid user.

SYN Flood works by establishing half-open connections to a node. When the target receives a SYN packet to an open port, the target will respond with a SYN-ACK and try to establish a connection. However, during a SYN flood, the three-way handshake never completes because the client never responds to the

server's SYN-ACK. As a result, these "connections" remain in the half-open state until they time out.

C. Ping of Death (POD)

Internet Control Message Protocol (ICMP) ping flood attacks used to be known as POD. IPv4 has a maximum packet size of 65,535 bytes, which can be sent between two devices. An unpatched system can be severely damaged by sending malformed or oversized packets via a simple ping command.

D. Denial of Sleep Attack

Denial-of-sleep (DoSL) attack is a special category of denial-of-service attack that prevents the battery powered sensor nodes from going into the sleep mode, thus affecting the network performance [94].

E. DHCP Starvation Attack

A DHCP starvation attack results in a denial of service (DoS) for legitimate clients making IP address requests from an overloaded DHCP server. To perform this attack, the attacker sends tons of bogus DHCP Discover messages with spoofed source MAC addresses. The DHCP server tries to respond to all these bogus messages, and as a result, the pool of IP addresses used by the DHCP server is depleted.

F. HTTP Header

HTTP headers are fields which describe which resources are requested, such as URL, a form, JPEG, etc. HTTP headers also inform the web server what kind of web browser is being used. Common HTTP headers are GET, POST, ACCEPT, LANGUAGE, and USER AGENT. The requester can insert as many headers as they want and can make them communication specific. DDoS attackers can change these and many other HTTP headers to make it more difficult to identify the attack origin. In addition, HTTP headers can be designed to manipulate caching and proxy services. For example, is it possible to ask a caching proxy to not cache the information.

G. HTTP POST Flood

An HTTP POST Flood is a type of DDoS attack in which the volume of POST requests overwhelms the server so that the server cannot respond to them all. This can result in exceptionally high utilization of system resources and consequently crash the server.

H. HTTP POST Request

An HTTP POST Request is a method that submits data in the body of the request to be processed by the server. For example, a POST request takes the information in a form and encodes it, then posts the content of the form to the server.

I. HTTPS POST Flood

An HTTPS POST Flood is an HTTP POST flood sent over an SSL session. Due to the use of SSL it is necessary to decrypt this request in order to inspect it.

J. HTTPS POST Request

An HTTPS POST Request is an encrypted version of an HTTP POST request. The actual data transferred back and forth is encrypted.

K. HTTPS GET Flood

An HTTPS GET Flood is an HTTP GET flood sent over an SSL session. Due to the SSL, it is necessary to decrypt the requests in order to mitigate the flood.

L. HTTPS GET Request

An HTTPS GET Request is an HTTP GET Request sent over an SSL session. Due to the use of SSL, it is necessary to decrypt the requests in order to inspect it.

M. HTTP GET Flood

An HTTP GET Flood is a layer 7 application layer DDoS attack method in which attackers send a huge flood of requests to the server to overwhelm its resources. As a result, the server cannot respond to legitimate requests from the server.

N. HTTP GET Request

An HTTP GET Request is a method that makes a request for information for the server. A GET request asks the server to give you something such as an image or script so that it may be rendered by your browsers.

O. ICMP Flood

Internet Control Message Protocol (ICMP) is primarily used for error messaging and typically does not exchange data between systems. ICMP packets may accompany TCP packets when connecting to a sever. An ICMP flood is a layer 3 infrastructure DDoS attack method that uses ICMP messages to overload the targeted network's bandwidth.

P. MAC Flood

A rare attack, in which the attacker sends multiple dummy Ethernet frames, each with a different MAC address, Network switches treat MAC addresses separately, and hence reserve some resources for each request. When all the memory in a switch is used up, it either shuts down or becomes unresponsive. In a few types of routers, a MAC flood attack may cause these to drop their entire routing table, thus disrupting the whole network under its routing domain.

2.3 The ML-Based Approaches

ML is an Artificial Intelligence application that allows a computer program to learn from input data [95]. It enables us to predict future data using historical data as input. As a result, the output's accuracy is solely determined by the quality of the historical data. ML techniques are now used to solve a variety of problems in a variety of fields. They are used in DDoS detection, email spam filtering, pattern, image recognition, search engine filtering, healthcare applications, and other applications.

ML algorithms can be broadly classified as Supervised learning algorithms, Unsupervised learning algorithms, and semi-supervised techniques.

2.3.1 The Supervised ML

Supervised learning algorithms are mainly used to solve classification and regression problems as it makes the detection or decision-making process easier. It uses the past learned data to predict future events. Our dissertation problem implements a supervised ML algorithm to classify network traffic into legitimate traffic and DDoS traffic. Here the classifier gets the input which is a set of feature values also called input vector and outputs the predicted value called class. Figure (2.13) shows the supervised learning classifier.

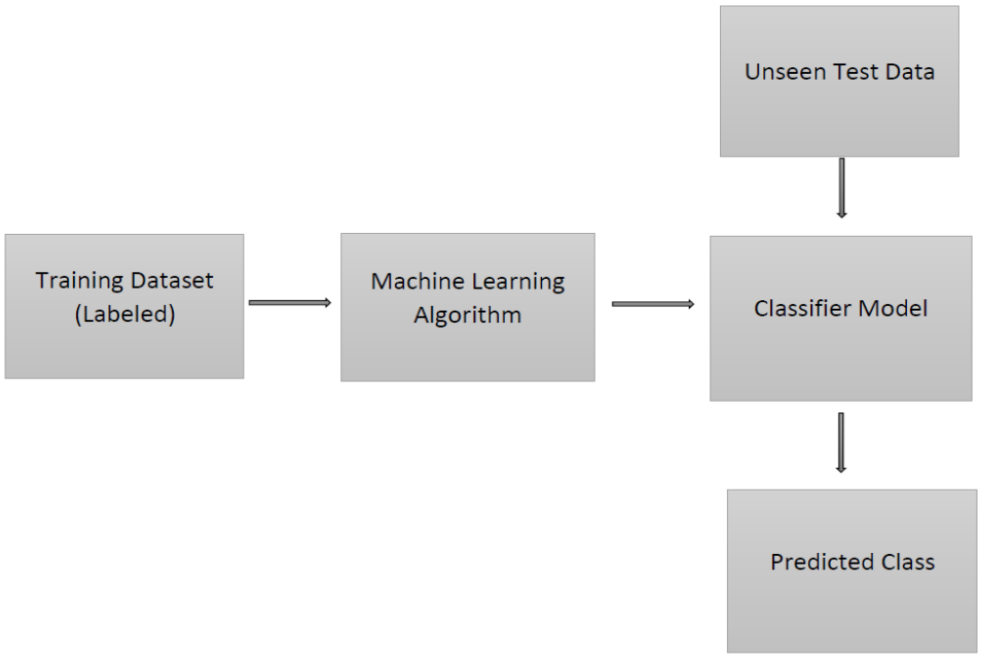


Figure 2-13: The supervised learning classifier [65]

Here the training data are given as an input to the learning algorithm which results in a classifier model. The performance of the classifier can be evaluated using unseen data. This dissertation applies six ML classifiers to complete the binary classification challenge.

A. J48 or C4.5

J48, often known as C4.5, is a popular ML method that is incorporated in the decision tree algorithm. Using the entropy notion, this method constructs a decision tree from a training dataset [96]. It varies from IDE3 in that it constructs

a decision tree, whereas J48 or C4.5 that accept both continuous and categorical characteristics. This approach has been used in investigations of DDoS identification by [97], [98].

Steps in C4.5 algorithm:

1. It begins with the original set S as the root node.
2. On each iteration of the algorithm, it iterates through the very unused attribute of the set S and calculates **Entropy(H)** and **Information gain (IG).**
3. It then selects the attribute which has the smallest Entropy or Largest Information gain.
4. The set S is then split by the selected attribute to produce a subset of the data.
5. The algorithm continues to recur on each subset, considering only attributes never selected before.

B. REP Tree

The REPT is a fast decision tree algorithm based on the C4.5 algorithm that can be used to create classification (discrete result) or regression trees (continuous outcome). It builds a regression/decision tree using information gain/variance and prunes it using reduced-error pruning (with back-fitting). This method has been employed in the detection of anomalies in studies such as [99].

C. Random Tree

An RT algorithm is a decision tree that is constructed using a set of random qualities (random). A decision tree consists of nodes and branches. A test attribute is represented by a node, and the results are represented by branches. The decision leaves represent the conclusion reached after computing all characteristics in the form of class labels [100]. This approach has been used in the identification of anomalies in research such as [99].

D. Decision Stump

Stump Decision [101] is an algorithm for creating basic binary DSs (1 level decision trees) for both nominal and numerical classification applications. It handles mission values by extending the third branch from the stump or treating'

missing' as a distinct attribute value. It may do regression (based on mean-squared error) or classification (based on entropy).

E. Random Forest

Because the RF classifier employs an ensemble of Decision Trees [101], it achieves high prediction performance for classification problems. The numerous decision trees help to categorize in such a manner that each tree in the forest decides which class the new instance should be allocated. The majority vote will determine the class designation for the new class. When the number of trees involved in decision-making is increased, accuracy improves. Before applying the classifier to the datasets, the number of trees must be specified.

F. Partial Decision Tree

There are many schemes to generate rules from decision trees. The C4.5 and RIPPER are two main schemes for rule learning. The both schemes operate in two stages. The C4.5 first induces an initial rule set and then it refines rule set using complex optimization stage by discarding the individual rule. The RIPPER do same thing by adjusting individual rules. These two schemes can be combined to produce good rule sets. This combination of two scheme of rule learning is called as Partial Decision Tree (PART). This combined scheme does not require any complex optimization stage. The algorithm to combine C4.5 and RIPPER is very simple, effective and straightforward. Initially, it built a pruned decision tree for current set of instances. The leaf (best) with largest coverage is converted into rule, and decision tree is discarded by removing covered instances form training dataset. This process is repeated for all set of instances of training dataset. This process is called as separate-and-conquers strategy. PART algorithm produces rule sets which are more accurate than RIPPER's rule set. PART's rule sets are as accurate as C4.5's rule set and the size of rule sets of PART are of less size of C4.5 rule set [102].

2.3.2 The Unsupervised ML

Unsupervised learning algorithm uses unlabelled input data to train the system. That is, the input data are not tagged with labels. It finds the hidden structure from the unlabelled input, and groups them as clusters showing the similarities. The initial performance of this type of learning algorithm is poor, but the system can tune itself to improve the performance.

2.3.3 The Semi-Supervised ML

Semi-supervised learning refers to a group of ML tasks and strategies that combine labelled and unlabelled data for training, often combining a small quantity of labelled data with a large amount of unlabelled data. Between unsupervised and supervised learning is semi-supervised learning. Many machine-learning researchers have shown that combining unlabelled data with a modest quantity of labelled data can enhance learning accuracy significantly over unsupervised learning without the time and cost of supervised learning. In a semi-supervised learning process, there are two steps: first, the general rule is examined using labelled data, and then the rule is utilized to infer unmarked data. Semi-supervised learning's performance is currently insecure and has to be improved [103].

A. The Semi-Supervised Clustering Algorithm

K-means algorithm is based on iterative relocation that partition a dataset into k clusters, locally minimizing the average squared distance between the cluster data points and the cluster centers. K-means algorithm, as a clustering method, has been successfully used to detect anomalies and DDoS [104]. The objective function based on this distance has been shown as Equation (2.1). The k-means semi-supervised uses the small amount of labelled data to guide the selection of initial cluster centers, and use other unlabelled data to train and form clusters [42]. Algorithm (2.1) shows the overall steps of the k-means algorithm.

Algorithm (2.1): k-means

Input (Data *D*, Number of clusters k)
Output (Clusters and Centroids)

1. Began
2. choose *k* data points as the initial centroids (cluster centers);
3. Repeat
4. For each data point x € *D* do;
5. Compute the distance from X to each centroid using Equation (2.1);
6. assign X to the closest centroid;
7. End For
8. Re-Compute the centroid using the current cluster memberships;
9. Until the stopping criterion is met;
10. End Algorithm

B. The Semi-Supervised Distance Equation

The semi-supervised clustering [105] mainly uses the Euclidean distance (see Equation 2.1), Manhattan distance, Murkowski distance, and Cosine distance in distance measurement.

$$Dist(X_i, X_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{id} - x_{jd})^2} \quad \ldots (2.1)$$

Where, *d*: number of dimensions (features).

X_i, X_j: represents two objects.

2.3.4 The Feature Selection Methods

One of the most common problems researchers' encounters is choosing which features are most important and thus relevant for use in detecting attacks. Feature selection is critical because it affects how well the system works. Too few features may lead to subpar detection accuracy, while too many may lead to excellent detection accuracy at the expense of an overly complex system that eats up more resources. This study employed two attractive features selection techniques.

A. The Variance Filter Feature Selection Technique

The Low Variance Filter method [24] was used to choose the features that were used in this study, since all of the attributes were numbers. The method was

used to exclude features with low variances that contributed little or nothing to the model's overall performance. Calculating the variance of each characteristic is involved (Equation 2.2).

$$\text{Variance } (\sigma^2) = \frac{\sum_{i=1}^{n}(X_i - \mu)^2}{N} \quad \dots (2.2)$$

where μ is the average of all the values that are associated with the attribute. The attribute values, denoted by X_i, are taken from a collection of data, where N is the total number of samples.

$$Mean() = \begin{cases} F(\frac{n+1}{2}), & when\ n\ is\ odd \\ f\left(\frac{n}{2}\right) + f(\frac{n+1}{2}), & otherwise \end{cases} \quad \dots (2.3)$$

where F contains all the flows for the interval

B. The Information Gain

Due to its usefulness and importance in detecting a class type, the Information Gain Ratio (IGR) [106] is also employed as a weight for attributes in this study (see Equation2.4).

$$IGR(Y, A_j) = \frac{H(Y) - H(Y|A_j)}{H(A_j)} \quad \dots (2.4)$$

Where *Y* represents the class and *Aj* the *jth* attribute. The entropy function, *H (.)*, is defined (see Equation 2.5):

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i) \quad \dots (2.5)$$

Given an input, the probabilities can be expressed as where *P(.)* is the probability operator and *i* is an index of the probabilities.

2.3.5 The Pruning Methods for Decision Tree

There are two types of pruning methods: post-pruning and pre-pruning. Although the ID3, C4.5 decision trees are accurate and efficient, they frequently produce very large trees that are incomprehensible to experts [8]. Pruning a

decision tree is a critical step in improving computational efficiency and classification accuracy. Pruning reduces the size of the tree, avoids unnecessary complexity, and prevents overfitting of data sets when classifying new data. Overfitting can result in an abnormally large number of rules, many of which have little predictive value for previously unseen data [14].

After the tree has grown, post pruning is performed. In practice, post-pruning methods outperform pre-pruning methods [37]. Backward pruning is another term for post-pruning. Create the decision tree first, then remove non-significant branches.

Pre-pruning is also known as forward or online pruning. Pre-pruning reduces the formation of insignificant branches. Pre-pruning a decision tree entails deciding when it is desirable to terminate some of the branches prematurely as the tree is generated.

The minimum no of object (minobj) is one of the pre-pruning methods. Whenever the split is made which yields a child leaf that represents less than minobj from the data set, the parent node and children node are compressed to a single node [107].

There are two methods for post-pruning, which are discussed in the following subsections.

A. RIPPER

One of the most efficient and widely used rule learning algorithms is RIPPER (Repeated Incremental Pruning to Produce Error Reduction). To rule induction, it employs a divide-and-conquer strategy. Ripper uses a technique known as Incremental Reduced Error Pruning (IREP) to generate an initial set of rules for each class. Then, in a subsequent optimization step, each rule in the current set is considered in turn and two alternative rules are created from them: a replacement rule and a revision rule [108].

B. The Minimum Description Length

The Minimum Description Length (MDL) concept is a strong inductive inference approach that serves as the foundation for statistical modeling, pattern recognition, and ML. It asserts that the optimal explanation, given a restricted collection of observable facts, is the one that allows for the most data compression. MDL approaches are especially well-suited for dealing with model selection, prediction, and estimation problems where the models under consideration might be arbitrarily complex and overfitting of the data is a severe concern. The basic idea of the MDL-based approach to pruning is that a subtree should be pruned if the description length of classification of training instances given the (whole) tree plus the description length of the (whole) tree is greater than if the subtree is pruned [109].

2.4 The Information Entropy Based Approach

Entropy is an information-theoretic important concept, it is defined by Shannon in 1948. It is a beneficial tool for inspecting the similarity and distribution of a given flow at a router in traditional networks [110]. It is a measuring of the uncertainty or unpredictability in a random variable. If the value of entropy is nearby to (0), this means there is a high likeness in the sample and low level of uncertainty. Otherwise, if the value of entropy is nearby to (1), this means there is a low likeness in the sample and a high level of uncertainty. The equation of entropy (2.6) as seen:

$$H(X) = -\sum_{i=1}^{n} p(xi)\, log\, p(xi) \quad \ldots (2.6)$$

Entropy measures have seen a great promise in discovering a various set of network anomalies. So, the entropy and the value of threshold which based on other measures will be used to detect and prevent flow table overloading attack in SDN.

Some researchers proposed a fast method for calculating entropy depend on the volume and type of network packet. This method uses the time interval,

which is called window size to calculate entropy values. In this way, false positive is lower, but if the attacker wanted to hide their bheavier by sending attack packets in the same speed normal packets, so this method doesn't appropriate [110].

Other researchers introduced a method based on the time interval as well. But this method has used three levels to determine a threshold. This method has decreased false negative and false positive in a network, but it has consumed more resources and time according to the view of researchers [111].

2.5 The Performance Evaluation

The performance and effectiveness of the proposed system are evaluated by several metrics. A confusion matrix is a table that is often used to describe the performance of the proposed system. A binary confusion matrix is described in Table (2.1).

Table 2-1: A Binary Confusion Matrix

		Predicted Classes	
		Anomaly	Legitimate
Actual Classes	Anomaly	True Positive (TP)	False Negative (FN)
	Legitimate	False Positive (FP)	True Negative (TN)

- TP: the number of anomaly records correctly classified.
- TN: the number of normal records correctly classified.
- FP: the number of normal records incorrectly classified.
- FN: the number of anomaly records incorrectly classified.

For the evaluation purpose, Accuracy, Precision, Recall and F1-score are applied. These metrics are calculated as follows:

Accuracy is showing the percentage of true detection over total traffic trace. To calculate accuracy, through the proportion between the number of

packets correctly classified whether these packets normal or attack over a total number of the packets which classified by the proposed system correctly and incorrectly, as shown in Equation (2.13).

$$Accuracy = \frac{number\ of\ true\ classifications}{total\ number\ of\ classifications} = \frac{TP + TN}{TP + FP + TN + FN} \quad \dots (2.13)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (2.14)$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad (2.15)$$

$$\text{F Score} = 2\ \frac{\text{Precision} * \text{Recall}}{\text{Precision+ Recall}} \quad (2.16)$$

The efficiency of the proposed system has been evaluated through the number of measures based on the confusion matrix. These measures have mentioned [112]:

Attack Detection Rate (ADR):

To calculate Attack Detection Rate (ADR), through the ratio between the total number of attack detected packets through the proposed system to the total number of attack packets, as shown in Equation (2.17).

$$\text{ADR} = \frac{\text{attack detected packets}}{\text{total number of attack packets}} = \frac{\text{T}N}{\text{TN} + \text{FN}} \quad \dots (2.17)$$

False Alarm Rate (FAR):

To calculate False Alarm Rate (FAR), through the proportion between the number of normal packets incorrectly categorized as attack packets to the total number of actual normal packets, it is also named **false discovery rate** (see Equation 2.18).

$$\text{FAR} = \frac{\text{number of misclassified normal packets}}{\text{total number of normal packets}} = \frac{\text{FP}}{FP + TP} \quad \ldots (2.18)$$

False Positive Rate (FPR):

To calculate **False Positive Rate (FPR),** through the proportion between the number of normal packets incorrectly categorized as attack packets to total packets which classified as attack packets, as shown in Equation (2.19).

$$FPR = \frac{FP}{FP+TN} \quad (2.19)$$

Finally, in order to success overloading detection algorithm, must be high accuracy, low FAR and high ADR.

2.6 The Advanced Message Queuing Protocol (AMQP)

AMQP is a message-oriented middle application layer protocol that is both open and free to use. AMQP is a messaging protocol that offers message routing, queuing, and reliability, in addition to security.

AMQP is a protocol that is used at the application layer and it enables a wide variety of messaging applications as well as communication patterns. As seen in Figure (2.14), due to the fact that it is a network protocol, the publishers, clients, and intermediates can all be located at separate workstations. It enables flow-controlled and message-oriented communication in addition to providing various message delivery guarantees, such as most of the time (where a message is only delivered once), at least once (where each message is guaranteed to be delivered but can be delivered multiple times), and exactly once (where the message will always arrive and only arrive once). In addition to this, it enables

authentication and/or encryption based on TLS [113]. It relies on a dependable transport layer technology like Transmission Control Protocol (TCP).

The communication protocol (AMQP) that has a key role in a proposed solution to enabling some of SDN controllers to work in distributed controllers' mechanism for overcoming the problem of a ***single controller of failure***

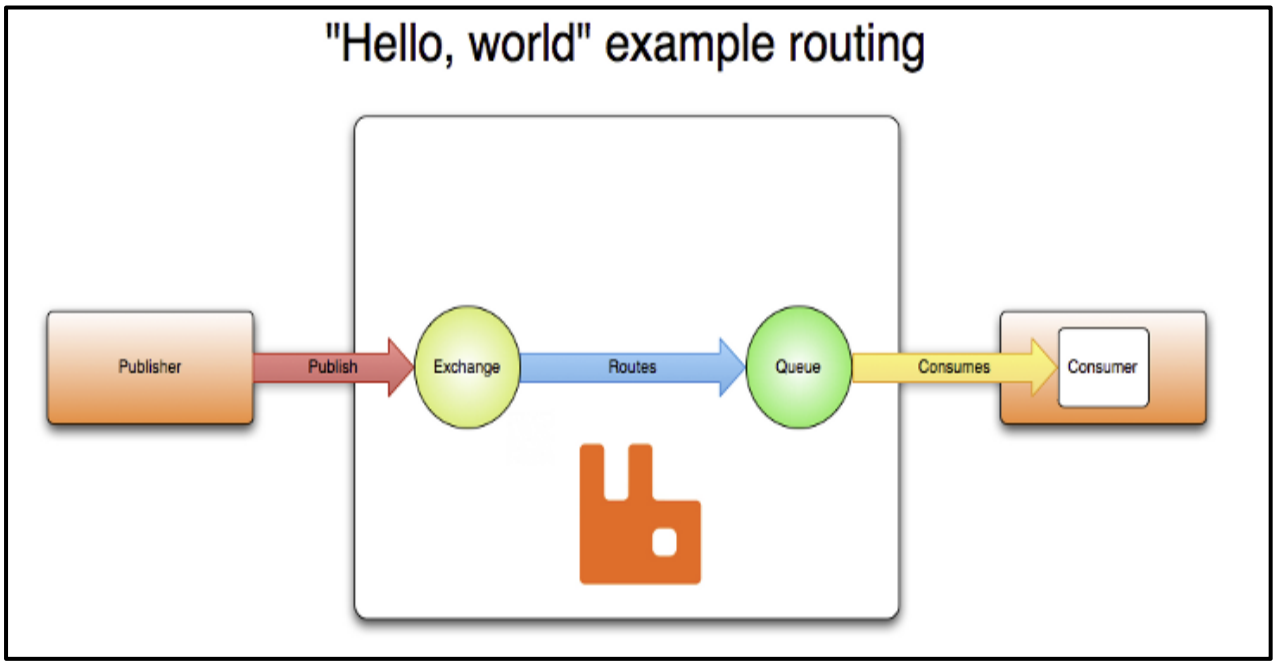


Figure 2-14: The AMQP based messages are published to exchanges [114]

2.7 The Tools Used to Execute SDN

The content of this section describes some of SDN controllers and different simulation tools.

In this dissertation uses Mininet simulator to the data plane of SDN and POX controller to emulate the control plane of SDN. In addition, the Miniedit to build the network topology, and Wireshark to analyze OpenFlow messages.

2.7.1 The SDN Controller

There are many popular controllers but a few of these available. These controllers have actually classified based on the programming language and the purpose of innovation. In Table (2.2) comparison between some types of controllers.

Table 2-2: The comparison between some types of controllers

POX [115]	**Ryu** [116]	**OpenDaylight** [117]
Lightweight and learning to write software for POX is rapid, this being the reason why it is mainly used in research purposes.	Less complex	Complex
Open-source designed as a platform for rapid development.	Open-source and supports multiple protocols.	Open-source and supports multiple protocols.
Original OpenFlow controller is NOX because it is no longer in development, so POX has used in the proposed system which considers an improved version from NOX.	Not Original OpenFlow controller	Not Original OpenFlow controller
Running on Python language and it works on Linux, so appropriate with environment of proposed system.	Running on Python programming language.	Running on Java programming language.
It has supported some versions but suitable with version that involve the problem.	It has supported multiple versions of an OpenFlow.	It has supported some versions of an OpenFlow.
Poor performance compared to other types of controllers running in java.	Poor performance compared to other types of controllers running in java.	Butter performance due it is written in Java, draw back it requires more time when learning to develop applications for it

2.7.2 The Network Emulators

It is so difficult to get devices like switches and routers that execute SDN functionalities, and if found these devices, they will be very expensive. In order to perform researchers their experiments, the virtual network emulators are the optimal solution. Mininet is the network emulator, it has created a virtual network

includes hosts, links, switches, and controllers. It has used in the experiment of the proposed system; it allows to execute big networks with high performance on a single computer [114]. Mininet has created developable Software-defined networks by utilizing lightweight virtualization mechanisms. Mininet has supported internet topology zoo (ITZ), which provides a database with many real-world topologies defined in a graphical notation [118].

There are some characteristics that directed to use Mininet:

1. Flexibility: This characteristic permits for adding new topologies in SDN by utilizing programming language and a specific operating system.
2. Scalability: choosing an appropriate environment for large networks that contain thousands of switches in a single computer.
3. Interactivity: Management and executing the network simulation should take place in real-time as it occurs in actual networks.
4. Applicability: Applications which will execute in the virtual environment, it must be executed in actual networks without any changing in source codes.
5. Shareable: When a developer creates the prototype, must easily be shared with other collaborators. As a result, they will modify and develop their experiments.

There are other network simulators such as EMULAB, it is OS-level virtualization technology, it is emulating many virtual nodes on a single computer but does not support an OpenFlow protocol [119]. The ns-3 is another simulation tool but only old versions supported OpenFlow protocol [120]. EstiNet has supported an OpenFlow protocol, it has used to simulate SDNs, It has used to test the performances and functions of OpenFlow controllers [121]. But performance of Mininet is better.

2.7.3 The Remote Access by MobaXterm

MobaXterm is the fundamental toolbox for computing control remotely. It is a terminal using for Windows, it supplies many functions tailored for IT administrators, webmasters, programmers and all users of computers who need for handling their jobs remotely in a simple method. When entering to MobaXterm, Linux environment appears inside windows. MobaXterm permits to create remote sessions, such as SSH, Telnet, VNC, FTP, SFTP, etc. A secure shell (SSH) is enabled data transfer in encryption way. Also, Linux commands can be used such as bash, cd, ls, grep, sed, etc. Every session will be saved automatically and displayed in the left sidebar. During login to the remote server by utilizing SSH, SFTP browser appears in the left sidebar. This browser permits drag and drop files to or from a remote server by a secure connection [122].

To access for Mininet remotely MobaXterm is used. The session can be established by clicking the session icon, then the IP address of the end server has put to connect remotely. Finally, login ID and password of Mininet have needed to access remotely. Figure (2.15) explains the home page of the MobaXterm.

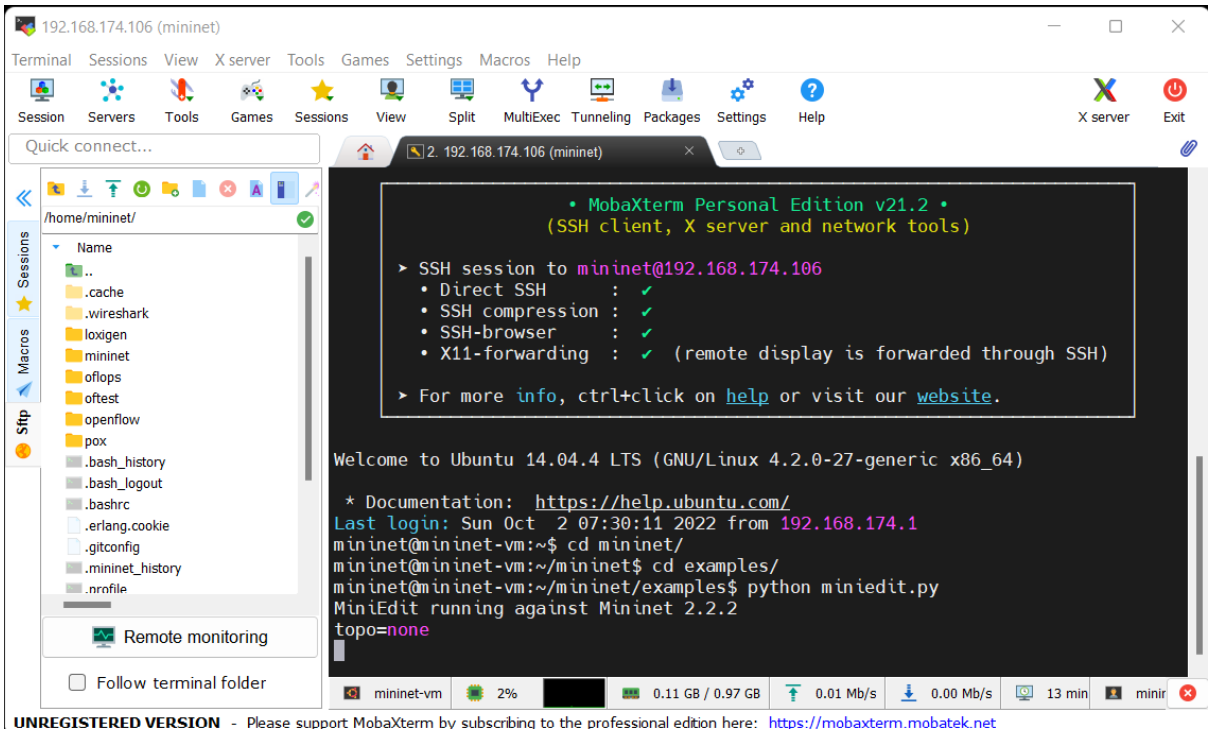


Figure 2-15: The home page of the MobaXterm

2.7.4 The Wireshark a Network Analyzer

Wireshark is the world's foremost and widely-used network protocol analyzer [123]. It is very helpful tools for observing and measuring Mininet network functionality. The function of Wireshark is capturing and analyses all network packets transmitted via network interfaces.

2.7.5 The Miniedit

The Miniedit is a simple GUI editor included in Mininet. It helps to understand how Mininet can be extended and how the topology looks like. Miniedit has a simple user interface that presents a canvas with a row of tool icons on the left side of the window, and a menu bar along the top of the window. The user interface of Miniedit GUI looks as in the image in Figure (2.16).

