

Final Defense for the degree of Masters in Computer Engineering

# **SDN Controller Placement and Optimization by Classification of Heavy Traffic and Ddos Attack Using SVM and Path Optimization Using DFS**



**Sahaj Shakya**  
**(2019-1-39-0020)**

**Nepal College of Information Technology**  
**Faculty of Science and Technology**  
**Pokhara University, Nepal**

**August, 2022**

Final Defense for the degree of Masters in Computer Engineering

# **Classification of Heavy Traffic and Ddos Attack Using SVM and Path Optimization Using DFS**

**Supervised by Prof. Roshan Chitrakar, Ph. D**

A thesis submitted in partial fulfillment of the requirement for the  
Degree of Master in Computer Engineering

**Sahaj Shakya  
(2019-1-39-0020)**

**Nepal College of Information Technology  
Faculty of Science and Technology  
Pokhara University, Nepal**

**August, 2022**

## ACKNOWLEDGEMENT

I would like to express my deep gratitude to the Masters department of engineering of NCIT, for providing me with the opportunity to do this project.

I would like to extend my sincere gratitude to my supervisor “**Prof. Dr Roshan Chitrakar, Ph. D**” for his valuable suggestions and guidance throughout the course of this project. I am also highly thankful to our program coordinator “**Mr. Saroj Shakya**” of Master’s in Computer Engineering for his constant support and guidance as well as to the department. Also, I would like to thank all my classmates and teachers for their constant support and suggestions on this project.

**Keywords:** SDN, DDOS, K-means Clustering, Genetic Algorithm, OpenFlow Controller, DFS

.....

Sahaj Shakya

2019-1-39-0020

08/21/2022

## ABSTRACT

Today's widely used networks are complex and difficult to manage. To implement high-level network guidelines in traditional IP-based networks, network administrators must configure individual network devices with manufacturer-specific commands. As a result, it becomes very difficult to implement the desired policies and reconfigurations of network devices in today's IP-based networks. Software Defined Networking (SDN) has been adopted for the Flexible Internet and has the potential to be used for the next generation of the Internet by using a controller to separate the control plane from the data plane. A distributed denial of service (DDoS) attack can make on the SDN controller unable to process legitimate flow requests from the switch. The main approach is to protect a controller from DDoS attacks is based on attack detection which results in a high rate of false-negative and false-positive results. Existing mitigation techniques are basically based on external and additional resource or network traffic analysis and tend to be computationally intensive or have a high rate of false positives and / or false positives. The management of 150 switches in a system can only handle up to 20,000 new flow requests. Therefore, the controller must do a lot of work to handle network traffic efficiently. However, additional computational overhead can occur for multiple parameters and optimizing the controllers' positions. The purpose methods reduce the overhead in three steps, firstly selecting the controller for overhead traffic, secondly using genetic algorithm to elect the leader and third select the optimize position for controller from clustered controllers.

**Keywords:** SDN, DDOS, K-means Clustering, Genetic Algorithm, OpenFlow Controller, DFS

## Table of content

Title	Page
Acknowledgement	i
Abstract	ii
Table of contents	iii
List of tables	v
List of figures	vi
Abbreviations/Acronyms	x
<b>CHAPTER 1</b>	
INTRODUCTION	
1.1. Types of Attacks	1
1.2. Attacks in SDN Layers	2
1.3. Nsl-KDD	4
1.4. Statement of the problem	7
1.5. Research Objectives	7
1.6. Significance/Rationale of the study	8
<b>CHAPTER 2</b>	
LITERATURE REVIEW	10
<b>CHAPTER 3</b>	
CONCEPTUAL MODEL	13
3.1 Conceptual Algorithm	13
3.1.1 Generated Traffic	13
3.1.2 Pre-processing unit	14
3.1.3 Main Controller	18
3.1.4 Secondary Controller	18
3.1.5 Decision Unit	19
<b>CHAPTER 4</b>	
METHODOLOGY	22
1.1 Training of model	23
1.2 Attack detection	29
1.3 Overhead Handling	31
I. Controller Stability and Overhead analysis	31
II. Optimal Path detection and Selections	32
<b>CHAPTER 5</b>	
RESULTS	40
5.1 Testing the Algorithm based on Simulated Environment	40

5.2 Training the Algorithm based on Simulated Environment	44
5.3 Dynamic Path Selections	46
5.4 Controller Performance	46
<b>CHAPTER 6</b>	
DATA ANALYSIS AND REPORTING	48
6.1 Validation of Model	48
6.2 Path Selections	52
6.3 Receiver operating characteristic curve and AURUC	52
6.4 Comparison with work performed by other authors	53
<b>CHAPTER 7</b>	
RECOMMENDATIONS	54
<b>CHAPTER 8</b>	
CONCLUSIONS AND DISCUSSIONS	55
REFERENCES	56

## List of Tables

	<b>Title</b>	<b>Page</b>
Table. 1-3.1	Features of NSL-KDD	5
Table. 3-1.1	IPV4 Packet	15
Table.4-1.1	Training dataset generated in tabulated form for two switches	28
Table. 5-1	Training dataset generated in tabulated form for two switches	41
Table. 6-1	Comparison of Algorithm with other Authors	53

## List of Figures

	<b>Title</b>	<b>Page</b>
Figure. 1-1	SDN Layers	1
Figure. 1-2	Attacks in SDN Layers	2
Figure 1-3.2	Parts of NSL-KDD Dataset	6
Figure 3-1	Conceptual Model	13
Figure 3-1.1	IPV4 Packet	14
Figure 4	Experimental Model from Algorithm	22
Figure 4-1.1	Mininet Topology with two controllers	23
Figure 4-1.2	One Hot Encoding Data	23
Figure 4-1.3	Correlation of Features	24
Figure 4-1.4	Open Flow Table with fields	25
Figure 4-1.5	Matching using MAC Address	26
Figure 4-1.6	Matching using IP Address	26
Figure 4-1.7	Matching using Protocol Address	26
Figure 4-1.8	Traffic on OVS Switch	26
Figure 4-1.9	Multiple controller Topology	27
Figure 4-1.10	Normal Traffic Generation	27
Figure 4-1.11	Attack Traffic Generation	27
Figure 4-1.12	Flow Table Topology	28
Figure 4-2.1	Attack Detection	29
Figure 4-2.2	Normal Traffic between two host	29
Figure 4-2.3	Attack traffic flow from Host 1	30
Figure 4-2.4	Attack traffic flow from Host 1 in case of two controllers	30
Figure 4-3	Selection Methodology for overhead detection	32
Figure 4-4	Multipath routing Topology	32



Figure 4-5	S1 Flow Table	33
Figure 4-6	S2 Flow Table	33
Figure 4-7	S3 Flow Table	33
Figure 4-8	Traffic Flow between H1 and H2	34
Figure 4-9	Traffic Generations between H1 and H2	34
Figure 4-10	S2 Flow Table with no load balancing	35
Figure 4-11	S3 Flow Table with no load balancing	35
Figure 4-12	Multipath Static Routing	35
Figure 4-13	Multipath Static Load Distributions	35
Figure 4-14	Multipath Weight Assignment	36
Figure 4-15	Multipath Flow Assignment Table	36
Figure 4-16	Multipath Flow Assignment	37
Figure 4-17	Multipath routing using DFS	37
Figure 4-18	Multipath routing using DFS in programs	39
Figure 5-1.1	Cluster of 84 fields using K Mean based on PCA on NSL-KDD	41
Figure 5-1.2	K mean Clustering Accuracy Calculations	42
Figure 5-1.3	Confusion Matrix of KNN Algorithm	42
Figure 5-1.4	Accuracy of KNN Algorithm	42
Figure 5-1.5	Accuracy, Precision, Recall and F- measures of KNN Algorithm	43
Figure 5-1.6	Confusion Matrix of SVM Classifier	43
Figure 5-1.7	Accuracy, Precision, Recall and F- measures of SVM Classifier	44
Figure 5-2.1	Distribution of traffic based on various features vectors or parameters	45
Figure 5-3.1	Multipath routing	46
Figure 5-3.2	Dynamic weight assignment switch flow	46
Figure 5-3.3	CPU Utilization for single controller	46
Figure 5-3.4	CPU Utilization for multiple controller	47
Figure 6-1	Confusion matrix using SVM Classifier	48

Figure 6-2	Precision, Recall and F1 score for above topology	48
Figure 6-3	Variance and correlation between features and traffic	49
Figure 6-4	Confusion matrix of Dataset	50
Figure 6-5	Confusion matrix of Dataset from Excel	50
Figure 6-6	AURUC of Dataset	51
Figure 6-7	AURUC of Dataset from Excel	51
Figure 6-8	ROC curve for two switch topology	52

## **List of abbreviation/acronyms**

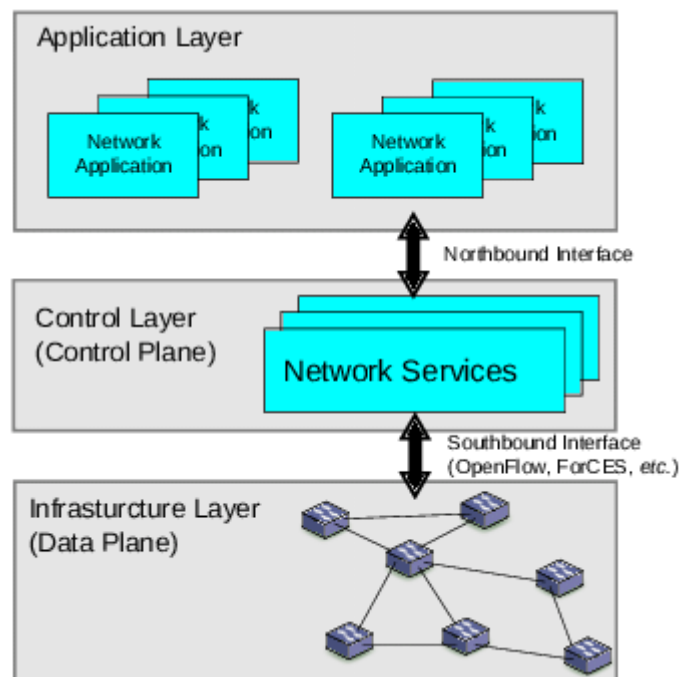
DDOS	Distributed Denial of Service
SDN	Software Defined Network
ANN	Artificial Neural Network
ANP	Analytic network process
SOM	Self-Organizing Map
DR	Detection Rate
TP	True Positive
FN	False Negative
FP	False Positive
SVM	Support Vector Machine
KNN	K Nearest Neighbor
SSIP	Speed if IP Sources
SFE	Speed of flow entries
RPE	Ratio of pair flow entries
DFS	Depth First Search

## CHAPTER 1

### INTRODUCTION

A distributed denial of service (DDoS) attack occurs when one or more attackers attempt to block the delivery of a service. This can be achieved by forcing access to virtually everything, including servers, devices, services, networks, applications, and even specific transactions within an application. A DoS attack is a system that sends malicious data and requests. DDoS attacks come from multiple systems. Impacts range from service interruptions to entire websites, applications, and even business failures, from minor annoyances. However, a system can be developed to mitigate such problems. The system can be trained based on dataset. For this methodology, NSL-KDD can be used.

Many researchers have proposed and implemented various models for IDS but they often generate too many false alerts due to their simplistic analysis. An attack generally falls into one of four categories [16].



**Figure 1-1 SDN Layers**

### 1.1 Types of Attacks

#### 1.1.1. Denial-of-Service(DoS):

Attackers tries to prevent legitimate users from using a service. For example, there are smurf, neptune, back, teardrop, pod and land.

#### 1.1.2. Probe:

The attacker attempts to obtain information about the target host. Port scans or sweeps of a specific range of IP addresses usually fall into this category. (e.g. saint, ipsweep, portsweep and nmap).

#### 1.1.3. User-to-Root(U2R):

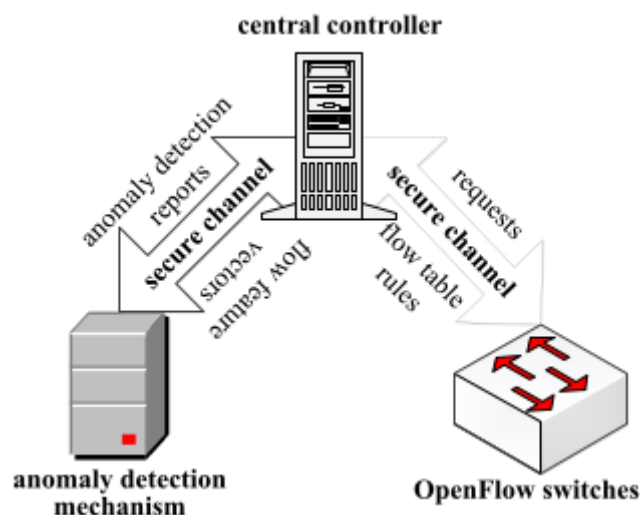
Attacker attempts to gain super user privileges by accessing the victim's computer locally. For example, these are buffer\_overflow, rootkit, landmodule and perl.

#### 1.1.4. Remote-to-Local(R2L):

The attacker does not have an account on the victim's computer and is trying to access it. For example, these are guess\_passwd, ftp\_write, multihop, phf, spy, imap, warezclient and warezmaster.

SDN controller causes potential threats due to its single point failure. The major vulnerabilities in SDN can be categorized into three major attacks based on the different plane of SDN architecture [17].

### 1.2 Attacks in SDN Layers



**Figure 1-2 Attacks in SDN Layers**

#### 1.2.1 Data Plane Attacks:

Space constraints in the data plane lead to buffer overflows and stream table overflows, which are the main causes of security issues challenges in data

plane attacks. It is difficult to identify malicious and true flows in the data plane due to the presence of a discharge switch and the role of the controller in decision-making. As data traffic increases, congestion on the data control plane link causes the control plane and data plane to shut down. Also, corruption of the SDN controller will corrupt the data plane resources. Therefore, a data plane attack depends on the security of the control plane [17].

#### 1.2.2 Control Plane Attacks:

Due to the centralized structure of the controller, the control plane is the focus of the majority of attacks. It comprises dangers from the application, scalability threats, and availability threats. The majority of unaddressed data plane issues result in control plane saturation attacks. The controller is in charge of performing a tailored security check of various applications, including application authentication and resource authorization for resources that have not yet been established. Furthermore, today's SDN controllers are incapable of handling network traffic on a 10 Gbps link in a high-speed network. DoS attacks in SDN are made easier by the SDN controller's lack of scalability and the unavailability of network resources. Multi-controller is not a good solution for DDoS attacks as it can result in cascading failure of all controllers if there is not optimizing mechanism for switching between them [17].

The DDoS attacks are categorized under the availability threat of the controller. The reasons for DDoS vulnerabilities are as follows:

- Due to the limited memory space available to buffer the information, buffer saturation occurs.
- Due to overhead in the controller caused by the controller's centralized architecture is known as controller saturation.
- due to limited TCAM memory which causes Flow Table overflow
- Due to control-data plane link's communication overhead generates a bottleneck for genuine users.

### 1.2.3 Application Plane Attacks:

It includes authentication and authorization issues, as well as access control and accountability concerns. Every request from the application to access network resources must be authenticated. Authentication of a large number of SDN apps, on the other hand, is difficult. Furthermore, due to a lack of access control and accountability, the malicious program can evade the SDN network architecture [17].

## 1.3 NSL-KDD

NSL-KDD is an up to date model of KDD cup99 statistics set which counseled to remedy a few troubles of preceding model. This statistics set is a powerful benchmark for researchers to evaluate special kinds of Intrusion detection system (IDS) methods, construct an Intrusion detection system (Host primarily based totally or Network primarily based totally), doing for a few experiments in Cyber safety region like smart there may be such a lot of advantages. And additionally, there are lot of statistics sets (ADFA-ID, ISCX-UNB etc) which use on this field [17].

There are 8 types of different data sets in the NSL-KDD.

