



**Faculty of Computer Science and Information Technology**

**Machine Learning Based Two Phase Detection and Mitigation  
Authentication Scheme for Denial-of-Service attacks in Software Defined  
Networks**

**Najmun Nisa**

**Doctor of Philosophy  
2025**

# Machine Learning Based Two Phase Detection and Mitigation Authentication Scheme for Denial-of-Service Attacks in Software Defined Networks

Najmun Nisa

A thesis submitted

In fulfilment of the requirements for the degree of Doctor of Philosophy

(Network Technologies)

Faculty of Computer Science and Information Technology

UNIVERSITI MALAYSIA SARAWAK

2025

## DECLARATION

I declare that the work in this thesis was carried out in accordance with the regulations of Universiti Malaysia Sarawak. Except where due acknowledgements have been made, the work is that of the author alone. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.



.....

Signature

Name: Najmun Nisa

Matric No.: 17010035

Faculty of Computer Science and Information Technology

Universiti Malaysia Sarawak

Date: 21-07-25

## **DEDICATION**

To the Almighty ALLAH (SWT) and the Holy Prophet Muhammad (P.B.U.H)

&

It's dedicated to my whole family for their endless support. Special dedication to my all friend's for their great support and guidance. Finally, to my mentor Assoc. Prof. Dr. Adnan Shahid Khan who always motivated, guided, and encouraged me in achieving this milestone of life.

## ACKNOWLEDGEMENT

In the Name of Allah, the Most Gracious, the Most Merciful. All the praises and thanks be to Allah Almighty, for everything. Without Allah's blessings, I would not have been able to stand where I stand today. I find myself short of words to acknowledge His bounties upon me. I am extremely grateful to my supervisor, Assoc. Prof. Dr. Adnan Shahid Khan, for his invaluable advice, continuous support, and patience during my PhD study. His immense knowledge and plentiful experience have encouraged me in all the time of my academic research. I would also like to thank Prof. Dr. Sohail Asghar for his support and kind suggestions towards my PhD work. I would also like to thank my co-supervisor, Dr. Azman Bujang Masli for his support and sincere advice.

My most profound gratitude goes to my late father Raja M. Ziarab, my mother Fazeelat Begum, my brother's especially elder brother's Mazhar Raja and Azhar Raja, sister Saeeda Zakir and my entire family members for their sense of understanding, encouragement, support, and prayers. I can't forget to acknowledge the sacrifices of my kid Owais Raja and my husband Aamir Raja for his guidance, moral support and for standing by me during the difficult times of my PhD. I want to express my love for my special friends: Gulmina Rextina, Dr. Nusrat Shaheen, Numera Ismail, Amber Madeeha Zeb, Sajida Kalsoom, Hafiza Sundus Farooq for always being there to give moral support, duas, and sympathy.

My sincere gratitude to the Centre for Graduate Studies, for the guidance and support given during my period of study in Universiti Malaysia Sarawak.

Finally, I would like to thank the management of Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak for giving me opportunity to complete my study here in Sarawak. Thank you all.

## ABSTRACT

Computer networks are becoming increasingly important to the delivery of modern society's basic services. As a result of this growing dependence, traffic and users worldwide have also grown. Therefore, Software Defined Networking (SDN) is one of the solutions administrators are employing to handle this expansion. Traditional distributed networking is replaced with SDN's more programmable centralised solution, which is built around the SDN controller. This enables administrators to respond more rapidly to fluctuating network conditions. However, due to this flexible SDN architecture, it makes it vulnerable to different attacks such as Denial of Service (DoS)/Distributed Denial of Service (DDoS), Man in the Middle (MITM), spoofing and intrusion etc. Among all these, DoS attacks have the most severe impact of any of these types of attacks since they are able to overwhelm the most important parts of SDN, which in turn causes performance issues such as switch memory, CPU utilization and control channel bandwidth, which can result in slower speed, more dropped packets, and less accurate detection. Due to its centralized control plane architecture, SDN is vulnerable to DoS/DDoS attack, which can compromise the entire network. To improve SDN's resilience and performance, early detection, mitigation strategies, and structured approaches are necessary. In this research, Two-Phase Authentication of Attack Detection (TPAAD) scheme is proposed and investigated for detection and mitigation of DoS attacks in SDN to increase the performance of the above-mentioned issues. This scheme incorporates machine learning techniques by utilizing Support Vector Machine (SVM) and K-Nearest Neighbors (KNN) classification algorithms to accurately identify and handle malicious network traffic following the initial packet filtration process that identifies abnormal traffic. Through effective separation of benign from harmful packet flows, this method lessens the burden on the control plane while