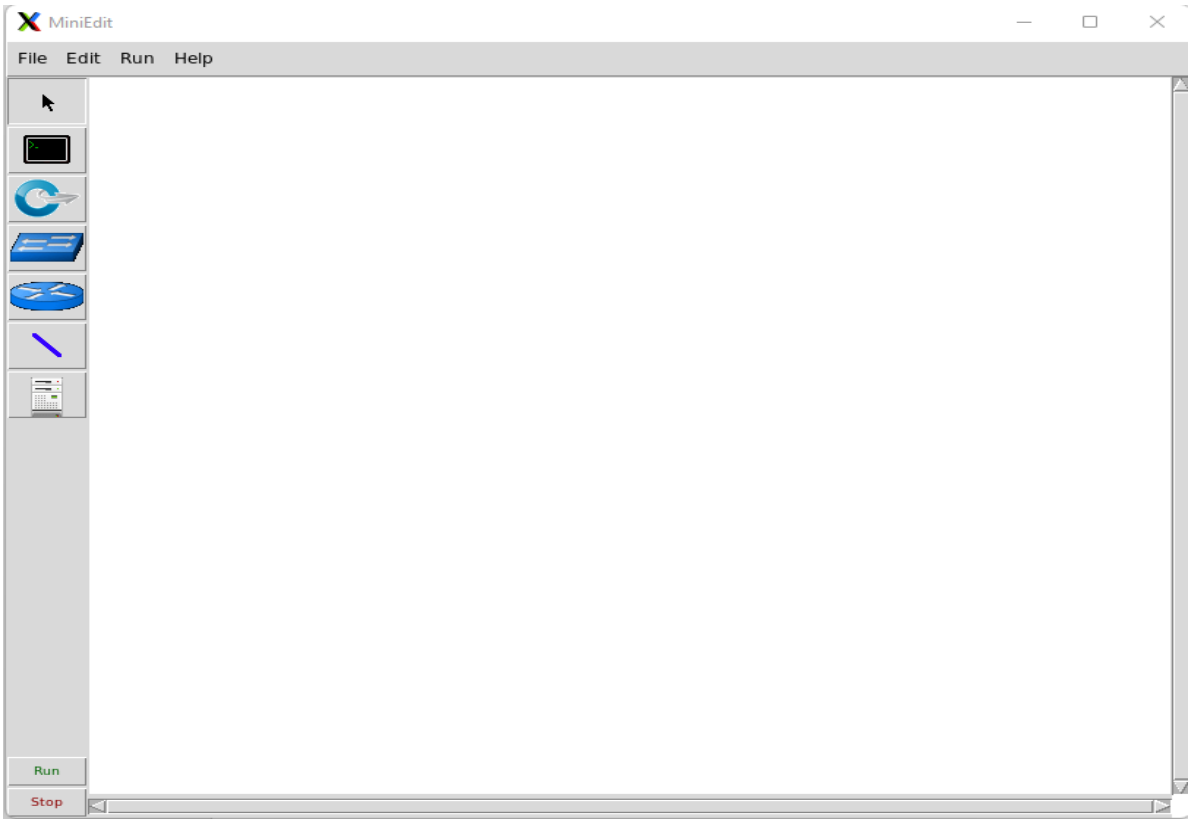


Figure 2-16: The user interface of Miniedit GUI

Chapter Three

The Proposed System Design

3.1 The Proposed System

This chapter explains the proposed system of detection and mitigation methods of DDoS attacks based on entropy values variation of new incoming flows and ML techniques, as shown in Figure (3.1) the proposed system consists of three main processes:

The first deals with incoming traffic using entropy-based detection, which is demonstrated in section (3.2).

The second deals with suspicious forwarded packets using ML techniques, which are demonstrated in sections (3.4) and (3.5).

The third deals with alleviating the effect of the DDoS attacks on the traffic through the mitigation process, which is demonstrated in section (3.6).

The focus of the study is on the security challenges in DDoS attack detection in an SDN environment. Two DDoS attack detection methods, Entropy and ML approaches are proposed, as shown in the main block diagram described in Figure (3.1).

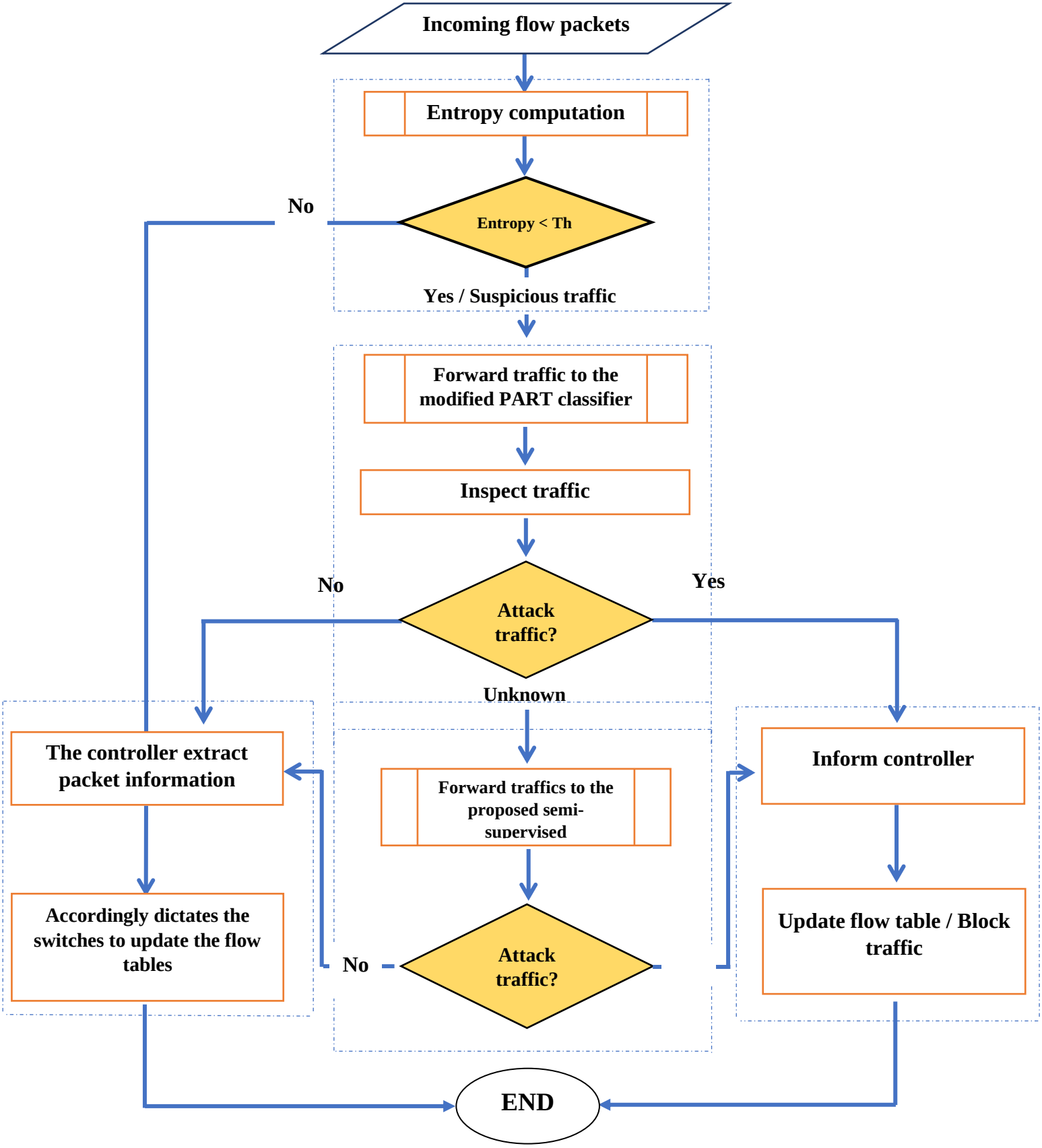


Figure 3-1: The main block diagram of the proposed system

3.1.1 The SDN Environment for Proposed System

To get the best security mechanism, the method of detection and mitigation of DDoS attacks must be executed. To achieve this method, a POX controller with the open virtual switch (OVS) will be used.

This system presents an idea to detect and prevent the attack which affects SDN. To identify the beginning of this attack, statistical measures have been used for network traffic analysis. To analyze network traffic, the technique of Sliding Window (SW) has been used to divide network flow into N windows of captured packets, each window with a particular size of consecutive packets.

The information of new incoming packets has stored in an associative way in (key, value) format, this is called a hash table which means the dictionary in python language. It is a data structure that stores data in an array. Each piece of information has been stored as a name, which is called a key. Each name is matched by only one piece of information called a value. Hash table gives better performance for lookup compared with search trees or any other tables assigned for search, due to the access of data becoming very fast if the index of the desired data is known. Thus, it becomes a data structure in which insertion and search operations are very fast irrespective of the size of the data.

To generate the packets flow, the “scapy” is used. The packets flow is processed by the proposed algorithm (3.1). Algorithm (3.1) explains the main steps of traffic-generated data by the scapy tool.

3.1.2 The Attacks Generating Algorithm

Algorithm (3.1) explains the generation of incoming packet attacks as explained in section (2.2), this tool is used to generate packets for computer networks. Algorithm (3.1) generates a huge number of packets with spoofed source IP addresses. Each IP address consists of four octets. The first octet is used to control the IP address generation to avoid getting some special addresses like

a private address, broadcast address, and loopback address. The algorithm sends 40 attack packets per second to the controller. The number 40 is calculated using Equation (3.1).

$$N = \frac{T}{NumPktsPerSec} \quad \ldots (3.1)$$

Where T is time, *NumPktsPerSec* is the number of packets per second.

Algorithm (3.1): The Attack Packets Generation

Input: Special_addresses, NumrPktsPerSec = 0.025
Output: Generated attacks packet flow
1**. Begin**
2. **Generate IP Packets;**
3. Special_addresses → [10,127,254,255,1,2,169,172,192] ;
4. **For IP Packets in [1.0.0.0 - 255.255.255.255] do;**
5. **While** First_octet ∉ Special_addresses **do**;
6. Generate a random source of IP addresses;
7. **For** each ***N*** successive packets **do**;
8. Send_P → Send attack packets for each 0.025 seconds;
9. **End For**
10. **End while**
11. **End For**
12. **End Algorithm**

The algorithm used to generate the packet flow of artificial attack traffic avoids generating special addresses using the first octet of IP addresses. The attack detection mechanism of the proposed system will be explained in sections (3.2), (3.3), and (3.4).

3.2 The Proposed Attack Detection Model Based on Entropy

The proposed entropy-based approach consists of two main steps:

The first step: The information taken from the incoming packets is gathered and passed to the collection, which is explained in algorithm (3.2).
The second step: is the determination if the packet is normal or suspicious based on entropy variation of new incoming flows by comparing their entropy with the **predefined threshold**. Then the suspected packets are diverted to the ML process for further inspection.

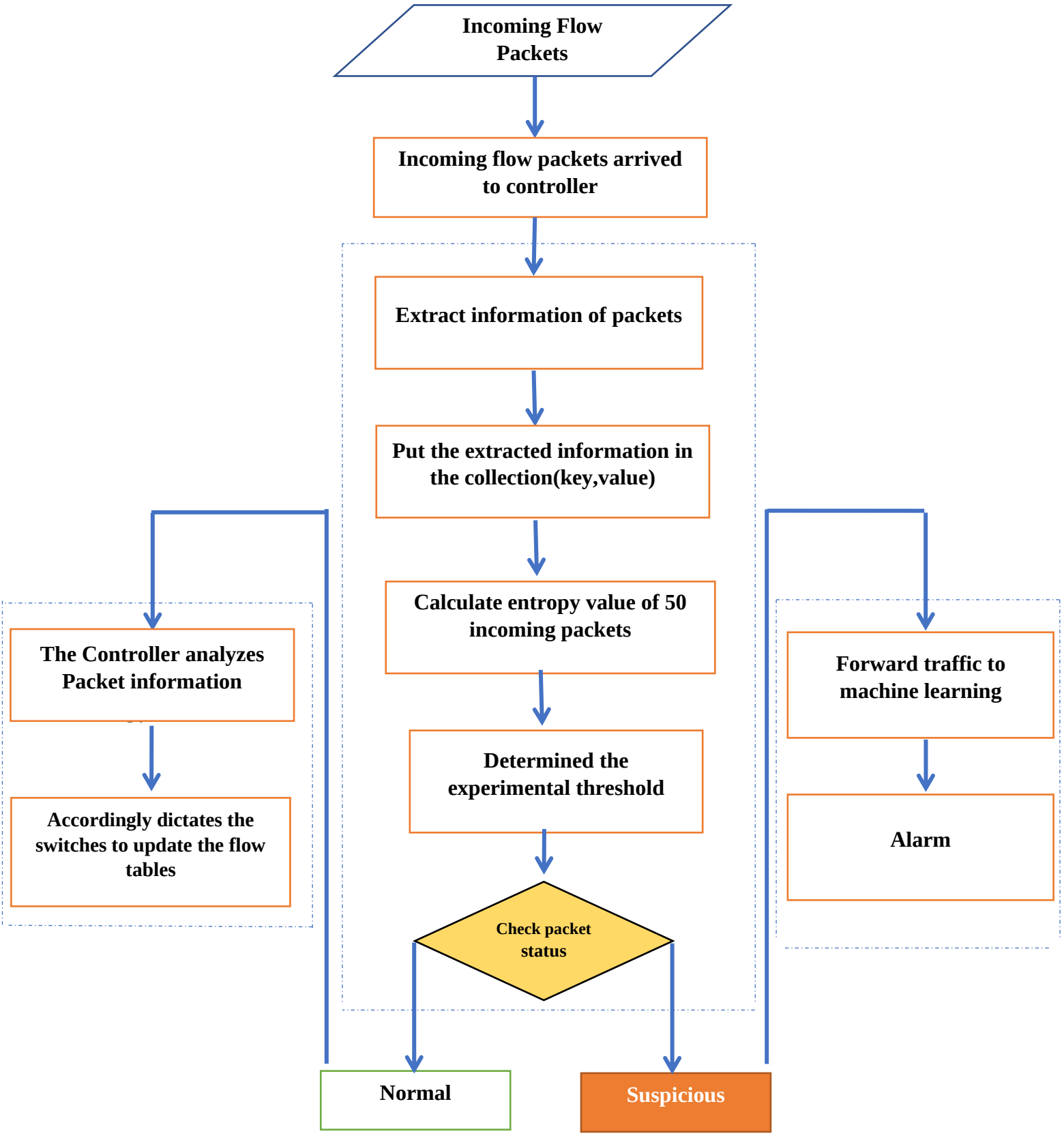


Figure 3-2: The block diagram of entropy-based as an indicator of suspicious traffic

3.2.1 The Entropy Computation

In order to measure the randomness of flows that are arriving at a network, entropy is proposed, which will be used to detect DDoS attacks. If the randomness of flows is high, the entropy value is high too, and vice versa. Two fundamental components are used to detect the attack with entropy calculations. The first component is the window size which is either the number of packets per second or the number of overall arrived packets. The entropy is calculated for each window to measure the uncertainty of incoming flows. The second component is the threshold, if the entropy value is less than the experimental threshold, a DDoS attack is announced. But if the entropy value exceeds the experimental threshold, the packet is normal.

Equation (2.4) describes the entropy computation. Where the applied values of n are the number of flow packets in the window, and $p(x_i)$ is the probability of every element in a window. Hence, $H(X)$ is the result of entropy for all elements in a window. The entropy approach operations can be represented in Figure (3.2).

3.2.2 The Utilizing SDN Specification

For each new incoming flow packet, the controller will install rule in the flow table of a switch, according to an OpenFlow switch specification. This table has contained rules to learn how the switch handling with the incoming packet. Information of rules include (ip-address, mac-address, and port) for each packet, and put actions for processing this packet through drop it or forwarding with priority. Incoming flow packets processing as to be seen in Figure (3.3).

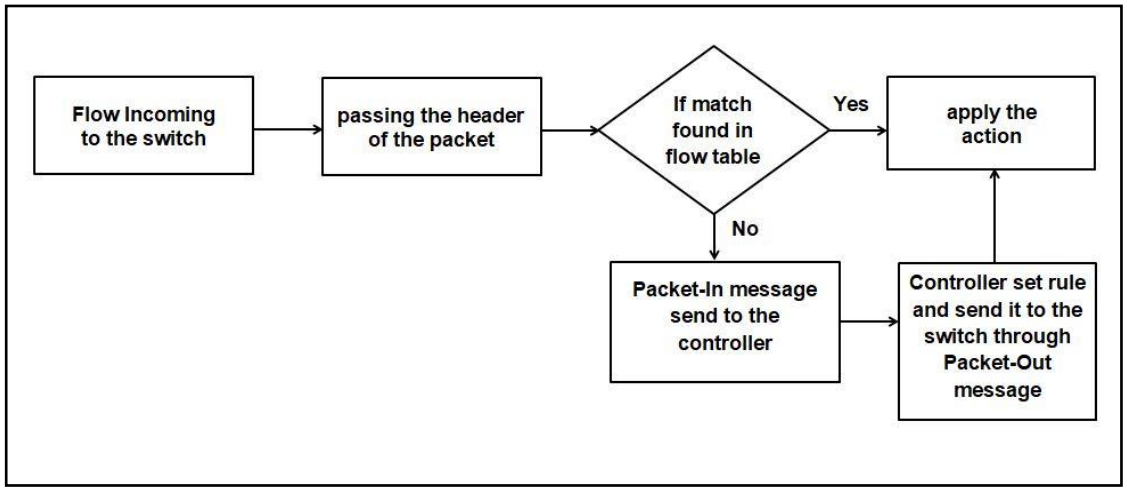


Figure 3-3: The incoming flow packets processing

When the controller receives Packet-In message from a switch, it looking for a match in a flow table of controller, if a match has existed, the controller has installed rule on the flow table of SDN switch. After that, the switch has learned how to process the packet. If a match has not existed, the controller sends Packet-Out to all switches to ask for a specific host. If one switch has known the specific host, it returns a message to the controller, and the controller records the rule on its own flow table to direct rule to the switch.

Another truth about new flows arriving at the controller, the destination host belongs to the network of the SDN controller. Hence, the attacker can generate multiple incoming flow packets to one host or more to instruct the controller to install rule in victim switch, so the switch will flood.

Dest_IP will be chosen instead of Src_IP, due to Src_IP has different in normal and attack state; hence randomness will be high. But in case chosen Dest_IP, the load of packets will be directed to specific hosts, hence randomness will be low.

As a result, the measure of randomness can be found by computing the entropy depending on the size of the window. For example, if there are 50 incoming flow packets, and each packet has sent to a specific destination, the

probability of per destination IP address must be Pi $= \frac{1}{\text{pk}} = \frac{1}{50}$. The value of entropy H according to Equation (2.4) must be $-\sum_1^{50} 0.02 \times \log_2 0.02 = \left(0.02 \times \frac{\log_{10} 0.02}{\log_{10} 2}\right) \times 50 = \left(0.02 \times \frac{-1.6989}{0.3010}\right) \times 50 = -(0.02 \times -5.6441) \times 50 = 0.1128 \times 50 = 5.64\ bits$

This means randomness very high due to the distribution of packets equal in the normal state. But for example, if there are 25 incoming flow packets heading to one destination and, entropy value will be decreased to 3.32.

$\left(\frac{1}{50} \times \log_2 \frac{1}{50}\right) \times 25 + \left(\frac{25}{50} \times \log_2 \frac{25}{50}\right) \times 1 = 3.32\ bits$

This means randomness becomes lower because of overload on a specific destination. But in reality, calculate entropy values in normal and attack state will be explained in chapter four

3.2.3 The Experimental Threshold

The experimental entropy calculator is used for a series of tests to choose the best value of the entropy threshold to in the design of the proposed system. According to many studies such as mentioned in [6], it's found that the value of (0.5) is with the wise value of the threshold with window size 50 packets.

Table (3.1) results gave a clear view of the best threshold value, suppose a have topology consistent 10 host, as fellow tests:

a) Test 1 (5 host) received the incoming packet only, each host received 10 packets.
b) Test 2 each packet received by single host; this means 10 hosts received 50 packets.
c) Test 3 (2 host) received incoming packets. Host 1 received 45 packets, and host 2 received 5 packets.
d) Test 4 (2 host); host 1 received 48 packets and host2 received 2 packets
e) Test 5 one host received whole 50 packets.

Table 3-1: The tests of the entropy value

No.	The probability ($p(h_n)$) of each host	Entropy value
Test 1	$P(h_1)$ =10/50, $p(h_2)$ =10/50, …, $p(h_5)$ =10/50	H(x) = 2.32
Test 2	$P(h_1)$ =5/50, $p(h_2)$ =5/50, …, $p(h_{10})$ =5/50	H(x) = 3.32
Test 3	$P(h_1)$ =45/50, $p(h_2)$ =5/50	H(x) = 0.47
Test 4	$P(h_1)$ =48/50, $p(h_2)$ =2/50	H(x)= 0.24
Test 5	$P(h_1)$ = 50/50	H(x) = 0

3.2.4 The Proposed Algorithm for Detecting Suspicious Traffic

To detect a suspicious traffic, the destination IP address must be monitored of incoming flows by creating a hash table as a dictionary in python language. If Dest_IP new in a hash table, this IP will be added to the table with counter one. If the IP reoccurs in this table only the counter will be increased. Algorithm (3.2) shows Dest_IP collections, and put them in the dictionary (hash table).

Algorithm (3.2): Dictionary Builder

Input: Occurrences of Dest_IPs
Output: Dictionary (Dest_IPs, Occurrences count)

```
1.  Begin
2.  Clear Dictionary
3.      For all incomings flow packets received by a controller do;
4.          Collect Dest_IP corresponding incoming packets in controller;
5.      End For
6.      For all Dest_IP ∈ window
7.          If Dest_IP ∉ Dictionary (Dest_IP, Occurrences count) do;
8.            Add Dictionary (Dest_IP, Occurrences count) = + 1;
9.          Else
10.           Update Dictionary (Dest_IP, Occurrences count + 1);
11.         End if
12.     End for
13. Return Dictionary
14. END Algorithm
```

After the hash table contains 50 packets, an entropy value is calculated. This is the mean of window size equal to 50, the reason for selecting 50 is to get the best results according to the number of hosts found in the network. If the

window size is very small, the error ratio will be high. While the error ratio will be low in other situations which means early detection of attack in case of small window size and lately detecting in case of large window size. Another reason to choose a window size of 50 is to reduce resource usage (e.g., RAM, CPU, etc.).

Equation (3.2) represents the hash table, **a1** means Dest_IP, and **b1** means the occurrences of Dest_IP, hence **W** is the widow size of the collection.

$$\boldsymbol{W = \{(a_1, b_1), (a_2, b_2), \ldots (a_n, b_n)\}} \quad \ldots (3.2)$$

Equation (3.3) represents the probability of the frequency of each Dest_IP.

$$\boldsymbol{Pi = \frac{ai}{N}} \quad \ldots (3.3)$$

Where **ai** represents the number of packets that arrived at specific Dest_IP, **N** represents the total number of packets. Therefore, the value of entropy can be calculated by Equations (3.2), (3.3).

Values of entropy relative stable in normal traffic, when there is a flooding attack in the network, the number of packets belong to hosts to the same destination increases. Hence, the values of entropy decrease sharply.

After choosing the threshold experimental value, the entropy value reduced to less than the threshold, the behavior indicates the occurrence of attack and vice versa. Algorithm (3.3) explains the main steps of entropy based suspected traffic detection.

Algorithm (3.3): The Detection of Suspicious Traffic

Input: Dictionary (Dest_IP, Occurrences count), Threshold (**δ**), Win_S
Output: Alarm the attack in progress

1. **Begin**
2. **Call** *Dictionary Builder Algorithm*
3. **For each** Dest_IP **in** the dictionary **do**;
4. **Find** the probability distribution of flows using Equation (3.3);
5. **End for**
6. **Find** the entropy of probability Dest_IP dictionary records using Equation (2.6)
7. **If** entropy value < δ
8. Alarm for suspicious traffic in the progress;
9. Call Forward the traffic to ML phase;
10. **Else**
11. Normal Traffic
12. **End if**
13. **END Algorithm**

3.2.5 Measures Used in Attack Detection

Entropy is a concept identified by Claude Shannon, it is an essential concept in information theory, as mentioned in section (2.4). Entropy is proposed to detect suspicious traffic. On another hand, statistical measures have been used to determine the threshold, which has been used to know if packet normal or attack, and SDN characteristics will be contributed to detect the attack.

3.3 The ML-Based Detection Approach

There are many ML techniques applicable to DDoS detection. This study applies 6 ML classifiers to perform the binary classification task and in addition to the proposed semi-supervised method for identifying the unknown DDoS traffic. This section and its subsections explain the main techniques of the ML stage, such as dataset, preprocessing, feature selection, and how to choose the suitable classifier. It also shows the semi-supervised method and the justification of its usage in the proposed system.

3.3.1 The Datasets

The CICIDS2017 dataset will be used in the current study. It consists of eighty-four features, it was downloaded from [124]. It is a popular and recent benchmark dataset, it used for training and validating the ML models. It was created on a testbed with three servers, one firewall, two switches, and ten terminals for the victim network and one router, one switch, and four terminals for the attacker network. Over the course of five working days, network traffic flows were recorded, and 84 features were retrieved, as shown in Table (3.2). In addition, the CICDDoS2019 dataset will be used to validate the proposed classifier.

Table 3-2: The features of the CICIDS2017 Dataset

No.	Feature Name	No.	Feature Name	No.	Feature Name
1	Flow ID	29	Fwd IAT Std	57	ECE Flag Count
2	Source IP	30	Fwd IAT Max	58	Down/Up Ratio
3	Source Port	31	Fwd IAT Min	59	Average Packet Size
4	Destination IP	32	Bwd IAT Total	60	Avg Fwd Segment Size
5	Destination Port	33	Bwd IAT Mean	61	Avg Bwd Segment Size
6	Protocol	34	Bwd IAT Std	62	Fwd Avg Bytes/Bulk
7	Timestamp	35	Bwd IAT Max	63	Fwd Avg Packets/Bulk
8	Flow Duration	36	Bwd IAT Min	64	Fwd Avg Bulk Rate
9	Total Fwd Packets	37	Fwd PSH Flags	65	Bwd Avg Bytes/Bulk
10	Total Backward Packets	38	Bwd PSH Flags	66	Bwd Avg Packets/Bulk
11	Total Length of Fwd Packets	39	Fwd URG Flags	67	Bwd Avg Bulk Rate

12	Total Length of Bwd Packets	40	Bwd URG Flags	68	Subflow Fwd Packets
13	Fwd Packet Length Max	41	Fwd Header Length	69	Subflow Fwd Bytes
14	Fwd Packet Length Min	42	Bwd Header Length	70	Subflow Bwd Packets
15	Fwd Packet Length Mean	43	Fwd Packets/s	71	Subflow Bwd Bytes
16	Fwd Packet Length Std	44	Bwd Packets/s	72	Init_Win_bytes_forward
17	Bwd Packet Length Max	45	Min Packet Length	73	Init_Win_bytes_backward
18	Bwd Packet Length Min	46	Max Packet Length	74	act_data_pkt_fwd
19	Bwd Packet Length Mean	47	Packet Length Mean	75	min_seg_size_forward
20	Bwd Packet Length Std	48	Packet Length Std	76	Active Mean
21	Flow Bytes/s	49	**Packet Length Variance**	77	Active Std
22	Flow Packets/s	50	FIN Flag Count	78	Active Max
23	Flow IAT Mean	51	SYN Flag Count	79	Active Min
24	Flow IAT Std	52	RST Flag Count	80	Idle Mean
25	Flow IAT Max	53	PSH Flag Count	81	Idle Std
26	Flow IAT Min	54	ACK Flag Count	82	Idle Max
27	Fwd IAT Total	55	URG Flag Count	83	Idle Min
28	Fwd IAT Mean	56	CWE Flag Count	84	Label

3.3.2 The Preprocessing

Downloading the CICIDS2017 dataset with its all features. Not a Number (NaN) values and duplicate columns were eliminated during the initial round of data cleaning. Two features in this file have the name Fwd Header Length, making it redundant; therefore, one of them is eliminated. The normal method of scaling is utilized. In order to acquire more precise results, features were selected based on the well-known utilized criteria. Figure (3.4) displays a sample of the CICIDS2017 dataset.

	A	B	C	D	E	F
1	Flow ID	Source IP	Source Por	Destination	Destination	Protocol
2	192.168.10.5	104.16.207.165	443	192.168.10.5	54865	6
3	192.168.10.5	104.16.28.216	80	192.168.10.5	55054	6
4	192.168.10.5	104.16.28.216	80	192.168.10.5	55055	6
5	192.168.10.1	104.17.241.25	443	192.168.10.1	46236	6
6	192.168.10.5	104.19.196.102	443	192.168.10.5	54863	6
7	192.168.10.5	104.20.10.120	443	192.168.10.5	54871	6
8	192.168.10.5	104.20.10.120	443	192.168.10.5	54925	6
9	192.168.10.5	104.20.10.120	443	192.168.10.5	54925	6
10	192.168.10.8	104.28.13.116	443	192.168.10.8	9282	6
11	192.168.10.5	104.97.123.193	443	192.168.10.5	55153	6
12	192.168.10.5	104.97.125.160	443	192.168.10.5	55143	6
13	192.168.10.5	104.97.125.160	443	192.168.10.5	55144	6
14	192.168.10.5	104.97.125.160	443	192.168.10.5	55145	6
15	192.168.10.5	104.97.139.37	443	192.168.10.5	55254	6
16	192.168.10.1	104.97.140.32	80	192.168.10.1	36206	6
17	192.168.10.2	121.29.54.141	443	192.168.10.2	53524	6
18	192.168.10.2	121.29.54.141	443	192.168.10.2	53524	6
19	192.168.10.2	121.29.54.141	443	192.168.10.2	53526	6
20	192.168.10.2	121.29.54.141	443	192.168.10.2	53526	6
21	192.168.10.2	121.29.54.141	443	192.168.10.2	53527	6
22	192.168.10.2	121.29.54.141	443	192.168.10.2	53528	6
23	192.168.10.2	121.29.54.141	443	192.168.10.2	53527	6
24	138.201.37.2	138.201.37.241	443	192.168.10.5	55035	6
25	144.76.121.1	144.76.121.178	443	192.168.10.5	55275	6
26	145.243.233	145.243.233.16	443	192.168.10.5	55277	6

Figure 3-4: Sample of CICIDS2017 dataset

3.3.3 The Feature Selection Methods

Feature selection methods play an important role in improving model performance, there is a tradeoff between the number of features and the time complexity of an algorithm, the more features model leads to the most time complexity and the most accuracy and vice versa. As indicated in section (3.3.2), the number of features in the data set is 84. CICID2017 is rich of features as compared to the rest of the datasets with a small number of features like the UCAL dataset [65]. Therefore, it is important to use feature selection techniques in order to reduce the number of features to suit the current study.

The features were chosen using a proposed hybrid feature selection method. The proposed method is composed of the low variance scores and the information gain. The low variance is used to exclude some features. Features

with little variance or close to zero should be ignored because they are not discernments of records. The data for this feature have nearly the same meaning when the variance is low, so this cannot be used as an efficient pattern in ML procedures. The proposed ML model dataflow is depicted in Figure (3.5). Preprocessing of the training data will be done initially by the proposed system. Algorithm (3.4) or (3.5) is used to select relevant features and eliminate worthless or not applicable features with SDN flow statistics.

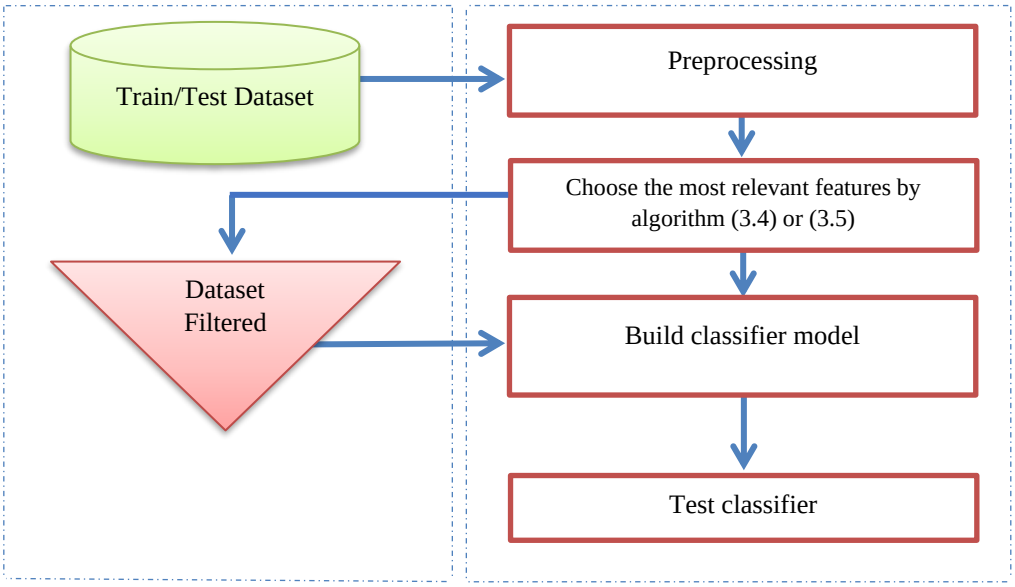


Figure 3-5: General framework of the ML

Algorithms (3.4, 3.5) are used to pick an optimal feature that distinguishes abnormal traffic from regular traffic. The stage of feature selection and engineering is crucial in the ML process. ML does not use all of the features in a dataset. Using whole features to develop a prediction model is not only inefficient in terms of system cost, but it also takes longer. First, data must be normalized, deduplicated, or corrected as needed for imbalanced data. In the actual world, datasets like DDoS attack traffic will be massive. And, in order to train such huge datasets with many features, we must remove irrelevant or worthless information from the dataset. Furthermore, insignificant or worthless features might degrade the detecting mechanism's performance. The attack classes in the CICIDS2017

dataset are somewhat unbalanced. The datasets include attributes that aren't relevant for detecting attacks.