- KDDTrain+.ARFF The full NSL-KDD train set with binary labels in ARFF format.
- KDDTrain+.TXT The full NSL-KDD train set including attack-type labels and difficulty level in CSV format.
- KDDTrain+\_20Percent.ARFF A 20% subset of the KDDTrain+.arff file.
- KDDTrain+\_20Percent.TXT A 20% subset of the KDDTrain+.txt file.
- KDDTest+.ARFF The full NSL-KDD test set with binary labels in ARFF format.
- KDDTest+.TXT The full NSL-KDD test set including attack-type labels and difficulty level in CSV format.
- KDDTest-21. ARFF A subset of the KDDTest+.arff file which does not include records with difficulty level of 21 out of 21.
- KDDTest-21.TXT A subset of the KDDTest+.txt file which does not include records with difficulty level of 21 out of 21.

No.	Feature Name	No.	Feature Name
1	Duration	23	<b>count</b>
2	Protocol_type	24	Srv_count
3	service	25	Serror_count
<b>4</b>	<b>flag</b>	26	Srv_serror_rate
<b>5</b>	<b>Src_bytes</b>	27	Rerror_rate
<b>6</b>	<b>Dst_bytes</b>	28	Srv_rerror_rate
7	land	29	Same_srv_rate
8	Wrong_fragment	30	diff_sev_rate
9	urgent	31	Srv_diff_host_rate
10	hot	32	Dst_host_count
11	Num_failed_logins	33	Dst_host_srv_count
12	Logged_in	34	Dst_host_same_srv_rate
13	Num_compromised	35	Dst_host_diff_srv_rate
14	Root_shell	36	Dst_host_same_src_port_rate
15	Su_attempted	37	Dst_host_same_diff_port_rate
16	Num_root	38	Dst_host_error_rate
17	Num_file_creations	39	Dst_host_srv_serror_rate
18	Num_shells	40	Dst_host_rerror_rate
19	Num_access_files	41	Dst_host_src_rerror_rate
20	Num_outbound_cmds		
21	Is_host_login		
22	Is_guest_login		

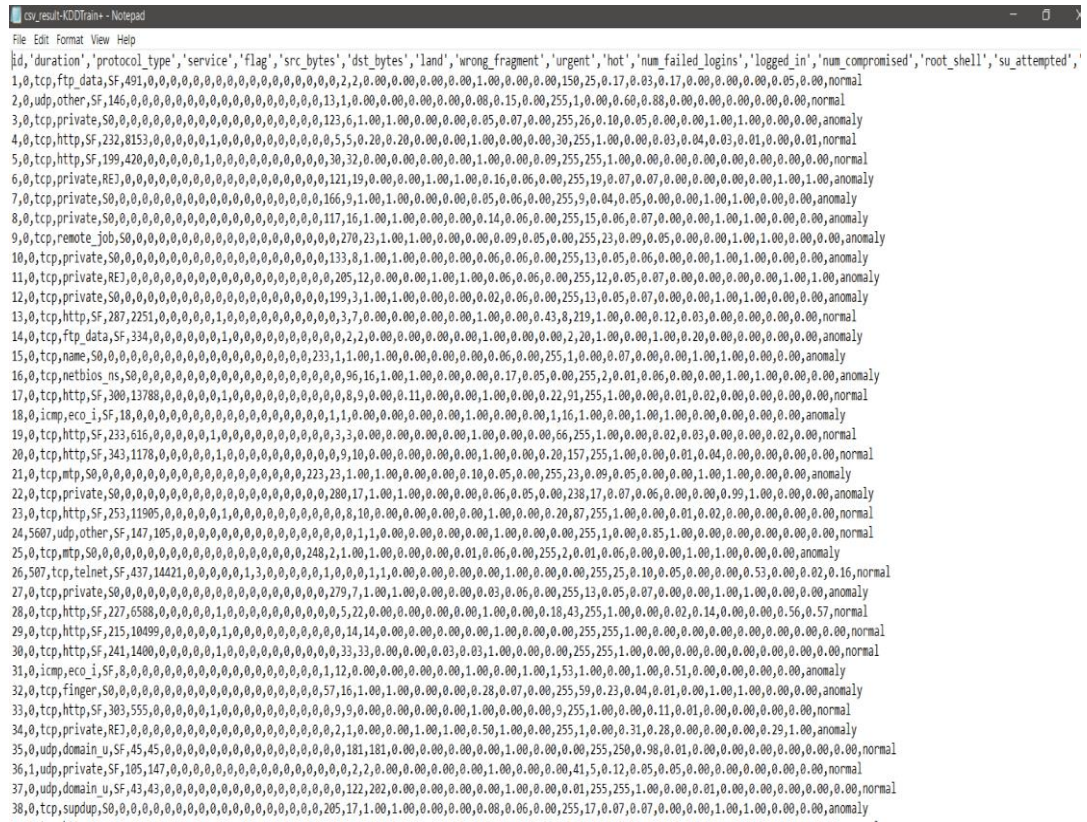
**Table 1- 3.1 Features of NSL-KDD**

Bolded text on the above tabular form are the labelled features chosen after the standardization and normalizations



Attacks in this Dataset is categorized into 4 parts.

- DoS- Denial of service.
- Probing- Surveillance and other probing attacks.
- U2R- Unauthorized access to local super user.
- R2L- Unauthorized access from a remote machine.



**Figure 1-3.2 Parts of NSL-KDD Dataset**

Above figure represents the labelled dataset of NSL-KDD Dataset that consists of features such as Durations, count, services, flags and so-on

## **1.4 Statement of the problem**

Distributed denial of service (DDoS) attacks are on the rise, and many organizations are either completely unprepared or poorly defended. The impact of a DDoS attack usually consists of loss of sales and customers, damage to the brand, and use as a smoke screen for further inbound attacks. Companies with business-critical online resources are not only exposed to greater losses, but are also more likely to be attacked by attackers. For this reason, enterprises need to be proactive in addressing DDoS threats [1].

In DDoS detection and prevention based on ANN, there are mainly two issues

### **1.4.1 Maximum Controller detection overhead**

The management of 150 switches in a system can handle up to 20,000 new flow requests. Therefore, the controller must do a lot of work to handle network traffic efficiently. However, additional computational overhead can occur for multiple parameters which might impact controller performance. The number of parameters used for DDoS detection can be reduced in an ANN-based way. However, it can also be improved by using multiple machine learning algorithms to reduce the complexity of the controller. Similarly, multiple OpenFlow controllers can be also used to mitigate the problem [2].

### **1.4.2 False Negative and False Positive Value**

False negative values are common when the attack rate parameter is set to a low value. However, an attack with a low packet flow rate will do less damage to the victim's system [3]. However, this problem can be mitigated by using multiple machine learning algorithms [4].

## **1.5 Research objectives**

The major objectives of this thesis are

- To mitigate the Single Point Failure and overhead of Controller
- To reduce the false positive detection in case of Slow Attack

## 1.6 Significance/Rationale of the study

With the rapid development in the field of IT infrastructure, the size of the network, its complexity has increased many times. This makes it increasingly difficult to guarantee key network characteristics such as integrity, confidentiality, authentication, information availability, and non-repudiation. In recent years, many researchers and industries have shifted their focus to developing more robust, scalable, and more secure networks. Today's widely used networks are complex and difficult to manage. To implement high-level network guidelines in traditional IP-based networks, network administrators must configure individual network devices with manufacturer-specific commands. As a result, it becomes very difficult to implement the desired policies and reconfigurations of network devices in today's IP-based networks [5].

The latest advances, such as SDN (Software Defined Networking), are a step towards the dynamic and centralized nature of networks compared to traditional network static distributed environments. Recently, Software Defined Networking (SDN) has been adopted for the Flexible Internet and has the potential to be used for the next generation of the Internet. The concept is to control the network through software. This approach facilitates network management and enables new applications in virtual environments. SDN was developed with extended network support from the central controller. SDN is a new network architecture that gives hope to an efficient network infrastructure. First, vertical integration is eliminated by separating the control plane (network control logic) from the data plane (routers and switches that route network traffic according to the network control logic). This isolation control then provides a simple packet forwarding device that simplifies flexibility, implementation speed, programmability, and network management with a logically centralized controller and network logic installed on the network switch. The SDN architecture can improve the security of the network with the help of a centralized controller, but it creates global transparency for the network and, if necessary, traffic routing rules. However, security issues still remain in the SDN. [4].

However, SDN still has its own concerns and challenges regarding network security, scalability, and support capabilities. Security is paramount because of all these issues. Since the central controller is responsible for managing the network, a

failure of this controller will affect the entire network. Central control and communication between the controller and the switch can be the target of advanced DDoS attacks.

Detecting flood-based distributed denial of service (DDoS) attacks is one of the biggest challenges for Internet security today. The DDoS attack mechanism is based on exploiting the huge resource asymmetry between the Internet and the victim's server limitations in handling a large number of fake requests. As a result, system resources are exhausted and the victim is taken from the Internet, so legitimate user requests are not processed [6].

With the advent and development of software-defined networks over the last decade, the ability to combat DDoS attacks in cloud environments has expanded. SDN has been determined to be an integral part of cloud and service providers that make networks programmable [7]. One of the problems is changing the packet header fields, similar to regular fields. As a result, it is very difficult to distinguish between legitimate regular traffic packets and useless packets sent from the compromised host to the victim. The second problem is the large number of packets that need to be analyzed. These are challenges that make detection difficult and slow response times.

The SDN controller can be a single point of failure. Security challenges are expected to increase as SDN technology becomes increasingly available. These SDN vulnerabilities focus on different layers of the SDN architecture. Analysis different vulnerabilities, security challenges can be summarized at different levels [4].

## CHAPTER 2

### LITERATURE REVIEW

Machine learning algorithms can use training data to automatically create classification models and use flow functions to classify network flows. In a real-time network environment, machine learning algorithms can detect known and unknown DDoS attacks. In the area of network security, the benefits of this SDN allow developers to easily and flexibly update their classification mechanisms to detect anomalous attacks on the network control plane. The SDN controller quickly collects and analyzes information from the switch and sends operational decisions back to the switch. Due to this flexible and effective process, the detection of SDN-based DDoS attacks has recently attracted the attention of the research community.

Initial was of mitigation mechanism uses HTTP-based and XML-based DDoS attacks preventions against cloud computing environments using a filter tree. This tree-based filtering scheme to prevent DDoS attacks uses five levels of filtering to mitigate the impact of HTTP-based and XML-based DDoS attacks. This tree-based DDoS attack prevention approach with filtering analyzes suspicious packets in the puzzle resolver to address issues caused by malicious data packets generated based on SOAP headers. This filter tree scheme identifies the IP address that was initially initiated by the malicious message to send the puzzle, and the puzzle sent is resolved to determine the actual client [8].

In case of “Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow”, the researcher has used multiple parameters for calculation of the current state of the network. Researchers have presented a lightweight method for detecting DDoS attacks based on traffic flow characteristics that extract such information with very little effort compared to traditional approaches [6]. This is made possible by the use of the NOX platform, which provides a programmatic interface that facilitates the processing of switching information. Other important contributions by researchers include high detection rates and very low false alarm rates achieved by flow analysis using self-organizing maps.

OpenFlow-based SDN protection against DDoS attacks was provided to prevent problems arising from smooth packet replay attacks. This OpenFlow-based SDN framework leverages the mutual benefits of the data plane and control plane used in the investigation to control the strength of DDoS attacks. This OpenFlow-based SDN framework is also important for managing and monitoring traffic flows that utilize SDN in the cloud. A fast and accurate

SDN-based DDoS attack prevention scheme has been proposed to solve the problems caused by flood-based DDoS attacks using the process of entropy variation. It has been found that this entropy coefficient of variation used in this accurate SDN-based DDoS attack defense scheme can enable an accurate and clear classification of normal and malicious traffic. The researcher found that the false positive rate of this entropy variable factor-based DDoS attack mitigation scheme was reduced by 12% compared to the mitigation framework focused on the OpenFlow-based monitoring process [9].

As per “Evaluating the Controller Capacity in Software “, an important issue with the OpenFlow architecture is the capacity of the controller. This can be defined as the number of switches that the controller can manage. This white paper models the switch-to-controller flow setup requirements as a batch arrival process  $M_k / M / 1$ . In addition, queuing theory is used to analyze controller performance to derive an equation for average flow uptime. In the situation of limited flow rate setting time, the number of switches is determined, which provides a way to evaluate the capacity of the regulator. The centralized controller is only responsible for creating policies. The researcher state that on management of 150 switches, only 20,000 new flow requests can be handled. However, this additional computational overhead for multiple parameters can impact controller performance. [1].

As per the paper “QoS improvement with an optimum controller selection for software-defined networks”, the controller has multiple functions that guide the network from the center point and respond to updates related to topology changes. However, the ability to support these features is strong on one controller and weak on another. Choosing the best SDN controller can be considered a multiple-criteria decision-making problem (MCDM), as multiple controllers and each controller have many features. Here the researchers have proposed a two-step approach for choosing an SDN controller. First, the researcher classifies the controllers using the Analytical Network Process (ANP) according to the qualitative characteristics that affect the performance of these controllers, and then perform a performance comparison to see the improvement in QoS. Controllers with high weights from feature-based comparisons are analyzed quantitatively by experimental analysis. The researcher's main contribution is to confirm the applicability of ANP to controller selection in SDN, taking into account feature and performance analysis in real-world Internet and Brite topologies [3]. Choosing the best controller with ANP results in faster topology discovery times and delays in normal and traffic load scenarios. Researchers have also seen

an increase in throughput with the controller, which makes good use of the central processing unit.

KNN classifies flows by measuring the distance between flow feature vectors. It's simple and effective [10]. Peng et al. We proposed a KNN-based method using a transductive confidence machine (TCM) for SDN anomaly detection [11]. Nam et al. Combined ANN and SOM to address the DDoS flood. Linear ANNs are very time consuming, so the Kth Dimensional tree (KD tree) stores training points in a tree structure for quick queries on ANNs [4]. However, the KD tree must create an index chain for all training units. Changes in training sessions affect detection accuracy [3].

Shin et al. Have proposed a system called Avant-Guard that can identify DDoS of TCP SYN floods including alarm services [12]. Line Switch then extended it to include a statistics collector. Also, Kotani et al. they proposed a packet filter mechanism to prevent attacks by Open flow for TCAM. The implementation here is focused only on software switches and open switch [13].

PATGEN, a Protocol to reduce the effects of DDoS Attacks using an advanced Genetic algorithm with optimized and new operators, thereby significantly reducing the effects of attacks and increasing the efficiency of the multi-controller SDN [14].

## CHAPTER 3:

### CONCEPTUAL MODEL

The Conceptual model is classified into two parts. One is Training and testing the algorithm on NSL-KDD Data set and another part is to use the algorithm on simulated Environment.

#### 3.1 Conceptual Algorithm

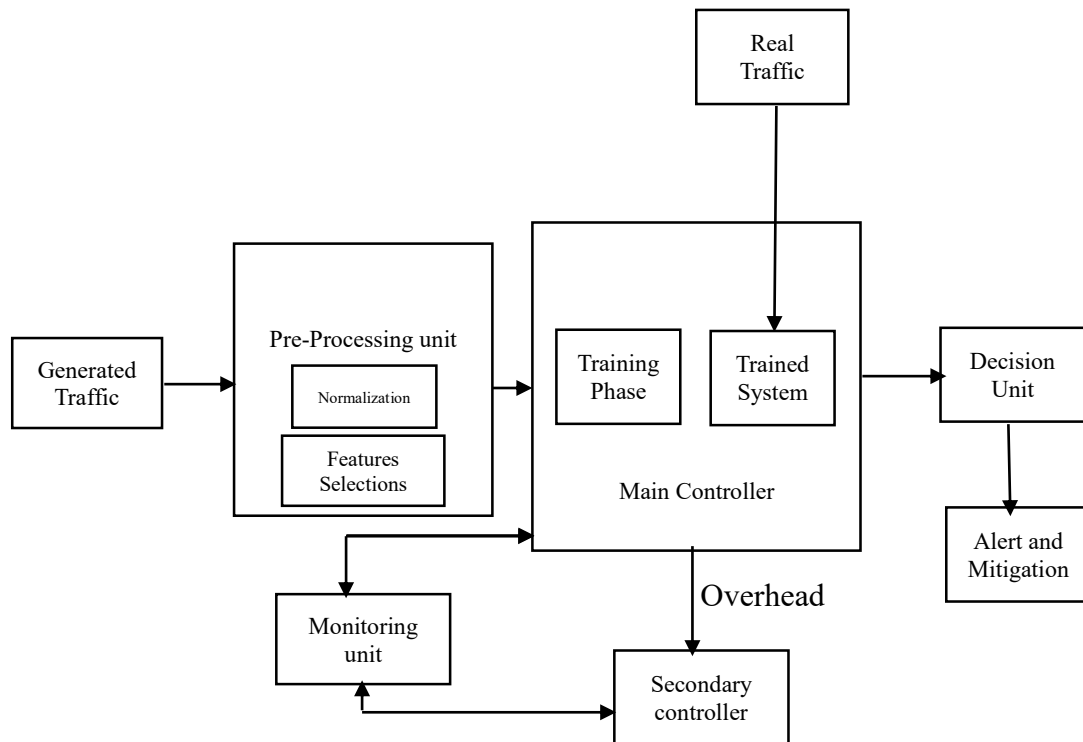


Figure 3-1 Conceptual Algorithm

##### 3.1.1 Generated traffic:

The normal traffic and attack traffic generated in controlled environment is the input to this block. The traffic is segregated in this block for further processing or precisely to pass to pre-processing unit. A Traffic has multiple packets that can be TCP or UDP depending upon their uses. A Packet contains general information's such as flag, source address, destination address and payload along with the protocol is uses. The general idea to difference between attack traffic and normal traffic can be measured by expectation of packet size. if the traffic causes others to be denied service. The first step is to analyze the requests that are being sent. Some attacks will try to establish a connection but never submit a complete request (slow loris). Once the features of an attack are studied, determining whether the requests are legitimate or malicious becomes simple.