preserving communication bandwidth. The results of the tests indicated increased performance in terms of fewer false positives, less CPU utilisation, less control channel bandwidth used, higher packet delivery ratio, and fewer flow requests made to the controller with 99.56% detection accuracy as compared to selected benchmarks. These findings demonstrate that the TPAAD scheme is more secure and effective than current approaches, operating efficiently and with little overhead, Specifically, it can accurately identify and mitigate DoS attacks in SDN.

**Keywords:** Denial of service, k-nearest neighbors, software defined networks, support vector machine, two-phase authentication of attack detection

***Pembelajaran Mesin Berdasarkan Pengesanan Dua Fasa dan Skim Pengesanan  
Tebatan untuk Rangkaian Ditakrifkan Perisian***

***ABSTRAK***

*Rangkaian komputer menjadi semakin penting untuk penyampaian perkhidmatan asas masyarakat moden. Akibat pergantungan yang semakin meningkat ini, trafik dan pengguna di seluruh dunia juga telah berkembang. Oleh itu, Software Defined Networking (SDN) ialah salah satu penyelesaian yang digunakan oleh pentadbir untuk mengendalikan pengembangan ini. Rangkaian teragih tradisional digantikan dengan penyelesaian terpusat SDN yang lebih boleh diprogramkan, yang dibina di sekeliling pengawal SDN. Ini membolehkan pentadbir bertindak balas dengan lebih pantas kepada keadaan rangkaian yang berubah-ubah. Walau bagaimanapun, disebabkan oleh seni bina SDN yang fleksibel ini, ia menjadikannya terdedah kepada serangan yang berbeza seperti Penafian Perkhidmatan (DoS)/Penolakan Perkhidmatan Tersebar (DDoS), Man in the Middle (MITM), penipuan dan pencerobohan dan lain-lain. Di antara semua ini, serangan DoS mempunyai impak yang paling teruk daripada mana-mana jenis serangan ini kerana ia dapat menyebabkan masalah kawalan dan penukaran SDN yang paling penting, yang mana pada bahagian yang paling penting untuk mengawal prestasi dan prestasi SDN. jalur lebar saluran, yang boleh mengakibatkan kelajuan yang lebih perlahan, lebih banyak paket yang tercicir dan pengesanan yang kurang tepat. Oleh kerana seni bina satah kawalan berpusatnya, SDN terdedah kepada serangan DoS/DDoS, yang boleh menjejaskan keseluruhan rangkaian. Untuk meningkatkan daya tahan dan prestasi SDN, pengesanan awal, strategi mitigasi dan pendekatan berstruktur adalah perlu. Dalam penyelidikan ini, skim Pengesanan Serangan Dua Fasa (TPAAD) dicadangkan dan disiasat untuk pengesanan dan mitigasi serangan DoS dalam SDN bagi meningkatkan prestasi isu-isu yang dinyatakan di atas. Skim ini menggabungkan teknik pembelajaran*