As a result, an employed the variance-based feature selection approach to identify useless features among 84 features as the initial phase in our ML process, as will be shown in chapter 4 section 4.3.

Algorithm (3.4): VarainceInfoGain

Input: D,F = Training dataset, processing n features f1, f2, f3... fn
Output: The selected features list

1. **Split** The dataset(D) train/test;
2. **ThresholdVaraince =** 2.06 (the value is chosen empirically);
3. **ThresholdInfoGain** = 0.6 (the value is chosen empirically);
4. **Perform** Features Filtering
   - (i) Calculate variance using Equation (2.2) and info gain scores of features using Equation (2.4);
   - (ii) Removed the features with the variance less than the **ThresholdVaraince** value;
   - (iii) Select the features with a variance higher than the **ThresholdVaraince** from D;
   - (iv) Filter the features with information gain higher than **ThresholdInfoGain**;
5. **For each feature F**
6. **Select** important features after applying the following criteria
   - (i) **Do Multiple** training/testing to delete less frequent or redundant features;
   - (ii) **Focus** on the features that give higher accuracy and test time decreases; // *this help to build a fast and accurate model*
   - (iii) **Discard** features that cannot be implemented in **real-time**; // *this help to avoid complexity in the proposed model*.
7. **End For**
8. **Create** a classifier model based on the training CICIDS2017 dataset, D;
9. **Test** the classifier by using the testing dataset, T;
10. **Evaluate** the performance and accuracy of the classifier;
11. **Sort** feature with their weight;
12. **Store** a List of features;

Algorithm (3.5): PrevWorksInfoGain

Input: D,F = Training dataset, processing n features f1, f2, f3... fn
Output: The selected features list

1. **Split** The dataset(D) train/test;
2. **ThresholdInfoGain** = 0.6 (the value is chosen empirically);
3. **Perform** Features Filtering
   (i) **Select** from the features based on previous works; // help to avoid unimportant features
   (ii) **Compute** info gain scores of features using Equation (2.4);
   (iii) **Filter** the features with information gain higher than **ThresholdInfoGain**;
4. **For each feature F**
5. **Select** important features after applying the following criteria
   (i) **Do Multiple** training/testing to delete less frequent or redundant features;
   (ii) **Focus** on the features that give higher accuracy and test time decreases; *// this help to build a fast and accurate model*
   (iii) **Discard** features that cannot be implemented in **real-time**; // *this help to avoid complexity in the proposed model*.
6. **End For**
7. **Create** a classifier model based on the training CICIDS2017 dataset, D;
8. **Test** the classifier by using the testing dataset, T;
9. **Evaluate** the performance and accuracy of the classifier;
10. **Sort** feature with their weight;
11. **Store** a List of features;

3.3.4 The Classifier Selection

An utilized the dataset to develop classification models using various ML approaches after identifying optimum feature subsets through either the algorithm (3.4) or (3.5). According to algorithm 3.6, exploiting the performance of several types of learning models, it uses J48, RF, REPT, PART, RT, and DS, which are standard supervised learning algorithms. The output of algorithm (3.6) is the best classifier that will be applied in real-time as an SDN app to detect DDoS attacks in an SDN environment.

Algorithm (3.6): ClassifierSelect

Input: **FL** = Features_List
Output: Features Subsets, Accuracy, and Testing time with **Fast and Accurate Model**

1. **Begin**
2. **For** every feature Fr in Feature_Ranked data
3. **Start** to Select from Feature Sets
4. SET1/Groups1 features
5. SET2/Groups2 features
6. SET3/Groups3 features
7. SET4/Groups4 features
8. **For each** Feature **in** SETs/Groups
9. **Feed** Selected features to RF, REPT, RT, DS, J48, PART using Algorithm (3.4)
10. **Feed** Selected features to RF, REPT, RT, DS, J48, PART using Algorithm (3.5)
11. **Apply Classifier**
12. C1 = Random Forest model
13. C2 = Random Tree model
14. C3 = REP Tree model
15. C4 = Decision Stump model
16. C5 = J48 model
17. C6 = PART model
18. **Calculate** Test time, Tree size, and Accuracy
19. **Compare** the Accuracy and testing time of C1, C2, C3, C4, C5, and C6
20. **END Algorithm**

3.3.5 The Modified PART

The PART algorithm consists of a decision tree with RIPPER, as explained in Section 2.3. Since it is lightweight, this is an encouraging factor for its real-time implementation of the proposed system.

A. The Minimum Description Length Based Post-Pruning

The MDL principle describes a way to minimize models. It is similar to combining the length (or in our case, the tree depth) and the cost into a new and improved cost function. The goal of MDL can be described as" to find regularity in the data (or in our case, the useless rule sets".

1. MDL: Explanation

MDL: prefer the hypothesis *h* that minimizes the space required to describe a theory plus the space required to describe the theory's mistakes.

MDL: minimize: Size(tree)+ size (misclassifications(tree))

2. **MDL: Formal Explanation**

Occam razor: prefer the shortest hypothesis

MDL: prefer the hypothesis *h* that minimizes

$$h_{MDL} = L_{C1}(h) + L_{C2}(D|h) \quad \ldots(3.4)$$

where Lc_x is the description length of *x* under encoding *C*.

3. **Example**

Let *H* be a set of decision trees of PART (hypotheses) and *D* be a set of training data labels. Then,

- $L_{c1}(h)$ is # bits to describe tree *h*.
- L_{C2}(D|h) is # its to describe d given *h*.
  - Note that $L_{C2}(D|h) = 0$ if all training instances are classified perfectly by *h*. It need only describe exceptions
- Hence h_{MDL} trades off tree for training errors.

B. The Minimum Number Object-Based Pre-Pruning

The different ranges of the minimum no of objects are set for a few examples and tested for accuracy. Figure (3.6) shows the proposed modified PART algorithm diagram. The proposed modified PART is a very accurate and short test time, as will be explained in Chapter 4, compared to other algorithms.

C. The PART Algorithm

This study modified the PART algorithm to improve real-time attack detection speed (test time). This contribution proposed modification is to choose an efficient burning method for non-significant PART trees as shown in algorithm (3.7). Reducing the decision tree means reducing the complexity time which is significant in real-time detection.

Algorithm (3.7): Modified PART

Input (Data, List features)

Output (fast and accurate rules set)

1. **Begin**
2. **ThresholdMinNumObj = 25 (the value is chosen empirically)**
3. **Pre-Pruning** by minNumObj with ThresholdMinNumObj value
4. **Create** rules set by C4.5
5. **Post-Pruning** the generated rules sets of C4.5 by RIPPER and MDL
6. **Produced** set with 44 rules set
7. **Call** evaluation function (Time, Accuracy, Tree Size)
8. **Validate** by second dataset (CICDDoS2019)
9. **Implement** the produced rules set in **real-time** as SDN Application
10. **End Algorithm**

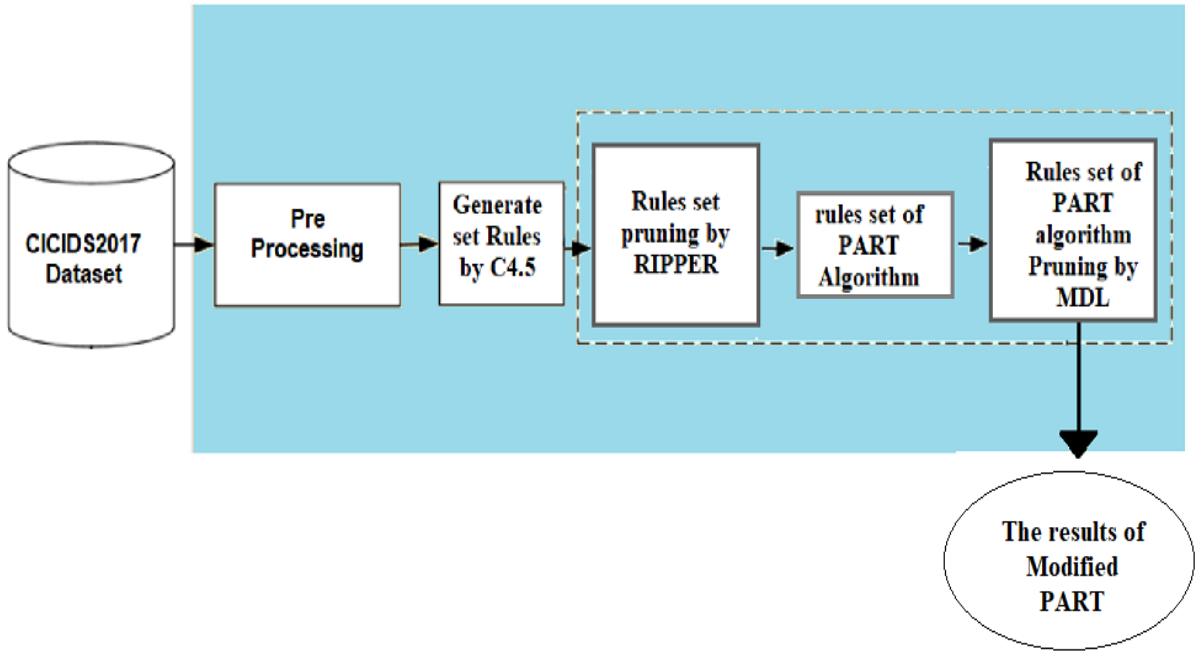


Figure 3-6: The modify PART algorithm diagram

3.3.6 The Semi-Supervised Learning Model

The proposed model uses the semi-supervised *k*-Means clustering algorithm to generate two centroids that can be used to classify traffic as either normal or DDoS. Starting with the CICIDS2017 dataset, the proposed model uses the K-Means algorithm to produce two clusters aggregated around two centroids

to split the traffic into two labels, which are infected and safe classes. Figure (3.7) shows the flow control of the semi-supervised framework.

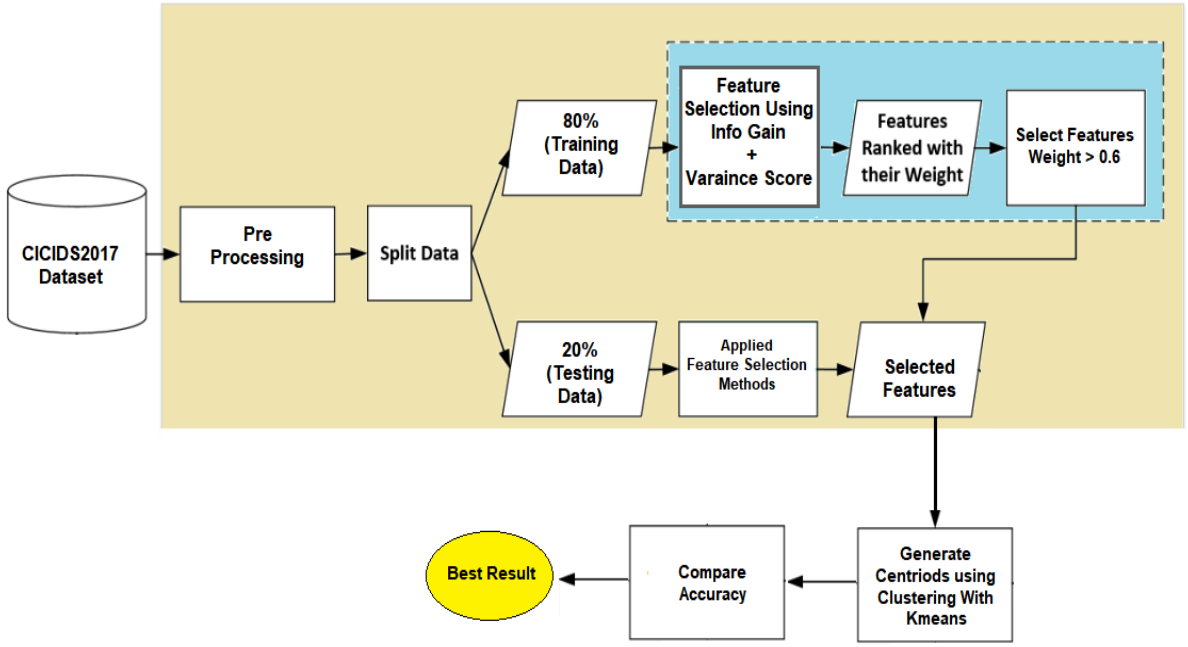


Figure 3-7: The proposed semi-supervised framework

3.3.7 The ML Performance

In this stage metric measures are applied to show the effect them in among 6 classifiers. The five major performance measures were used to evaluate the performance of classifiers. Fit time, Accuracy, Recall, Precision, and F1 scores, which are explained in section (2.5). Accuracy represents the algorithm's accuracy in identifying attacks over both regular and attack traffic. Recall reflects the age of real assault traffic that is accurately detected. Precision refers to the detection of an assault over projected positive cases. The F1 score, which combines recall and accuracy, is also computed.

Fit time, which indicates the classifier's fitting time during the testing step, is also calculated together with the other four evaluation criteria.

3.4 The Application of the Proposed System in the SDN Environment

The following subsections gives more details about the application of the proposed system in the SDN environment.

3.4.1 The Proposed Network Topology

The Mininet is selected as the network emulation platform for the first section of our experiment, which employs POX SDN as the controller. The detecting application will be a separate module integrated within the L3 Learning module. The controller will now be capable of installing flows and verifying DDoS assaults. During this testing phase, OF-switch is used to simulate the behavior of an edge switch in an SDN network. A client in the SDN network generates both normal and attack traffic aimed at a victim node in the Mininet network.

3.4.2 The System Architecture

The initial packet is transferred from one client to another during typical communication between clients in an SDN environment. The switch's flow table will initially have no entry indicating where the incoming packet should be sent. The packet will be sent to the controller by default. The switch will be instructed by the controller to forward the packet to the appropriate destination port. The controller will also give an entry rule for that source and destination in the switch's flow table. Without the intervention of the controller, packets with the same source and destination will be routed automatically in the future.

3.4.3 The POX Modules

The detecting application will operate as stand-alone modules on POX. The controller will be able to execute a variety of functions, including installing flows, forwarding packets, and verifying DDoS attacks, over the employment of many modules.

3.4.4 The Learning Module of the POX Controller

The L3 learning module in the POX controller is the most critical component. It's a basic layer 3 learning module that connects the network's nodes. The L3 learning module processes the incoming packet and keeps track of the couplings between the OVS ports and the connected clients' MAC addresses. This

information is used to create a rule that substitutes a destination MAC when a packet is sent to the destination port. The module creates ARP requests if no binding is identified. It is combined with a detection algorithm to identify a malicious traffic flow in addition to a connection.

3.4.5 The DDoS Detection and Mitigation Algorithms in SDN

Design of detection system based on three part which are entropy indicator, rules of modified PART as classifier, and proposed semi-supervised. This design is written in Python and will be implemented within the POX controller.

3.5 The Proposed Algorithm for Attack Mitigation

To defend against the DDoS attack, the proposed model has applied in the SDN controller. If ML classifies the traffic as an attack, the controller will be blocking these packets destined to a specific host until terminating the attack, and values of entropy are rising of the flow of incoming packets. Hence, this model creates the flow rule to block flows that belong to certain Dest_IP, as described in figure (3.8).

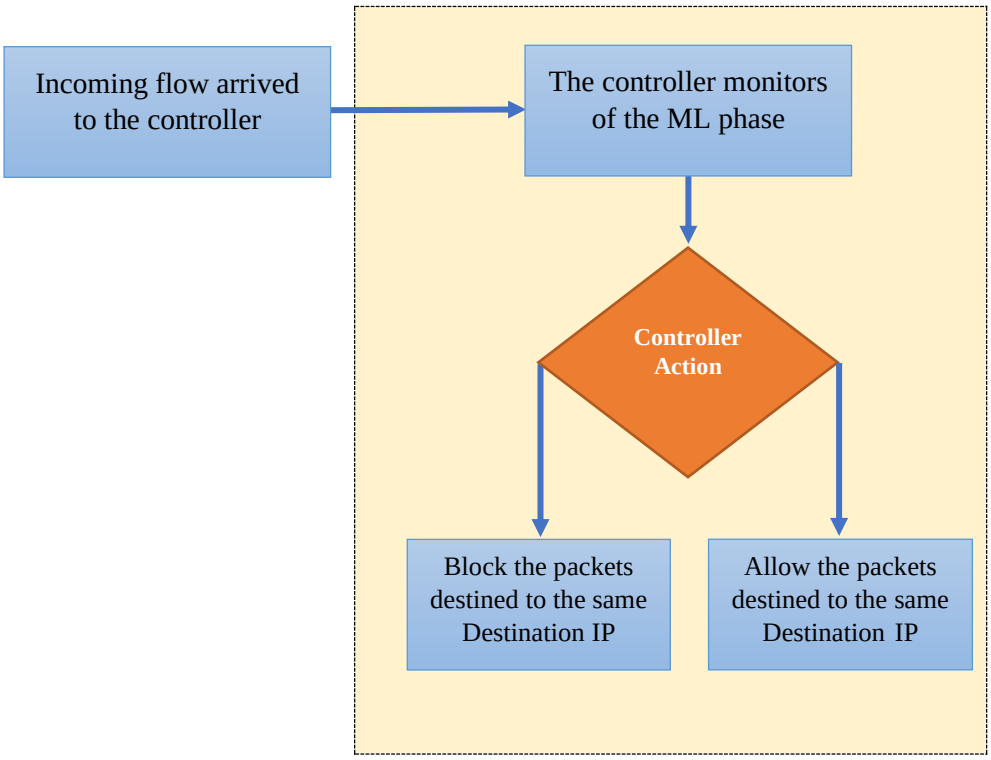


Figure 3-8 : The mitigation process

The steps of the mitigation model are shown in Algorithm (3.8). The rule has been created in the controller to drop the flows directed to the victim switch, and the remainder flow continuously within the network. This rule determines by the administrator of the network. The administrator knows the maximum number of incoming packets that arrived for each server in the network. When an attack occurs, the number of incoming packets exceeds the normal rate. Hence, the Administrator knowing there is an attack at the network, then the prevention phase will be applied.

Algorithm (3.8): DDoS Attack Mitigation

11. **Input:** Destination IP (Dest_IP)
12. **Output:** Add rule to drop attack packets

1. Begin
2. **S[i]** ← **S**witchesIDs**;**
3. **rule** ← **(Packet Information, Action);**
4. Identify the frequently of **Dest_ IP;**
5. **For** S[i] ∈ P **do;** // P is the SDN POX controller
6. Create the rules (Dest_IP, Drop)**;**
7. Write the flow rule in $\mathbf{S_i}$**;**
8. **End for**
9. **END Algorithm**

3.6 FSCA: Flat Distributed SDN Controllers Using AMQP

This section demonstrates that the proposed FSCA, Flat distributed SDN Controllers utilizing AMQP, which is able to manage distributed SDN controllers. FSCA-based concepts for multiple SDN controllers are presented in order to manage multiple controllers and overcome the challenges of a single point of failure in SDN controller. One of some prevalent types of distributed SDN controllers is the flat model. FSCA is a flat distributed SDN controller model that has been validated in multiple experiments and shown to prevent a single point of failure in a multi-domain controller environment.

The proposed FSCA is efficient East/West communication using AMQP to enable any SDN controller to be run in distributed controllers' topology. The main goal of this proposal, focus on building a strategy of distributed controllers or the so-called multiple controllers in an SDN environment. The main key features and contributions of the proposed strategy in addition to its support avoiding the problem of single point of failure, can be listed as fellow:

1. It is compatible with more than one type of controller (OpenDaylight, ONOS, POX, etc...).
2. It works in Python Language.
3. It supports a flat distributed controller's model.
4. It is cross-platform coordination.
5. Easy to implement.
6. Publish/subscribe via standard protocol (AMQP) in application-to-application mode among POX controllers.

Topology is built which is consisting of one server (Linux OS) a platform for AMQP, two POX controllers, 2 switches, and two hosts. Created topology by using Mininet. Figure (3.9) shows the proposed network topology.

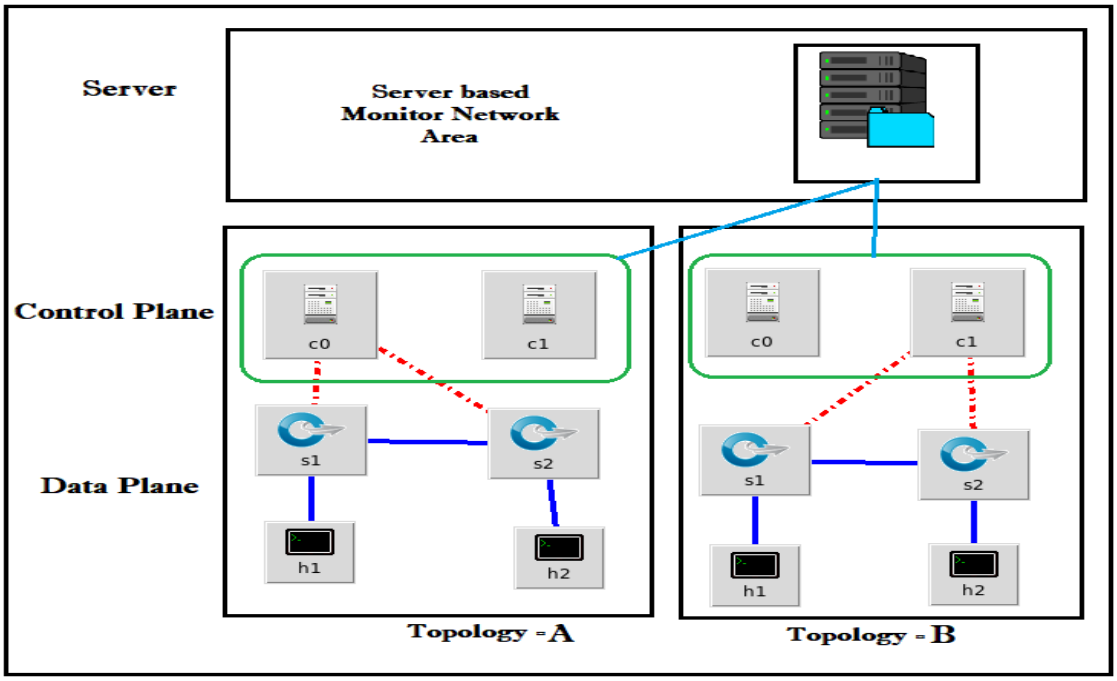


Figure 3-9: The Proposed Topologies Based on East/West Communication

3.7 The summary

The study will be explained that applied in SDN controller to detect suspicious traffic and forwarding it to upper inspection by ML techniques as the following:

1- The destination IP address has been chosen, due to the attacker has controlled on a number of hosts in the network. The attacker generates a massive number of packets, the header of packets contains the various source IP address and various mac address, which represent the mechanism of the DDoS attack. These packets have destined to one host or more belong to victim switch at a rate more than normal. By monitoring the Dest_IP, it will be contributed as a first step in detecting the attack.

2- When incoming packets arrived at the network, the switch doesn't know how to process these packets, because information of these packets doesn't find in the flow table of the switch. So, the switch directs these packets to SDN controller. In SDN controller, the destination IP address of incoming packets have extracted.

3- The dictionary has created in SDN controller to aggregate destination IP address of packets, a hash table or dictionary contains two fields, the first field represents Dest_IP, and the second field represents recurrence of this Dest_IP.

4- The maximum capacity of the hash table or dictionary is 50 packets, which means that the window size is 50, because of arriving restricted number from new packets to every host. Another reason to get the best value and faster result.

5- Compute the entropy value for every 50 packets, in the normal case entropy value will be high, while in attack case entropy value will be decreased.

6- A threshold experimental has chosen based on equations Mean, Standard Deviation and Confidence Interval.

7- If entropy value less than the threshold, it means there is suspicious traffic.

8- The controller monitors distribution of the new incoming packets in the network. If the controller classified these packets as an attack (specious traffic), it will forward these packets to ML phase (Modified PART and proposed semi-supervised methods). The reason to pass the suspicious traffic to ML techniques to get high decision accuracy by further inspect these packets (low false positive) using rules set generated from Modified PART.

9- If ML classifies the traffic as an attack, the controller will be blocking these packets destined to a specific host until terminate the attack, and values of entropy are rising.

10- Finally, the proposed FSCA is efficient East/West communication using AMQP to enable any SDN controller to be played in distributed controllers' topology.

Chapter Four

The Results Discussion and Analysis

4.1 Overview

This chapter demonstrates the results of the proposal of detection and mitigation methods for DDoS attacks. The chapter also discusses the results proposed system and explains the methodology presented in chapter three.

4.2 The Hardware and Software Requirements

The proposed system is implemented on the Mininet simulator and POX controller. Simulations and evaluations were carried out on a variety of platforms. The main platform for testing our detection system is a Windows 11 machine with 11th Gen Intel(R) Core (TM) i5-1135G7 @ 2.40GHz processor and 16 GB RAM. There are two virtual machines in this environment (Mininet and POX controller).

Mininet is selected as the network emulation platform for the first section of the experiments, which employs POX as the SDN controller. The detecting application will be a separate module integrated within the L3 Learning module. The controller will now be capable of installing flows and verifying DDoS attacks.

4.3 The Entropy-based Attack Detection Approach

The entropy phase will be discussed in this section, where it will highlight both how a DDoS attack is generated and how it is detected early by the entropy approach in the SDN network environment.

4.3.1 The Generate Traffic

Scapy traffic generator was used to create UDP packets and also spoofed the src_IP address. Scapy is a program of forceful interactive packet manipulation written in python and it can forge packets of different protocols. The Scapy has generated UDP packets for DDoS attacks with a random source IP address; also, it focuses on the destination IP address. The normal packets have been generated from the host to all hosts in the network topology by using Scapy. When the packet is a DDoS packet, the attacker will be focused on the victim host. The

UDP packet has been chosen because of no need to institute a 3-way handshake process between the source and destination. Figure (4.1) illustrates the packet generation for DDoS. Traffic generated by the algorithm (3.1) is presented in Section (3.2).

Figure 4-1: The DDoS attack packet generation by Scapy

4.3.2 The Monitor Traffic and Entropy Value

Traffic, as described in the previous section, is generated via the scapy tool. On the other hand, a monitor SDN application has been developed that monitors the traffic and displays it to the network administrator, as in Figures (4.2) and (4.3), it displays traffic information as packet address information as well as entropy values, and this helps to review the working and analysis of the proposed system. It should be noted that the SDN application developed has been integrated into one of the POX controller modules. The l3_learning module has been modified by adding codes for the proposed system algorithms.

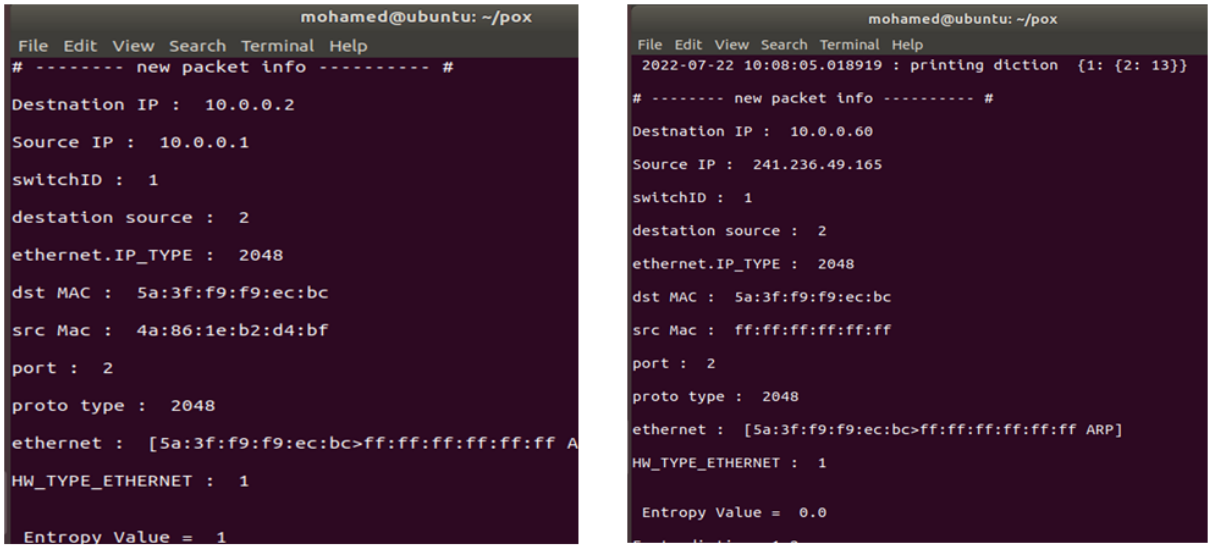


Figure 4-2 :The capture of normal (A) and DDoS (B) packets info by monitoring SDN App

Figure (4.3) shows capturing the DDoS traffic by Wireshark tool on target machine (destination IP=10.0.0.2) with spoofed IP addresses, which is shown under source column.

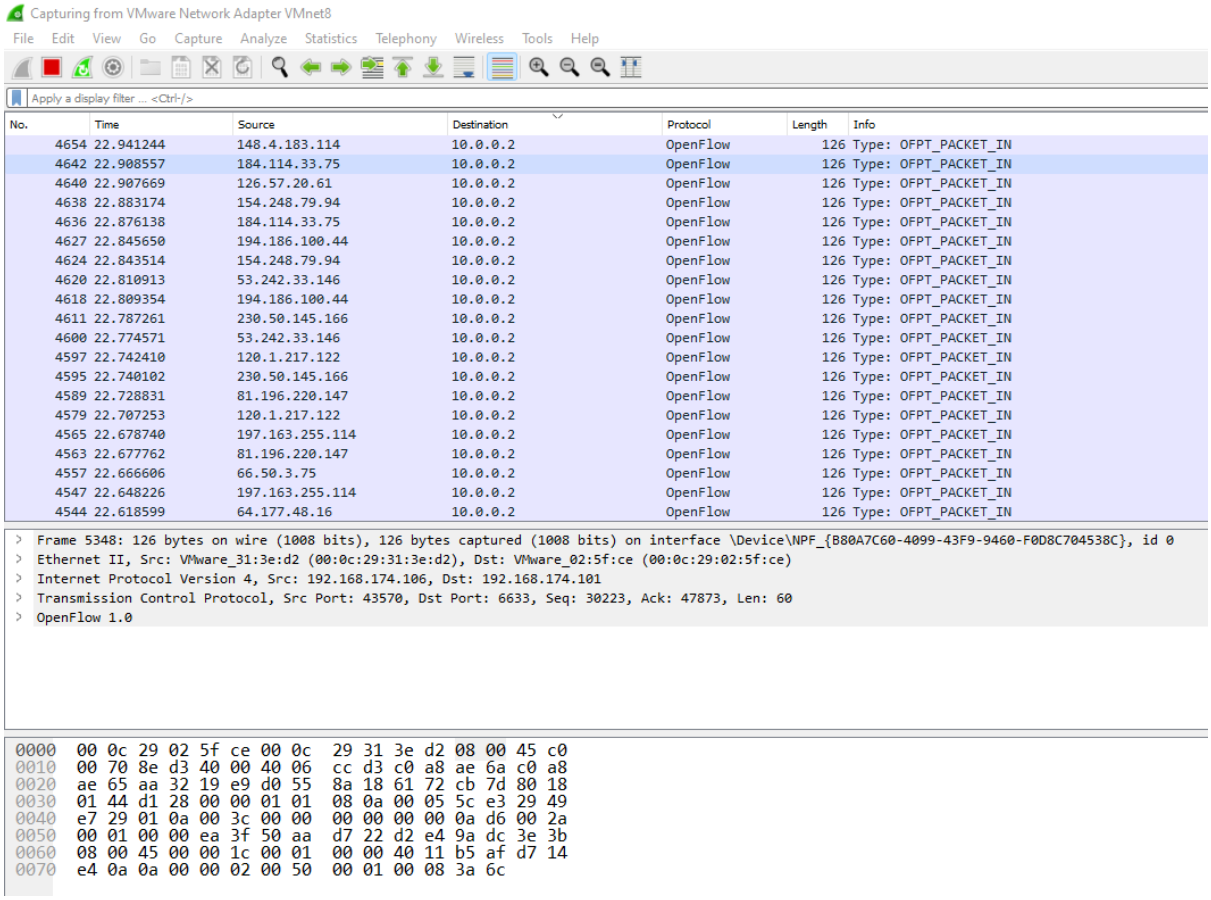


Figure 4-3: The capture of DDoS packets info monitoring traffic SDN App

4.4 The ML-based DDoS Attacks Detection

Detection of DDoS attacks by entropy generates false negative alerts. This motivates the inclusion of the ML process to alleviate the effect of a false alert.