Mathematically,

TCP:

Size of Ethernet frame = 24

Bytes Size of IPv4 Header (without any options) = 20 bytes

Size of TCP Header (without any options) = 20 Bytes

Total TCP Segment =  $24 + 20 + 20 = 64$  bytes

UDP:

Size of Ethernet frame = 24

Bytes Size of IPv4 Header (without any options) = 20 bytes

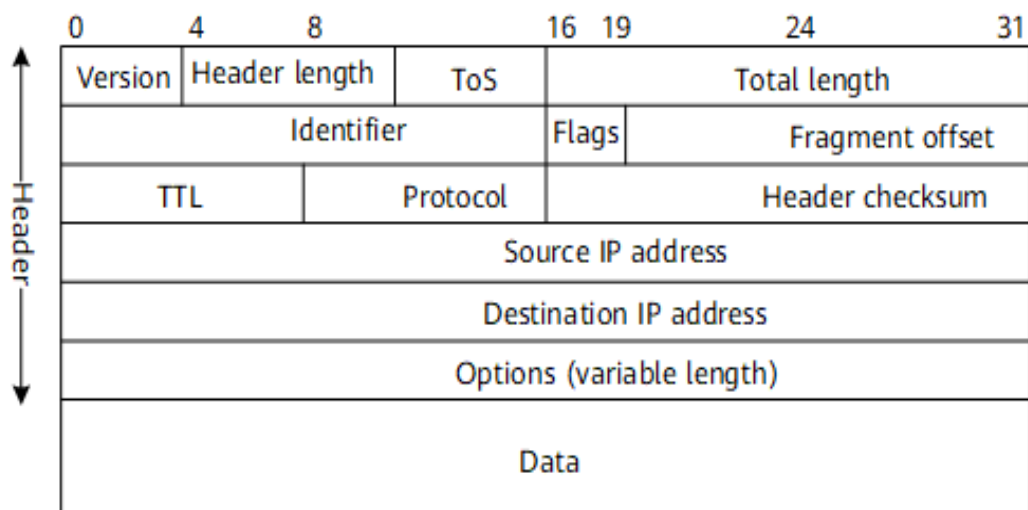
Size of UDP header = 8 bytes

Total IP Packet with an empty UDP Datagram=-  $24 + 20 + 8 = 52$  bytes

Malicious traffic will be static in size. Malicious packets generally will have the basic minimum needed for the protocol and will have very little or no payload like above. Wherease, Real traffic will have larger segments and will vary.

### 3.1.2 Pre-processing unit:

#### A. Data Analysis



*Figure 3-1.1 IPV4 Packet*

The Training set is divided as:

- Feature 'protocol\_type'
- Feature 'service'
- Feature 'flag'
- Feature 'label'

Similarly, distribution of categories in Features can be done as

S.N	Application layer Protocol Type	Transport Layer	Flag
1	Http	TCP	SF
2	Ftp	TCP	S0
3	SMTP	TCP, UDP	SF

***Table 3-1.1 IPV4 Packet***

A traffic analysis depends upon Protocol type, flag value and how traffic is flowed through the network. TCP FLAG flood is a DDoS attack that saturates bandwidth and resources on devices in its route, causing network activity to be disrupted. Stateful defenses can be brought down by repeatedly sending ALL TCP Flags packets towards a target. Similarly, in a UDP Flood, attackers use a vast range of source IPs to transmit tiny faked UDP packets at a high rate to random ports on the victim's system. This uses the network's bandwidth. Since all the features and flag value could not be analyzed, the value should be normalized and only necessary features should be selected.

## **B. Features Normalizations**

LabelEncoder is used to labels a value between 0 and  $n^{\text{th}}$  number of class. LabelEncoder is to use to categories the data to find any kind of relationship between them.

One Hot Encoding converts the data to prepare it for the algorithm and get better prediction results. Each categorical column is converted to a new caterogial columns to find further relation between it. In Short Normalization is done by normalizing the standardized data into [0, 1]. It is assumed that  $X_{ij}$  is the normalized value of  $X'_{ij}$  [13].

$$X''_{ij} = \frac{X'_{ij} - X_{min}}{X_{max} - X_{min}} \quad (3.1.2)$$

Where,

$$X_{min} = \min[X'_{ij}]$$

$$X_{max} = \max[X'_{ij}]$$

Since all the attack traffic is set to 1, the objective is to find the correlated between the traffic, flag and the protocol for which the conditions should be valid.

### C Features Selection

To improve the effectiveness of data mining algorithms, feature selection is critical. The majority of the data contains features that are irrelevant, redundant, or noisy. Feature selection is a technique in data mining for dimension reduction that involves picking a subset of original features based on particular criteria. It is an important and widely used approach. It brings about palpable results for applications by reducing the amount of features, removing unnecessary, redundant, or noisy features, and speeding up a data mining method, boosting learning accuracy, and leading to higher model comprehensibility. For feature reduction, there are two main ways. A Wrapper analyzes the usefulness of features using the intended learning process, whereas a filter evaluates features using heuristics based on the data's general properties. Although the wrapper approach is thought to create superior feature subsets, it is slower than a filter. In this two feature subset selection methods are used Correlation-based Feature Selection (CFS) and Information Gain (IG) to compare our method [19].

#### I. Correlation-based Feature Selection

CFS has two concepts. One is the feature-classification (r<sub>cfi</sub>) correlation and another is the feature-feature (r<sub>fifj</sub>) correlation. These two ideas are based on the theory that "good feature subsets comprise features that are substantially linked with the classification but not with each other." The feature-classification correlation shows how closely a characteristic is linked to a certain class. The correlation between two characteristics is known as the feature-feature correlation [19].

$$Ms(k) = \frac{K(r_{ef})^c}{\sqrt{k+k(k-1)(r_{ff})^c}} \quad (3.1.4.1)$$

Which is average feature classification correlation to average feature to feature correlation

## D Post Processing

After the Normalizations and Features classification is completed. In this block, the input traffic is operated on by statistical analysis to generate various feature vectors also known as representative parameters like Speed of IP sources (SSIP), Speed of flow entries (SFE) and Ratio of pair flow entries (RPE) based on the selected features to generate a set of samples known as training sample dataset [23].

### I. Speed of IP sources (SSIP):

The parameter is given by the formula

$$SSIP = \frac{\text{sumIP}_{src}}{T} \quad (3.2.2.1)$$

where, the total number of IP sources incoming in each flow is  $\text{sumIP}_{src}$ , and the sampling time intervals are T. The time period T is set to three seconds, causing the detection system to monitor and gather data from flows every three seconds, as well as store the number of source. IPs during that time. For the machine learning system to predict attacks, the controller must have enough data of both regular and attack traffic. The SSIP for benign traffic is usually low, whereas the count for attacks is frequently larger.

### II. Speed of flow entries (SFE):

The parameter is given by the formula

$$SFE = \frac{N}{T} \quad (3.2.2.2)$$

where, N gives the total number of flow entries entering into a switch of a network and T denotes a particular time period. This is an extremely relevant feature of attack traffic detection, since in the event of DDOS attacks, the number of flow entries increases significantly in a fixed time interval as opposed to normal traffic flows.

### III. Ratio of pair flow entries (RPE):

It is given by the formula

$$RPE = \frac{srcIP_s}{N} \quad (3.2.2.3)$$

where, N represents the total number of Ips, and srcIPs represents the number of collaborative IPs in the network flow. In each sampling, the representative flow table entries in every sampling are calculated to form a sample set P, which is written as  $Q = (P, S)$ , where P represents flow table entries of triplet-characteristics value matrix, and S denotes the marker vector for category corresponding to P: “0” denotes the normal condition and “1” denotes the attack condition. The generated sample set is then fed to SVM classifier to train the model for further detection of real time traffic. There is a training set  $D = [(P_1, s_1), \dots, (P_n, s_n)]$ , where  $P_i$  is the characteristic vector, whereas  $s_i$  is the corresponding class label. In this experiment,  $s_i$  belongs to  $(-1, +1)$  takes the value of either 1 or 0.

#### 3.1.3 Main controller:

It is the block where the main operation of classification of DDoS traffic from benign or normal traffic takes place. The block can be further divided into training phase unit and trained module. Training phase unit optimizes the algorithm for further clustering of incoming traffic. In order to determine the type of the traffic received from the network; the characteristics discussed in the theoretical section needs to be modeled. The parameters are calculated on basis of incoming traffic as Trained module is block to which real time incoming traffic is passed for classification into normal traffic and attack traffic.

#### 3.1.4 Secondary controller:

If there is maximum overhead or load on the main controller, the controlling operation is transferred to this secondary controller and all the incoming traffic passes through it consequently.

### 3.1.5 Decision unit:

This block makes the decision to either classify the incoming traffic as an attack and detect the port from which the attack traffic is coming if there is DDoS attack on the network. There are various roles that a controller can play. The three roles are:

#### I. Equal

By default, all equal controllers in a network linked to the switch have the ability to add and delete flows from it. It means that all of the switch's controllers have complete control over the flow updates and modifications. The PACKET IN Message must be sent to all Controllers by the Switch. Also, all of the controllers' PACKET OUT, FLOW UPDATES, and so on are processed by the switch.

#### II. Master:

Similar to a switch, except only one master can be connected to it at any given time, and no other flows can intercept the communication between the switch and the master. It means that the MASTER controller will be in charge of the switch openflow dataplane management. Only the MASTER controller will receive control signals from the switch.

#### III. Slave:

It has a read-only limitation to the switch openflow tables. Slave plays backup role for MASTER. it also receives the HELLO and KEEPALIVE Message. But it cannot send and receive the Control message.

Since the default role of each controller is "Equal". The controller's job will be performed by the SDN switch. This is due to the fact that a single switch can be connected to multiple controllers. With respect to that switch, each controller can play a different role. A single controller can serve as both a slave and a master for different switches. This is why the controller's role is present in the switch, because the controller code is effectively the same.

#### IV. ROLE\_REQUEST and RESPONSE:

When the controller comes up, it will send the ROLE REQUEST with ROLE MASTER, other controller should send a role as SLAVE. Switch

will communicate with MASTER. The role request is done on the basis of stability values.

The stability of the controller can be used to determine the state of controller overhead. If Stability is 1, the controller is assumed to be insecure, and the entire node is optimized. For controllers having a stability of less than 1 [25]. The flag is set to 0.

The stability parameter is given by

$$Stability = \frac{\text{Number of completed requests of controller}}{\text{Number of current requests reaching the controller}} \quad (3.2.5)$$

In case of higher traffic, the controller needs to select an optimal path for which the traffic can be distributed based on the load. Load balancing in case of SDN depends upon various parameters such as Bandwidth, Reference of bandwidth, Speed, Performance of Controller. Basic concept of load balancing is to assign weights to the path and select the path based on the loads assigned. However, such approach is a static approach that depends upon the load assigned by the user and traffic is dynamic in nature. An approach to mitigate such issue is to dynamically assigned to load based on the requirements. For assigned the load, the controller need to find the overall path of flow and decided with route to take based on the parameters such as Bandwidth, Speed and Performance of each hardware. Basic Approach is divided into three step such that

- I. DFS Path Finding
- II. Path Calculations

The method for cost calculation can be used by:

- use OSPF to calculate the link cost.
- Add up the link costs for each and every link on the journey.

Cost Calculation is given by

$$bw(p) = \left(1 - \frac{pw(p)}{\sum_{i=0}^{i=n} pw(i)}\right) \times 100 \quad (3.2.6)$$

Where for a path (p):

- bw is the bucket weight,  $0 \leq bw(p) < 10$

- $pw$  is the path cost
- $n$  is the total number of paths available

### III. Install Flows

For OpenFlow to operate, a network of OpenFlow switches must be configured and managed by an SDN Controller (Ryu). There are two options:

- letting the controller decide how to route traffic, and
- teaching the switches to decide how to route traffic on their own by installing OpenFlow rules on the switches.

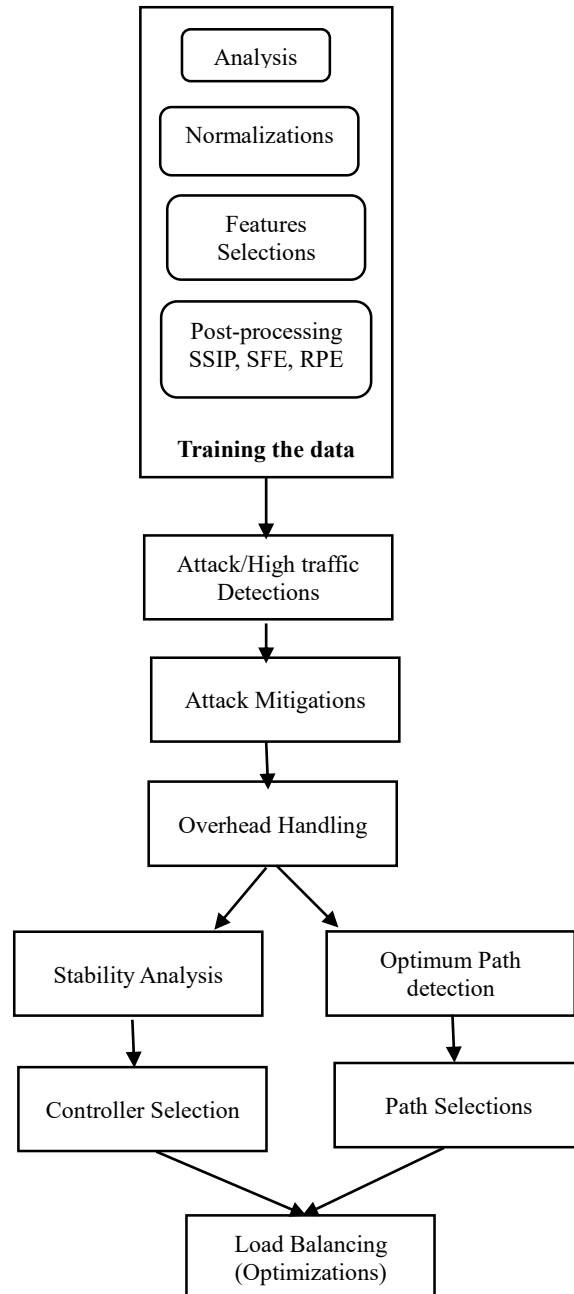
The second choice is more effective because it eliminates the need for the switch to continually ask the controller what to do when a packet arrives at the switch.



## CHAPTER 4

### METHODOLOGY

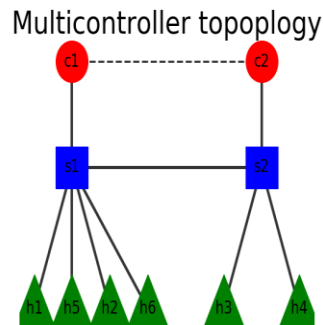
The working principle of the project can be broadly classified into four different subsequent stages: training of model, detection of attack traffic, mitigation of DDoS attack and controller switching on overhead detection.