mesin dengan menggunakan algoritma klasifikasi Mesin Vektor Sokongan (SVM) dan K-Nearest Neighbors (KNN) untuk mengenal pasti dan mengendalikan trafik rangkaian berniat jahat dengan tepat berikutan proses penapisan paket awal yang mengenal pasti trafik abnormal. Melalui pengasingan berkesan aliran paket jinak daripada berbahaya, kaedah ini mengurangkan beban pada satah kawalan sambil mengekalkan lebar jalur komunikasi. Keputusan ujian menunjukkan peningkatan prestasi dari segi kurang positif palsu, kurang penggunaan CPU, kurang jalur lebar saluran kawalan yang digunakan, nisbah penghantaran paket yang lebih tinggi dan lebih sedikit permintaan aliran dibuat kepada pengawal dengan ketepatan pengesanan 99.56% berbanding dengan penanda aras yang dipilih. Penemuan ini menunjukkan bahawa skim TPAAD adalah lebih selamat dan berkesan daripada pendekatan semasa, beroperasi dengan cekap dan dengan sedikit overhead, Secara khusus, ia boleh mengenal pasti dan mengurangkan serangan DoS dengan tepat dalam SDN.

**Kata kunci:** Penafian perkhidmatan, jiran k-hampir, rangkaian yang ditentukan perisian, mesin vektor sokongan, pengesanan dua fasa pengesanan serangan

## TABLE OF CONTENTS

|                                            | Page |
|--------------------------------------------|------|
| <b>DECLARATION</b>                         | i    |
| <b>DEDICATION</b>                          | ii   |
| <b>ACKNOWLEDGEMENT</b>                     | iii  |
| <b>ABSTRACT</b>                            | iv   |
| <b><i>ABSTRAK</i></b>                      | vi   |
| <b>TABLE OF CONTENTS</b>                   | viii |
| <b>LIST OF TABLES</b>                      | xii  |
| <b>LIST OF FIGURES</b>                     | xiii |
| <b>CHAPTER 1 INTRODUCTION</b>              | 1    |
| 1.1 Introduction                           | 1    |
| 1.2 Research Background                    | 1    |
| 1.3 Problem Statement                      | 7    |
| 1.4 Research Objectives                    | 10   |
| 1.5 Scope of Research                      | 11   |
| 1.6 Research Significance and Contribution | 12   |
| 1.7 Thesis Organization                    | 14   |
| <b>CHAPTER 2 LITERATURE REVIEW</b>         | 15   |
| 2.1 Introduction                           | 15   |

|       |                                                                     |    |
|-------|---------------------------------------------------------------------|----|
| 2.2   | Security in Traditional Network                                     | 15 |
| 2.3   | SDN Architecture                                                    | 17 |
| 2.3.1 | Open Flow Protocol                                                  | 20 |
| 2.3.2 | SDN Security in Broader Network Security                            | 25 |
| 2.3.3 | SDN Geographical and Industrial Perspective                         | 26 |
| 2.4   | Security Challenges in SDN                                          | 27 |
| 2.4.1 | DoS/DDoS Attacks on the Control Plane                               | 31 |
| 2.4.2 | DoS/DDoS Attacks on the Data Plane                                  | 33 |
| 2.4.3 | DoS/DDoS Attacks on the Application Plane                           | 34 |
| 2.5   | Machine Learning in SDN                                             | 35 |
| 2.5.1 | Machine Learning Types                                              | 36 |
| 2.5.2 | Supervised Learning Algorithms                                      | 40 |
| 2.5.3 | Comparative Analysis of Machine Learning Techniques in SDN Security | 43 |
| 2.6   | Analysis on Current Approaches                                      | 44 |
| 2.6.1 | SECOD Approach                                                      | 44 |
| 2.6.2 | Flood Defender Approach                                             | 50 |
| 2.6.3 | DAISY Approach                                                      | 57 |
| 2.6.4 | Scalable Memory Based Approach                                      | 63 |
| 2.7   | Related Works                                                       | 66 |
| 2.8   | Summary                                                             | 74 |