The ML section presents the results and performance of ML models in varieties steps, see Figure (4.4). It entails several feature selections that were utilized to determine which features were beneficial and important. After the classifier models have been created, their performance is compared, and choose the best classifier. Furthermore, optimize the best classifier model and then it should be used in the SDN environment.

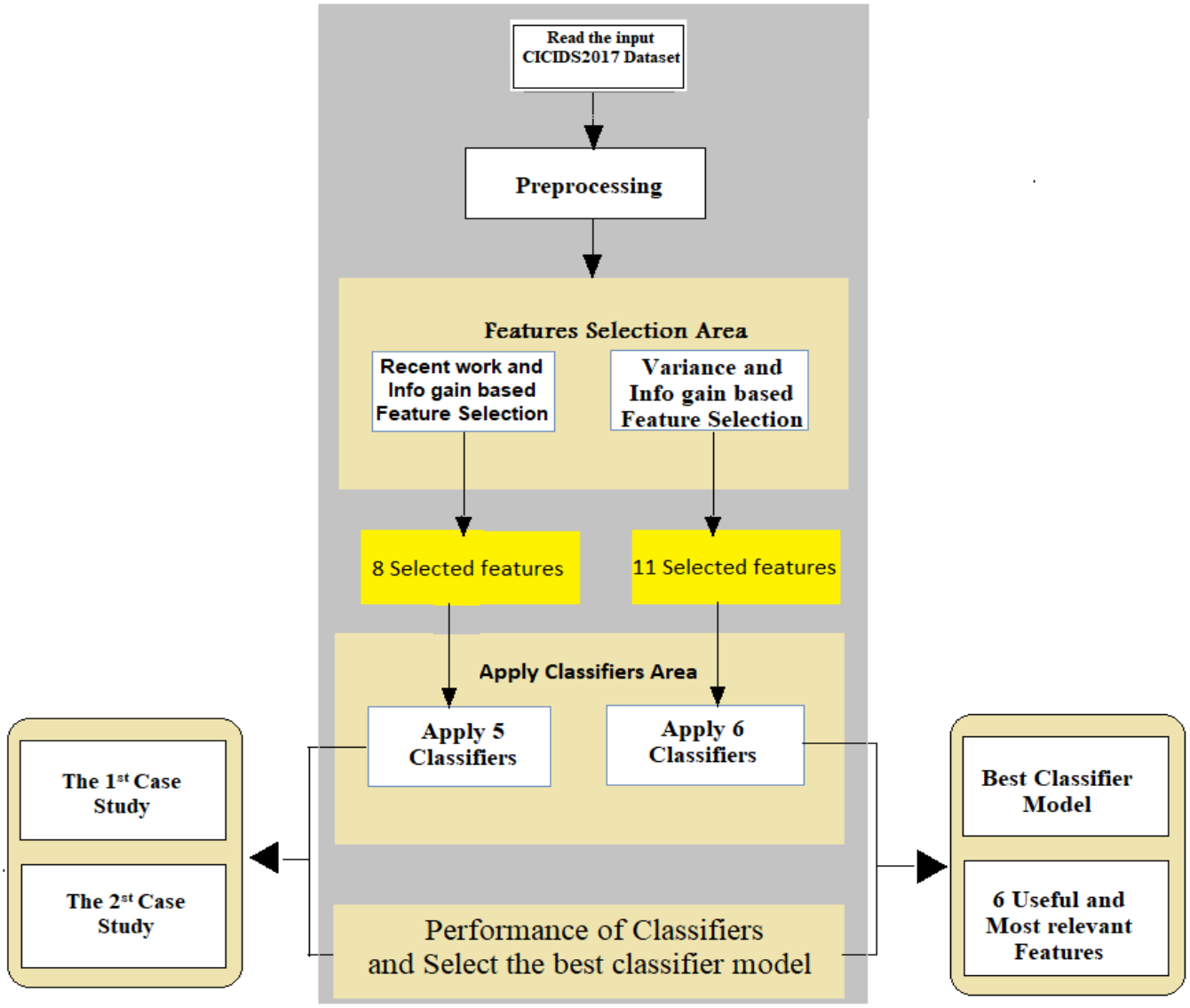


Figure 4-4: The dataflow of the ML process

4.4.1 The Datasets

As mentioned in the Chapter 3, the CICIDS2017 dataset will be used for building ML models and testing them. CICDDoS2019 will also be used to verify the proposed model. Since the two datasets do not differ much in features, the first dataset features will be reviewed in this section. The CICIDS2017 dataset has more than 225,000 records.

4.4.2 The Feature Selection Methods

The feature selection is essential for removing unnecessary features that increase the time complexity of the model during testing and may lead to model failure. This section presents two dependent mechanisms of feature selection as will be demonstrated in the next sections. Due to this, two terms (GROUPS and SETS) will be used to describe each outcome referred to as multiple features of the previous two mechanisms of features selection.

A. The Information Gain and Recent Works Based on Feature Selection

Table (4.1) shows the values of each feature based on information gained and recent works, which is one of the methods to select appropriate features in this study.

Table 4-1: The information gain ranks of features with previous works

No	Name of feature	Information gain	Previous works
1	SourcePort	0.396	[125]
2	DestinationPort	0.776	[126],[127], [128] ,[129],[125]
3	Protocol used	0.2	[130],[131]
4	FlowBytes/s	0.309	[126], [130], [128] ,[129]
5	FlowPackets/s	0.287	[126], [130]
6	PacketLengthMean	0.568	[126], [128] ,[129]
7	PacketLengthVariance	0.532	[126],[132],[133], [128] ,[129]
8	AveragePacketSize	0.81	[126],[133],[134], [128] ,[129],[125]

B. The proposed hybrid feature selection method

Information gain and variance scores based proposed hybrid features selection method. Table (4.2) shows the values of each feature based on variance score, which is one of the methods to choose the most appropriate attributes.

Table 4-2: The variance score of features

No.	Feature Name	Variance Score	No.	Feature Name	Variance Score
1	Bwd PSH Flags	0.00	43	Total Length of Fwd Packets	10558623.00
2	Fwd URG Flags	0.00	44	Subflow Fwd Bytes	10558623.00
3	Bwd URG Flags	0.00	45	Bwd Packet Length Max	13727939.86
4	CWE Flag Count	0.00	46	Max Packet Length	14539997.38
5	Fwd Avg Bytes/Bulk	0.00	47	Init_Win_bytes_backward	18659983.81
6	Fwd Avg Packets/Bulk	0.00	48	Init_Win_bytes_forward	64605923.71
7	Fwd Avg Bulk Rate	0.00	49	Destination Port	390246093.89
8	Bwd Avg Bytes/Bulk	0.00	50	Bwd Packets/s	395848197.39
9	Bwd Avg Packets/Bulk	0.00	51	Source Port	531639178.98
10	Bwd Avg Bulk Rate	0.00	52	Total Length of Bwd Packets	1538077992.04
11	Timestamp	0.00	53	Subflow Bwd Bytes	1538077992.04
12	RST Flag Count	0.00	54	Fwd Packets/s	12247878939.37
13	ECE Flag Count	0.00	55	Flow Packets/s	13248986014.97
14	FIN Flag Count	0.00	56	Active Std	44215016300.83
15	Fwd PSH Flags	0.03	57	Flow IAT Min	577311298613.20
16	SYN Flag Count	0.03	58	Active Min	615064093428.86
17	URG Flag Count	0.12	59	Active Mean	636684372346.84
18	PSH Flag Count	0.23	60	Active Max	810423031803.74
19	ACK Flag Count	0.25	61	Flow IAT Mean	7298619788130.22
20	Down/Up Ratio	2.05	62	Fwd IAT Min	14403752184805.30
21	Protocol	15.07	63	Bwd IAT Min	16154690164265.70
22	min_seg_size_forward	17.36	64	Packet Length Variance	16940970009011.70
23	act_data_pkt_fwd	150.55	65	Bwd IAT Mean	21034826248906.10
24	Total Fwd Packets	237.87	66	Bwd IAT Std	29984141125230.80
25	Subflow Fwd Packets	237.87	67	Fwd IAT Mean	35220592892771.20
26	Min Packet Length	248.62	68	Flow IAT Std	58107370685071.70
27	Total Backward Packets	473.30	69	Fwd IAT Std	116345399015843.00
28	Subflow Bwd Packets	473.30	70	Idle Std	162738330705309.00
29	Bwd Packet Length Min	2548.29	71	Bwd IAT Max	261748754441930.00
30	Fwd Packet Length Min	26674.78	72	Flow Bytes/s	285120776678114.00
31	Fwd Header Length	141218.67	73	Idle Min	393272307102719.00
32	Fwd Packet Length Mean	254916.91	74	Idle Mean	477554839285659.00
33	Avg Fwd Segment Size	254916.91	75	Bwd IAT Total	483320409299892.00
34	Bwd Header Length	261904.23	76	Flow IAT Max	712981673992564.00
35	Packet Length Mean	312553.11	77	Idle Max	724754436283845.00
36	Average Packet Size	391996.45	78	Fwd IAT Max	755628397616387.00
37	Fwd Packet Length Std	635864.42	79	Flow Duration	993786170715473.00
38	Bwd Packet Length Mean	1255127.93	80	Fwd IAT Total	999081952607325.00
39	Avg Bwd Segment Size	1255127.93	81	Flow ID	
40	Packet Length Std	1611779.33	82	Source IP	
41	Bwd Packet Length Std	3003986.63	83	Destination IP	
42	Fwd Packet Length Max	3474976.89	84	Label	

Note the variance values for all the data ranges (0 to 9.99E+14) for (Bwd PSH Flags and Fwd IAT Total) features respectively. The variance scores of features are listed in Table (4.2). Dismiss the 20 attributes with the lowest variance value. The 20 lowest scores of variances are listed in Table (4.3).

Table 4-3: 20 Features from the CICIDS2017 Dataset are the lowest variance

No.	Feature Name	Variance Score	No.	Feature Name	Variance Score
1	BwdPSHFlags	0.00	11	Timestamp	0.00
2	FwdURGFlags	0.00	12	RST Flag Count	0.00
3	BwdURGFlags	0.00	13	ECE Flag Count	0.00
4	CWEFlagCount	0.00	14	FIN Flag Count	0.00
5	FwdAvgBytes/Bulk	0.00	15	Fwd PSH Flags	0.03
6	FwdAvgPackets/Bulk	0.00	16	SYN Flag Count	0.03
7	FwdAvgBulkRate	0.00	17	URG Flag Count	0.12
8	BwdAvgBytes/Bulk	0.00	18	PSH Flag Count	0.23
9	BwdAvgPackets/Bulk	0.00	19	ACK Flag Count	0.25
10	BwdAvgBulkRate	0.00	20	Down/Up Ratio	2.05

When an applied variance features selection technique, eliminated 20 useless features. So, remain 64 features in the CICIDS2017 dataset. After then, employed the information gain-based feature selection approach to identify useful features among 84 features as the initial phase in our ML process.

Table 4-4: The information gains the rank of features

No.	Attribute Name	Score attribute	No.	Attribute Name	Score attribute	No.	Attribute Name	Score attribute
1	Source IP	0.97903	29	Packet Length Variance	0.531761	57	Protocol	0.200314
2	Destination IP	0.955359	30	Fwd Packet Length Std	0.504997	58	URG Flag Count	0.192125
3	Subflow Fwd Bytes	0.939343	31	Bwd IAT Max	0.50402	59	Down/Up Ratio	0.198318
4	Total Length of Fwd Packets	0.939343	32	Bwd IAT Total	0.503071	60	min_seg_size_forward	0.157849
5	Average Packet Size	0.80995	33	Bwd Packet Length Std	0.501199	61	Flow IAT Min	0.112749
6	Total Length of Bwd Packets	0.782456	34	Bwd Packets/s	0.475092	62	Idle Max	0.047888
7	Subflow Bwd Bytes	0.782456	35	Bwd IAT Mean	0.472621	63	Idle Mean	0.046425

8	Bwd Packet Length Mean	0.781841	36	Subflow Bwd Packets	0.470555	64	PSH Flag Count	0.045662
9	Avg Bwd Segment Size	0.781841	37	Total Backward Packets	0.470555	65	Idle Min	0.042428
10	Fwd Header Length	0.778016	38	Max Packet Length	0.427961	66	Fwd PSH Flags	0.041217
11	Destination Port	0.77582	39	Fwd Packets/s	0.411763	67	SYN Flag Count	0.041217
12	Bwd Packet Length Max	0.760317	40	Source Port	0.39621	68	Idle Std	0.035837
13	Init_Win_bytes_forward	0.708411	41	Bwd Packet Length Min	0.394222	69	Active Std	0.031044
14	Avg Fwd Segment Size	0.706064	42	Bwd IAT Std	0.339751	70	ACK Flag Count	0.006794
15	Fwd Packet Length Mean	0.706064	43	Flow IAT Std	0.331599	71	FIN Flag Count	0.002481
16	Fwd Packet Length Max	0.701009	44	Flow Bytes/s	0.308851	72	RST Flag Count	0.000144
17	Bwd Header Length	0.682524	45	Timestamp	0.304741	73	ECE Flag Count	0.000144
18	Flow ID	0.649026	46	Flow IAT Max	0.298929	74	Bwd Avg Packets/Bulk	0
19	Fwd IAT Max	0.630467	47	Flow Duration	0.298854	75	Fwd Avg Bytes/Bulk	0
20	Fwd IAT Total	0.62688	48	Flow Packets/s	0.287355	76	Bwd Avg Bulk Rate	0
21	Fwd IAT Mean	0.614771	49	Flow IAT Mean	0.275637	77	Fwd Avg Packets/Bulk	0
22	Total Fwd Packets	0.585548	50	Bwd IAT Min	0.259063	78	Fwd Avg Bulk Rate	0
23	Subflow Fwd Packets	0.585548	51	Active Min	0.240495	79	CWE Flag Count	0
24	Init_Win_bytes_backward	0.579844	52	Active Mean	0.235187	80	Bwd PSH Flags	0
25	act_data_pkt_fwd	0.571137	53	Fwd IAT Min	0.22963	81	Fwd URG Flags	0
26	Fwd IAT Std	0.569454	54	Active Max	0.228967	82	Bwd URG Flags	0
27	Packet Length Mean	0.568029	55	Fwd Packet Length Min	0.22525	83	Bwd Avg Bytes/Bulk	0
28	Packet Length Std	0.531846	56	Min Packet Length	0.200749	84	Label	0

Table (4.4) listed the scores of features by information gain feature selection method. In this study, the variance scores and the information gained were used to discover the perfect list of features, according to hybrid feature selection. By applying variance to exclude useless features with a variance score less than 3.0. In addition, discarding features with a minimum weight of 0.2 from the information gain, then 14 selected features are produced by algorithm (3.4) features selection set-up.

Table 4-5: 14 Features selected

No.	Feature name	Description of features
1	SourceIP	Source IP address
2	DestinationIP	Destination IP address
3	SourcePort	Source port number
4	DestinationPort	Destination port number
5	Protocol	Types of protocols (e.g : tcp or udp)
6	FlowDuration	Duration flow in MS
7	TotalFwdPackets	Total packet in the forward direction
8	FlowPackets/s	Number of packets bytes per second
9	FlowIATMean	Mean of the time between two packets sent in the flow
10	FlowIATStd	Standard deviation of the time between two packets sent in the flow
11	FwdIATTotal	Total of the time between two packets sent in the forward direction
12	FwdIATMean	Mean of the time between two packets sent in forward direction
13	FwdIATStd	Std of the time between two packets sent in forward direction
14	FwdPackets/s	Number of forward packets per second

According to the features selection set-up algorithm (3.4), produce 14 features as shown Table (4.5).

4.4.3 The Selection of Classifiers Algorithms and Their Performance

The process of selecting the appropriate classifier is essential; thus, the search for the classifier must be not only efficient but also fast and efficient.

In the current study, two mechanisms were used to select the features, and aggregated as groupings (GROUPS, SETS), as shown in Tables (4.1) and (4.7), respectively. Correspondingly, the results of ML are discussed in the subsections, which are divided into the methods of feature selection and data flow distribution. It will be properly explained in the subsections that follow.

A. Discussions of the Results

This subsection produces results based on the first groups of features. It is also presenting the results of multiple tests, each test has its own method and

techniques that it has adopted to reach the final results in the field of DDoS attack detection using ML. It is depending on grouping features in Table (4.6).

The 1st Case Study

Features were selected based on information gain and recent works recommendations. The features were chosen as shown subsection (4.3.4) - A. Table (4.6) lists the feature groups.

Table 4-6: The Feature Groups

S. No	Number of features group	Features
1	Feature group 1	Source Port, Destination Port, Protocol, Flow Bytes/s, Flow Packets/s, Packet Length Mean, Packet Length Variance, Average Packet Size
2	Feature group 2	Source and Destination Ports, Protocol, Flow Bytes/s, Flow Packets/s, Packet Length Mean, Packet Length Variance
3	Feature group 3	Destination Port, Protocol, Flow Bytes/s, Packet Length Mean, Packet Length Variance
4	Feature group 4	Destination Port, Flow Bytes/s, Packet Length Mean, Packet Length Variance

The four major performance measures criteria were used to evaluate the performance of classifiers. There are four of them: Accuracy, Recall, Precision, and F1 scores.

The evaluation criteria are used to calculate fit time (testing time), which shows the classifier's fitting time during the testing stage. The tables show how well classifiers perform in terms of group concern. The performance of classifiers is shown in Table (4.7). (Feature group 1). In this experiment, the first test mode of split data was used. In this scenario, all eight attributes are considered while categorizing the items. When the data is analyzed, four classifiers, REPT, RTree, RF, and J48, are shown to have an accuracy of better than 99%. The J48 classifier has a 99.95% accuracy. The RF classifier able know all fake packets with a 99.95% accuracy score, but it takes longer to test than DT. DS classifier has the lowest accuracy score of 97.7%, yet it is still respectable.

Table 4-7: Performance of Classifiers on Feature Group 1 with 8 Selected Features

Algorithm	Accuracy%	Recall%	Precision%	F1 score%	Fit Time
REPT	99.9513	1.00	1.00	1.00	0.36
DS	81.5791	1.00	0.76	0.86	0.21
RT	99.961	1.00	1.00	1.00	0.13
RF	99.969	1.00	1.00	1.00	3.99
J48	99.9566	1.00	1.00	1.00	0.13

Table (4.8) shows the classifier's performance with feature group 2. The first test mode of split data was employed in this experiment. The Decision Tree classifier obtains the greatest accuracy of 99.81% when using feature group 2 with 7 specified features. The RF, RT, and REPT classifiers all have the same accuracy score for feature groups 2 and 1. Except for one exception, reducing the number of features from eight to seven has no discernible effect on classifier performance. It's also worth noting that with feature group 2, the accuracy of the J48, RF, and REPT classifiers reduces marginally.

Table 4-8: Classifier Performance on Feature Group 2 with 7 Selected Features

Algorithm	Accuracy%	Recall%	Precision%	F1 score%	Fit Time
REPT	99.8219	0.999	0.998	0.998	0.11
DS	81.5791	1.00	0.76	0.86	0.08
RT	99.7821	0.998	0.998	0.998	0.15
RF	99.8379	0.999	0.998	0.999	2
J48	99.81%	0.998	0.998	0.998	0.21

Table (4.9) shows the performance of classifiers using feature groups 3 and 5 chosen features. A third test mode with split data was employed in this experiment. The classifiers REPT and J48 attain the greatest accuracy of 99.44% and 99.44%, respectively. About feature group 2, the accuracy of RF and RT classifiers falls.

Table 4-9: Classifier Performance on Feature Group 3 with 5 Selected Features

Algorithm	Accuracy%	Recall%	Precision%	F1 score%	Fit Time
REPT	99.44%	0.998	0.992	0.995	0.26
DS	81.19%	1.000	0.858	0.751	0.51
RT	99.25%	1.00	0.992	0.993	0.33
RF	99.28%	0.994	0.993	0.994	2.58
J48	99.44%	0.998	0.992	0.995	0.23

Table (4.10) shows the performance of classifiers using feature groups 4 and 5 chosen features. In this case, the REPT and Decision Tree classifiers attain the maximum accuracy of 99.44% and 99.44%, respectively. About feature group 2, the accuracy of RF and RT classifiers falls. Classifier fitting time for the RF classifier is the longest for all three feature SETS, followed by the REPT classifier.

Table 4-10: Classifier Performance on Feature Group 4 with 5 Selected Features

Algorithm	Accuracy%	Recall%	Precision%	F1 score%	Fit Time
REPT	99.48	0.993	0.995	0.994	0.38
DS	81.6164	1.00	0.755	0.86	0.27
RT	99.4631	0.996	0.995	0.995	0.67
RF	99.467	0.996	0.995	0.995	2.78
J48	99.5127	0.999	0.993	0.996	0.23

For eight attributes, Figures (4.5–4.8) provides a comparison of classifiers through accuracy, precision, recall, and F1 Measure.

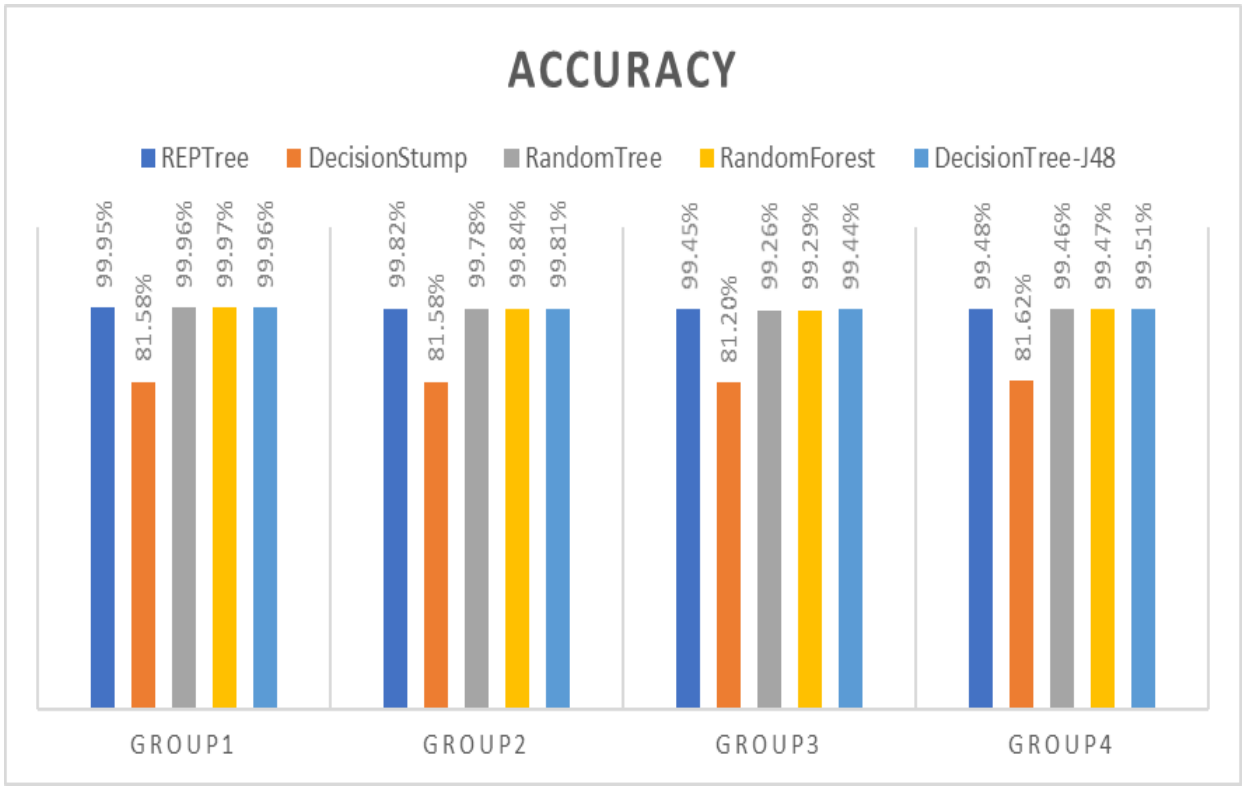


Figure 4-5: Classifier's accuracy for eight Feature Groups

Figure (4.5) depicts the evaluation of classifiers for the eight-feature group 1. The RF classifier achieves the best accuracy of 99.995% for feature group 1 with all 8 features. With 8 features, the REPT, Decision Tree, and RT classifiers get the maximum accuracy. Algorithm accuracy on average (excluding DS) with feature groups 1, 2, 3, and 4 is 99.96%, 99.81%, 99.36%, and 99.48%, respectively. Feature group 1 with all eight features has the best overall classifier accuracy.

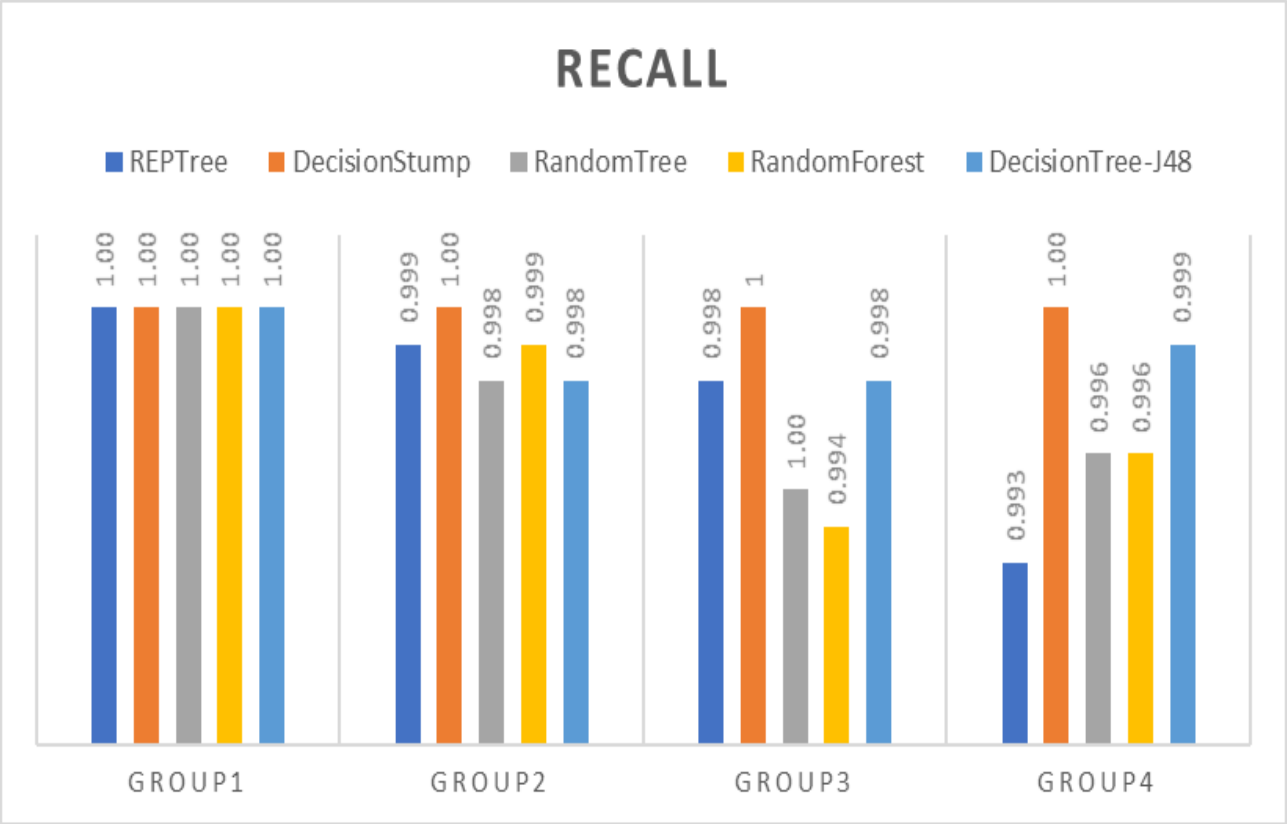


Figure 4-6: Classifier Recall for Eight Feature Groups

Figure (4.6) depicts classifier recall for the three feature groupings. With eight feature groups, the recall of classifiers J48, DS, REPT, and RF remains the same. With four groups, the DS classifier obtains higher recall. Across eight feature groups, the recall of all classifiers is nearly constant. RT classifiers achieve the greatest recall for feature groups 1 and 3. Feature groups do not affect the DS classifier's recall. The average recall of algorithms with feature group 1, feature group 2, feature, and feature group 4 are 100%, 99.88%, 99.70%, and 99.68%, respectively. Feature group 1 achieves the highest overall recall.

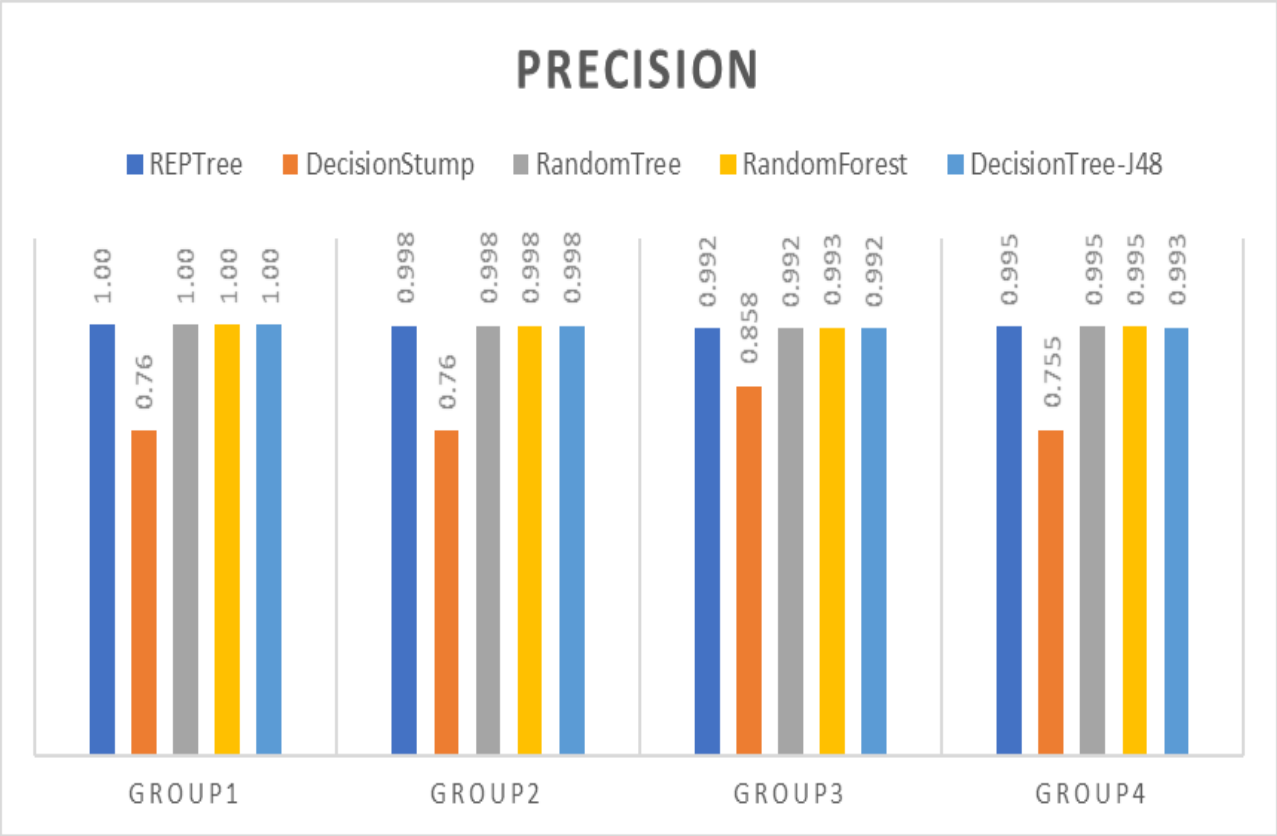


Figure 4-7: Precision of Classifiers for three Feature Groups

In terms of accuracy, as shown in Figure (4.5), all classifiers except DS score 100% with feature group 1. Except for DS, all classifiers achieve higher precision with feature group 2. The majority of algorithms are capable of detect DDoS attacks with great accuracy.

Except for the DS, the average precision of algorithms with feature groups 1, 2, 3, and 4 is 100%, 99.80%, 99.2%, and 99.4%, respectively. Feature group 3 achieves the highest overall recall.

Figure (4.8) depicts the F-measure scores of classifiers for the four feature groups. With feature group 1, RT, RF, and Decision Tree classifiers acquire a higher F measure. Feature groups have an impact on these algorithms. Average F-measure for all classifiers for feature group 1, feature group 2, feature group 3, and feature group 4 are 99.98%, 99.83%, 99.43%, and 99.50% respectively. Feature group 1 has the greatest overall classifier F-measure score.