**Figure 4 Experimental Model from Algorithm**

## 4.1 Training of model:

### A. Data Analysis



**Figure 4-1.1 Mininet Topology with two controllers**

Two ryu controllers and five openflow switches were used to create a network with hosts connected to both switches. The number of hosts per switches can be varied. The topology was created on Mininet based on python programming language.

### B. Features Normalizations

	protocol_type	service	flag
0	tcp	ftp_data	SF
1	udp	other	SF
2	tcp	private	S0
3	tcp	http	SF
4	tcp	http	SF
-----			
	protocol_type	service	flag
0	1	20	9
1	2	44	9
2	1	49	5
3	1	24	9
4	1	24	9

**Figure 4-1.2 One Hot Encoding Data**

For repeated label such as tcp is again assigned 1 and udp is assigned 0. This is done so that Alphabetic relation can be represented by numerical value so that predication and connection of features can easily be done. Similarly, for attack detection label encoding is done to represent the respective column to new column such that “Normal” and “Attack” so that new table with respective features can be represented. For simplicity all the normal traffic is assumed as 0 and rest of all the traffic are assumed to be 1 for easy correlation of flags with the traffic.

### C. Feature Selections

Features selection can be done by correlation of features so that strong correlated features can be reduced to a single feature. There will be an error in the correlation

[illegible]

and “UDP” can be taken. Similarly, Destination host flag is observed highly correlated with Attack state, which represents that only destination traffic needs to be analyzed. Similarly flag ‘SF’ is highly correlated with destination host bit, so Flag “SF” can be used to evaluate the destination packets time. Hence, Strong correlated data can be assumed as the threshold value of correlation for features selections.

#### **D. Post Processing**

In case of SDN, analysis of features can be done from flow entire in flow tables both reactively (in reaction to packets) and proactively (in advance) using the OpenFlow switch protocol. Each flow table in the switch has a collection of flow entries, which are made up of match fields, counters, and instructions to apply to matched packets. The matching process begins with the first flow table and may extend to other flow tables in the pipeline. Flow entries are used to match packets in order of priority, with the first matching entry in each table being used. The instructions associated with the specific flow entry are performed if a matching entry is identified. If no match is discovered in a flow table, the outcome is determined by the table-miss flow entry's setting.: for example, the packet may be forwarded to the controllers over the OpenFlow channel, dropped, or may continue to the next flow table. Packet forwarding, packet modification, and group table processing are all described in the instructions. Flow entries can be routed to a specific port. This is normally a physical port, but it could also be a switch-specified logical port (such as link aggregation groups, tunnels, or loopback interfaces) or a reserved port defined by this specification.

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags
--------------	----------	----------	--------------	----------	--------	-------

**Figure 4-1.4 Open Flow Table with fields**

A flow table entry is identified by its match fields and priority: the match fields and priority taken together identify a unique flow entry in a specific flow table.

A typical example of Match can be

## I. Match using Mac

```
cookie=0x0, duration=4.742s, table=0, n_packets=2, n_bytes=196, priority=1,in_port=s1-eth2,d_l_src=00:00:00:11:12,d_l_dst=00:00:00:11:11 actions=output:s1-eth1
cookie=0x0, duration=4.738s, table=0, n_packets=1, n_bytes=98, priority=1,in_port=s1-eth1,d_l_src=00:00:00:11:11,d_l_dst=00:00:00:11:12 actions=output:s1-eth2
cookie=0x0, duration=5.781s, table=0, n_packets=29, n_bytes=3102, priority=0 actions=CONTROLLER:65535
```

**Figure 4-1.5 Matching using MAC Address**

## II. Match using Ip address

```
cookie=0x0, duration=12.927s, table=0, n_packets=2, n_bytes=196, priority=1,ip,nw_src=192.168.1.1,nw_dst=192.168.1.2 actions=output:s1-eth2
cookie=0x0, duration=12.918s, table=0, n_packets=2, n_bytes=196, priority=1,ip,nw_src=192.168.1.2,nw_dst=192.168.1.1 actions=output:s1-eth1
cookie=0x0, duration=12.959s, table=0, n_packets=37, n_bytes=3844, priority=0 actions=CONTROLLER:65535
```

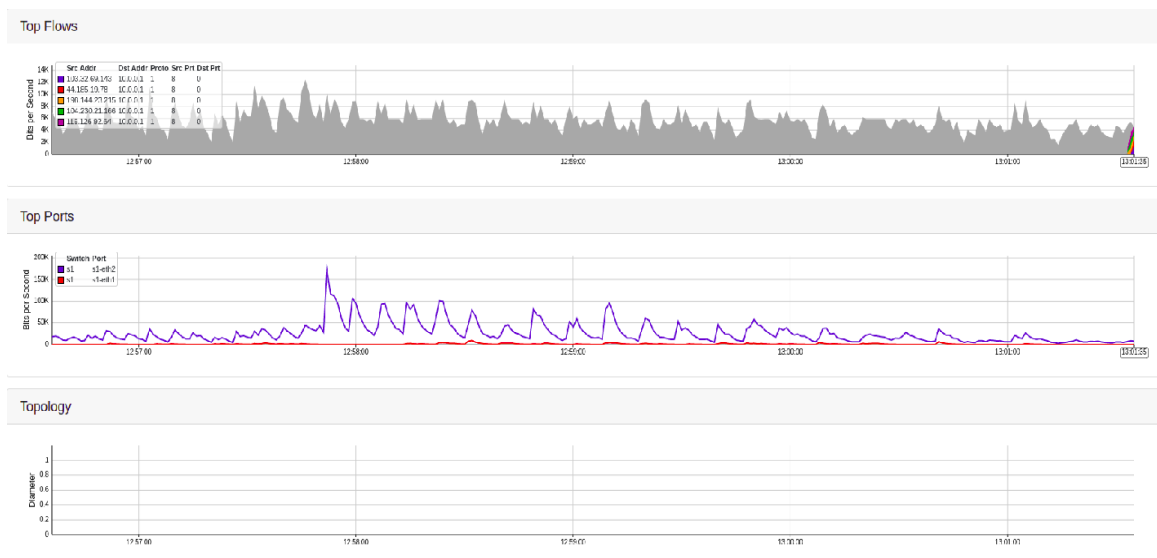
**Figure 4-1.6 Matching using IP Address**

## III. Match using Protocol

```
cookie=0x0, duration=3.933s, table=0, n_packets=238752, n_bytes=11069190464, priority=1,tcp,nw_src=192.168.1.2,nw_dst=192.168.1.1,tp_src=37304,tp_dst=5001 actions=output:s1-eth1
cookie=0x0, duration=3.906s, table=0, n_packets=192421, n_bytes=12699810, priority=1,tcp,nw_src=192.168.1.1,nw_dst=192.168.1.2,tp_src=5001,tp_dst=37304 actions=output:s1-eth2
cookie=0x0, duration=31.495s, table=0, n_packets=43, n_bytes=4309, priority=0 actions=CONTROLLER:65535
```

**Figure 4-1.7 Matching using Protocol Address**

During this step, the model is trained using both normal traffic and attack traffic. This can be done by analyzing the parameters such as timeouts, flags and Priority Fields. Similarly, if the match fields don't match the topology such kinds of traffics can be labeled as Malicious Traffics.



**Figure 4-1.8 Traffic on OVS Switch**

The incoming traffic to the switch is first stored. As the incoming traffic has lots of parameters, the traffic is applied to statistical analyzer which only generates a data



set consisting of useful and meaningful features. The data set consisting of these feature vectors is known as training data.

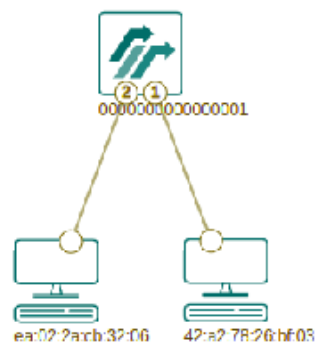


Figure 4-1.9 Multiple controller Topology

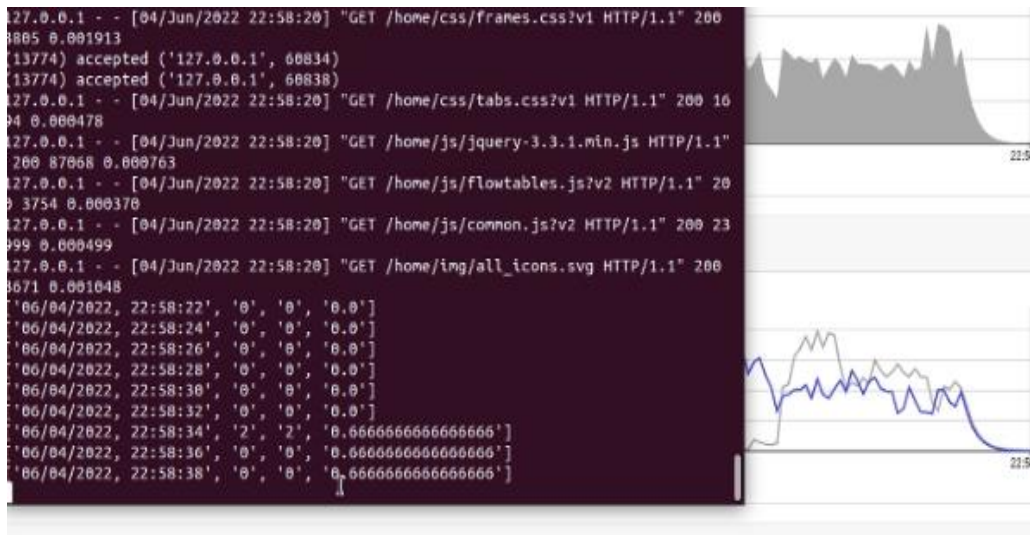


Figure 4-1.10 Normal Traffic Generation

Generally, the model is trained using for both normal and attack traffic. First normal traffic is generated and it's SFE and SSIP value is calculated and similarly RPE is calculated to differentiate weather the traffic is Normal or Attack.

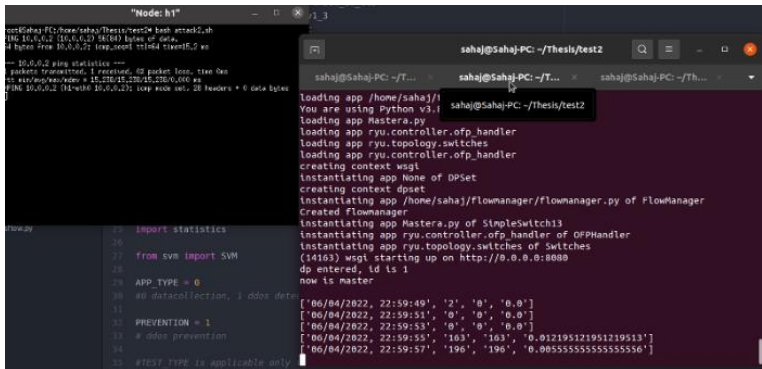


Figure 4-1.11 Attack Traffic Generation

Out of the multiple host, one host is assuming as an attacker and rest are normal host user. DDos attack is simulated using hping3 and iperf.

Controller is responsible to check the flow table periodically to insure that the packets are flooded or not. Similarly, each port of the OVS switch are also periodically observed for continuous packet monitor and flow control.

	PRIORITY	MATCH FIELDS	COOKIE	DURATION	IDLE TIMEOUT	HARD TIMEOUT	INSTRUCTIONS	PACKET COUNT	BYTE COUNT	FLAGS
<input type="checkbox"/>	65535	eth_dst = 01:80:c2:00:00:0e eth_type = 35020	0	19	0	0	OUTPUT:CONTROLLER	0	0	0
<input type="checkbox"/>	0	ANY	1	19	0	0	OUTPUT:CONTROLLER	14	1132	0

**Figure 4-1.12 Flow table for above Topology**

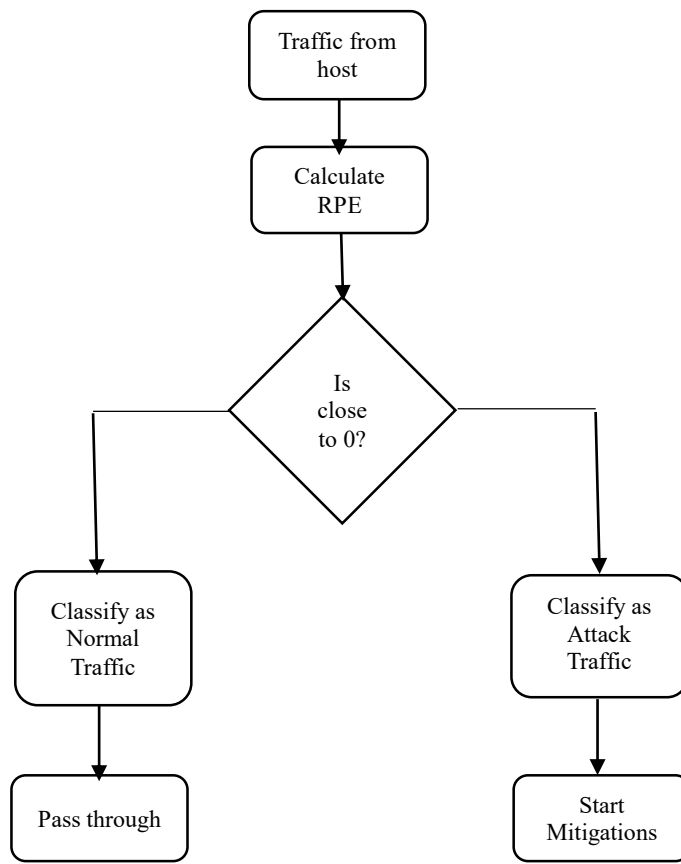
After collecting the normal traffic, the result along with its SSIP and SFE are stored for continuous comparisons. The dataset is passed to the algorithm and with the help of these data samples, data point can be classified based on similarity in the specific group of neighboring data points. The dataset generated is shown below.

The generated table consists of all the parameters that are used for the classification of traffic as normal and from the tabulated sample.

Time	Sfe	Ssip	RPE	Flowcount
03/31/2022, 12:25:01	7	4	1	7
03/31/2022, 12:25:06	6	2	1	13
03/31/2022, 12:25:11	6	3	1	19
03/31/2022, 12:25:16	8	1	1	27
03/31/2022, 12:26:21	177	176	0.011364	177
03/31/2022, 12:26:26	482	482	0.00304	659
03/31/2022, 12:26:31	374	374	0.001938	1033
03/31/2022, 12:26:36	326	326	0.001473	1359

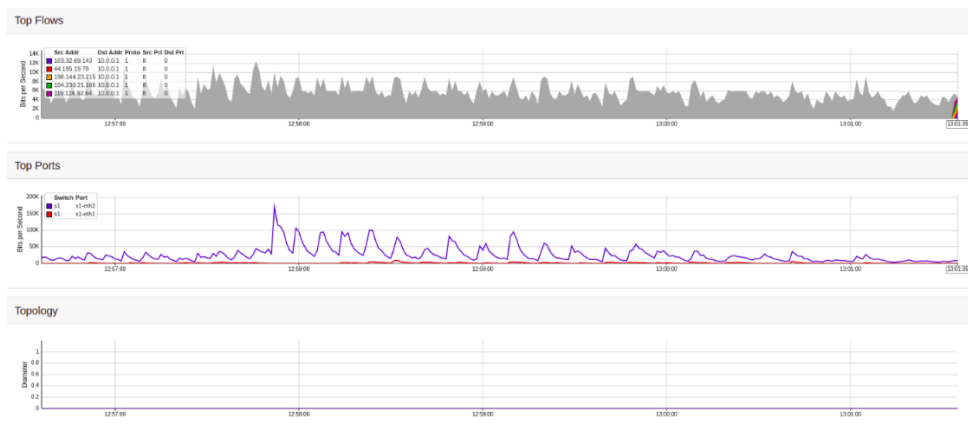
**Table 4.1.1 Training dataset generated in tabulated form for two switches**

## 4.2 Attack detection:



**Figure 4-2.1 Attack Detection**

Based on the training of the model, the model predicts the nature of incoming traffic based on various feature vectors mainly RPE and classifies it as either benign traffic or attack traffic based on which side of neighboring point the data lies on [24].