|                                                                  |     |
|------------------------------------------------------------------|-----|
| <b>CHAPTER 3 METHODOLOGY</b>                                     | 87  |
| 3.1 Introduction                                                 | 87  |
| 3.2 Model Overview                                               | 89  |
| 3.3 Proposed System Architecture                                 | 90  |
| 3.3.1 Two Phase Authentication of Attack Detection (TPAAD)       | 91  |
| 3.3.2 Measure Performance using Confusion Matrix                 | 112 |
| 3.3.3 Mitigation Module                                          | 114 |
| 3.4 Simulation Setup                                             | 120 |
| 3.4.1 Tools used                                                 | 122 |
| 3.5 Evaluation Metrics                                           | 123 |
| 3.6 Summary                                                      | 124 |
| <b>CHAPTER 4 RESULTS ANALYSIS</b>                                | 126 |
| 4.1 Introduction                                                 | 126 |
| 4.2 Experimental Approach                                        | 126 |
| 4.2.1 Hardware and Operating System Configuration                | 126 |
| 4.2.2 Virtual Machine Environment Setup                          | 127 |
| 4.2.3 Experimental Setup                                         | 129 |
| 4.3 Performance Analysis and Comparison with Existing Techniques | 130 |
| 4.3.1 CPU Utilization                                            | 130 |
| 4.3.2 Control Channel Bandwidth                                  | 133 |

|                                                 |                                  |            |
|-------------------------------------------------|----------------------------------|------------|
| 4.3.3                                           | Packet-In messages               | 137        |
| 4.3.4                                           | Evaluation of Latency            | 141        |
| 4.3.5                                           | Attack Identification False Rate | 144        |
| 4.3.6                                           | Detection Accuracy               | 146        |
| 4.4                                             | Summary                          | 147        |
| <b>CHAPTER 5 CONCLUSION AND RECOMMENDATIONS</b> |                                  | 149        |
| 5.1                                             | Conclusion                       | 149        |
| 5.2                                             | Research Contributions           | 150        |
| 5.3                                             | Future Work                      | 151        |
| <b>REFERENCES</b>                               |                                  | <b>152</b> |

## LIST OF TABLES

|                                                                                           | <b>Page</b> |
|-------------------------------------------------------------------------------------------|-------------|
| Table 2.1 Machine Learning Types                                                          | 39          |
| Table 2.2 DAISY's Parameters                                                              | 59          |
| Table 2.3 Related Work Overview                                                           | 75          |
| Table 3.1 Classification Record                                                           | 102         |
| Table 3.2 Confusion Matrix for Two Class Scenario                                         | 113         |
| Table 4.1 Simulation Parameters for Proposed System                                       | 130         |
| Table 4.2 CPU Utilization difference in % between Normal and DoS Attack Traffic           | 132         |
| Table 4.3 Control Channel Bandwidth difference in % between Normal and DoS Attack Traffic | 134         |
| Table 4.4 Packet Delivery Ratio Difference (%)                                            | 140         |
| Table 4.5 Response Time Diff in % between Normal and DoS Attack Traffic                   | 144         |
| Table 4.6 Detection Accuracy                                                              | 146         |

## LIST OF FIGURES

|                                                            | <b>Page</b> |
|------------------------------------------------------------|-------------|
| Figure 1.1 Transformation from Traditional Network to SDN  | 3           |
| Figure 2.1 SDN Architecture                                | 18          |
| Figure 2.2 SDN Controller to OpenFlow Switch Communication | 22          |
| Figure 2.3 Possible SDN Attack Points                      | 31          |
| Figure 2.4 Workflow of SECOD algorithm (Wang et al., 2021) | 48          |
| Figure 2.5 FloodDefender Architecture (Gao et al., 2018)   | 55          |
| Figure 2.6 Working of DAISY (Imran et al., 2019)           | 61          |
| Figure 2.7 Flow Diagram of (Revathi et al., 2022)          | 64          |
| Figure 3.1 Flow Chart of TPAAD                             | 90          |
| Figure 3.2 Propose TPAAD Solution                          | 92          |
| Figure 3.3 Proposed Methodology for Classification         | 103         |
| Figure 3.4 Proposed System Visualtization Diagram          | 122         |
| Figure 4.1 Network Model for TPAAD                         | 129         |
| Figure 4.2 CPU utilization with normal traffic             | 131         |
| Figure 4.3 CPU utilization with DoS attack                 | 133         |
| Figure 4.4 Control Channel Bandwidth without DoS Attack    | 135         |
| Figure 4.5 Control Channel Bandwidth with DoS Attack       | 137         |
| Figure 4.6 Packet-In messages without DoS attack           | 138         |
| Figure 4.7 Packet-In messages with DoS Attack              | 140         |
| Figure 4.8 Response time without DoS Attack                | 142         |
| Figure 4.9 Response time with DoS Attack                   | 143         |
| Figure 4.10 Attack identification and false-positive rate  | 145         |
| Figure 4.11 Attack Detection Accuracy                      | 147         |