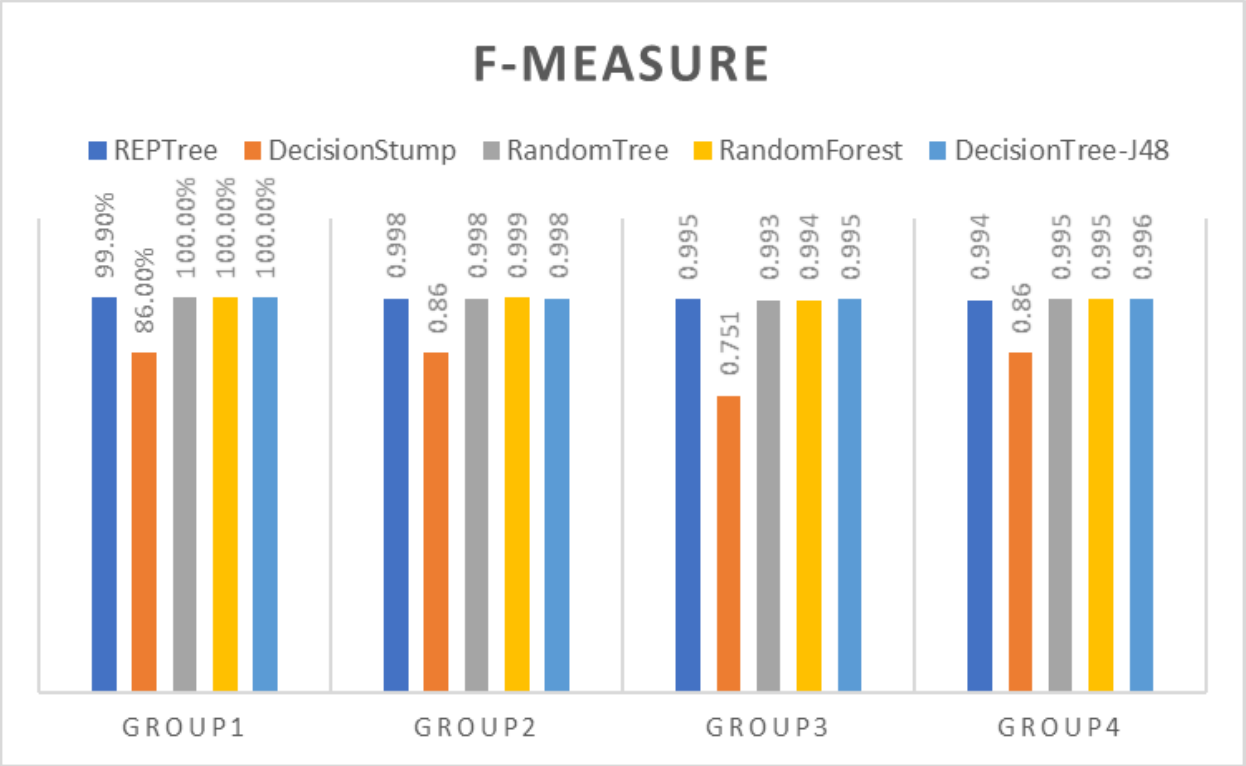


Figure 4-8: The F-Measure for four different feature groups

After examining the four performance measures obtained by the five classifiers for the four feature groups, it is discovered that feature group 1 with all features achieves the highest overall classifier performance. According to the experimental results, the CICIDS2017 dataset characteristics chosen and employed in this work are capable of identifying DDoS attacks with an average accuracy of 99.65%, a recall rate of 99.82% on average, the average accuracy was 99.62%, while the average F1 score was 98.68%. Table 7summarizes the best accuracy score for each feature group. The RF classifier achieved the maximum accuracy of 99.96 % when all eight characteristics were used.

While conducting the classification with six features, the J48 and REPT classifiers detected attacks with 99.44 % and 99.45 % accuracy, respectively. Using only four significant characteristics for classification, the J48 and REPT classifiers obtained 99.51% and 99.48 % accuracy, respectively. With the four best-ranked characteristics – 'Destination Port, Flow Bytes/s, Packet Length Mean, Packet Length Variance', J48 is capable of identifying DDoS attacks -

HTTP request flooding attacks, TCP SYN flooding attacks, UDP flooding attacks, and ICMP flooding attacks. These characteristics may be applied to the development of lightweight models, it is used in multi-stage classification systems for first-stage classification. Table (4.11) shows best accuracy of classifier model in each group features (1,2,3,4).

Table 4-11: The best feature group accuracy score

No	Number of features group	Features	The best classifier based on the rate of accuracy
1	Features group 1	Source Port, Destination Port, Protocol, Flow Bytes/s, Flow Packets/s, Packet Length Mean, Packet Length Variance, Average Packet Size	99.96% with RF classifier
2	Features group 2	Source and Destination Ports, Protocol, Flow Bytes/s, Flow Packets/s, Packet Length Mean, Packet Length Variance	99.94% with RF classifier
3	Features group 3	Destination Port, Protocol, Flow Bytes/s, Packet Length Mean, Packet Length Variance	99.44% with Decision tree-J48 and REPT classifiers
4	Features group 4	Destination Port, FlowBytes/s, PacketLengthMean, Packet Length Variance	99.51% with Decision tree-J48 classifier

Table (4.12) highlights the accuracy of classifiers acquired in previous works.

Table 4-12: The previous work on classification is compared

No	Authors	Dataset	Accuracy of the classifier %	No. Features
1	K. Kurniabudi et al [128] , 2020	CICIDS2017	99.79% 96.47%	15 4
2	D. Stiawan et al [126], 2021	CICIDS2017	99.79%	22
3	S. D. Çakmakçı et al [134], 2020	CICIDS2017	99.55%	10
4	Y. M. Swe et al [132],2021	CICIDS2017	99.50%	unknown
5	**The Proposed J48**	CICIDS2017	99.51% 99.96%	4 8

In comparison to previous works, the proposed model using the top four features based on feature score derived was able to accurately detect DDoS attacks. Based on tests, the proposed J48 model takes short time to test and good

perform. As indicated in Table (4.12) for prior efforts, group 4 performed with an accuracy of 99.51%.

The 2nd Case Study Based Paper # 2

Table (4.13) shows the performance of classifiers depending on 7 selected features – group 2 in Table (4.11). The test mode of split data 50:50 ratio was employed in this experiment. In this case, all seven features are taken into account for categorization. When the data are analyzed, it is discovered that four classifiers, REPT, RT, RF, and PART classifier, have an accuracy greater than 99%. The accuracy of the PART classifier is 99.77%. The RF classifier can detect all fraudulent traffic and get an accuracy score of 99.84%, however, it requires more testing time than the PART. DS has the lowest, but still respectable, accuracy score of 81.58%.

Table 4-13: The performance of classifiers with 7 selected features

No.	Algorithm	Accuracy%	Recall%	Precision%	F1 score%	Fit Time
1	REPT	99.79%	0.999	0.998	0.998	0.24
2	DS	81.58%	1.00	0.76	0.86	0.08
3	RT	99.78%	0.998	0.998	0.998	0.15
4	RF	99.84%	0.999	0.998	0.999	2
5	PART	99.77%	0.997	0.998	0.998	0.12

Evaluation metrics are used to perform comparison of classifiers as shown in figure 4.9.

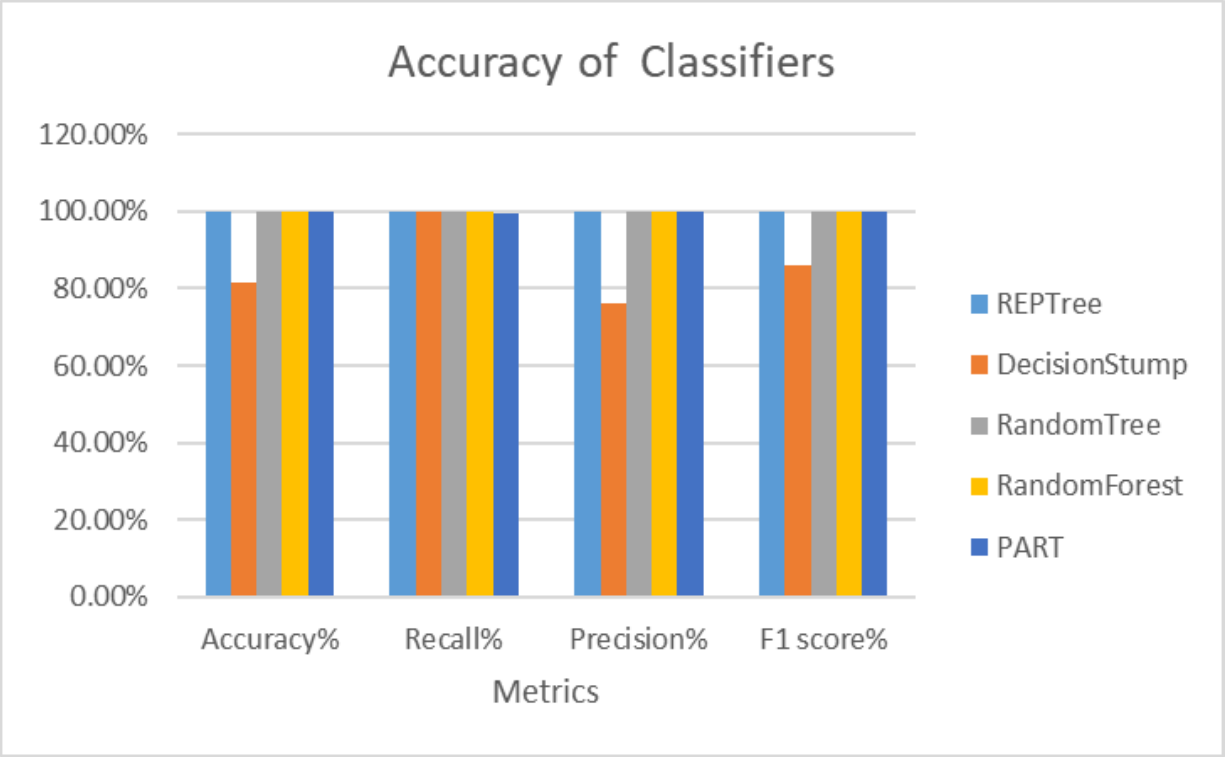


Figure 4-9: The performance comparison

B. The Information Gain – Variance Score Based Enhance Results

Information gain and variance giving 14 features as shown in Table (4.5). these features are most relevant and applicable with SDN flow statistics. After serios of tests using of the 14 features, generated four SETS of them, as shows in Table (4.14). The column entitled "Number of features in SETS" indicates the numbers of feature from Table (4.5).

Table 4-14: The features SETS

No	SET No.	Number Features in SETS	No. features
1	Features SET 1	4,5,6,7,8,9,10,11,12,13,14	11
2	Features SET 2	4,7,8,9,10,11,12,13,14	9
3	Features SET 3	4,5,7,8,11,14	6
4	Features SET 4	7,11,12,13,14	5

Using the three performance measures (Testing time, Tree size, and Accuracy), the performance of classifiers and feature selection was evaluated.

Accuracy reflects the accuracy of the algorithm in detecting attacks over both normal and infected packets.

This work used test mode by split 50% train, remainder test. It is used to help of choosing the suited of ML classifiers. This mode can aid in the development of a lightweight classifier model as well as the selection of important features, because make balancing between accuracy and testing time.

Test time, which indicates the model rules' testing time during the testing step, is also calculated together with the other three evaluation criteria. Table (4.15) presents the accuracy obtained from several classifiers (feature SETS). Within the context of this experiment, the 50:50 ratio split mode of split data was utilized.

When the data are analyzed, it is found that there are five classifiers that have an accuracy that is more than 99% in features SETS (1,2,3). These classifiers include the REPT, PART, RT, RF, and the J48 classifier. The RF classifier has a high level of accuracy, reaching 99.8%. However, in order to get an accuracy score of 99.8%, the RF classifier takes more testing time than the other classifiers.

Table 4-15: The accuracy of classifiers on feature SETS with 8 selected features

ALGORITHM	SET1	SET2	SET3	SET4
REPT	99.85%	99.84%	99.76%	98.25%
DS	82.58%	82.58%	82.58%	82.58%
RT	99.86%	99.89%	99.78%	98.30%
RF	99.93%	99.92%	99.84%	98.54%
J48	99.87%	99.87%	99.77%	98.30%
PART	99.87%	99.82%	99.77%	97.87%

Table (4.16) shows the classifier's spent time in testing rules with feature four SETS. The RF classifier achieves the highest test time when using feature four SETS. While, the DS that lowest testing time lowest accuracy score for all feature SETS.

A decrease in testing time is observed when the number of features is reduced from eleven to five. It should also be noted that feature SET3 contains six features, the test time of the PART, and REPT classifiers lowest value than another four classifiers.

Table 4-16: The classifier testing time (ms) on feature sets

ALGORITHM	SET1	SET2	SET3	SET4
REPT	0.23	0.11	0.08	0.12
DS	0.27	0.08	0.12	0.14
RT	0.19	0.15	0.17	0.11
RF	2.82	2.74	3.21	12.59
J48	1.34	0.23	0.17	0.15
PART	0.23	0.18	0.1	0.29

Table 4-17: The classifier tree size on feature SET 3 with 6 selected features

Algorithm	Tree size
REPT	289
DS	22
RT	1179
RF	1599
J48	325
PART	65

For features SETS, Figures (4.10), (4.11), and (4.12) provide a comparison of classifier accuracy, testing time, tree size depending outcomes values as shown in tables (4.15), (4.16), and (4.17).

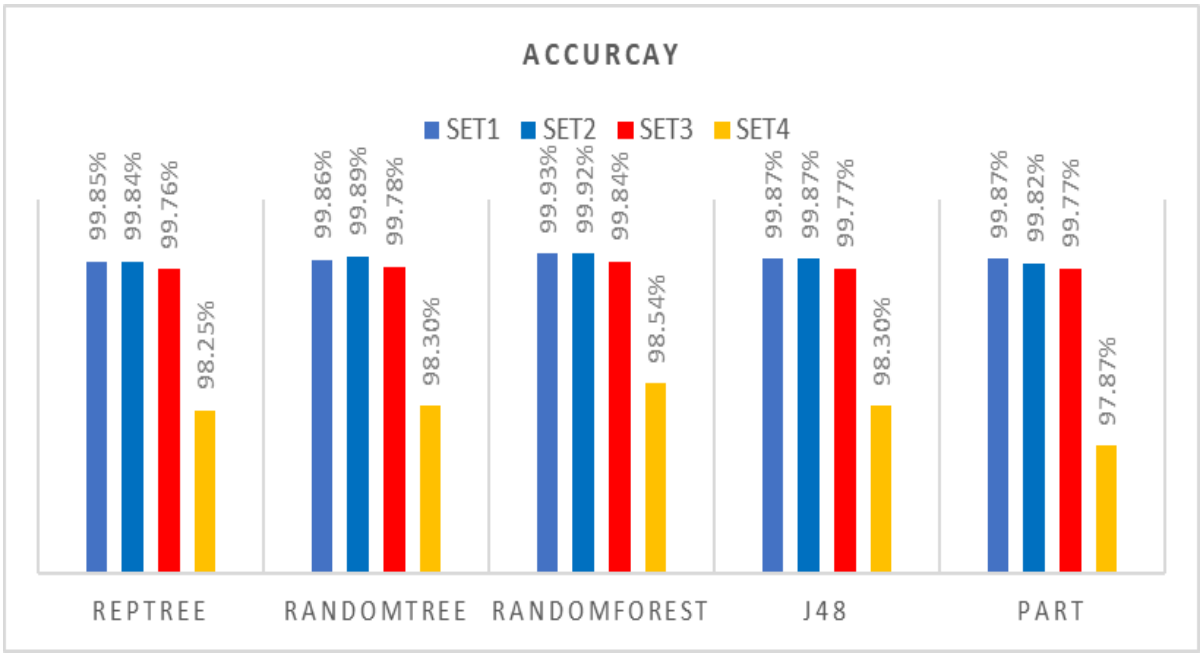


Figure 4-10: The classifier's accuracy for four feature SETS

Figure (4.10) illustrates the evaluation of classifiers for each of the four feature SETS. The RF and PART classifiers achieve the best accuracy of 99.93% and 99.87%, respectively, for feature SET 1 with all 11 features. With 11 features, the PART, REPT, J48, and RT classifiers get the highest accuracy.

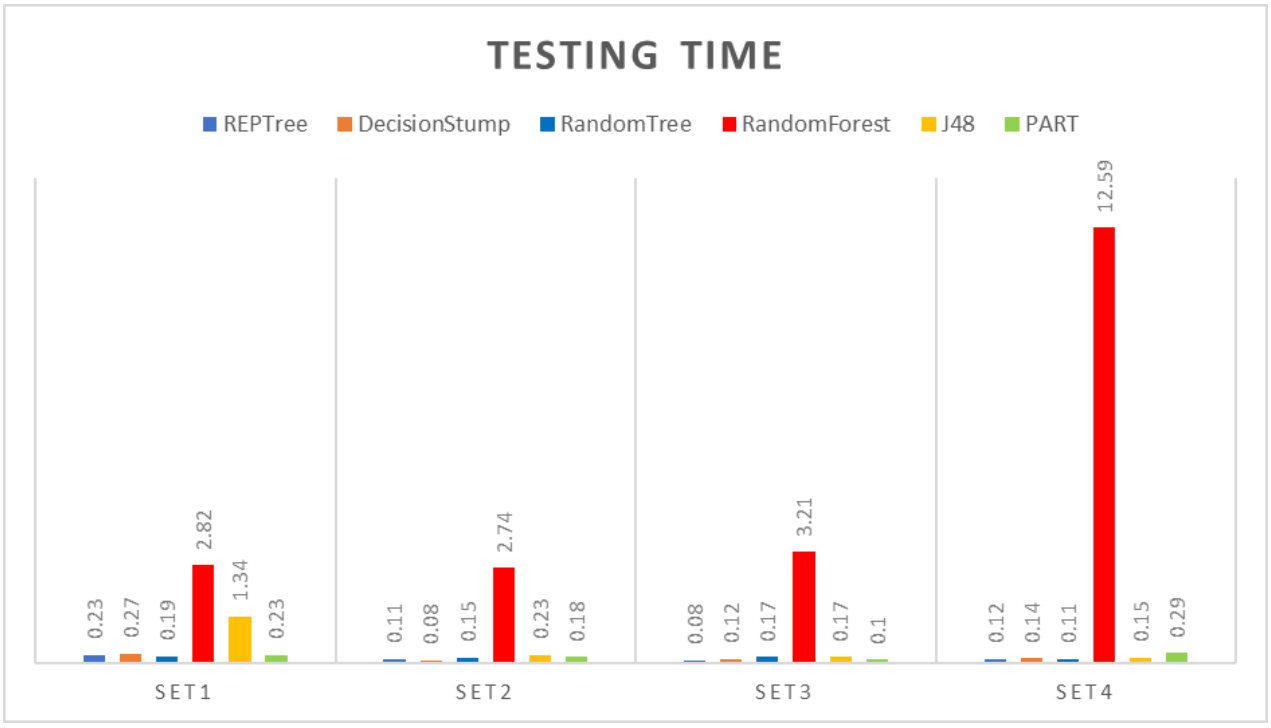


Figure 4-11: The classifier testing time period for four feature SETS

Figure (4.11) depicts classifier testing time for the four feature SETS. The testing time has huge impact of performance in real-time constant, when it is increase lead to decrease the performance and verse versa. With feature SET4, the testing time of classifiers J48, DS, REPT, and PART remains the same but RF was registered highest testing time.

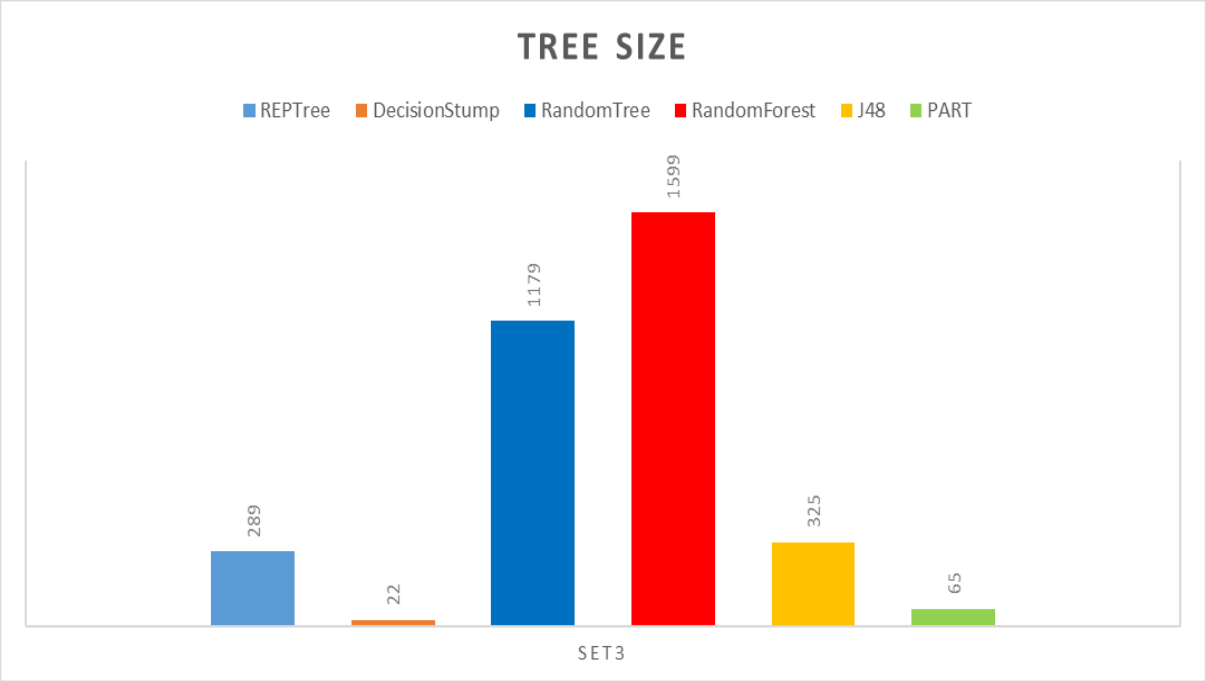


Figure 4-12: The classifiers' tree size

The figure (4.13) represents the tree size, which is the number of rules that will be approved for each classifier. With a dimension that explains to us what is the best classifier in terms of accuracy according to the form, it has become necessary to Identify the best classifier in terms of test time and the fewest number of rules according to the result of performance in the Figures (4.11) and (4.12), also shows that the classifiers have different accuracy and different tree sizes. So, in this case, must create a balance to find an efficient classifier with high accuracy and small tree size, so it becomes a high performance when applied in real-time detection of DDoS attacks.

Analyzing the results of the six classifiers on the three performance measures for each of the four feature SETS, find that feature SET 1 with all

features gets the best overall classifier performance. In this study, use features selected from the CICIDS2017 dataset using algorithm (3.4), according to the experiments, DDoS attacks can be identified with 99.87% reliability.

The best accuracy score for each feature SETs is summarized in Table (4.18). When all 11 features were used, the RF classifier achieved the highest accuracy of 99.92%.

While conducting the classification with six features, found that PART is lightweight and high accuracy. According n tests of SET3, as shown in figures (4.10), (4.11), and (4.12), this result has been proved that the PART classifier is the best depending on achieved 99.77% accuracy by using 65 rules and testing time 0.1 ms, while RF achieved 99.53% with SET 4 by using 1599 and testing time 3.21 Ms.

Table 4-18: Accuracy score is determined by the best feature.

S. No	Number of features set	Features	The most effective classifier, determined by its high accuracy
1	Feature SET 1	4,5,6,7,8,9,10,11,12,13,14	RF ; 99.92%
2	Feature SET 2	4,7,8,9,10,11,12,13,14	RF ; 99.92%
3	Feature SET 3	4,5,7,8,11,14	PART ; 99.77% and RF; 99.83%
4	Feature SET 4	7,11,12,13,14	RF ; 98.53%

C. The Modified PART

This section presents the modified PART results. Experiments were performed on all feature sets, and it was found that the best compromised results can be obtained by reducing the tree size to speed up the decision process and the same time keep the accuracy in accepted levels.

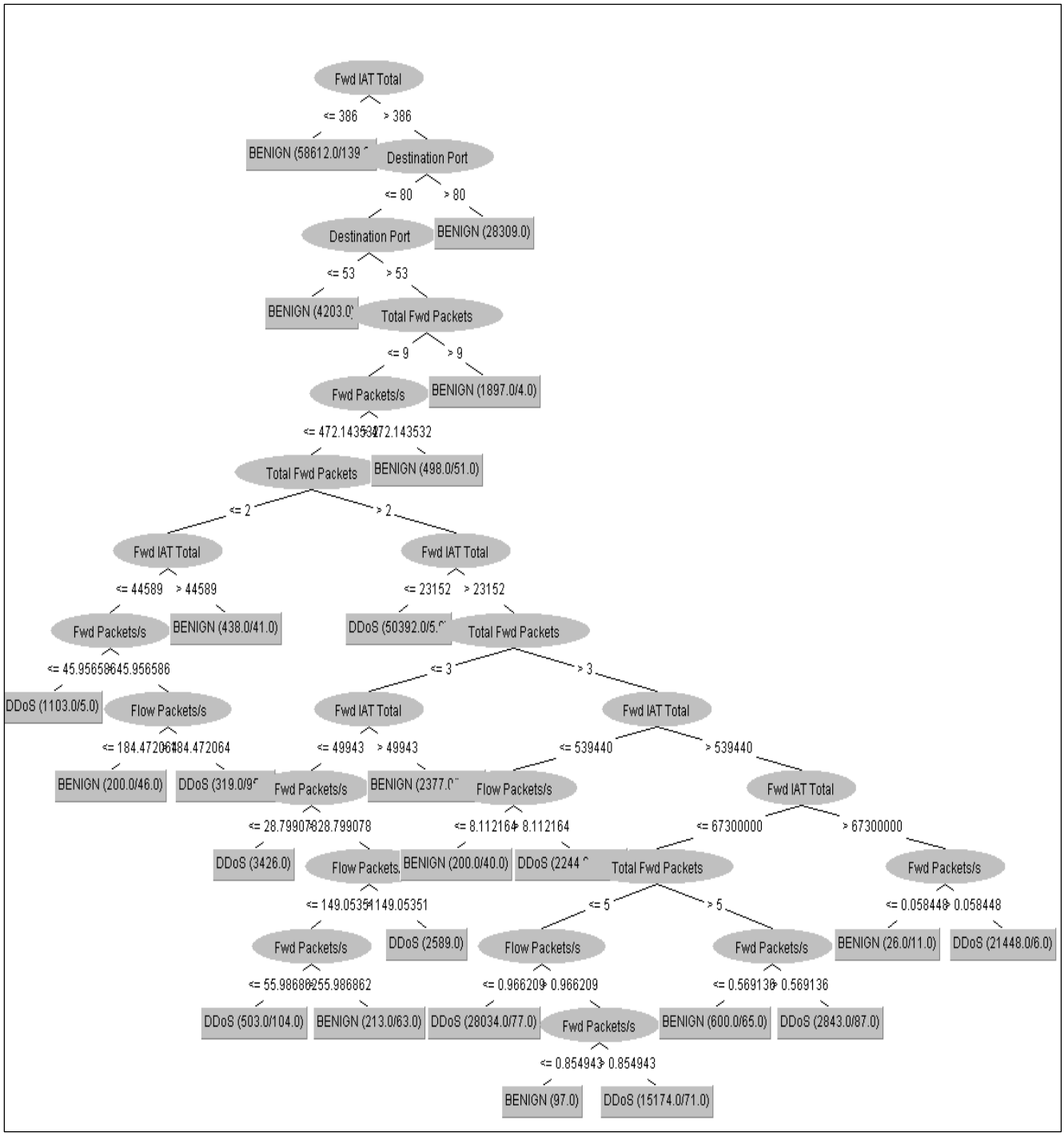


Figure 4-13: The final outcome of the training phase (decision tree)

Applying the modified PART on SET 3 features gives the accuracy of 99.39%. Figure (4.14) shows a sample the size of the tree (rules set) and Figure (4.15) shows the time test.

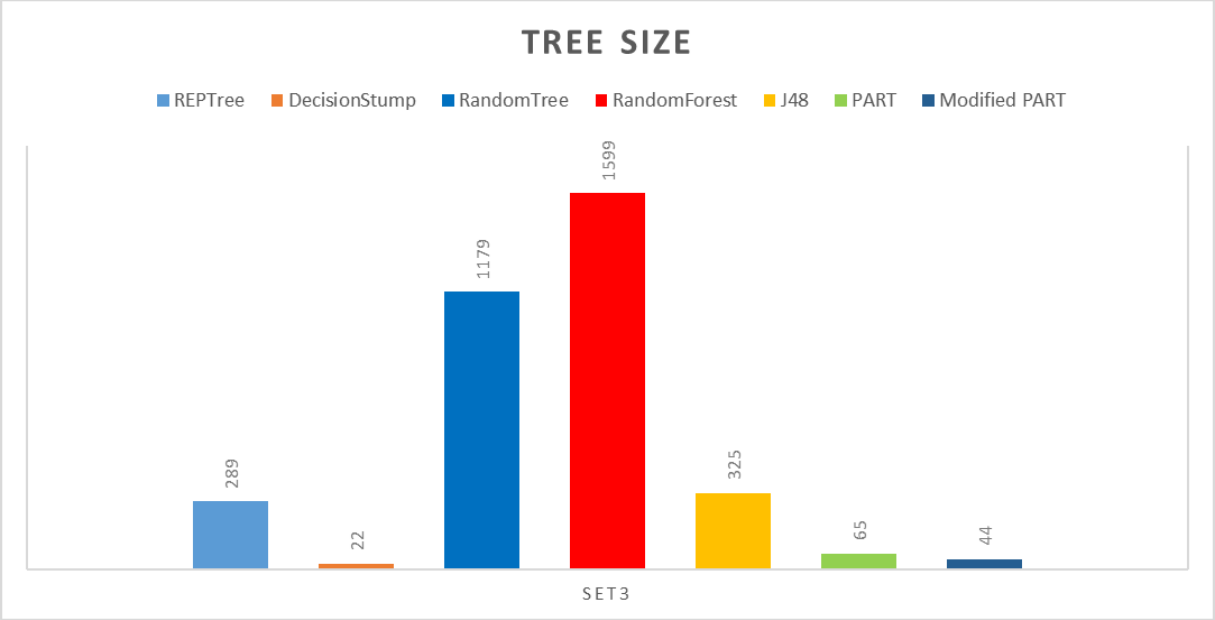


Figure 4-14: The tree size of modified PART (number rules)

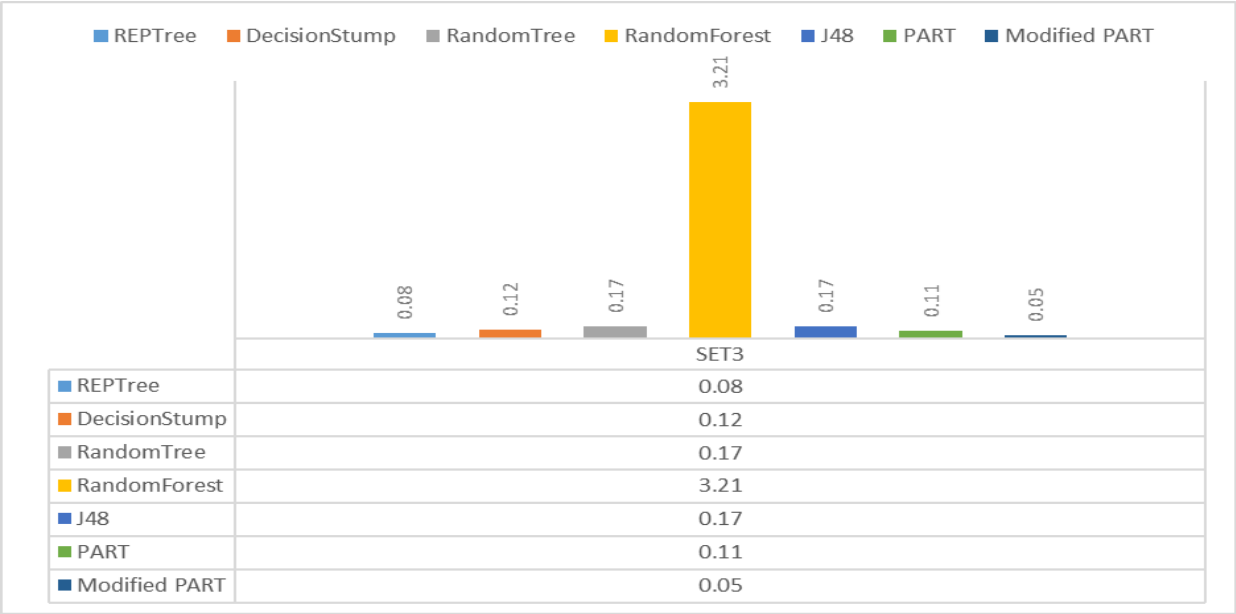


Figure 4-15: The time testing SET3 features using Modified PART

The performance of the modified PART model indicates that it is better compared to the original and outperforms the remaining classifiers, as described in the preceding sections. Therefore, the modified PART-generated set of rules will be adopted as a model for detecting DDoS attacks in the SDN environment.