**Figure 4-2.2 Normal traffic Flow between 2 host**

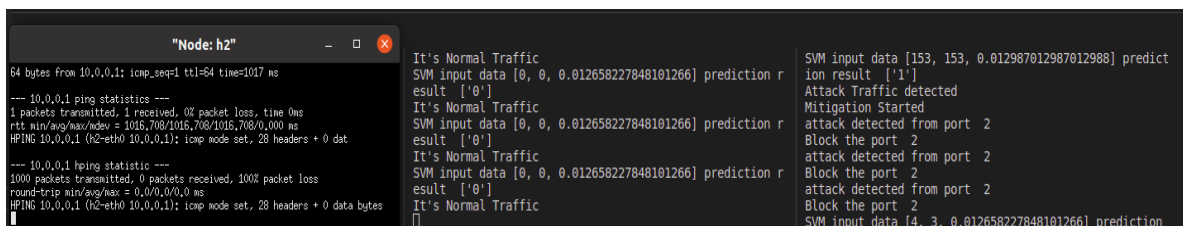


In case of normal traffic, the traffic towards the switches are easily pass through which can be done by analyzing the delay and timeout parameter obtained from SSIP, SFE and RPE. If a traffic from any host incoming to the host is found to have low SSIP as well as low SFE, the traffic is classified as normal traffic and it is allowed to pass through the network without any tempering. In case of the abnormal traffic, the traffic can be observed on flow table parameters [24].



**Figure 4-2.3 Attack traffic Flow from Host 1**

For example, a packet incoming to switch from host h1 connected to port1 of traffic has high SSIP, the point lies above the hyperplane generated from training of model and it is classified as attack traffic in case of SVM. For another layer of surety, another feature like Speed of flow entries is also plotted. If it also classifies the traffic as attack traffic, the host h1 is decided to be the attacker on the network.



**Figure 4-2.4 Attack traffic Flow from Host 1 in case of two Controllers**

The number of hosts were varied from 10 to 20 for multiple iteration of traffic generation. First the normal traffic was generated for 200 seconds while taking sample for every 2 seconds followed by attack traffic for 200 seconds taking sample for every 2 seconds. The process of normal traffic and attack traffic is repeated

multiple times. Since the traffic is Attack, the “RPE” value is close to 1 and the SVM classifier detects and interprets it as attack traffic.

The normal traffic is generated using ping command whereas the attack traffic is generated using hping3 and iperf depending on requirement of either automatic traffic or manual traffic. The traffic from both switch is collected and stored in csv file [24].

### **4.3 Overhead Handling:**

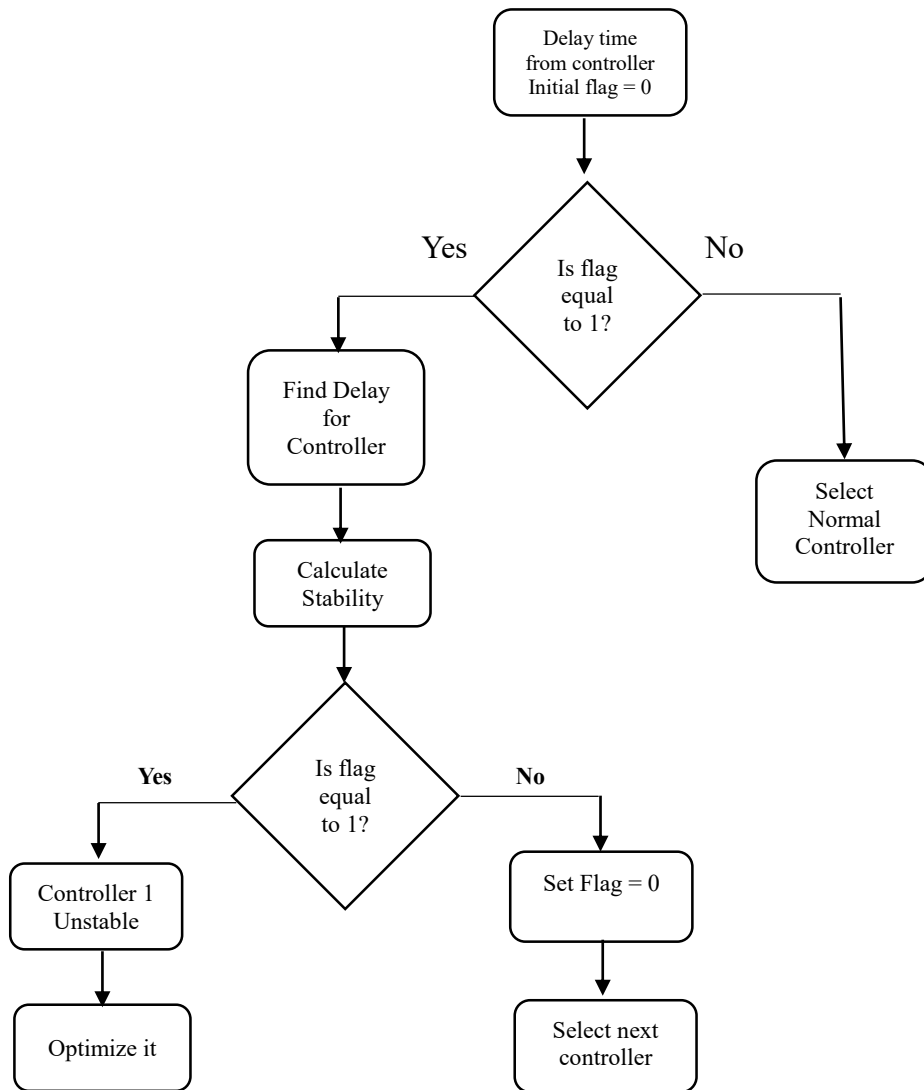
Even if the network is under attack, the controller can protect the hosts on the network by blocking the attacker as explained in earlier steps. But, if the controller is attacked, then the problem of overhead arises which is solved by using multi controller environment. In this project, the multi controller network has 2 independent controllers, one operating a main controller and another one being a standby secondary controller. If the controller is in normal operation, there is zero or no delay in receiving of the packet and the flag is set to 0. When an attack is performed on the controller, there is a certain delay in incoming traffic and the flag is set to 1 which is a sign of attack on controller. But as delay can occur due to other factors too, stability check is performed to confirm about the attack. Under attack scenario, the incoming traffic is extremely high compared to traffic processed by the controller due to the arrival of large amount of fake traffic. This result in the controller being unstable which is confirmation of attack on controller. Once the controller is on attack, the operation is transferred to the standby secondary controller from the main controller and main controller is terminated from the network for certain duration to protect the hosts in the network.

Overhead calculation is done on the basis of two aspect

#### **I. Controller Stability and Overhead analysis**

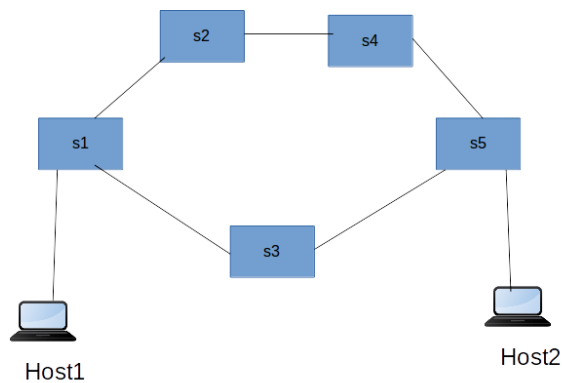
Overhead is calculated based on time period of flow of packers. A flag of set for every controller. Initially flag is set to 0. If flag is equal to 1 then the delay of the flow for particular controller is calculated. Similarly, the stability for all controller is determined. If Stability is equal to 1, the controller is assumed unstable and the whole node is optimized. For controller with stability not equal to 1. The flag is set to 0. The methodology elects the controller with highest performance as Master and loads are

transferred to slave controller if overhead is detected. If controller cannot control the overhead, the slave controller becomes master and so-on [25].



**Figure 4-3 Selection Methodology for overhead detection**

## II. Optimal Path detection and Selections



**Figure 4-4 Multipath routing Topology**

Above topology consist of two path.

Path1: H1-S1-S2-S4-A5-H2

Path2: H1-S3-S5-H2

When there is no flow, there will not be any table miss entry and flow table will have no Flows which can be verified by checking individual flow tables such that

#### For S1 Flow

```
cookie=0x0, duration=10.441s, table=0, n_packets=1, n_bytes=70, priority=10,in_port="s1-eth3" actions=group:50  
cookie=0x0, duration=10.441s, table=0, n_packets=6, n_bytes=531, priority=10,in_port="s1-eth1" actions=output:"s1-e  
th3"  
cookie=0x0, duration=10.441s, table=0, n_packets=5, n_bytes=424, priority=10,in_port="s1-eth2" actions=output:"s1-e  
th3"  
cookie=0x0, duration=10.441s, table=0, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
```

**Figure 4-5 S1 Flow Table**

Here n-packets value shows that no packets are flow through the individual port of the switch S1.

#### For S2 Flow

```
cookie=0x0, duration=18.237s, table=0, n_packets=17, n_bytes=1666, priority=1,in_port="s2-eth2",dl_src=00:  
00:02,dl_dst=00:00:00:00:00:01 actions=output:"s2-eth1"  
cookie=0x0, duration=17.255s, table=0, n_packets=17, n_bytes=1666, priority=1,in_port="s2-eth1",dl_src=00:  
00:01,dl_dst=00:00:00:00:00:02 actions=output:"s2-eth2"  
cookie=0x0, duration=61.729s, table=0, n_packets=20, n_bytes=1789, priority=0 actions=CONTROLLER:65535
```

**Figure 4-6 S2 Flow Table**

Here n-packets value shows that no packets are flow through the individual port of the switch S2.

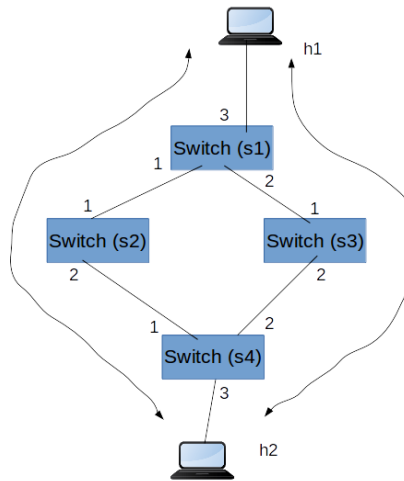
#### For S3 Flow

```
cookie=0x0, duration=21.883s, table=0, n_packets=1, n_bytes=42, priority=1,in_port="s3-eth2",dl_src=00:00:00:00:  
02,dl_dst=00:00:00:00:00:01 actions=output:"s3-eth1"  
cookie=0x0, duration=16.713s, table=0, n_packets=0, n_bytes=0, priority=1,in_port="s3-eth1",dl_src=00:00:00:00:00:  
01,dl_dst=00:00:00:00:00:02 actions=output:"s3-eth2"
```

**Figure 4-7 S3 Flow Table**

Here n-packets value shows that no packets are flow through the individual port of the switch S3.

Switch S4, S5 also follows the same case.



**Figure 4-8 Traffic flow between H1 and H2**

In absence of load balancing when a higher traffic is sent from Host 1 to Host 2, the packets are distributed to both paths.

```
mininet> h1 iperf -c h2 -b 1m -P 10 -t 30
-----
Client connecting to 192.168.1.2, TCP port 5001
TCP window size: 85.3 KByte (default)
-----
[ 3] local 192.168.1.1 port 53414 connected with 192.168.1.2 port 5001
[ 5] local 192.168.1.1 port 53418 connected with 192.168.1.2 port 5001
[ 4] local 192.168.1.1 port 53416 connected with 192.168.1.2 port 5001
[ 8] local 192.168.1.1 port 53424 connected with 192.168.1.2 port 5001
[ 7] local 192.168.1.1 port 53422 connected with 192.168.1.2 port 5001
[ 6] local 192.168.1.1 port 53420 connected with 192.168.1.2 port 5001
[ 9] local 192.168.1.1 port 53426 connected with 192.168.1.2 port 5001
[10] local 192.168.1.1 port 53428 connected with 192.168.1.2 port 5001
[11] local 192.168.1.1 port 53430 connected with 192.168.1.2 port 5001
[12] local 192.168.1.1 port 53432 connected with 192.168.1.2 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0-30.1 sec  3.75 MBytes 1.05 Mbits/sec
[ 7] 0.0-30.1 sec  3.75 MBytes 1.05 Mbits/sec
[ 6] 0.0-30.1 sec  3.75 MBytes 1.05 Mbits/sec
[10] 0.0-30.1 sec  3.62 MBytes 1.01 Mbits/sec
[11] 0.0-30.1 sec  3.62 MBytes 1.01 Mbits/sec
[12] 0.0-30.1 sec  3.62 MBytes 1.01 Mbits/sec
[ 5] 0.0-30.1 sec  3.75 MBytes 1.05 Mbits/sec
[ 4] 0.0-30.1 sec  3.75 MBytes 1.05 Mbits/sec
[ 8] 0.0-30.1 sec  3.75 MBytes 1.05 Mbits/sec
[ 9] 0.0-30.1 sec  3.75 MBytes 1.05 Mbits/sec
[SUM] 0.0-30.1 sec  37.1 MBytes 10.3 Mbits/sec
```

**Figure 4-9 Traffic Generations between H1 and H2**

Here, 10 Session packets with individual size of 1Mbits/sec is generated between two host. Since there are no proper load distributions, the packet

flows in all direction for the path between Host 1 and Host 2 which can be verified by checking flow tables of two paths i.e. via S2 and S3 switches.

### For S2 Flow

```
cookie=0x0, duration=59.612s, table=0, n_packets=57, n_bytes=5586, priority=1, in_port="s2-eth2", dl_src=00:00:00:00:00:02, dl_dst=00:00:00:00:00:01 actions=output:"s2-eth1"
cookie=0x0, duration=58.630s, table=0, n_packets=57, n_bytes=5586, priority=1, in_port="s2-eth1", dl_src=00:00:00:00:00:01, dl_dst=00:00:00:00:00:02 actions=output:"s2-eth2"
cookie=0x0, duration=103.104s, table=0, n_packets=20, n_bytes=1789, priority=0 actions=CONTROLLER:65535
```

**Figure 4-10 S2 Flow Table with no load balancing**

Here, in case of 10 sessions of packet generations, around 5000 bytes of data and 57 numbers of packets are passed through Switch S2 which can be seen on attributes n\_packets and n\_bytes.

### For S3 Flow

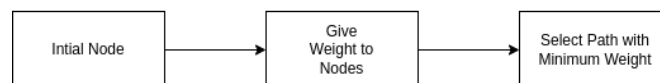
```
cookie=0x0, duration=96.351s, table=0, n_packets=6, n_bytes=252, priority=1, in_port="s3-eth2", dl_src=00:00:00:00:00:02, dl_dst=00:00:00:00:00:01 actions=output:"s3-eth1"
cookie=0x0, duration=91.181s, table=0, n_packets=5, n_bytes=210, priority=1, in_port="s3-eth1", dl_src=00:00:00:00:00:01, dl_dst=00:00:00:00:00:02 actions=output:"s3-eth2"
cookie=0x0, duration=139.827s, table=0, n_packets=27, n_bytes=2102, priority=0 actions=CONTROLLER:65535
```

**Figure 4-11 S3 Flow Table with no load balancing**

Here, in case of 10 sessions of packet generations, around 200 bytes of data and 30 numbers of packets are passed through Switch S3 which can be seen on attributes n\_packets and n\_bytes.

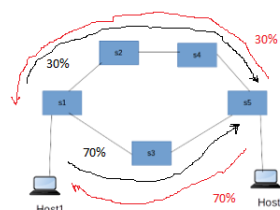
In absence of optimization of path, packet is passed through every possible connection.

## 1. Static Path Selections



**Figure 4-12 Multipath Static Routing**

In case of static load balancing, weights are assigned for each path and traffic flows are defined on the group table.