## **LIST OF ABBREVIATIONS**

|       |                                              |
|-------|----------------------------------------------|
| 5G    | Fifth Generation Cellular Communication      |
| API   | Application Programming Interface            |
| DA    | Discriminant Analysis                        |
| DCP   | User Datagram Protocol                       |
| DDoS  | Distributed Denial of Service                |
| DoS   | Denial of Service                            |
| DT    | Decision Tree                                |
| FNN   | Feedforward Neural Network                   |
| FRL   | Flow Request Limit                           |
| GLM   | Generalized Linear Model                     |
| IoT   | Internet of Things                           |
| KNN   | K-Nearest Neighbour                          |
| MITM  | Man in the Middle                            |
| MLP   | Multilayer Perception                        |
| NBI   | North Bound Interface                        |
| ONF   | Open Networking Foundation                   |
| RL    | Reinforcement Learning                       |
| SBI   | South Bound Interface                        |
| SDN   | Software Defined Networking                  |
| SL    | Supervised Learning                          |
| SSL   | Semi Supervised Learning                     |
| SVM   | Support Vector Machine                       |
| TCP   | Transmission Control Protocol                |
| TPAAD | Two Phase Authentication of Attack Detection |
| UL    | Un Supervised Learning                       |

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 Introduction**

Traditional networks have faced challenges in terms of scalability, adaptability, and efficiency, particularly in light of the rapid growth of data originating from cloud computing, IoT, and big data (Alashhab et al., 2022; Parashar et al., 2019). Network administration has been transformed by the introduction of Software Defined Networking (SDN), which centralizes control, enhances programmability, and fosters innovation. As a result of SDN's decoupling of the control and data planes, a central controller is now able to manage the network and choose intelligent routing strategies based on set network policies. This centrally administered policy-based method streamlines network administration and permits quick network service customisation (Jimenez et al., 2021; Xie et al., 2019; Zhang et al., 2018). Given this context, the first chapter draws attention to the unique security challenges posed by SDN-based networks, especially denial of service (DoS) and distributed denial of service (DDoS) attack. The problem to be researched is defined, along with the significance of this research. This chapter also discusses the objectives, scope, and contribution of this research. The research investigates the impact of DoS/DDoS attacks on SDN networks, and it focuses on detection and mitigation part at control plane.

### **1.2 Research Background**

Before the advent of SDN, it was difficult for data centres to deal with irregular patterns of data traffic due to the lack of predictability in the patterns. The increased resource requirements brought on by those data traffic patterns exceed the capabilities of legacy

network architectures. Cloud computing, the Internet of Things (IoT), and big data have all contributed to an increase in the amount of demand that is being placed on existing networks to provide high-performance, flexible, and scalable services (Masoudi et al., 2016). However, traditional networks are typically inflexible, with hardware configurations that are difficult to modify and limited options for programming. For example, additional devices may be required to scale the network or support a new routing protocol. Traditional devices have restricted programmability, allowing only the predefined capabilities provided by the vendor and preventing any modification or automation of network operations. SDN separates the control plane from the hardware, allowing for simple modification of network configurations. It is possible to manage and optimize the network in real time through programming without requiring hardware changes. This has led to a renewed focus on SDN as an alternative approach to network administration (Rahouti et al., 2022; Yan et al., 2015).