D. The Semi-supervised

The performance of the detection engine can also be measured by its accuracy. The machine's ability to predict traffic based on its actual conditions is indicated by its accuracy. In other words, the capacity of a machine to precisely classify a class. Figure (4.15) and Table (4.19) present values of generated centroids of the proposed method.

Table 4-19: The values of generated centroids

No.	Feat. Name	Centroid1 (Normal)	Centroid2 (DDoS)
1	Destination Port	80	443
2	Protocol	6	6
3	Total Fwd Packets	3	49
4	Flow Packets/s	8.879743	751.330481
5	Fwd IAT Total	687	143726
6	Fwd Packets/s	3.329904	340.881422

The proposed method is providing two optimum centroids from test1 to classify traffic into normal and DDoS attack. Table (4.19) displays K-means accuracy performance.

Table 4-20: The accuracy of K-means and Canopy algorithms

Test No.	K-means Accuracy	Canopy Accuracy
Test1 (2 centroids)	79.60%	72.30%
Test2 (4 centroids)	68.90%	65.70%
Test3 (6 centroids)	42.10%	55.90%

The results as shown in Table (4.19), shows that the test 1 was the best choice to achieved accuracy with 2 centroids that labeled into normal and another with DDoS. Figure (4.16) present performance comparison between the proposed K-means and Canopy.

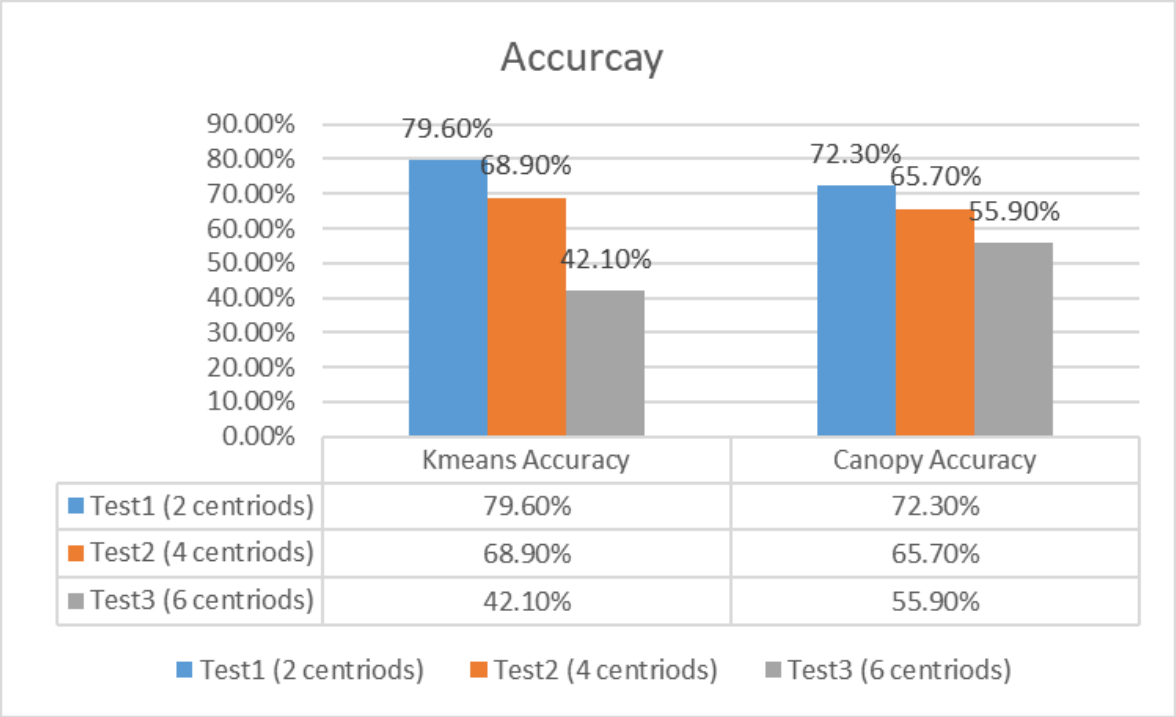


Figure 4-16: The performance of the proposed method and Canopy

In this study, the classify unknown DDoS attacks using a semi-supervised ML method. It starts with traffic statistics that aren't labeled that are gathered. K-Means clustering algorithm group the data that doesn't have labels. The scheme used a representative subset of the benchmark CICIDS2017 dataset with new normal and attack centroids to test how well labels were given.

4.5 The Proposed System in The SDN Environment

The following subsections present the proposed system implementation in SDN

4.5.1 The Proposed Topology

Mininet is selected as the network emulation platform for the first section of our experiment, which employs POX as the SDN controller. The detecting application will be a separate module integrated within the L3 Learning module. The controller will now be capable of installing flows and verifying DDoS attacks. During this testing phase, OF-switch is used to simulate the behavior of an edge switch in an SDN network. A client in the SDN network generates both regular and attack traffic aimed at a victim node in the Mininet network.

4.5.2 The System Architecture

The initial packet is transferred from one client to another during typical communication between clients in an SDN environment. The switch's flow table will initially have no entry indicating where the incoming packet should be sent. The packet will be sent to the controller by default. The switch will be instructed by the controller to forward the packet to the appropriate destination port. The controller will also give an entry rule for that source and destination in the switch's flow table. Without the intervention of the controller, packets with the same source and destination will be routed automatically in the future.

4.5.3 The Covered Attack Types

Entropy and ML techniques-based detection methods on SDN are designed to detect various attack types.

According to Kaspersky reports [135]–[138], the most common type of attack by a wide margin is UDP flooding in 2019,2020,2021 and first quarter in 2022. In that regard, the implementation is designed based on detect to UDP and TCP protocol-based attacks such as TCP flood and UDP flood. However, our method can adapt to any flooding attacks. For future work, we aim to make this study large scale attack detection module covered all types of attack.

4.5.4 The Environment Setup and Test

During this testing phase, OF-switch is used to simulate the behavior of an edge switch in an SDN network. A client in the SDN network generates both regular and attack traffic aimed at a victim node in the Mininet network. A virtual host may be attacked with the Mininet detection mechanism. The IP addresses issued to customers span between 10.0.0.1 and 10.0.0.8. As shown in Figure (4.16), 8 hosts, 3 open virtual switches (OVS), and a single POX controller are installed for this proposed system.

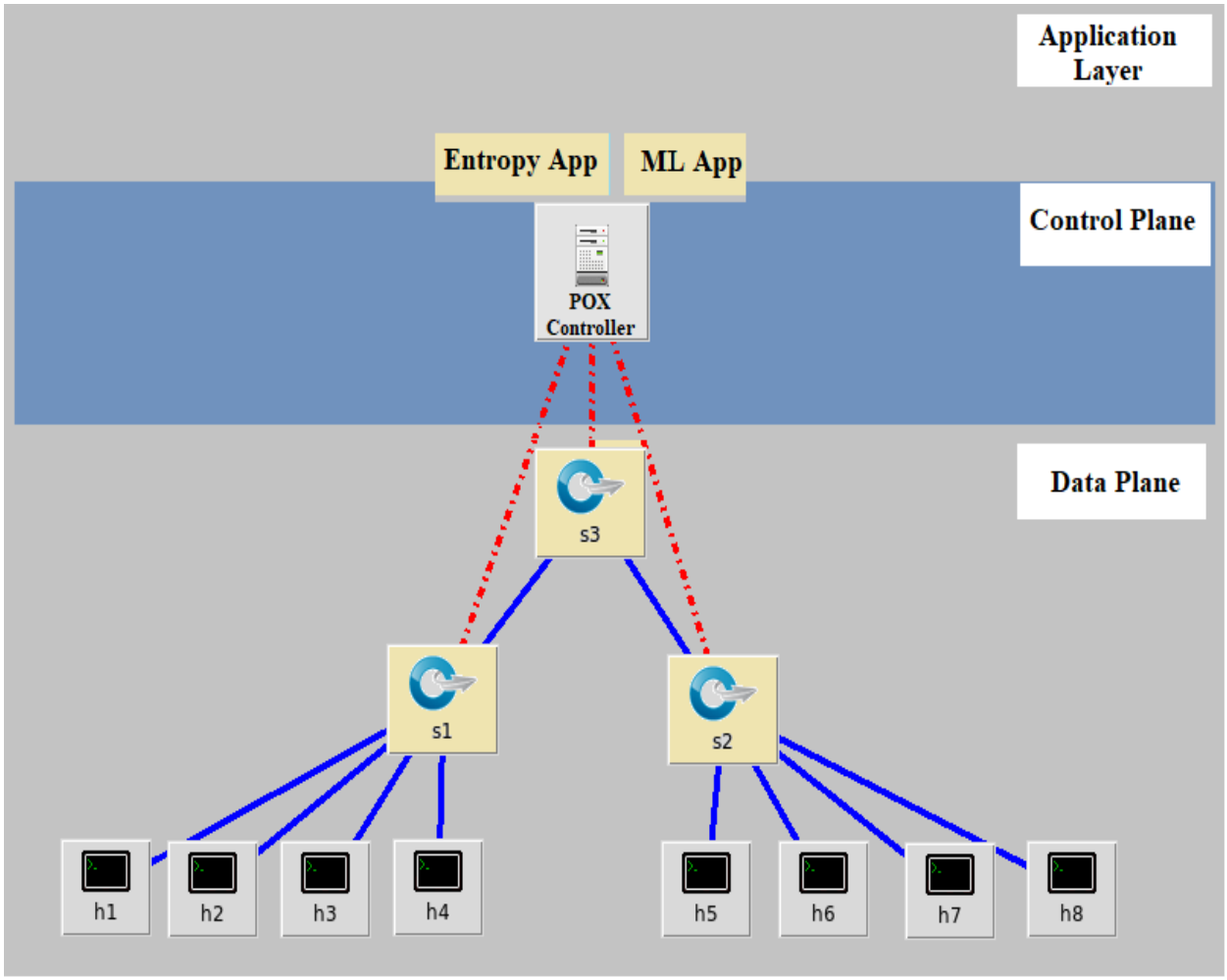


Figure 4-17: The Mininet topology of 3 switches and 8 hosts with a single POX controller

A. The Implement Entropy

Despite the fact that typical networks consist of twelve of devices, the previous test was designed to simulate comparable problems in large networks due to limited resources. A built a virtual network with the same limitations as our testbed so that could develop and test our solution in a real-time setting. You can import functions like sendp, IP, UDP, Ethernet, and TCP with the Scapy module. It is used to make UDP packets with different payloads. In steady, normal traffic, it will generate 4 packet per second (inter=0.25). Figure (4.1) shows how Scapy script generates traffic in a virtual host (h1).

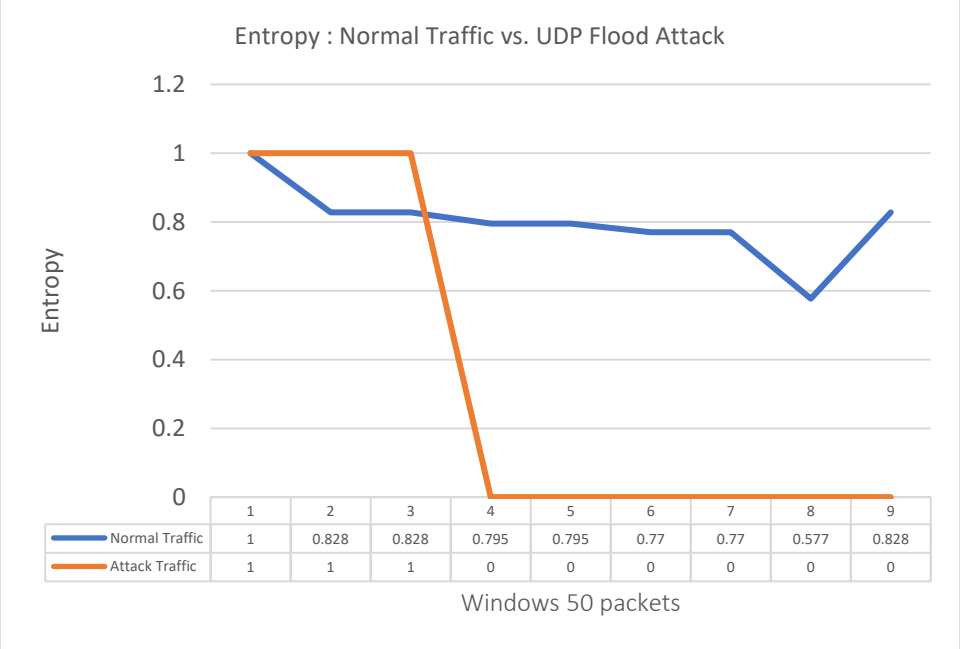
Before launching an attack with Scapy, launch normal traffic with Scapy to a predetermined threshold value that serves as a boundary between normal and attack traffic (DDoS attack). After several testing, the average entropy for a

normal traffic is calculated using normal traffic simulation. The purpose is to calculate the average normal entropy value for the proposed network topology and window size 50 packets to predefine the suitable threshold value.

In contrast to attack packets, which have short-time flows with tiny amounts of traffic (inter=0.025), normal traffic is described as traffic with lengthy duration flows. On the network, traffic with a 0.5 second traffic interval was begun (1/0.5=2 packets/sec) defines the traffic rate.

When a count of 50 packets in the window and then calculate the entropy and compare it with threshold that set and entropy value lower than threshold, which is mean there are suspicious traffic and then call ML phase to deep inspect packets. The graph in Figure (4.18) shows how the entropy changes during traffic. During this traffic flow, the lowest threshold was 0.577, and the highest point was 1 in first traffic.

The topology and parameters used to mimic attack traffic with numerous victims are the same as those used to simulate normal traffic. The tested a 0.025 attack rate for the victim attack. Figure (4.18) depicts the DDoS traffic interval: normal traffic and a UDP flood attack on entropy.



	1	2	3	4	5	6	7	8	9
Normal Traffic	1	0.828	0.828	0.795	0.795	0.77	0.77	0.577	0.828
Attack Traffic	1	1	1	0	0	0	0	0	0

Figure 4-18: The entropy values of normal traffic and DDoS attacks

The attack was discovered, as expected. The developed SDN app, as expected, detects the attack in less than one second. The DDoS attack is identified within first second with a higher traffic rate of 40 packets/sec. The entropy levels for the 25% attack rate are slightly greater, ranging from 0.511 to 0.00.

A DDoS attack's average detection rate is between 0.445 and 0.501 seconds with 96% as average accuracy. It's safe to state that the algorithm (3.3) does a good job of discriminating between normal and malicious traffic. It's vital to keep in mind that this simulation was created expressly for the test. The network and its traffic will grow in a real-world setting. If this is the case, the threshold must be changed to avoid false positive detection.

It is also clear that the entropy attack is detected in a short time compared to other techniques such as ML, but it suffers from a slight lack of accuracy of the results because some normal traffic goes to the same destination, whereas this method becomes unfair to some legitimate requests. Therefore, ML with the entropy method will be adopted to complement each other, as implement and described in next subsection.

B. The Implement Both of ML and Entropy

After applying entropy of attack detection, where it gave accuracy reached to 96%, vulnerabilities were identified that could be overcome using ML. A SDN application has been developed that includes the work of entropy as well as ML together, one complementary to the other.

The entropy will act as an indicator of suspicious traffic. If the entropy value reaches the predefined threshold, the traffic will move to the ML phase for further examination.

The entropy runs after the window size is completed with 50 packets, as mentioned in the previous section. In the ML stage, the POX controller collects packets based on the number of seconds. Every 1 seconds, packets arriving at the POX controller are collected and extracted each feature values in SET3 features

in the Table (4.18) will be calculated. After the feature values are obtained, they will be passed to the ruleset of the modified PART. Propose Modified PART produce 44 sets of rules which is can be to detect DDoS, as demonstrated in ML section.

After a series of experiments within the SDN environment, a sample of experiments was taken, which are three experiments in terms of accuracy, the first was 97.76%, the second was 97.81%, and the third was 97.89%. Based on the experiments studied, the accuracy rate was 97.82%.

Incorrectly classified packets 2.72% were split between wrong decision and unknown packets. An ***unknown packet*** means that the rule sets of the classifier failed to classify it as normal packets or DDoS attacks.

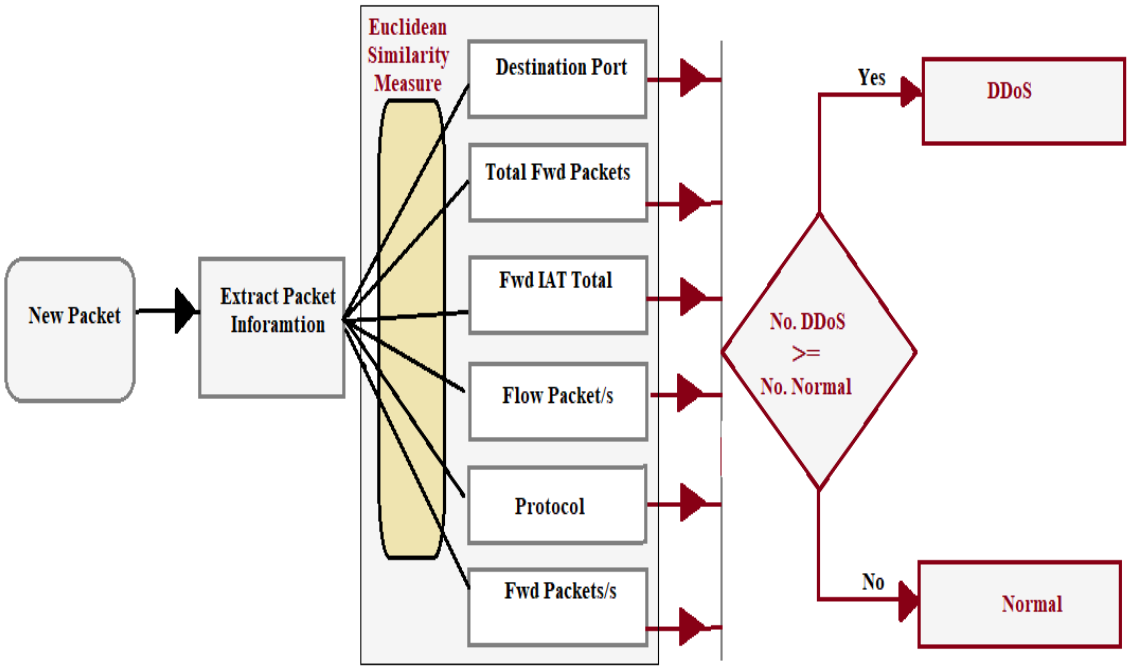


Figure 4-19 : The semi-supervised method in real-rime

In order to solve the problem of classifying unknown packets, the ***semi-supervised*** method is adopted. This method was developed as mentioned in Section 4. The accuracy improved to 98.06%, the improve rate about 0.24%, when added the semi-supervised method with 44 rules set of modified PART together.

The accuracy of each stage as it is used in the SDN environment is shown in Figure (4.18) along with the increase in accuracy brought on by the application of the aforementioned techniques.

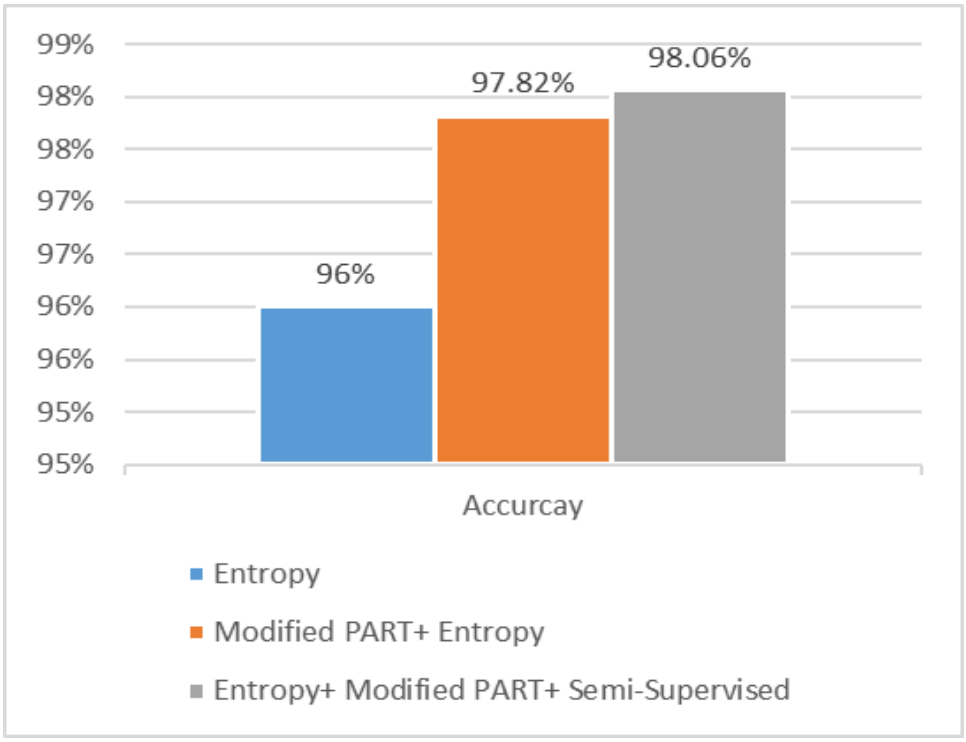


Figure 4-20: The performance comparison of entropy and ml approaches

4.5.5 The mitigation results

After the attack has been confirmed by ML stage of the proposed system, the mitigation module is called, and the attack traffic is dropped to prevent the breakdown of the switches in the network. Figure (4.21) shows the output of the execution of the mitigation module of the POX controller to drop the attack traffic.

Figure 4-21 : The execution of mitigation module

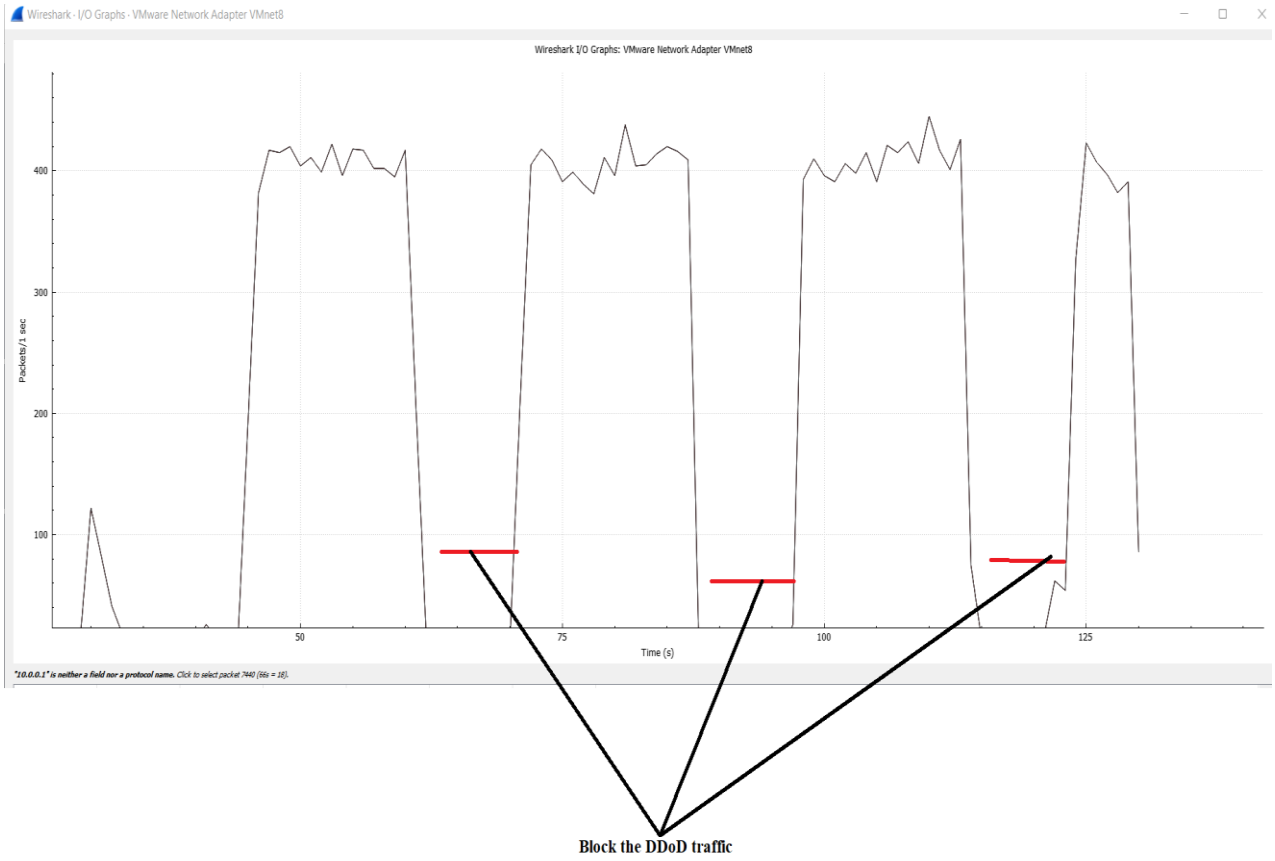


Figure 4-22 : The time(s) of block DDoS traffic

Figure (4.22) shows the block traffic about 5 second and then allow, check the traffic again and when find that the network is still under DDoS attacks, re block again.

4.6 The comparisons of the proposed system against related works

Table (4.21) shows comparisons of the proposed system against the recent related works.

Table 4-21 : The porposed system is compared with recent works

No	Authors	Network	Dataset	Accuracy of the classifier %	No. Features
1	K. Kurniabudi et al [128] , 2020	unknown unknown	CICIDS2017	99.79% 96.47%	15 4
2	D. Stiawan et al [126], 2021	unknown	CICIDS2017	99.79	22
3	S. D. Çakmakçı et al. [134], 2020	unknown	CICIDS2017	99.55%	10
4	Y. M. Swe et al. [132],2021	unknown	CICIDS2017	99.50%	unknown
5	C. Fan et al. [139], 2021	Mininet simulated	Simulated generation	91.25%	unknown
6	A. Maheshwari et al. [140], 2022	Mininet simulated	CICDDoS2019	99.41%	unknown

7	Y. Liu et al. [141], 2022	Mininet simulated	CICIDS2017	98.98%	unknown
#	**The Proposed System**	**Offline**: WEKA and Python application **Realtime**: Mininet Simulated	CICIDS2017	99.77% 99.96% 98.06%	6 11 6

4.7 The Experiment and Results of FSCA

In this study, presented the proposed FSCA, Flat Distributed SDN Controllers using AMQP: To manage multi controllers and address the difficulties of a single SDN controller, the concept of distributed SDN controller based on FSCA is introduced.

In this section will present the tools used in the experiment setup, how to apply enable POX controller to work as multiple controller strategy and the results.

To apply the proposed method, a test of the proposed network was carried out as follows: -

- **Before the Proposal Application**

When a user stops the controller manually or by any external intervention. It was undoubtedly noted that any *ping* does not work among SDN infrastructure devices (switches and hosts), and this is normally happened when the controller becomes out of service, this situation is considered one of the problems facing SDN network. Consequently, this challenge attracts more research to overcome this problem. Therefore, in this study, the problem handled by adding one extra controller working as back-to-back backup.

- **After the Proposal Application**

Using AMQP architecture, the two running services are configured (sender.py and receive.py). The first acts as sender and the second as a receiver. Both of them are written in Python, one for sending the queue and the other for receiving it.

After the implementation of proposal, the solution become as follows:

1- When the primary controller becomes down for any reason. The server detects the failures and responding by waking up the second controller within a milli second.
2- The server notifies the SDN switches (on the data plane side) to reconnect to the woken controller.
3- It is worthy noted that the network is controlled through the second controller to enable it to use the AMQP services to avoid the headache of single point of failure.

4.7.1 The Experiment Configuration

The initial section of our experiment is to select the Mininet is a network simulation tool for SDN. Mininet is a Linux feature that uses lightweight virtualization in the Linux kernel to isolate CPU bandwidth, and process groups, and merge them with virtual Ethernet lines. As a result, the system grows faster and contains more hosts than another emulator. The POX controller was employed in our experiment. It is lightweight and fast, and it is written in Python. POX improved on the preceding NOX controller [142]. It is worth mentioning at the beginning of clarifying the proposed method of work is to mention what are the tools that enabled us to implement the proposed method, as it consists of network tools which are (POX controller, Mininet). In addition to that, a new service was added, which is a mechanism for applying communication among SDN controllers. A communication protocol (AMQP) is used to make a connection between the controllers to overcome the single point of failure.

It is noteworthy that the experiments in this work went through three modes, as fellow:

A. ***In The First mode***, As can be seen in Figure (4.23 - A), it is completely usual to carry out one's duties on the network without experiencing any difficulties Pinging between the two hosts, h1 and h2, reveals that both of them are accessible to one another. On the controller side, the POX controller, which

includes an SDN application and is represented in Figure (4.23 - B), the control plane monitors any connection between data plane devices or between the data plane and the SDN controller.

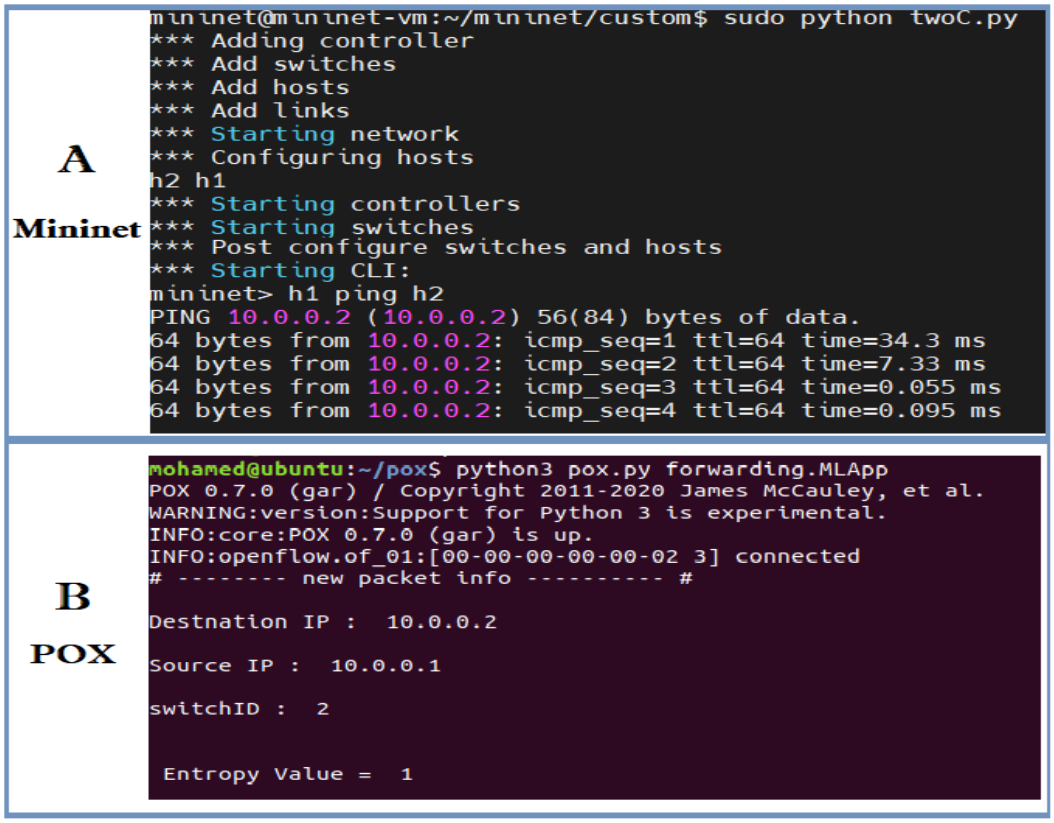


Figure 4-23: Normal mode –Startup topology (A) (Mininet) with POX controller (B)

B. ***The Second mode*** is the occurrence of a single point of failure without adopting our proposed solution, and as a result, the network becomes unreachable overall SDN terminal, as shown in Figure (4.24), which displays a ping that is unreachable across hosts on the SDN network.

```
*** Checking and Adding controller
Unable to contact the remote controller at 192.168.174.102:6633
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
h2 h1
*** Starting controllers
*** Starting switches
*** Post configure switches and hosts
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h2 -> X
h1 -> X
*** Results: 100% dropped (0/2 received)
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
From 10.0.0.1 icmp_seq=4 Destination Host Unreachable
From 10.0.0.1 icmp_seq=5 Destination Host Unreachable
From 10.0.0.1 icmp_seq=6 Destination Host Unreachable
From 10.0.0.1 icmp_seq=7 Destination Host Unreachable
```

Figure 4-24: Ping unreachable among hosts of SDN network

C. ***The Third mode***, A single point of failure scenario is explained with an implementation of the proposal, as shown in Figures (4.25) and (4.26). The topology is configured at the Mininet site of the primary POX controller when the connection with the primary fails, and it will be connected to the secondary POX controller. AMQP-based messages are sent between the server and the POX controllers, enabling instructions to run using the Mininet network and are developed to notify the module-based alert network administrator of the failure of the primary controller.