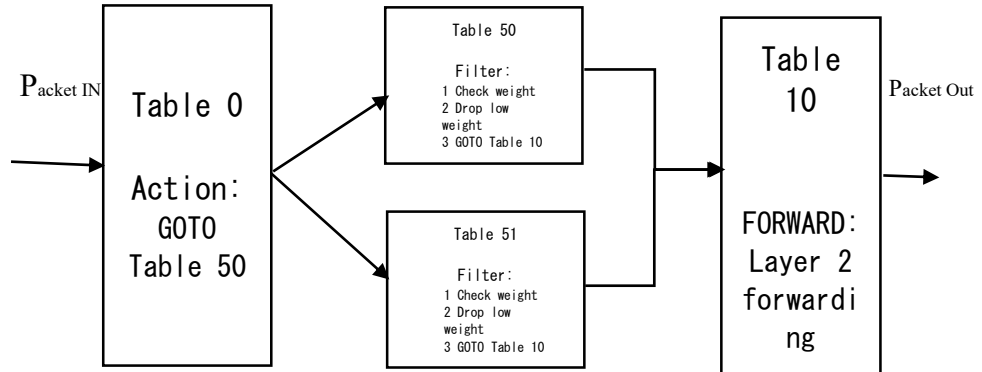


**Figure 4-13 Multipath Static Load Distributions**

```
[
  {
    "dpid": 1,
    "type": "SELECT",
    "group_id": 50,
    "buckets": [
      {
        "weight": 30,
        "actions": [
          {
            "type": "OUTPUT",
            "port": 1
          }
        ]
      },
      {
        "weight": 70,
        "actions": [
          {
            "type": "OUTPUT",
            "port": 2
          }
        ]
      }
    ]
  },
  {
    "dpid": 5,
    "type": "SELECT",
    "group_id": 51,
    "buckets": [
      {
        "weight": 30,
        "actions": [
          {
            "type": "OUTPUT",
            "port": 1
          }
        ]
      },
      {
        "weight": 70,
        "actions": [
          {
            "type": "OUTPUT",
            "port": 2
          }
        ]
      }
    ]
  }
]
```

**Figure 4-14 Multipath Weight Assignment**

Here weights are assigned in terms of percentage of packet distribution for individual ports of multi path point. Packet follows the path with higher weights however this only defined flow distributions. For packet to flow, flow should be defined in the group table.



**Figure 4-15 Multipath Flow Assignment Table**

Static Load balancing consist of two process

#### a. Load Assignment

In S1:

Make a group table with the type = SELECT so that one bucket will be chosen for each bucket (based on weight - vendor implementation). and processing will begin on the chosen bucket). Create two buckets. The



packet will be sent from one bucket to the S2 port and from another bucket to the S3 port.

In S3:

Make a group table with the type = SELECT so that one bucket will be chosen for each bucket (based on weight - vendor implementation). and processing will begin on the chosen bucket). Create two buckets. The packet will be sent from one bucket to the S4 port and another bucket to the S3 port. Proactively in S1 and S5, we provide group tables and flow

## b. Bucket Creation

Create two buckets for Host 1 and Host 2 define the load flow in case of multiple path selections.

Bucket 50:

- If Packet is from S2 to S1, send it through H1 (Path 1 for S1)
- If packet is from S3 to S1, send it through H1 (Path 2 for S1)

Bucket 51:

- If Packet is from S5 to S4, send it through H2
- If packet is from S5 to S3, sent it through H2

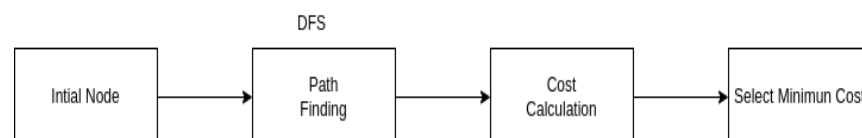
The path can be verified by check the flow table for table miss entry.

```
cookie=0x0, duration=241.650s, table=0, n packets=514, n bytes=33748, priority=1, in_port="s3-eth2", dl_src=00:00:00:00:00:02, dl_dst=00:00:00:00:00:01 actions=output:"s3-eth1"
cookie=0x0, duration=236.480s, table=0, n packets=518, n bytes=6587630, priority=1, in_port="s3-eth1", dl_src=00:00:00:00:00:01, dl_dst=00:00:00:00:00:02 actions=output:"s3-eth2"
cookie=0x0, duration=285.126s, table=0, n packets=33, n bytes=2596, priority=0 actions=CONTROLLER:65535
```

**Figure 4-16 Multipath Flow Assignment**

Here all the packets are flown through the S3 thereby balancing the load through the shortest path.

## 2. Dynamic Path Selections



**Figure 4-17 Multipath routing using DFS**

Multipath routing is a routing technique that looks for several pathways via a network structure to reach a target. It is possible to distribute network



traffic equitably across several paths in the network by offering numerous routes to a destination, or by a process known as load-balancing, which raises the effectiveness of network utility. Mutlipath algorithm in this thesis consist of three major approachs

### **A. DFS Path Finding**

In order to seek additional potential vertex positions in a graph, the Depth First Search (DFS) path discovery algorithm first locates the deepest vertex in the graph, then moves backward down the graph using a stack to find further potential vertex locations. The ability to use a stack in this method is a useful feature for multipath routing in particular since it enables modification of the algorithm to find every route that might connect two vertices.

### **B. Path Calculations**

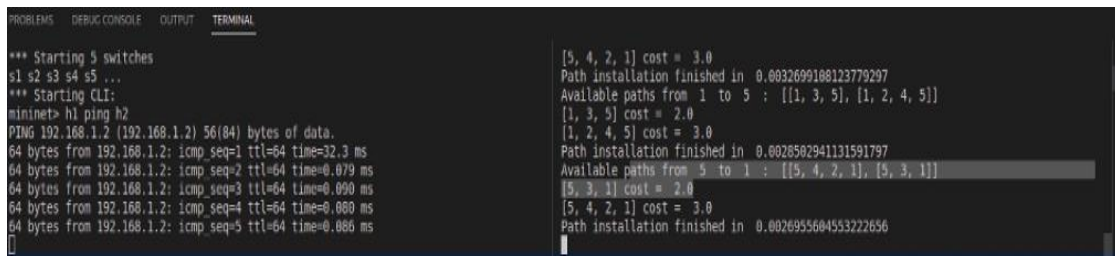
Here, 1 Gbps is taken as the reference bandwidth. An unweighted list of pathways is provided by the DFS algorithm. There is a simple method that can be used by:

- use OSPF to calculate the link cost.
- Add up the link costs for each and every link on the journey.

### **C. Install Flows**

Flow installation follow these procedures to accomplish this.

- Describe potential pathways from source to destination.
- Repeat the process for each switch that contains a path. List each switch port with a path.
- If a path is present on more than one switch port, install an (OFPGT SELECT) group table flow else just install a regular flow.
- In order to construct a group table, include the following information while creating buckets, which are collections of actions such as bucket weight refers to the weight of the bucket, watch port refers to the port to be monitored, watch group refers to additional group tables to be watched (not necessary), and actions refers to typical openflow action, such as the output port.



```
PROBLEMS  DEBUG CONSOLE  OUTPUT  TERMINAL

*** Starting 5 switches
s1 s2 s3 s4 s5 ...
*** Starting CLI:
mininet> h1 ping h2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data:
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=32.3 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.079 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=0.090 ms
64 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=0.080 ms
64 bytes from 192.168.1.2: icmp_seq=5 ttl=64 time=0.086 ms

[5, 4, 2, 1] cost = 3.0
Path installation finished in 0.0032699108123779297
Available paths from 1 to 5 : [[1, 3, 5], [1, 2, 4, 5]]
[1, 3, 5] cost = 2.0
[1, 2, 4, 5] cost = 3.0
Path installation finished in 0.0028502941131591797
Available paths from 5 to 1 : [[5, 4, 2, 1], [5, 3, 1]]
[5, 3, 1] cost = 2.0
[5, 4, 2, 1] cost = 3.0
Path installation finished in 0.0026955604553222656
```

**Figure 4-18 Multipath routing using DFS in programs**

In case of static weight assignment, Cost function can be used to determine the dynamic weight value based on the traffics for the network. DFS is use for determining the path of the network. For each node, cost functions can have evaluated based on reference bandwidth (vendor depended) and current traffic. The path with the minimum cost is assigned with the highest weight on the group table and is chosen the best optimum path.

Using the cost value from the cost function, the weight can be dynamically assigned based on cost for multi path.

## CHAPTER 5

### RESULTS

Similar to the Conceptual model, the results can also be classified into two parts. One is Training and testing the algorithm on NSL-KDD Data set and another part is to use the algorithm on simulated Environment. The results are calculated based on Accuracy, error, Precisions, recall and F1-measures where

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (5.1)$$

$$\alpha - \text{error} = \frac{FN}{FN+TP} \quad (5.2)$$

$$Precision = \frac{\sum TP}{\sum TP+FP} \quad (5.3)$$

$$Recall = \frac{\sum TP}{\sum TP+FN} \quad (5.4)$$

$$F - \text{Measure} = \frac{2*Precision*Recall}{Precision+Recall} \quad (5.5)$$

#### 5.1 Testing the Algorithm based on Simulated Environment

The normal traffic is generated using ping command whereas the attack traffic is generated using hping3 and iperf depending on requirement of either automatic traffic or manual traffic. The traffic from both switch is collected and stored in csv file. The dataset generated is shown below.

Time	Sfe	Ssip	Rfip	Flowcount
03/31/2022, 12:25:01	7	4	1	7
03/31/2022, 12:25:06	6	2	1	13
03/31/2022, 12:25:11	6	3	1	19
03/31/2022, 12:25:16	8	1	1	27
03/31/2022, 12:26:21	177	176	0.011364	177
03/31/2022, 12:26:26	482	482	0.00304	659

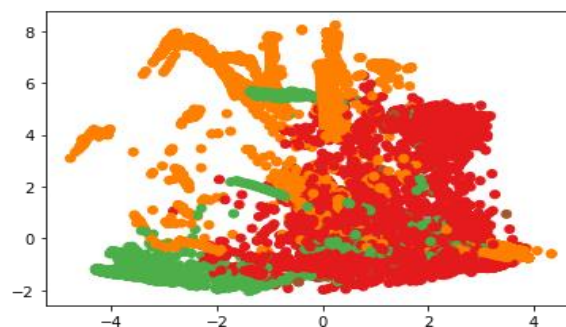
03/31/2022, 12:26:31	374	374	0.001938	1033
03/31/2022, 12:26:36	326	326	0.001473	1359
03/31/2022, 12:26:41	326	326	0.001188	1685
03/31/2022, 12:38:14	2	1	1	7
03/31/2022, 12:38:19	2	1	1	9
03/31/2022, 12:38:24	14	3	1	23

**Table 5-1 Training dataset generated in tabulated form for two switches**

The generated table consists of all the parameters that are used for the classification of traffic as normal and from the tabulated sample, the SVM model is trained for classification of real-time traffic. The distribution of feature vectors is seen to be skewed which estimates that there are outliers in the data and affects the performance of the statistical modeling. The skewness in data is handled by SVM. The distribution of data is seen as:

#### I. Using K means Clustering

As per the algorithm, the detection and training was initially carried using K means clustering. Since the clustering data was of varying size and density, K-means algorithm shows average result from predictions. For training the dataset, the total fields to be analyzes was upto 84 different field.



**Fig 5-1.1 Cluster of 84 fields using K mean cluster based on PCA on NSL-KDD**

As the number of dimension increase, the mean is distance between the features decreased. However, such problems can be avoided using Spectral clustering by

reducing the dimension size of features by using PCA, the accuracy of the algorithm decreases.

The accuracy for K means was calculated for the Attack labelled “Ddos” for K=4

```
acc = sklearn.metrics.accuracy_score(Y_Df_test,Y_Df_pred)
print(acc)

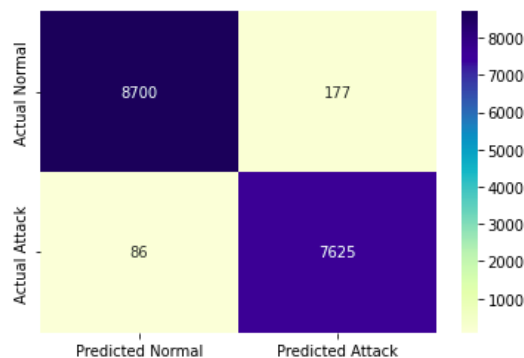
0.7764815471965933
```

**Fig 5-1.2 K mean Clustering Accuracy Calculations on NSL-KDD**

The Accuracy of K-mean Clustering for K=4 was found to be around 77%.

## II. Using Knn Algorithm

For training the data set, the total fields to be analyzes was up to 84 different field. The data set was initially divided into training and testing set. The accuracy for the logarithm was tested using KneighborsClassifier with metric as ‘minikwoski’ and neighboring value of 5.



**Fig 5-1.3 Confusion Matrix of KNN Algorithm on NSL-KDD**

The accuracy of Knn classifier was found to be the best however the precision and F-measure based on the train and test data was found to be weak.

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=5, p=2, metric='minkowski')
knn.fit(X_train_std, y_train)

print('The accuracy of the knn classifier is {:.2f} out of 1 on training data'.format(knn.score(X_train_std, y_train)))
print('The accuracy of the knn classifier is {:.2f} out of 1 on test data'.format(knn.score(X_test_std, y_test)))

The accuracy of the knn classifier is 0.99 out of 1 on training data
The accuracy of the knn classifier is 0.99 out of 1 on test data
```

**Fig 5-1.4 Accuracy of KNN Algorithm on NSL-KDD**

```

from sklearn.model_selection import cross_val_score
from sklearn import metrics
accuracy = cross_val_score(knn, X_test_std, y_test,
print("Accuracy for Knn: %0.5f (+/- %0.5f)" % (accu
precision = cross_val_score(kmeans, X_test_std, y_t
print("Precision for Knn: %0.5f (+/- %0.5f)" % (pre
recall = cross_val_score(kmeans, X_test_std, y_test
print("Recall for Knn: %0.5f (+/- %0.5f)" % (recall
f = cross_val_score(kmeans, X_test_std, y_test, cv=
print("F-measure for Knn: %0.5f (+/- %0.5f)" % (f.m

Accuracy for Knn: 0.98740 (+/- 0.01022)
/usr/local/lib/python3.7/dist-packages/sklearn/metr:
_warn_prf(average, modifier, msg_start, len(result
/usr/local/lib/python3.7/dist-packages/sklearn/metr:
_warn_prf(average, modifier, msg_start, len(result
/usr/local/lib/python3.7/dist-packages/sklearn/metr:
_warn_prf(average, modifier, msg_start, len(result
Precision for Knn: 0.38067 (+/- 0.55342)
Recall for Knn: 0.53732 (+/- 0.81974)
F-measure for Knn: 0.42514 (+/- 0.62018)

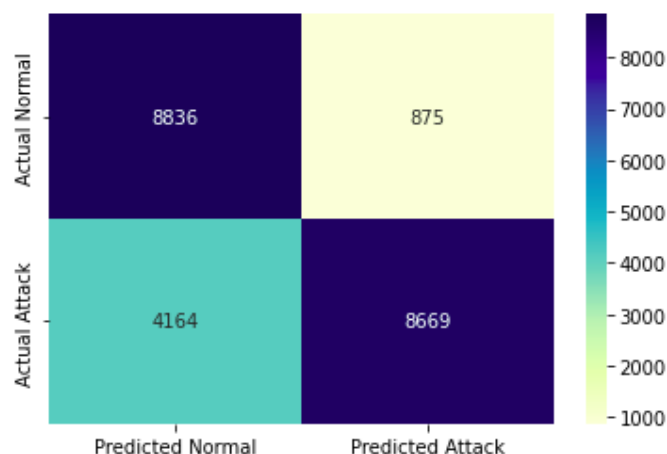
```

**Fig 5-1.5 Accuracy, Precision, Recall and F-measure of KNN Algorithm on NSL-KDD**

Since KNN model has the best accuracy but the precision and F-measures are found to be weak.

### III. Using SVM Classifier

For training the data set, the total fields to be analyzes was up to 84 different field. The data set was initially divided into training and testing set. The accuracy for the logarithm was tested using both kernel as ‘rbf as nonlinear” and Linear. For lower dimension, the Linear function had better accuracy and was linearly separable but as feature dimension increases, the difference in the space dimension starts to break and becomes more nonlinear. Hence RBG was use to map the complex boundaries.



**Fig 5-1.6 Confusion Matrix of SVM Classifier on NSL-KDD**

The accuracy of svm.SVC classifier was found to be the best as well as the precision and F-measure based on the train and test data was found to be best. So similar algorithm was chosen for classification between Normal and Attack traffic in case of simulation of real world data. However the imbalanced is due to the dataset having higher attack rate (55%).