SDN's centralised, programmable, and dynamic approach to network management offers more efficient and adaptable routing, improved network visibility and control, and faster network innovation. SDN allows for a centralized controller to govern the network and make intelligent routing decisions based on network-wide policies by decoupling the control and data planes. By defining and pushing policies to all devices on the network from the controller, centralization also simplifies network administration (Alshamrani et al., 2017; Rafique et al., 2020; Zhao et al., 2019).

Major cloud providers like Google, Amazon, and Microsoft have all implemented SDN as it gains traction in the business and service provider networks. In 2011, the Open Networking Foundation (ONF) was founded to advance the use of SDN and provide open standards for related technologies. Moreover, the focus on safety, scalability, and network virtualization highlights their significance in SDN research (Lara et al., 2013).

In traditional network, devices such as switches and routers have been in charge of both the routing of traffic as well as managing the network operations that are performed by the network (Al Mtawa et al., 2021; Han et al., 2018; Urrea et al., 2021). A router, for instance, manages access control lists, decides the optimal way to send data packets among devices, and ensures that network traffic is appropriately divided using virtual local area networks. These capabilities are carefully incorporated into the router's hardware and software, therefore assigning the device the dual responsibility of traffic direction and execution of these supplementary network operations (Urrea et al., 2021). SDN, on the other hand, separates the network's control plane from the data plane, enabling network managers to manage the network centrally and programmatically as shown in Figure 1.1.

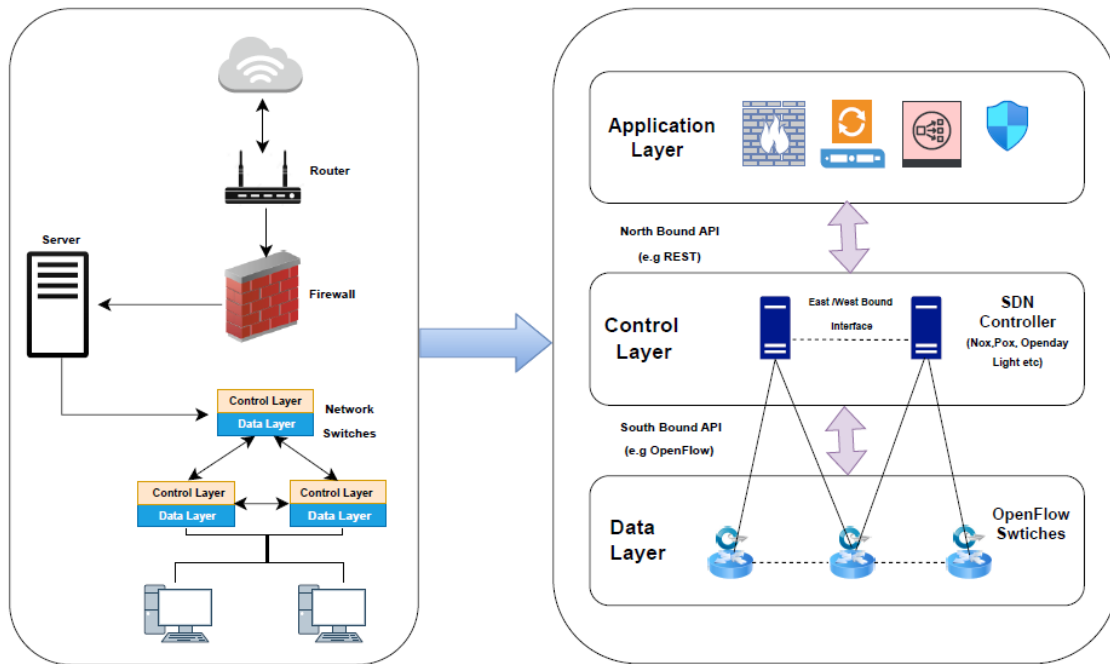


**Figure 1.1** Transformation from Traditional Network to SDN

The transition from a conventional network to a SDN requires a fundamental reconsideration of the management of network operations. Hardware components such as routers and switches in traditional networks serve as both the control plane and the data