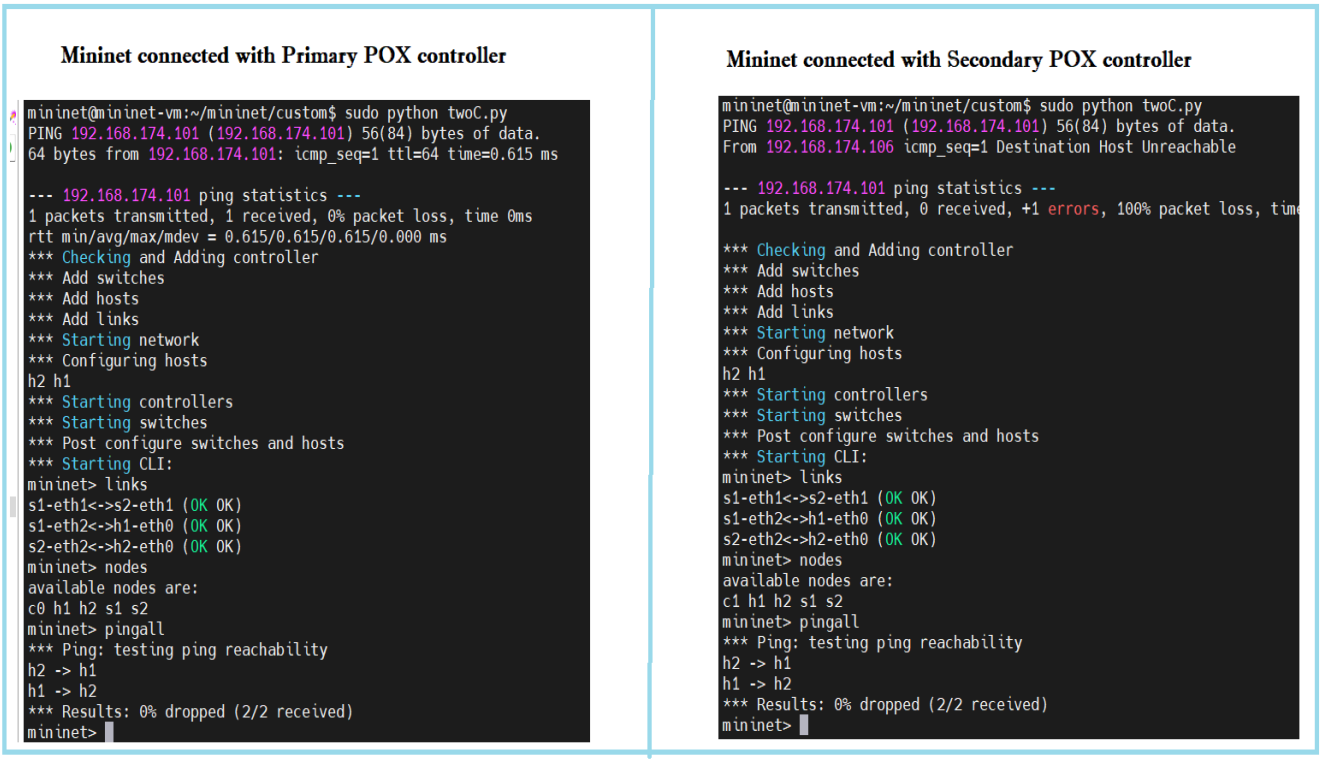


Figure 4-25: Minnet either connect to A) Primary or to B) Secondary POX controller with shown reachable ICMP

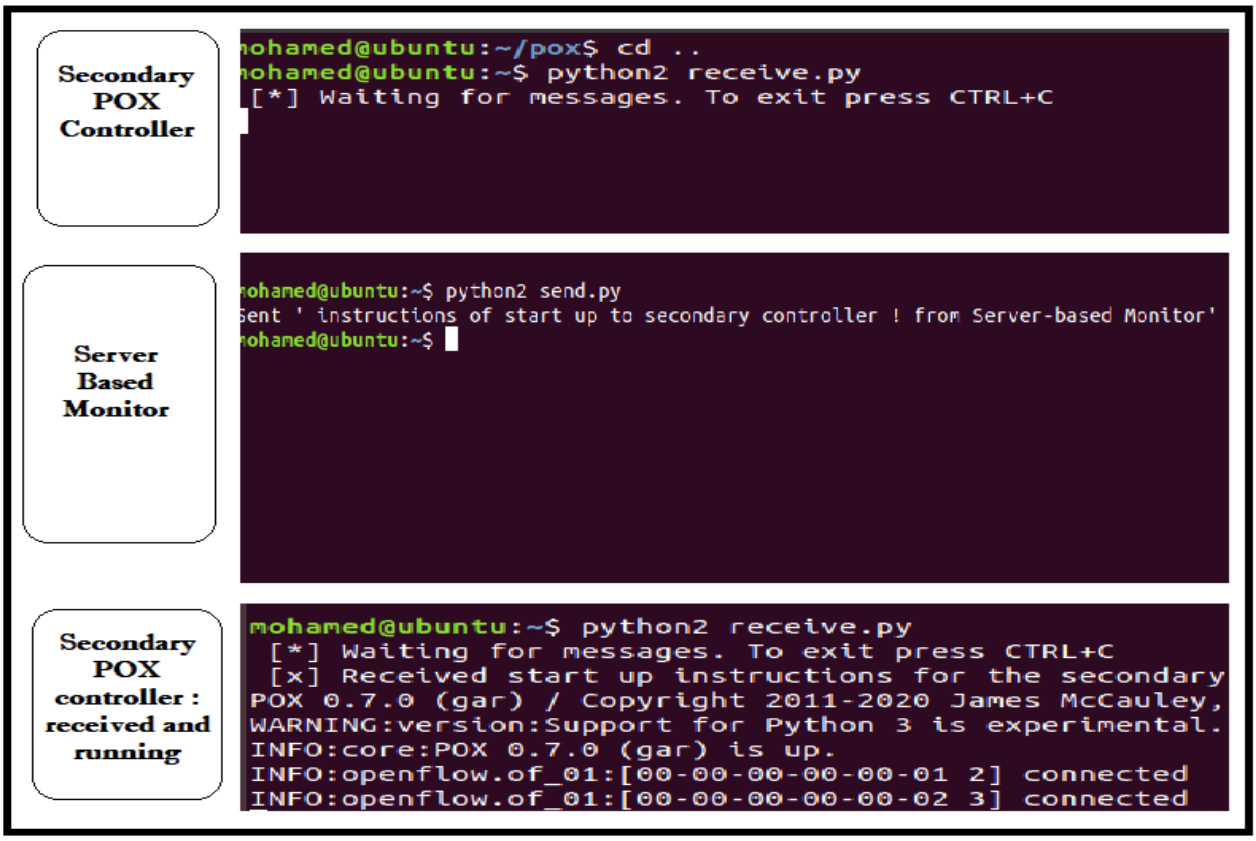


Figure 4-26: Running of proposed method between secondary controller and server

In the control plane, employed server to check POX controllers and running anyone when another controller becomes down or unreachable for any reason.

Figure (4.26) shows the procedure that shows the steps of the running of the proposed method.

Figure (4.27) depicts the graph of the response time with POX calculation from three modes of the test as mentioned in the methodology section. The graph shows the three cases:

a) The normal work of the network from any defect it is going through, which is coloured blue under the name (Before Time),
b) the work of the network during a malfunction, which is the stop of the controller, as shown in the graph in orange color under the name (Issue: Time) that the network has become unreachable,
c) and the third case is the occurrence of the same defect as above, but with implementation, our proposed, show it in the gray outline under the name (after time).

13.

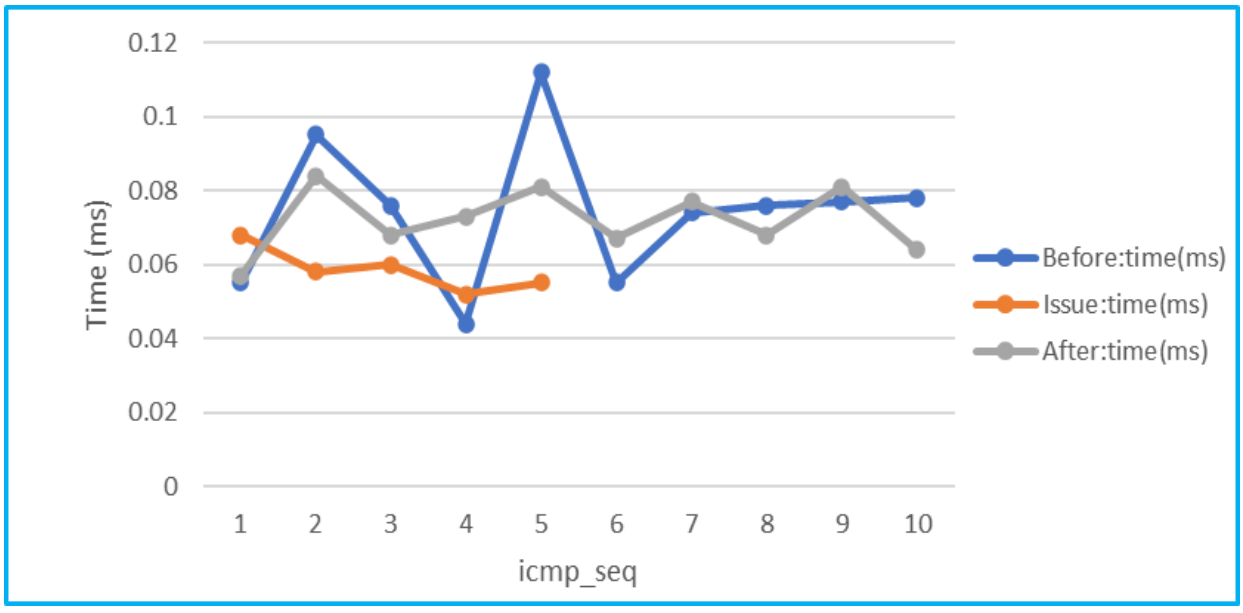


Figure 4-27: Throughput of proposed topology over 3 experiments

The third case proved that the proposed solution can address any obstacle that occurs to the controller and replace it with another controller to ensure that the network is operating normally.

4.7.2 The summary of proposed FSCA

A distributed SDN controller exists to replace the early installation of a single SDN controller. One of the primary goals of this work is to enable the construction of a suitable connection in an SDN environment using software implemented by multiple SDN controllers. The controller has become the one that manages the SDN devices and is considered the brain of the network and is a central software, and this possibility puts the network at risk of a single point of failure. Additionally, some controllers do not support a distribution strategy. This encouraged the construction of the proposal in this study. After several experiments, especially the three modes, it was clear in previous the results section, that the proposed model FSCA is an effective flat distributed SDN controller model, it has been validated in many experiments, which showed its ability to avoid a single point of failure in a multi-control plane environment where it was run with POX Controller.

4.8 The Performance of the POX Controller Under a DDoS Attack

To demonstrate the results of the testing phase, it will be clarified as SDN environment and the remaining components where it is a form of flood attack, such as the scapy tool. Here, multiple packets are sent to the network device to stop the service or reduce its performance. The CPU and RAM consumption level, as well as the delay rate of packets passing through the attack link, are then calculated. Subsections (4.8.1, 4.8.2 and 4.8.3) will presents the tests, metrics, and results with discussion.

4.8.1 Testbed

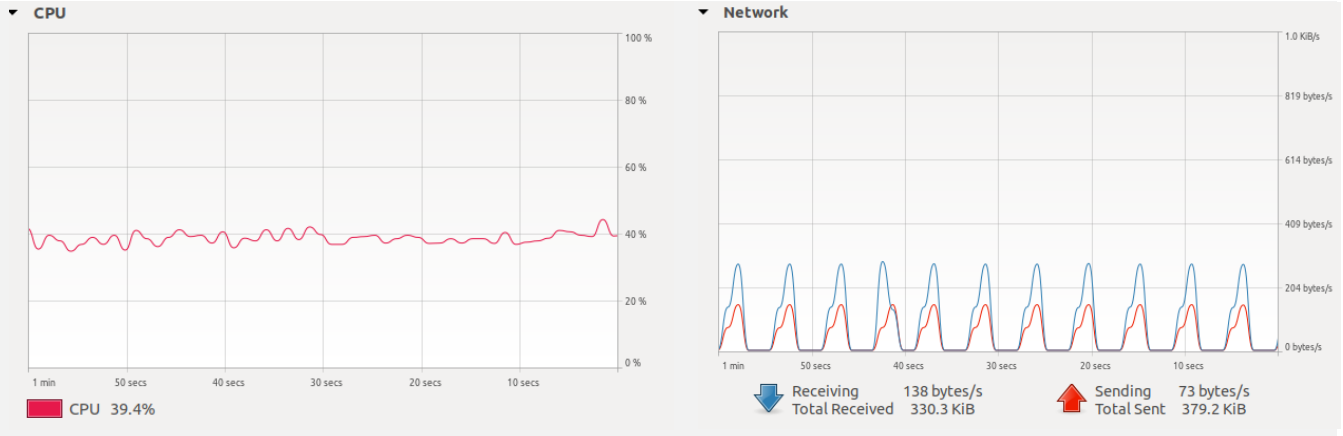
The tests in an emulated environment, it is used a laptop with the next specifications: Windows 11, Core i5 11th, 16 GB RAM, two virtual machines for

Ubuntu-16.04.2-server-amd64, and Mininet 2.2.2. The testbed topology is depicted in Figure (4.30). The POX controller was located in a virtual machine with 1 core and 4 GB RAM delegated. The topology defined in Mininet is 3 switches, and 2 hosts. The Mininet was located in a virtual machine with 1 core and 1 GB RAM delegated.

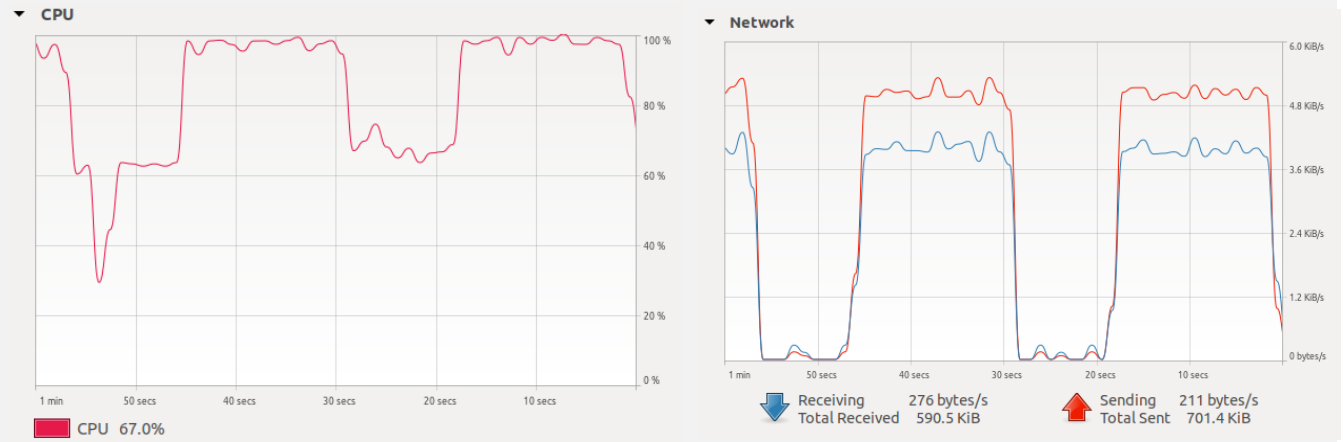
In each attacking host, set-up permitted to reach a maximum rate on average equal to 100 packets/sec, that is 4.2 kilobytes/sec per attacking host. In spite of the scapy library being slow for creating and sending spoofed packets, it is enough for seeing the DDoS attack consequences in this testbed.

The performance metrics were measured based on the number of devices deploying the DoS attack. The measures were taken over 5 minutes from one host deploying the attack with a spoofed IP address. The DDoS attack always started at first second of the simulation time. The average rate of packets arriving at the controller to be processed in each attack scenario is described in Table (4.22).

Before Attack



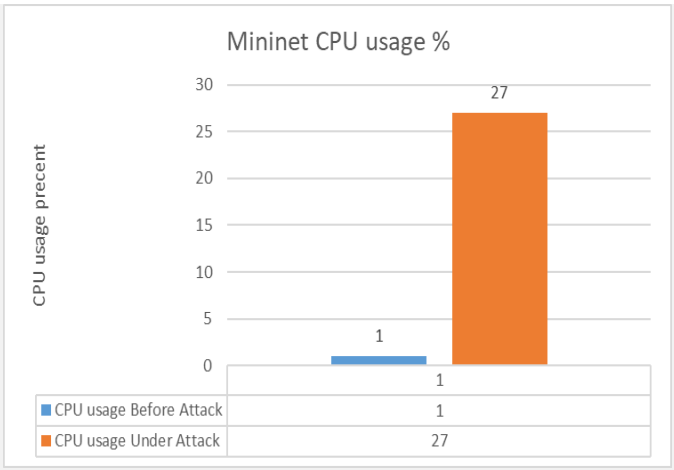
Under attacks



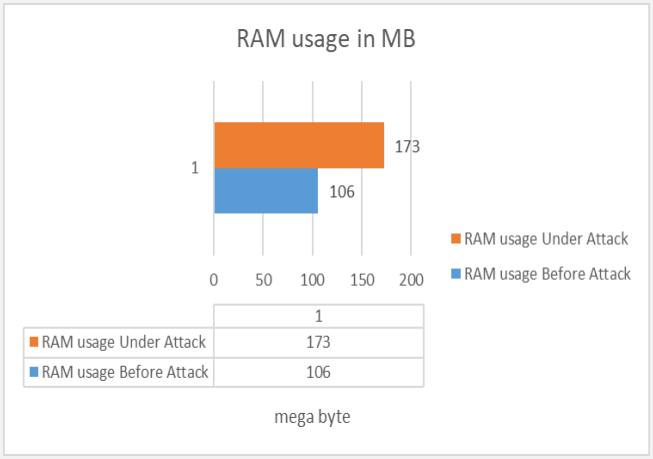
(A) POX CPU usage (B) POX network Utilize

Figure 4-28: Before and after attack on control plane of POX controller

Data Plane



Compare Mininet CPU Usage



Compare Mininet RAM Usage

Figure 4-29: CPU and RAM of Mininet before and under attack

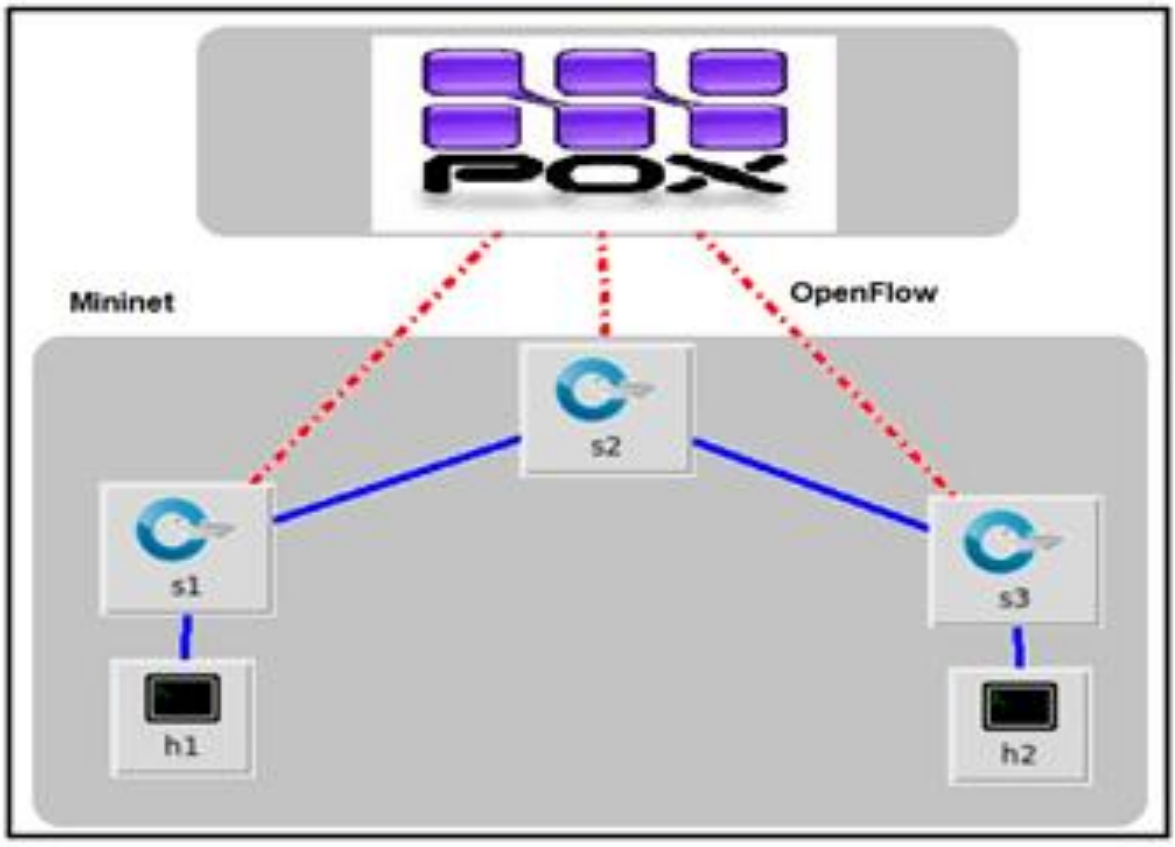


Figure 4-30: Evaluation Topology

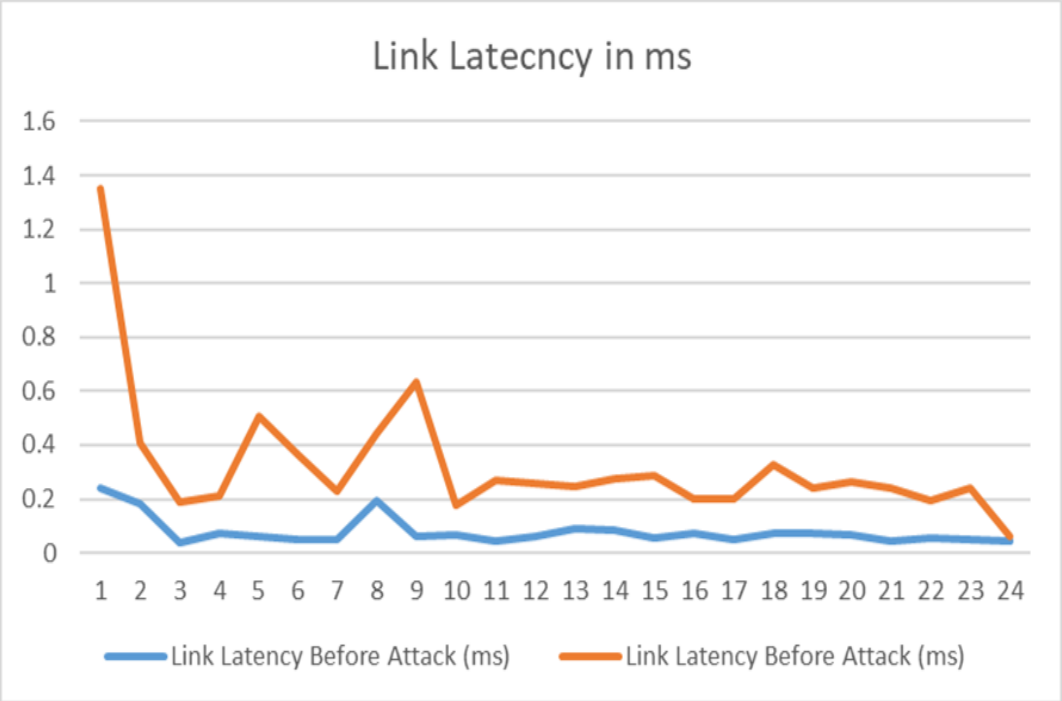


Figure 4-31: The link latency in ms at data plane (Mininet)

Table 4-22: Average link latency between h1 and h2

Before Attack	Under Attack
0.0661 ms	0.3354 ms

4.8.2 The Metrics

To show the effect of the attack on a network, both the CPU usage and the data transfer rate on the control plane were measured before and after the attack.

On the other hand, both the RAM and CPU usage were calculated and link latency between data plane hosts, also before and under DDoS attacks.

The metrics chosen to measure the attack performance are Mininet CPU consumption, the link latency, and RAM usage in data plane. Additionally, POX controller CPU consumption, and network bandwidth usage in the control plane. To measure the CPU consumption was used the Linux monitor tool were each 2 seconds the consumed CPU percentage at that moment, is taken and stored.

The link latency is the time spent by one packet to pass a link; this paper measured the link latency between host 1 and host 2 in two situations (before and under attack (see Figure (4.31))). The strategy developed via a monitor of Mininet CLI to measure the link latency is depicted in Figure (4.32).

Each 2 seconds a UDP packet with an arbitrary Eth Type value is sent from the controller to pass through the link 2. when the package finishes passing the link, it is returned to the controller due to the flow rule instanced before at the switch. Knowing the total time spent by the UDP packet turning around ping command between host 1 and host 2, the link latency is calculated.

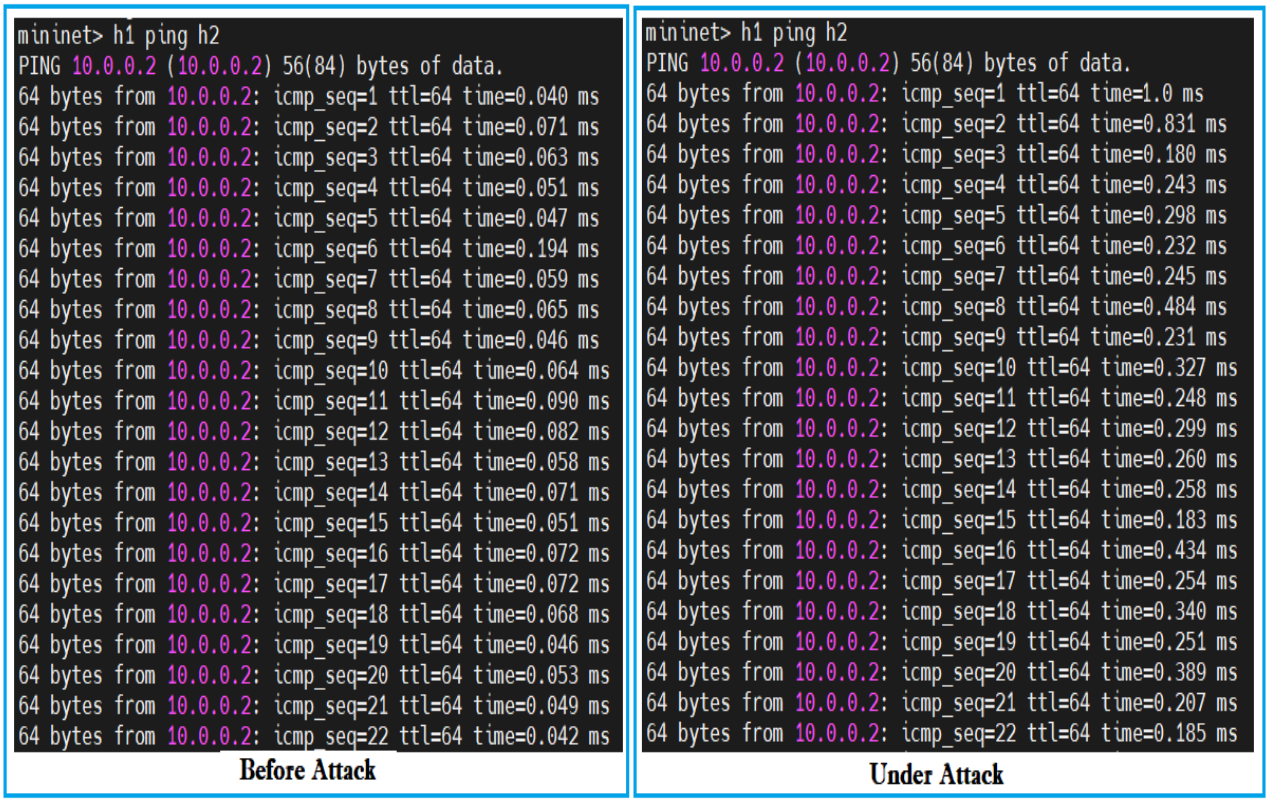


Figure 4-32: The link latency calculation by PING command between host1 and host 2

4.8.3 The Results Discussion of DDoS Effect

Figures (4.28, 4.29, and 4.30) depict the performance metrics under the DDoS attack depending of the number of attacking hosts. When the attack is running, we can clearly see an increase in the average values of the metrics when the number of attacking hosts in the DDoS attack increase.

On the other hand, as the link latency and flow rule instance latency were measured by monitoring on the Mininet terminal, we have noticed that the application is not recording data after 20 second of simulation time because at any moment the application falls due to a jam in the controller (see blue and

orange vertical line in right side of Figure (4.28)). This was a sign to know the moment that the controller begins to fail. Knowing that with 4 attacking hosts we got the lowest time around 5 minutes.

Analyzing the CPU consumption in the three cases (left-hand side of Figures (4.28) and (4.29)), at the moment that the number of attacking hosts is increased, the consumption fluctuation decreases and it tends to achieve the constant maximum value, saturating the controller. The case with attacking, the controller has reached more than 98% of CPU consumption, being our best case. On the other hand, the time spent by one UDP packet to pass-through through the link between host1 and host2 is increased significantly under the attack (see Figures (4.31) and (4.32)), which means a huge delay in the packet's circulation on the network, especially when the attack is deployed.

Chapter Five

The Conclusions and Suggested Future Works

5.1 The Conclusions

Despite the fact that SDN is an attractive technology that will likely have a bright future. It is attractive for attackers to target, the SDN controller to flood and make it down, making the network unusable. Unlike traditional networks, the SDN is more exposed to attacks due to centralization of control and the rest of conclusion can be listed as fellows.

1- The new proposal of mixed model of entropy and ML leads to a compromised solution fast as entropy and accurate as ML.
2- The new proposal consumes very low rate of the controller resources because it makes benefit from the already exist SDN flow statistics to make its decision which makes it lightweight procedure.
3- The pruning of decision tree is trade-off between response time which depends on tree size and the accuracy. So, it is network administrator responsibility to choose the suitable values.
4- The reduction of training dataset features from 84 to 14 leads to avoiding a high dimensionality and keeping the proposed system at high performance level when compared to the recent related works.
5- The proposed system is accurate and efficient, as evidenced by its usefulness when applied to real-time data on the network, with a detection accuracy of up to 98%.
6- The conventional networks resources utilization is at level normally, so the use of defending software never forms and extra burden network performance while in SDN environment the network resources are extensively used, so the margin left for defending software is very narrow which requires the defending software to be very lightweight.

5.2 The Suggested Future Works

We recommend that, as part of future development:

1. Add more upgrades to the modified controller which is able to detect the most types of SDN DDoS attacks to enable it to detect another type of attacks like (Port Scan, …).
2. Using training datasets especially made for SDN environment to involve more SDN relevant features.
3. Applying the proposed FSCA system on other SDN controllers (e.g., RYU, …).
4. Use dynamic window size instead of fixed windows size (50).
5. Use dynamic threshold instead of predefined threshold (0.5).
6. Develop model to detection of low-rate DDoS attack in SDN environment.
7. Block port of open vSwitch (in Mininet) when DDoS attack detection.
8. The decision tree pruning process needs more investigation to find a mathematical model to control the level of pruning instead of using empirical threshold.