```
from sklearn.model_selection import cross_val_score
from sklearn import metrics
accuracy = cross_val_score(clf_SVM_Df, X_Df_test,
print("Accuracy for SVM: %0.5f (+/- %0.5f)" % (ac
precision = cross_val_score(clf_SVM_Df, X_Df_test
print("Precision for SVM: %0.5f (+/- %0.5f)" % (p
recall = cross_val_score(clf_SVM_Df, X_Df_test, Y
print("Recall for SVM: %0.5f (+/- %0.5f)" % (reca
f = cross_val_score(clf_SVM_Df, X_Df_test, Y_Df_t
print("F-measure for SVM: %0.5f (+/- %0.5f)" % (f
print("train_time for SVM:%.3fs\n" %train1)
print("test_time for SVM:%.3fs\n" %test1)

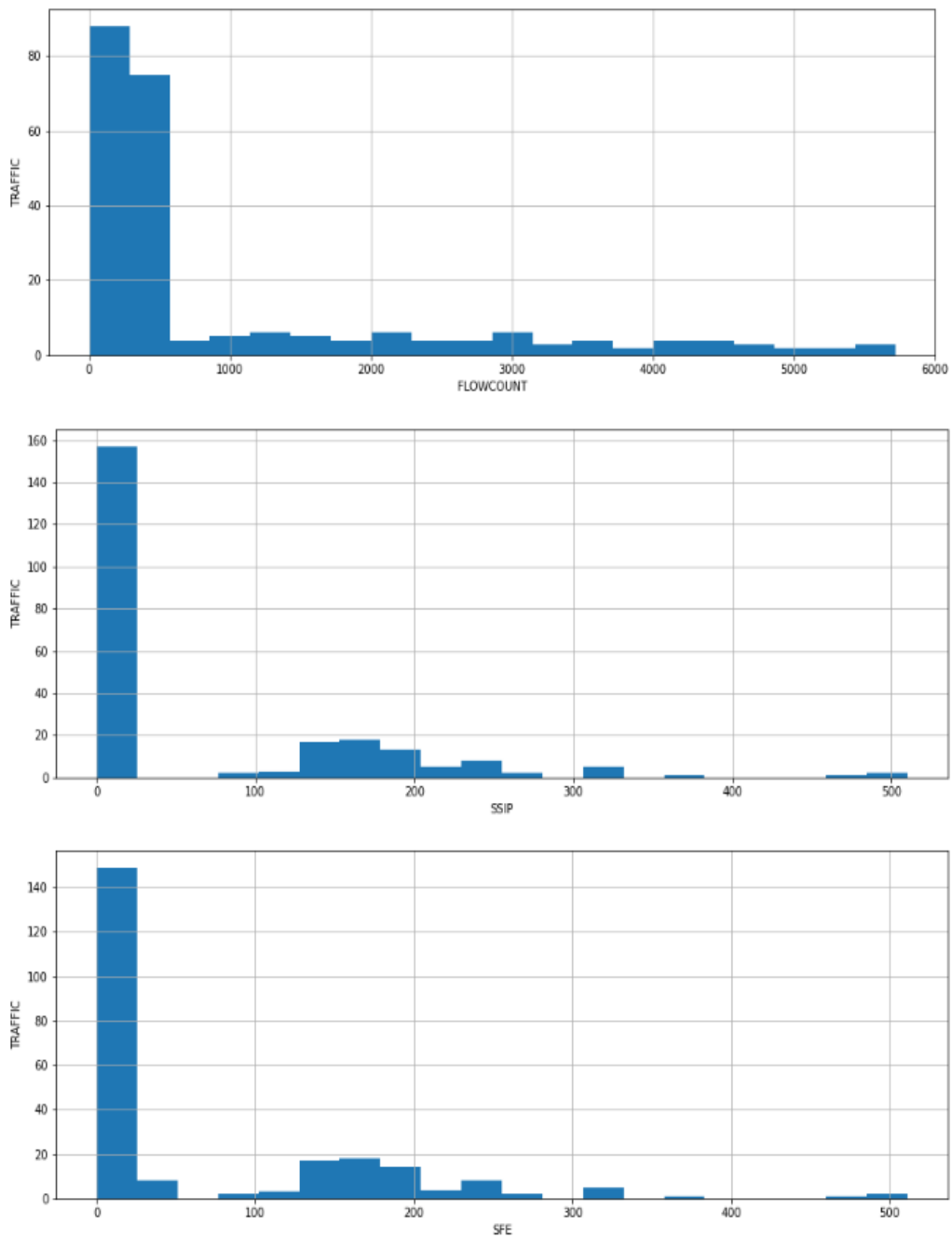
Accuracy for SVM: 0.96842 (+/- 0.00722)
Precision for SVM: 0.95891 (+/- 0.00982)
Recall for SVM: 0.98683 (+/- 0.00625)
F-measure for SVM: 0.97266 (+/- 0.00618)
train_time for SVM:279.161s

test_time for SVM:16.158s
```

**Fig 5-1.7 Accuracy, Precision, Recall and F-measure of SVM Classifier**

## **5.2 Training the Algorithm based on Simulated Environment**

Two ryu controllers and five openflow switches were used to create a network with hosts connected to both switches. The number of hosts per switches can be varied. The topology was created on Mininet based on python programming language.



**Figure 5-2.1 Distribution of traffic based on various feature vectors or parameters**

The distribution of feature vectors is seen to be skewed which estimates that there are outliers in the data and affects the performance of the statistical modeling. The skewness in data is handled by SVM. On basis of parameters the decision boundary with hyperplane is calculated by the process similar to that performed for single controller and single switch configuration.



### 5.3 Dynamic Path Selections

```

PROBLEMS  DEBUG CONSOLE  OUTPUT  TERMINAL

*** Starting 5 switches
s1 s2 s3 s4 s5 ...
*** Starting CLI:
mininet> h1 ping h2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data:
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=32.3 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.079 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=0.090 ms
64 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=0.080 ms
64 bytes from 192.168.1.2: icmp_seq=5 ttl=64 time=0.086 ms

[5, 4, 2, 1] cost = 3.0
Path installation finished in 0.0032699108123779297
Available paths from 1 to 5 : [[1, 3, 5], [1, 2, 4, 5]]
[1, 3, 5] cost = 2.0
[1, 2, 4, 5] cost = 3.0
Path installation finished in 0.0028502941131591797
Available paths from 5 to 1 : [[5, 4, 2, 1], [5, 3, 1]]
[5, 3, 1] cost = 2.0
[5, 4, 2, 1] cost = 3.0
Path installation finished in 0.0026955604553222656

```

**Figure 5-3.1 Multipath routing**

In case of static weight assignment, Cost function can be used to determine the dynamic weight value based on the traffics for the network. DFS is use for determining the path of the network. For each node, cost functions can have evaluated based on reference bandwidth (vendor depended) and current traffic. The path with the minimum cost is assigned with the highest weight on the group table and is chosen the best optimum path.

```

mahaj@mahaj-PC:~/Thesis/loadbalancing_selfsel$ sudo ovs-ofctl -O OpenFlow13 dump-flows s3
cookie=0x0, duration=27.385s, table=0, n_packets=62, n_bytes=3720, priority=65
S3S,d1_dst=01:00:c2:00:00:0e,d1_type=0x80cc actions=CONTROLLER:65535
cookie=0x0, duration=22.215s, table=0, n_packets=22, n_bytes=2156, tp,nw_src=192.168.1.1,nw_dst=192.168.1.2 actions=output:"s3-eth2"
cookie=0x0, duration=22.215s, table=0, n_packets=2, n_bytes=84, priority=1,arp,arp_spa=192.168.1.1,arp_tpa=192.168.1.2 actions=output:"s3-eth2"
cookie=0x0, duration=22.215s, table=0, n_packets=22, n_bytes=2156, tp,nw_src=192.168.1.2,nw_dst=192.168.1.1 actions=output:"s3-eth1"
cookie=0x0, duration=22.215s, table=0, n_packets=2, n_bytes=84, priority=1,arp,arp_spa=192.168.1.2,arp_tpa=192.168.1.1 actions=output:"s3-eth1"
cookie=0x0, duration=27.273s, table=0, n_packets=39, n_bytes=5362, priority=1,ipv6 actions=drop
cookie=0x0, duration=27.391s, table=0, n_packets=2, n_bytes=128, priority=0 actions=CONTROLLER:65535

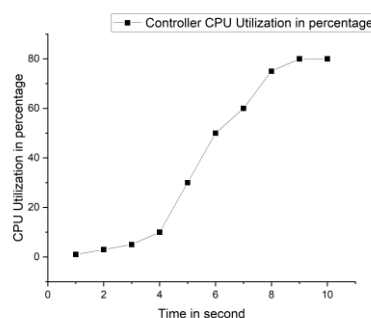
mahaj@mahaj-PC:~/Thesis/loadbalancing_selfsel$ sudo ovs-ofctl -O OpenFlow13 dump-flows s2
cookie=0x0, duration=34.991s, table=0, n_packets=79, n_bytes=4748, priority=65
S3S,d1_dst=01:00:c2:00:00:0e,d1_type=0x80cc actions=CONTROLLER:65535
cookie=0x0, duration=29.021s, table=0, n_packets=0, n_bytes=0, tp,nw_src=192.168.1.1,nw_dst=192.168.1.2 actions=output:"s2-eth2"
cookie=0x0, duration=29.021s, table=0, n_packets=0, n_bytes=0, priority=1,arp,arp_spa=192.168.1.1,arp_tpa=192.168.1.2 actions=output:"s2-eth2"
cookie=0x0, duration=29.021s, table=0, n_packets=0, n_bytes=0, tp,nw_src=192.168.1.2,nw_dst=192.168.1.1 actions=output:"s2-eth1"
cookie=0x0, duration=29.021s, table=0, n_packets=0, n_bytes=0, priority=1,arp,arp_spa=192.168.1.2,arp_tpa=192.168.1.1 actions=output:"s2-eth1"
cookie=0x0, duration=34.973s, table=0, n_packets=41, n_bytes=5606, priority=1,ipv6 actions=drop
cookie=0x0, duration=34.997s, table=0, n_packets=3, n_bytes=174, priority=0 actions=CONTROLLER:65535

```

**Figure 5-3.2 Dynamic weight assignment switch flow**

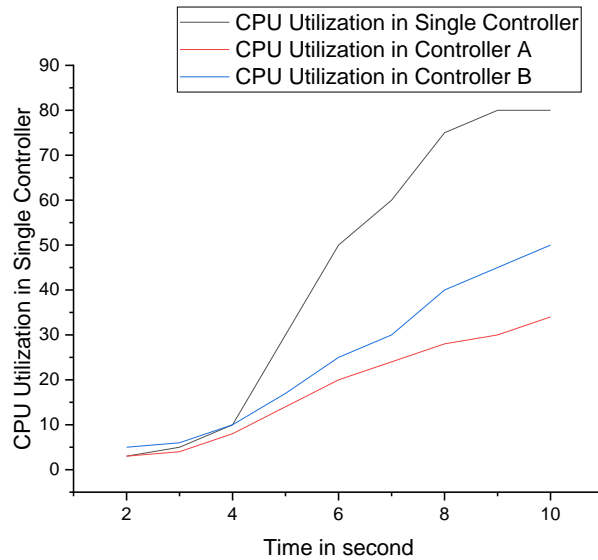
From the above flow table, the packet is transferred from path H1-S1-S3-S5-H2, as the packet on S2 is 0 and around 3000 in S3.

### 5.4 Controller Performance



**Figure 5-3.4 CPU Utilization for single controller**

In case of higher traffic, the complexity of controller also increases. From above fig, it is seen that as traffic increases, the CPU utilization of the controller also increases.



**Figure 5-3.4 CPU Utilization for multiple controller**

In case of higher traffic, the traffics are divided into two controllers which reduces the complexity and increases the performance of controller.

## CHAPTER 6

### DATA ANALYSIS AND REPORTING

All the topologies used in the project are analyzed for the accuracy in detection of attack traffic and also the decision rate is evaluated. The accuracy was detected for different kernel types with varying penalty parameter. The accuracy of the SVM model on data generated from this topology was calculated with linear kernel and penalty parameter set to 0.25.

Accuracy = 96.27%

The training set and testing set were split into ratio of 3:2 with respective accuracy of:

Training Set Accuracy = 96.50%

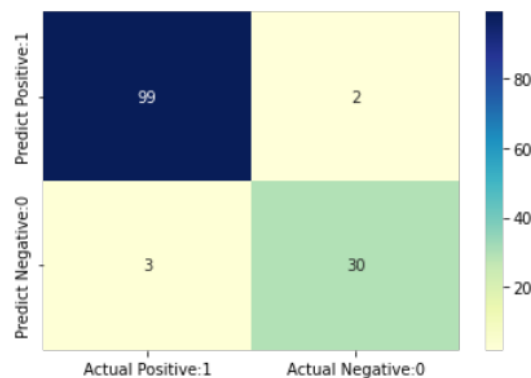
Test Set Accuracy = 96.27%

Since, two values are comparable, there is no chance of over fitting or under fitting.

Null accuracy = 75.37%

Since, the accuracy of model is better than null accuracy, the model is doing good job is classifying the labels.

The confusion matrix for the model is shown below:



**Figure 6-1: Confusion matrix using SVM Classifier**

	precision	recall	f1-score	support
0	0.97	1.00	0.98	86
1	1.00	0.94	0.97	50
accuracy			0.98	136
macro avg	0.98	0.97	0.98	136
weighted avg	0.98	0.98	0.98	136

**Figure 6-2 Precision, Recall and F1 score for above topology**

	SFE	SSIP	RFE	TYPE
count	642.00	642.00	642.00	642.00
mean	27.12	26.32	0.65	0.27
std	47.17	47.58	0.48	0.44
min	0.00	0.00	0.00	0.00
25%	0.00	0.00	0.00	0.00
50%	2.00	0.00	1.00	0.00
75%	40.00	40.00	1.00	1.00
max	197.00	197.00	1.00	1.00

**Figure 6-3 Variance and correlation between features and traffic**

The outcomes are:

True Positives(TP) = 99

True Negatives(TN) = 30

False Positives(FP) = 2

False Negatives(FN) = 3

Thus,

$$\text{Classification Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = 96.27 \%$$

Also,

$$\text{Classification Error} = \frac{FP + FN}{TP + TN + FP + FN} = 3.73 \%$$

$$\text{Precision} = \frac{TP}{TP + FP} = 98.02 \%$$

$$\text{True Positive Rate} = \frac{TP}{TP + FN} = 97.06 \%$$

True positive rate is also known as detection rate.

$$\text{False Positive Rate} = \frac{FP}{FP + TN} = 6.25 \%$$

## 6.1 Validation of the model

```
cm = confusion_matrix(y_test, y_pred_test)

print('Confusion matrix\n\n', cm)

print('\nTrue Negative(TN) = ', cm[0,0])

print('\nTrue Positive(TP) = ', cm[1,1])

print('\nFalse Negative(FN) = ', cm[0,1])

print('\nFalse Positive(FP) = ', cm[1,0])
```

Confusion matrix

```
[[85  3]
 [ 6 35]]
```

True Negative(TN) = 85

True Positive(TP) = 35

False Negative(FN) = 3

False Positive(FP) = 6

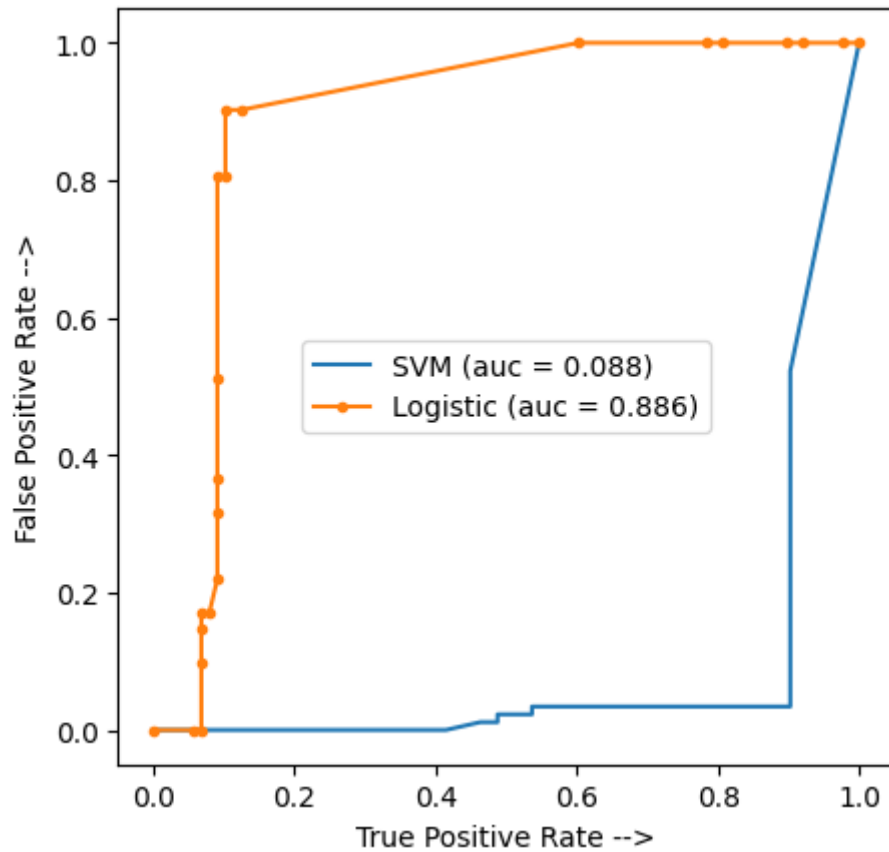
**Figure 6-4 Confusion matrix of dataset**

Above figure shows confusion matrix of the dataset using SVM algorithm.

	A	B	C	D	E	F	G	H	I	J	K
	Actual	Predict									
2	0	0									
3	0	0									
4	0	0									
5	0	0									
6	1	1							T	Y	N
7	1	1							F	35	6
8	1	1								3	85
9	0	0									
10	1	1								Y	N
11	1	1							T	TP	FP
12	1	1							F	FN	TN

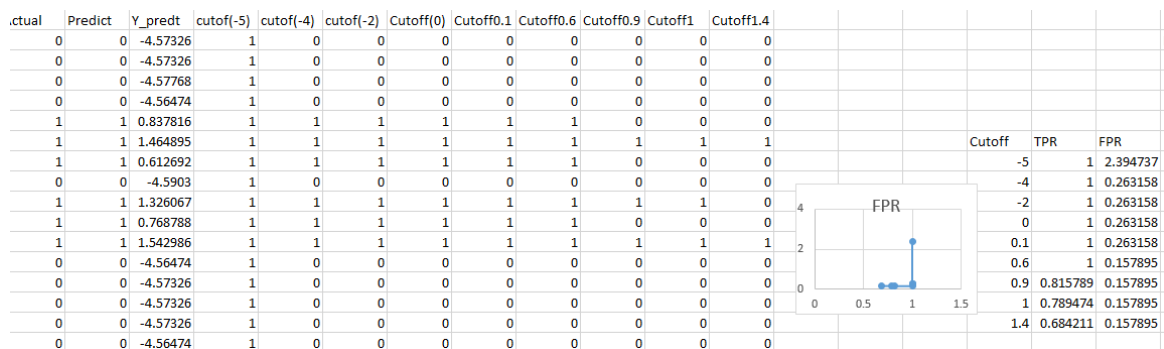
**Figure 6-5 Confusion matrix of dataset from excel**

Above figure shows confusion matrix of the dataset using SVM algorithm obtained from excel which appears to be same as obtained from the model above.



**Figure 6-6 AURUC of dataset**

Above figure shows AU-ROC of the dataset using SVM algorithm for above model.



**Figure 6-7 AURUC of dataset from excel**

Above figure shows AU-ROC of the dataset using SVM algorithm obtained from excel which appears to be same as obtained from the model above.

## 6.2 Path Selections:

$$pw1 = (s1-s2) + (s2-s4) + (s4-s5) = 3$$

$$pw2 = (s1-s3) + (s3-s5) = 2$$

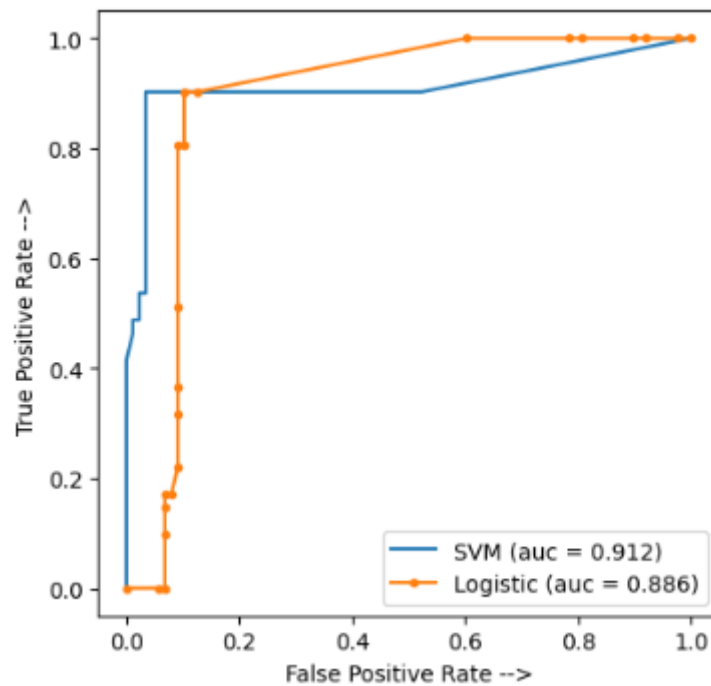
$$bw1 = (1 - \frac{3}{5}) * 100 = 40 \%$$

$$bw2 = (1 - \frac{2}{5}) * 100 = 60 \%$$

The shorter path (lower pw), is assigned a higher bucket weight.

## 6.3 Receiver operating characteristic curve and Area Under the ROC Curve

A ROC AUC of 1 indicates a flawless classifier, whereas a ROC AUC of 0.5 indicates a fully random classifier.



**Figure 6-8 ROC curve for two switches topology**

From the curve, ROC AUC = 0.94

The above AU-ROC represents that the for the accuracy below 91.2% the model will have close to 0 False positive Rate. If the accuracy of the model is increase from 91% the rate of false positive increases. k-fold cross-validation is a powerful

technique for assessing performance of the model. However, it fails in this case because we have an imbalanced dataset.

#### 6.4 Comparison with work performed by other authors and algorithm

While comparing the accuracy result of a single switch IPv4 topology to the results present in paper “Machine Learning Based Botnet Detection” [9],” A protocol for cluster confirmations of SDN controllers against DDoS attacks [2]”, the outcome of was found to be better than K means, Decision tree and K median while comparable to other methods. The accuracy of the model can be increased by training the model for longer period.

Model	Accuracy
Statistical Analysis + Linear SVM	97.79 %
J48 – decision tree	97.60 %
GA+GD	82.74 %
K median	76.17 %
Capacity Based Greedy Approach	80 %
Random Approach	81.36 %

**Table 6.1 Comparison of Algorithm Accuracy with other Author**

However, the comparison of algorithm between the data set and simulation environment may not be feasible due to

- Smaller Dataset which led to less training and testing time
- Delay between the packet flow
- Performance of the workstation



## CHAPTER 7

### RECOMMENDATIONS

The superiority of this algorithm is due to using relatively little strain on the controller CPU utilization cycle when compared to previous papers. In general, the study of Salah is used as the basis for calculating the average of the results of the simulated experiment, which is significant when the network traffic varies greatly [26]. Furthermore, CPU utilization on the Linux machine is monitored to assess the added effect of the DDoS attack mitigation method on the RYU controllers. If the network topology is divided into the access, aggregation, and core levels, this algorithm would be more effective and reliable defense against DDoS attacks in the access level. To deploy in aggregate and core levels, further features and customizations are necessary. One of the presumptions was that, when static IP configuration for routers is taken into account, DDoS attacks are carried out via IP spoofing. However, if the network is set with a non-static IP address, tracking the target source of the attacks is a difficult operation that which can be done on the future. Routers typically utilize a fixed IP address for a wide area network (WAN), and this form of detection is quite effective. Similarly, the DDos only work for the when the payload is fixed in nature. Similarly, there might be improvement in performance when the controller selection is based on an election algorithm. This algorithm is only feasible in homogenous environment. In case of heterogeneous environment, middleware is necessary.

## **CHAPTER 8**

### **CONCLUSIONS AND DISCUSSIONS**

This study presents an improved method for classification of higher and attack traffic, placement of controller, optimization of path link and controller in the SDN control plane. In order to prevent overload on some of the controllers, the requests (from the switches to controllers) are to be divided among them and thereby giving an intelligence on switch to control the traffic. In order to do this, control plane architecture with a master controller and slave controllers that are capable of carrying out independent load actions is used. As a result, it is possible to run the entire load balancing procedure concurrently and decrease the complexity of the time frame. Classification algorithm such as SVM is used to separate Ddos attack from higher traffic. Furthermore, higher traffics are handled by using multiple controller introducing master and slave configuration which reduces the chances of single controller failure. To further reduce the load on controller and improve the controller performance, optimum path is selected. In the near future, this algorithm could be paired with election algorithm to select the optimal master controller. Furthermore, this algorithm paired with election algorithm is intended to be used in both homogenous and heterogeneous controller environment for better accuracy.

## References

- [1] L. Yao, P. Hong and W. Zhou, "Evaluating the controller capacity in software defined networking," 2014 23rd International Conference on Computer Communication and Networks (ICCCN), 2014, pp. 1-6, doi: 10.1109/ICCCN.2014.6911857.
- [2] Ali J, Roh Bh, Lee S, "QoS improvement with an optimum controller selection for software-defined networks," 2019, PLOS ONE 14(5): e0217631, doi: 10.1371/journal.pone.0217631
- [3] Nam, Tran Manh et al. "Self-organizing map-based approaches in DDoS flooding detection using SDN." 2018 International Conference on Information Networking (ICOIN) (2018): 249-254.
- [4] Jagdeep Singh, Sunny Behal, Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions, Computer Science Review, Volume 37, 2020, 100279, ISSN 1574-0137, doi: 10.1016/j.cosrev.2020.100279.
- [5] P. Preamthaisong, A. Auyporntrakool, P. Aimtongkham, T. Sriwuttisap and C. So-In, "Enhanced DDoS Detection using Hybrid Genetic Algorithm and Decision Tree for SDN," 2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2019, pp. 152-157, doi: 10.1109/JCSSE.2019.8864216.
- [6] R. Braga, E. Mota and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," IEEE Local Computer Network Conference, 2010, pp. 408-415, doi: 10.1109/LCN.2010.5735752.
- [7] Harikrishna, P., Amuthan, A. SDN-based DDoS Attack Mitigation Scheme using Convolution Recursively Enhanced Self Organizing Maps. Sādhana 45, 104 (2020). Doi: 10.1007/s12046-020-01353-x
- [8] Karnwal, Tarun & Sivakumar, T. & Gnanasekaran, Aghila. (2012). A filter tree approach to protect cloud computing against XML DDoS and HTTP DDoS attack. Advances in Intelligent Systems and Computing. 182. 10.1109/SCEECS.2012.6184829.
- [9] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," 2015 International Conference on Computing, Networking and Communications (ICNC), 2015, pp. 77-81, doi: 10.1109/ICCNC.2015.7069319.
- [10] S. Dong and M. Sarem, "DDoS Attack Detection Method Based on Improved KNN with the Degree of DDoS Attack in Software-Defined Networks," in IEEE Access, vol. 8, pp. 5039-5048, 2020, doi: 10.1109/ACCESS.2019.2963077.

- [11] H. Peng, Z. Sun, X. Zhao, S. Tan and Z. Sun, "A Detection Method for Anomaly Flow in Software Defined Network," in *IEEE Access*, vol. 6, pp. 27809-27817, 2018, doi: 10.1109/ACCESS.2018.2839684.
- [12] Seungwon, Shin & Yegneswaran, Vinod & Porras, Phillip & Gu, Guofei. (2013). AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks. 10.1145/2508859.2516684.
- [13] Ambrosin, Moreno & Conti, Mauro & De Gaspari, Fabio & Poovendran, Radha. (2015). LineSwitch: Efficiently Managing Switch Flow in Software-Defined Networking while Effectively Tackling DoS Attacks. ASIACCS 2015 - Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security. 10.1145/2714576.2714612.
- [14] Iranmanesh, Amir & Naji, Hamid. (2021). A protocol for cluster confirmations of SDN controllers against DDoS attacks. *Computers & Electrical Engineering*. Volume 93. 10.1016/j.compeleceng.2021.107265.
- [15] Islam, Md & Estivill-Castro, Vladimir & Rahman, Md Anisur & Bossomaier, Terry. (2017). Combining K-Means and a Genetic Algorithm through a Novel Arrangement of Genetic Operators for High Quality Clustering. *Expert Systems with Applications*. 91. 402-417. 10.1016/j.eswa.2017.09.005.
- [16] Upendran, V., and R. Gopinath. "FEATURE SELECTION BASED ON MULTI-CRITERIA DECISION MAKING FOR INTRUSION DETECTION SYSTEM." (2021).
- [17] J. E. Varghese and B. Muniyal, "An Efficient IDS Framework for DDoS Attacks in SDN Environment," in *IEEE Access*, vol. 9, pp. 69680-69699, 2021, doi: 10.1109/ACCESS.2021.3078065.
- [18] Alzahrani, Abdulsalam & Alenazi, Mohammed. (2021). "Designing a Network Intrusion Detection System Based on Machine Learning for Software Defined Networks". *Future Internet*. 13. 111. 10.3390/fi13050111.
- [19] Sang-Hyun Choi and Hee-Su Chae. I. International Conference Data Mining, Civil and Mechanical Engineering (ICDMCME'2014), Feb 4-5, 2014 Bali (Indonesia). "Feature Selection for efficient Intrusion Detection using Attribute Ratio".
- [20] Cherubin, Giovanni and Chatzikokolakis, Konstantinos and Jaggi, Martin. "Exact Optimization of Conformal Predictors via Incremental and Decremental Learning". Creative Commons Attribution 4.0 International. doi.org/10.48550/arXiv.2102.03236

- [21] H. Peng, Z. Sun, X. Zhao, S. Tan and Z. Sun, "A Detection Method for Anomaly Flow in Software Defined Network," in *IEEE Access*, vol. 6, pp. 27809-27817, 2018, doi: 10.1109/ACCESS.2018.2839684.
- [22] Gao, Zhiqiang and Nirwan Ansari. "Differentiating Malicious DDoS Attack Traffic from Normal TCP Flows by Proactive Tests." *IEEE Communications Letters* 10 (2006): n. pag.
- [23] J. E. Varghese and B. Muniyal, "An Efficient IDS Framework for DDoS Attacks in SDN Environment," in *IEEE Access*, vol. 9, pp. 69680-69699, 2021, doi: 10.1109/ACCESS.2021.3078065.
- [24] Kumar Singh, Vishal (2020) DDOS attack detection and mitigation using statistical and machine learning methods in SDN. Master's thesis, Dublin, National College of Ireland.
- [25] Amir Iranmanesh, Hamid Reza Naji, A protocol for cluster confirmations of SDN controllers against DDoS attacks, *Computers & Electrical Engineering*, Volume 93, 2021, 107265, ISSN 0045-7906, <https://doi.org/10.1016/j.compeleceng.2021.107265>.
- [26] Sufiev, H.; Haddad, Y. A dynamic load balancing architecture for SDN. In *Proceedings of the IEEE International Conference on the Science of Electrical Engineering (ICSEE)*, Eilat, Israel, 16–18 November 2016; pp. 1–3.