References

[1] H. Kim and N. Feamster, “Improving network management with software defined networking,” *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 114–119, 2013.

[2] K. Bhushan and B. B. Gupta, “Distributed denial of service (DDoS) attack mitigation in software defined network (SDN)-based cloud computing environment,” *J. Ambient Intell. Humaniz. Comput.*, vol. 10, no. 5, pp. 1985–1997, 2019.

[3] H. Polat, O. Polat, and A. Cetin, “Detecting DDoS attacks in software-defined networks through feature selection methods and machine learning models,” *Sustain.*, vol. 12, no. 3, Feb. 2020, doi: 10.3390/su12031035.

[4] Y. Jarraya, T. Madi, and M. Debbabi, “A survey and a layered taxonomy of software-defined networking,” *IEEE Commun. Surv. tutorials*, vol. 16, no. 4, pp. 1955–1980, 2014.

[5] A. S. Jose, L. R. Nair, and V. Paul, “Towards Detecting Flooding DDOS Attacks Over Software Defined Networks Using Machine Learning Techniques,” *Rev. GEINTEC-GESTAO Inov. E Tecnol.*, vol. 11, no. 4, pp. 3837–3865, 2021.

[6] M. I. Kareem and M. N. Jasim, “The Current Trends of DDoS Detection in SDN Environment,” in *2021 2nd Information Technology To Enhance e-learning and Other Application (IT-ELA)*, 2021, pp. 29–34.

[7] R. T. Kokila, S. T. Selvi, and K. Govindarajan, “DDoS detection and analysis in SDN-based environment using support vector machine classifier,” in *2014 Sixth International Conference on Advanced Computing (ICoAC)*, 2014, pp. 205–210.

[8] Y. Park, N. V. Kengalahalli, and S.-Y. Chang, “Distributed security network functions against botnet attacks in software-defined networks,” in *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2018, pp. 1–7.

[9] S. M. Mousavi and M. St-Hilaire, “Early detection of DDoS attacks against SDN controllers,” in *2015 International Conference on Computing, Networking and Communications (ICNC)*, 2015, pp. 77–81.

[10] R. Wang, Z. Jia, and L. Ju, “An entropy-based distributed DDoS detection mechanism in software-defined networking,” in *2015 IEEE Trustcom/BigDataSE/ISPA*, 2015, vol. 1, pp. 310–317.

[11] A. S. da Silva, J. A. Wickboldt, L. Z. Granville, and A. Schaeffer-Filho, “ATLANTIC: A framework for anomaly traffic detection, classification,

and mitigation in SDN," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, 2016, pp. 27–35.

[12] J. Boite, P.-A. Nardin, F. Rebecchi, M. Bouet, and V. Conan, "Statesec: Stateful monitoring for DDoS protection in software defined networks," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, 2017, pp. 1–9.

[13] R. Sanjeetha, S. Srivastava, R. Pokharna, S. Shafiq, and A. Kanavalli, "Mitigation of DDoS attack instigated by compromised switches on SDN controller by analyzing the flow rule request traffic," *Int. J. Eng. Technol.(UAE)*, vol. 7, pp. 46–49, 2018.

[14] A. Koay, A. Chen, I. Welch, and W. K. G. Seah, "A new multi classifier system using entropy-based features in DDoS attack detection," in *2018 International conference on information networking (ICOIN)*, 2018, pp. 162–167.

[15] K. Kalkan, L. Altay, G. Gür, and F. Alagöz, "JESS: Joint entropy-based DDoS defense scheme in SDN," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2358–2372, 2018.

[16] G.-C. Hong, C.-N. Lee, and M.-F. Lee, "Dynamic threshold for DDoS mitigation in SDN environment," in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2019, pp. 1–7.

[17] R. Li and B. Wu, "Early detection of DDoS based on $\varphi $-entropy in SDN networks," in *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 2020, vol. 1, pp. 731–735.

[18] Y. Shen, C. Wu, D. Kong, and M. Yang, "Tpdd: A two-phase ddos detection system in software-defined networking," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[19] Z. Abou El Houda, L. Khoukhi, and A. S. Hafid, "Bringing intelligence to software defined networks: Mitigating DDoS attacks," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 4, pp. 2523–2535, 2020.

[20] N. M. AbdelAzim, 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," *Egypt. Informatics J.*, vol. 22, no. 1, pp. 85–90, 2021.

[21] A. Mishra, N. Gupta, and B. B. Gupta, "Defense mechanisms against DDoS attack based on entropy in SDN-cloud using POX controller," *Telecommun. Syst.*, vol. 77, no. 1, pp. 47–62, 2021.

[22] R. M. A. Ujjan, Z. Pervez, K. Dahal, W. A. Khan, A. M. Khattak, and B. Hayat, "Entropy based features distribution for anti-ddos model in sdn," *Sustainability*, vol. 13, no. 3, p. 1522, 2021.

[23] A. Anil, T. A. Rufzal, and V. Adat Vasudevan, "DDoS Detection in Software-Defined Network Using Entropy Method," in *Proceedings of the Seventh International Conference on Mathematics and Computing*, 2022, pp. 129–139.

[24] L. V. Filter and P. C. A. Filter, "Seven Techniques for Dimensionality Reduction," 2014.

[25] G. Lucky, F. Jjunju, and A. Marshall, "A lightweight decision-tree algorithm for detecting DDoS flooding attacks," in *2020 IEEE 20th international conference on software quality, reliability and security companion (QRS-C)*, 2020, pp. 382–389.

[26] P. Maniriho, L. J. Mahoro, E. Niyigaba, Z. Bizimana, and T. Ahmad, "Detecting Intrusions in Computer Network Traffic with Machine Learning Approaches," *Int. J. Intell. Eng. Syst.*, vol. 13, no. 3, 2020, doi: 10.22266/ijies2020.0630.39.

[27] S. Peneti and E. Hemalatha, "DDOS Attack Identification using Machine Learning Techniques," in *2021 International Conference on Computer Communication and Informatics (ICCCI)*, 2021, pp. 1–5.

[28] M. H. Zaib, F. Bashir, K. N. Qureshi, S. Kausar, M. Rizwan, and G. Jeon, "Deep learning based cyber bullying early detection using distributed denial of service flow," *Multimed. Syst.*, pp. 1–20, 2021.

[29] M. Ahsan, R. Gomes, M. M. Chowdhury, and K. E. Nygard, "Enhancing Machine Learning Prediction in Cybersecurity Using Dynamic Feature Selector," *J. Cybersecurity Priv.*, vol. 1, no. 1, pp. 199–218, 2021.

[30] D. Kshirsagar and S. Kumar, "A feature reduction based reflected and exploited DDoS attacks detection system," *J. Ambient Intell. Humaniz. Comput.*, vol. 13, no. 1, pp. 393–405, 2022.

[31] S. U. N. Guozi, W. Jiang, G. U. Yu, R. E. N. Danni, and L. I. Huakang, "DDoS attacks and flash event detection based on flow characteristics in SDN," in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2018, pp. 1–6.

[32] J. Li, Z. Zhao, R. Li, and H. Zhang, "Ai-based two-stage intrusion detection for software defined iot networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2093–2102, 2018.

[33] V. Deepa, K. M. Sudar, and P. Deepalakshmi, "Design of ensemble

learning methods for DDoS detection in SDN environment," in *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, 2019, pp. 1–6.

[34] M. Myint Oo, S. Kamolphiwong, T. Kamolphiwong, and S. Vasupongayya, "Advanced support vector machine-(ASVM-) based detection for distributed denial of service (DDoS) attack on software defined networking (SDN)," *J. Comput. Networks Commun.*, vol. 2019, 2019.

[35] T. V Phan and M. Park, "Efficient distributed denial-of-service attack defense in SDN-based cloud," *IEEE Access*, vol. 7, pp. 18701–18714, 2019.

[36] M. P. Novaes, L. F. Carvalho, J. Lloret, and M. L. Proenca, "Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment," *IEEE Access*, vol. 8, pp. 83765–83781, 2020.

[37] W. G. Gadallah, N. M. Omar, and H. M. Ibrahim, "Machine Learning-based Distributed Denial of Service Attacks Detection Technique using New Features in Software-defined Networks," *Int. J. Comput. Netw. Inf. Secur.*, vol. 13, no. 3, pp. 15–27, 2021.

[38] L. Zhou, Y. Zhu, Y. Xiang, and T. Zong, "A novel feature-based framework enabling multi-type DDoS attacks detection," *World Wide Web*, pp. 1–23, 2022.

[39] S. Lysenko, O. Savenko, and K. Bobrovnikova, "DDoS Botnet Detection Technique Based on the Use of the Semi-Supervised Fuzzy c-Means Clustering.," in *ICTERI Workshops*, 2018, pp. 688–695.

[40] M. Idhammad, K. Afdel, and M. Belouch, "Semi-supervised machine learning approach for DDoS detection," *Appl. Intell.*, vol. 48, no. 10, pp. 3193–3208, 2018.

[41] O. Y. Al-Jarrah, Y. Al-Hammdi, P. D. Yoo, S. Muhaidat, and M. Al-Qutayri, "Semi-supervised multi-layered clustering model for intrusion detection," *Digit. Commun. Networks*, vol. 4, no. 4, pp. 277–286, 2018.

[42] Y. Gu, K. Li, Z. Guo, and Y. Wang, "Semi-supervised K-means DDoS detection method using hybrid feature selection algorithm," *IEEE Access*, vol. 7, pp. 64351–64365, 2019.

[43] E. K. Viegas, A. O. Santin, V. V Cogo, and V. Abreu, "A reliable semi-supervised intrusion detection model: One year of network traffic anomalies," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[44] S. Y. Khamaiseh, A. Al-Alaj, and A. Warner, “FloodDetector: Detecting Unknown DoS Flooding Attacks in SDN,” in *2020 International Conference on Internet of Things and Intelligent Applications (ITIA)*, 2020, pp. 1–5.

[45] M. Aamir and S. M. Ali Zaidi, “Clustering based semi-supervised machine learning for DDoS attack classification,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 33, no. 4, pp. 436–446, May 2021, doi: 10.1016/J.JKSUCI.2019.02.003.

[46] S. R. Akula, “Semi supervised machine learning approach for DDOS detection,” *Int. J. Innov. Res. Educ.*, vol. 8, no. 1, pp. 27–35, 2021.

[47] M. Najafimehr, S. Zarifzadeh, and S. Mostafavi, “A hybrid machine learning approach for detecting unprecedented DDoS attacks,” *J. Supercomput.*, pp. 1–31, 2022.

[48] O. Aouedi, K. Piamrat, G. Muller, and K. Singh, “Intrusion detection for Softwarized Networks with Semi-supervised Federated Learning,” in *ICC 2022-IEEE International Conference on Communications*, 2022, pp. 1–6.

[49] A. Tootoonchian and Y. Ganjali, “Hyperflow: A distributed control plane for openflow,” in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, 2010, vol. 3.

[50] J. Stribling *et al.*, “USENIX Association NSDI ’09: 6th USENIX Symposium on Networked Systems Design and Implementation 43 Flexible, Wide-Area Storage for Distributed Systems with WheelFS.”

[51] T. Koponen *et al.*, “Onix: A distributed control platform for large-scale production networks,” 2010.

[52] “Apache ZooKeeper.” https://zookeeper.apache.org/ (accessed May 14, 2022).

[53] J. Medved, R. Varga, A. Tkacik, and K. Gray, “Opendaylight: Towards a model-driven sdn controller architecture,” in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, 2014, pp. 1–6.

[54] “MD-SAL Release 9.0.0-SNAPSHOT OpenDaylight Project,” 2022.

[55] “GitHub - ktoso/akka-raft: A toy project implementing RAFT on top of Akka Cluster (not prod ready).” https://github.com/ktoso/akka-raft (accessed May 14, 2022).

[56] P. Berde *et al.*, “ONOS: towards an open, distributed SDN OS,” in *Proceedings of the third workshop on Hot topics in software defined networking*, 2014, pp. 1–6.

[57] F. Benamrane and R. Benaini, “An East-West interface for distributed SDN control plane: Implementation and evaluation,” *Comput. Electr. Eng.*, vol. 57, pp. 162–175, 2017.

[58] B. Lee, S. H. Park, J. Shin, and S. Yang, “IRIS: the Openflow-based recursive SDN controller,” in *16th International Conference on Advanced Communication Technology*, 2014, pp. 1227–1231.

[59] S. Hassas Yeganeh and Y. Ganjali, “Kandoo: a framework for efficient and scalable offloading of control applications,” in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 19–24.

[60] “DDS-RTPS.” https://www.arc-it.net/html/standards/standard1306.html (accessed May 14, 2022).

[61] K. Kirkpatrick, “Software-defined networking,” *Commun. ACM*, vol. 56, no. 9, pp. 16–19, 2013.

[62] U. Rahamathullaha and E. Karthikeyanb, “Distributed denial of service attacks prevention, detection and mitigation–A review,” 2021.

[63] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, “Security in software defined networks: A survey,” *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2317–2346, 2015.

[64] Y. Xu and Y. Liu, “DDoS attack detection under SDN context,” in *IEEE INFOCOM 2016-the 35th annual IEEE international conference on computer communications*, 2016, pp. 1–9.

[65] J. R. Dennis and X. Li, “Machine-Learning and Statistical Methods for DDoS Attack Detection and Defense System in Software Defined Networks.” MS thesis College of Eng. and Sc Ryerson Univ., Toronto, 2018.

[66] J. Ashraf and S. Latif, “Handling intrusion and DDoS attacks in Software Defined Networks using machine learning techniques,” in *2014 National software engineering conference*, 2014, pp. 55–60.

[67] S. Ahmad and A. H. Mir, “Scalability, consistency, reliability and security in SDN controllers: a survey of diverse SDN controllers,” *J. Netw. Syst. Manag.*, vol. 29, no. 1, pp. 1–59, 2021.

[68] C. Prabha, A. Goel, and J. Singh, “A Survey on SDN Controller Evolution: A Brief Review,” in *2022 7th International Conference on Communication and Electronics Systems (ICCES)*, 2022, pp. 569–575.

[69] “(PDF) Software Defined Network, Controller Comparison | Bhargavi Goswami and Saleh Asadollahi - Academia.edu.” https://www.academia.edu/32668975/Software_Defined_Network_Contro

ller_Comparison (accessed Nov. 01, 2022).

[70] P. Goransson, C. Black, and T. Culver, *Software defined networks: a comprehensive approach*. Morgan Kaufmann, 2016.

[71] N. Xue, D. Guo, J. Zhang, J. Xin, Z. Li, and X. Huang, "OpenFunction for Software Defined IoT," in *2021 International Symposium on Networks, Computers and Communications (ISNCC)*, 2021, pp. 1–8.

[72] J. Benabbou, K. Elbaamrani, and N. Idboufker, "Security in OpenFlow-based SDN, opportunities and challenges," *Photonic Netw. Commun.*, vol. 37, no. 1, pp. 1–23, 2019.

[73] K. Nam and K. Kim, "A study on sdn security enhancement using open source ids/ips suricata," in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, 2018, pp. 1124–1126.

[74] T. Dargahi, A. Caponi, M. Ambrosin, G. Bianchi, and M. Conti, "A survey on the security of stateful SDN data planes," *IEEE Commun. Surv. Tutorials*, vol. 19, no. 3, pp. 1701–1725, 2017.

[75] S. Kaur, J. Singh, and N. S. Ghumman, "Network programmability using POX controller," in *ICCCS International conference on communication, computing & systems, IEEE*, 2014, vol. 138, p. 70.

[76] K. A. Rasol, "Flexible architecture for the future internet scalability of SDN control plane," 2022.

[77] E. Amiri, E. Alizadeh, and K. Raeisi, "An efficient hierarchical distributed SDN controller model," in *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*, 2019, pp. 553–557.

[78] "ONF SampleTap-An Educational Journey Developing a simple OpenFlow application on OpenDaylight," 2014.

[79] Y. E. Oktian, S. G. Lee, H. J. Lee, and J. H. Lam, "Distributed SDN controller system: A survey on design choice," *Comput. Networks*, vol. 121, pp. 100–111, Jul. 2017, doi: 10.1016/J.COMNET.2017.04.038.

[80] M. Paliwal, D. Shrimankar, and O. Tembhurne, "Controllers in SDN: A review report," *IEEE access*, vol. 6, pp. 36256–36270, 2018.

[81] M. P. Singh and A. Bhandari, "New-flow based DDoS attacks in SDN: Taxonomy, rationales, and research challenges," *Comput. Commun.*, vol. 154, pp. 509–527, 2020.

[82] N. Z. Bawany, J. A. Shamsi, and K. Salah, "DDoS attack detection and mitigation using SDN: methods, practices, and solutions," *Arab. J. Sci.*

Eng., vol. 42, no. 2, pp. 425–441, 2017.

[83] F. Khashab, J. Moubarak, A. Feghali, and C. Bassil, "DDoS attack detection and mitigation in SDN using machine learning," in *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, 2021, pp. 395–401.

[84] S. Asadollahi and B. Goswami, "Experimenting with scalability of floodlight controller in software defined networks," in *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, 2017, pp. 288–292.

[85] N. Dayal, P. Maity, S. Srivastava, and R. Khondoker, "Research trends in security and DDoS in SDN," *Secur. Commun. Networks*, vol. 9, no. 18, pp. 6386–6411, 2016.

[86] M. I. Kareem and M. N. Jasim, "Fast and accurate classifying model for denial-of-service attacks by using machine learning," *Bull. Electr. Eng. Informatics*, vol. 11, no. 3, pp. 1742–1751, 2022.

[87] K. N. Mallikarjunan, K. Muthupriya, and S. M. Shalinie, "A survey of distributed denial of service attack," in *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, 2016, pp. 1–6.

[88] O. Igbe, O. Ajayi, and T. Saadawi, "Detecting denial of service attacks using a combination of dendritic cell algorithm and the negative selection algorithm," in *2017 IEEE International Conference on Smart Cloud (SmartCloud)*, 2017, pp. 72–77.

[89] A. F. Alsirhani, "DDOS DETECTION MODELS USING MACHINE AND DEEP LEARNING ALGORITHMS AND DISTRIBUTED SYSTEMS." 2021.

[90] P. Biondi, "Packet generation and network based attacks with scapy," *CanSecWest/core05*, 2005.

[91] A. R. Yusof, N. I. Udzir, and A. Selamat, "Systematic literature review and taxonomy for DDoS attack detection and prediction," *Int. J. Digit. Enterp. Technol.*, vol. 1, no. 3, pp. 292–315, 2019.

[92] J. Singh and S. Behal, "Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions," *Comput. Sci. Rev.*, vol. 37, p. 100279, 2020.

[93] "CISA | DEFEND TODAY, SECURE TOMORROW."

[94] M. Gunasekaran and S. Periakaruppan, "GA-DoSLD: Genetic algorithm based denial-of-sleep attack detection in WSN," *Secur. Commun. Networks*, vol. 2017, 2017, doi: 10.1155/2017/9863032.

[95] A. K. Tyagi and P. Chahal, “Artificial intelligence and machine learning algorithms,” in *Research Anthology on Machine Learning Techniques, Methods, and Applications*, IGI Global, 2022, pp. 421–446.

[96] A. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” *IEEE Commun. Surv. tutorials*, vol. 18, no. 2, pp. 1153–1176, 2015.

[97] A. B. Dehkordi, M. Soltanaghaei, and F. Z. Boroujeni, “The DDoS attacks detection through machine learning and statistical methods in SDN,” *J. Supercomput.*, vol. 77, no. 3, pp. 2383–2415, 2021.

[98] O. Rahman, M. A. G. Quraishi, and C.-H. Lung, “DDoS attacks detection and mitigation in SDN using machine learning,” in *2019 IEEE World Congress on Services (SERVICES)*, 2019, vol. 2642, pp. 184–189.

[99] M. J. Awan *et al.*, “Real-time DDoS attack detection system using big data approach,” *Sustainability*, vol. 13, no. 19, p. 10743, 2021.

[100] A. Niranjan, D. H. Nutan, A. Nitish, P. D. Shenoy, and K. R. Venugopal, “ERCR TV: Ensemble of random committee and random tree for efficient anomaly classification using voting,” in *2018 3rd International Conference for Convergence in Technology (I2CT)*, 2018, pp. 1–5.

[101] R. S. Khairy, A. S. Hussein, and H. T. S. Alrikabi, “The Detection of Counterfeit Banknotes Using Ensemble Learning Techniques of AdaBoost and Voting,” *Int. J. Intell. Eng. Syst.*, vol. 14, no. 1, p. 2021, doi: 10.22266/ijies2021.0228.31.

[102] D. P. Gaikwad and R. C. Thool, “Intrusion detection system using bagging with partial decision treebase classifier,” *Procedia Comput. Sci.*, vol. 49, pp. 92–98, 2015.

[103] “Semi-supervised learning - Wikipedia.” https://en.wikipedia.org/wiki/Semi-supervised_learning (accessed Jun. 15, 2022).

[104] W. L. Al-Yaseen, Z. A. Othman, and M. Z. A. Nazri, “Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system,” *Expert Syst. Appl.*, vol. 67, pp. 296–303, 2017.

[105] Y. Qin, S. Ding, L. Wang, and Y. Wang, “Research Progress on Semi-Supervised Clustering,” *Cognit. Comput.*, vol. 11, no. 5, pp. 599–612, Oct. 2019, doi: 10.1007/S12559-019-09664-W.

[106] J. R. Quinlan, “Induction of decision trees,” *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

[107] N. Patel and S. Upadhyay, “Study of various decision tree pruning methods with their empirical comparison in WEKA,” *Int. J. Comput. Appl.*, vol. 60, no. 12, 2012.

[108] W. W. Cohen, “Fast effective rule induction,” in *Machine learning proceedings 1995*, Elsevier, 1995, pp. 115–123.

[109] I. Kononenko, “The minimum description length based decision tree pruning,” in *Pacific Rim International Conference on Artificial Intelligence*, 1998, pp. 228–237.

[110] A. El Gamal and Y.-H. Kim, “Elements of network information theory,” 2012.

[111] J. Zhang, Z. Qin, L. Ou, P. Jiang, J. Liu, and A. X. Liu, “An advanced entropy-based DDOS detection scheme,” in *2010 International Conference on Information, Networking and Automation (ICINA)*, 2010, vol. 2, pp. V2-67.

[112] P. Natesan, P. Balasubramanie, and G. Gowrison, “Improving attack detection rate in network intrusion detection using adaboost algorithm with multiple weak classifiers,” *J. Inf. &COMPUTATIONAL Sci.*, vol. 9, no. 8, pp. 2239–2251, 2012.

[113] A. Ćatović, N. Buzađija, and S. Lemes, “Microservice development using RabbitMQ message broker,” *Sci. Eng. Technol.*, vol. 2, no. 1, pp. 30–37, 2022.

[114] R. L. S. De Oliveira, C. M. Schweitzer, A. A. Shinoda, and L. R. Prete, “Using mininet for emulation and prototyping software-defined networks,” in *2014 IEEE Colombian conference on communications and computing (COLCOM)*, 2014, pp. 1–6.

[115] A. Bagewadi and R. M. Babu, “Towards an Ethernet Learning Switch and Bandwidth Optimization using POX Controller,” *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 3, no. 7, pp. 7531–7535, 2014.

[116] A. L. Stancu, S. Halunga, A. Vulpe, G. Suciu, O. Fratu, and E. C. Popovici, “A comparison between several Software Defined Networking controllers,” in *2015 12th international conference on telecommunication in modern satellite, cable and broadcasting services (TELSIKS)*, 2015, pp. 223–226.

[117] S. Badotra and J. Singh, “Open Daylight as a Controller for Software Defined Networking.,” *Int. J. Adv. Res. Comput. Sci.*, vol. 8, no. 5, 2017.

[118] M. Großmann and S. J. A. Schuberth, “Auto-Mininet: Assessing the Internet topology zoo in a software-defined network emulator,” *Messung,*

Mellierung un Bewertung von Rechensystemen, vol. 7, pp. 1–10, 2013.

[119] M. Hibler *et al.*, "Large-scale virtualization in the emulab network testbed," 2008.

[120] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM Demonstr.*, vol. 14, no. 14, p. 527, 2008.

[121] S.-Y. Wang, C.-L. Chou, and C.-M. Yang, "EstiNet openflow network simulator and emulator," *IEEE Commun. Mag.*, vol. 51, no. 9, pp. 110–117, 2013.

[122] R. S. Selvakumari, R. Sinthuja, and G. Subasree, "Instantaneous Electronics Information Board."

[123] "Wireshark · Go Deep." https://www.wireshark.org/ (accessed Oct. 02, 2022).

[124] "IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB." https://www.unb.ca/cic/datasets/ids-2017.html (accessed Sep. 06, 2022).

[125] T. Mourouzis and A. Avgousti, "Intrusion Detection with Machine Learning Using Open-Sourced Datasets," *arXiv Prepr. arXiv2107.12621*, 2021.

[126] D. Stiawan, M. Y. Bin Idris, A. M. Bamhdi, and R. Budiarto, "CICIDS-2017 dataset feature analysis with information gain for anomaly detection," *IEEE Access*, vol. 8, pp. 132911–132921, 2020.

[127] E. M. Zeleke, H. M. Melaku, and F. G. Mengistu, "Efficient Intrusion Detection System for SDN Orchestrated Internet of Things," *J. Comput. Networks Commun.*, vol. 2021, 2021.

[128] K. Kurniabudi, D. Stiawan, D. Darmawijoyo, M. Y. Bin Idris, B. Kerim, and R. Budiarto, "Important Features of CICIDS-2017 Dataset For Anomaly Detection in High Dimension and Imbalanced Class Dataset," *Indones. J. Electr. Eng. Informatics*, vol. 9, no. 2, pp. 498–511, 2021.

[129] A. Abbas, M. A. Khan, S. Latif, M. Ajaz, A. A. Shah, and J. Ahmad, "A New Ensemble-Based Intrusion Detection System for Internet of Things," *Arab. J. Sci. Eng.*, pp. 1–15, 2021.

[130] S. Sarraf, "Analysis and detection of ddos attacks using machine learning techniques," *Am. Sci. Res. J. Eng. Technol. Sci.*, vol. 66, no. 1, pp. 95–104, 2020.

[131] R. B. Adhao and V. K. Pachghare, "Performance-Based Feature Selection

Using Decision Tree," in *2019 International Conference on Innovative Trends and Advances in Engineering and Technology (ICITAET)*, 2019, pp. 135–138.

[132] Y. M. Swe, P. P. Aung, and A. S. Hlaing, "A Slow DDoS Attack Detection Mechanism using Feature Weighing and Ranking."

[133] A. Yulianto, P. Sukarno, and N. A. Suwastika, "Improving adaboost-based intrusion detection system (IDS) performance on CIC IDS 2017 dataset," in *Journal of Physics: Conference Series*, 2019, vol. 1192, no. 1, p. 12018.

[134] S. D. Çakmakçı, T. Kemmerich, T. Ahmed, and N. Baykal, "Online DDoS attack detection using Mahalanobis distance and Kernel-based learning algorithm," *J. Netw. Comput. Appl.*, vol. 168, p. 102756, 2020.

[135] "DDoS attacks in Q4 2019 | Securelist." https://securelist.com/ddos-report-q4-2019/96154/ (accessed Sep. 20, 2022).

[136] "DDoS attacks in Q4 2020 | Securelist." https://securelist.com/ddos-attacks-in-q4-2020/100650/ (accessed Sep. 20, 2022).

[137] "Kaspersky Q4 2021 DDoS attack report | Securelist." https://securelist.com/ddos-attacks-in-q4-2021/105784/ (accessed Sep. 20, 2022).

[138] "Kaspersky DDoS report, Q1 2022 | Securelist." https://securelist.com/ddos-attacks-in-q1-2022/106358/ (accessed Sep. 20, 2022).

[139] C. Fan, N. M. Kaliyamurthy, S. Chen, H. Jiang, Y. Zhou, and C. Campbell, "Detection of DDoS attacks in software defined networking using entropy," *Appl. Sci.*, vol. 12, no. 1, p. 370, 2021.

[140] A. Maheshwari, B. Mehraj, M. S. Khan, and M. S. Idrisi, "An optimized weighted voting based ensemble model for DDoS attack detection and mitigation in SDN environment," *Microprocess. Microsyst.*, vol. 89, p. 104412, 2022.

[141] Y. Liu, T. Zhi, M. Shen, L. Wang, Y. Li, and M. Wan, "Software-defined DDoS detection with information entropy analysis and optimized deep learning," *Futur. Gener. Comput. Syst.*, vol. 129, pp. 99–114, 2022.

[142] S. H. Darekar, M. Z. Shaikh, and H. B. Kondke, "Performance Evaluation of Various Open Flow SDN Controllers by Addressing Scalability Metric Based on Multifarious Topology Design on Software-defined Networks: A Comprehensive Survey," in *Proceedings of Third International Conference on Intelligent Computing, Information and Control Systems*, 2022, pp. 327–338.

الخلاصة

تمثل الشبكات المعرفة بالبرمجيات (SDN) واحدة من أكثر التقنيات الممتازة في تسهيل إدارة الشبكة وتكوينها عن طريق زيادة قابلية برمجة الشبكة. تتميز SDN بإدارتها وتحكمها المركزيين ، لذا فهي تصبح أكثر عرضة لهجمات رفض الخدمة الموزعة (DDoS) للتأثير على الشبكة بأكملها. تعد هجمات DDoS أحد التهديدات الخطيرة للشبكات ، حيث يسهل تشغيلها ويصعب اكتشافها. يستغل الهجوم وحدة تحكم SDN ليتم إغراقها بالحزم الواردة من اجهزة switches . هناك العديد من الأساليب لاكتشاف آثار هجمات DDoS والتخفيف من حدتها.

في هذه الاطروحة، تم اقتراح نظام يتكون من تقنيات الإنتروبيا والتعلم الآلي (ML) للتغلب على نقاط الضعف في كليهما وتعزيز شبكة SDN ضد هجمات الـ DDoS.

يتكون النظام المقترح من مستويين من اكتشاف الهجوم ، الأول هو اكتشاف الهجوم بشكل سريع باستخدام تقنية الانتروبيا والثاني هو الفحص العميق للحزم باستخدام تقنية التعلم الآلي ML المقترحة. المستوى الثاني هو حصيلة تدريب النموذج باستخدام ستة خوارزميات ML ، وهي كلا من (random forest, REP tree, decision stump, random tree, decision tree ، partial decision tree) بالإضافة إلى نموذج مقترح شبه خاضع للإشراف ثم اختيار افضل مصنف للتطوير وهو PART. تم تدريب واختبار جميع نماذج ML المقترحة باستخدام مجموعة البيانات المعروفة "CICIDS2017" والتي تشتهر بمجموعة البيانات المعيارية الحالية.

من اجل اداء افضل للخوارزميات المستخدمة ، تم اختيار الخواص المهمة الموجودة في مجموعة البيانات المعتمدة.

حيث تم تحقيق نتائج بدقة عالية وصلت الى نسبة 99.92٪ و 99.77٪ باستخدام 6 و 11 خاصية على التوالي وذلك باعتماد خوارزمية PART . تم التحقق من صحة النموذج المقترح أيضًا باستخدان مجموعة البيانات الشهيرة "CICDDoS2019" والتي تظهر دقة 99.99٪ و 99.73٪ لهجمات UDP و SYN على التوالي. يتم إجراء عمليات المحاكاة في محاكي Mininet مع جهاز التحكم POX و Data Plane Switches. حقق النظام المقترح في بيئة الـ SDN معدل اكتشاف مرتفعًا بنسبة 98.06٪. بالإضافة إلى ذلك ، تم استخدام إستراتيجية فعالة مقترحة للاتصال بين اكثر من POX Controller للتغلب على لضمان ادامة عمل الشبكة في حال حصل فشل في وحدة تحكم SDN.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية تكنولوجيا المعلومات
قسم شبكات المعلومات

نموذج مطور لاكتشاف ومنع هجوم منع الخدمة الموزعة في بيئة الشبكات المعرفة برمجياً بالاعتماد على تقنيات الإنتروبيا والتعلم الآلي

أطروحة

مقدمة الى مجلس كلية تكنولوجيا المعلومات للدراسات العليا بجامعة بابل
في استيفاء جزئي لمتطلبات درجة دكتوراه فلسفة في تكنولوجيا المعلومات
/ شبكات المعلومات

من قبل الطالب
محمد ابراهيم كريم خليل

باشراف
ا.م.د مهدي نصيف جاسم علوان

2022 م 1444 هـ