plane, respectively as shown in Figure 1.1. The control plane determines the destination of incoming traffic, while the data plane actually transmits the data. This comprehensive integration renders the network inflexible, challenging to expand, and difficult to oversee as each device must be individually set up and controlled. By contrast, SDN separates the control plane from the data plane, consolidating network intelligence in a distinct controller that operates exclusively through software. The controller facilitates comprehensive network administration by enabling centralized and programmed control of network resources. Consequently, network managers are able to enforce policies, streamline traffic flows, and quickly deploy new services throughout the network with enhanced adaptability and effectiveness.

In SDN, the data plane and the control plane are separated from one another, unlike in conventional networks; the switch manages the data plane, while a distinct entity known as the controller is responsible for maintaining the control plane (Ahmad et al., 2015; Dargahi et al., 2017; Rawat et al., 2016). In SDN, managers have more control over the enforcement of network regulations includes security protocols as well as traffic administration rules by adjusting the application layer. This time- and resource-saving strategy significantly benefits the customization of network administration, such as tailoring security policies, optimizing traffic flows, and automating network management tasks like the IoT (Ahmad et al., 2022; Flauzac et al., 2015).

To better manage and control rapidly expanding networks, SDN removes the licencing cost as a restriction by allowing administrators to use their own programmes to customise network management above generic hardware (Shariff, 2021). Open and standardised interfaces, dynamic configuration, and centralised control are all features of SDN. It simplifies the creation, upkeep, and use of networks without exposing their inner

workings. Cost-effective and interoperable SDN-based devices that use a vendor-neutral southbound protocol, such as OpenFlow, are available; after analyzing network data, the SDN controller issues OpenFlow directives to the switches to manage traffic efficiently (Kreutz et al., 2014).

SDN is useful in many contexts, including data centres. To maximise bandwidth utilisation in data centres that deal with massive data and a wide variety of users, SDN provides dynamic routing based on real-time input. By centralizing the SDN, it becomes easier to transmit security policies to endpoints. The IoT relies on massive, dense networks to enable ubiquitous real-time broadcasts. SDN is being explored to enhance security, optimize traffic management, and improve scalability in IoT environments (Xu et al., 2017).

The integration of SDN has brought numerous benefits to organizations and industries, but it has also introduced security challenges, particularly in the form of DDoS/DoS attacks. These attacks can cause significant disruption and harm the normal functioning of SDN. Addressing this issue is critical for ensuring the security and reliability of SDN (Alsmadi et al., 2015; Dacier et al., 2017).

While SDN is enhancing network administration with more flexibility and centralized control, it is still vulnerable to DoS/DDoS attacks (Tian et al., 2019). Traditional approaches to network security, such as static rules and manual configurations, do not work with SDN systems. These methodologies are unable to adjust to the real-time identification and reaction to complex and evolving threats. Therefore, innovative responses to these problems are required. This thesis explores the use of machine learning methods to identify and mitigate SDN DoS/DDoS attacks. These techniques enable the learning from data, adaptation to emerging threats, and enhancement of SDN network resilience and protection. However, there are several challenges associated with detecting and mitigating DoS/DDoS

attacks in SDN. Firstly, the rapidly evolving nature of these attacks makes it difficult to develop effective defences that can keep up with the changing threat landscape (Gao et al., 2020). Secondly, the large volume of network traffic in SDN makes it challenging to accurately identify malicious traffic and distinguish it from legitimate traffic (Imran et al., 2019). Thirdly, the design and implementation of effective mitigation strategies is complicated by the centralized nature of SDN (Shu et al., 2016). Considering these challenges, this PhD thesis focuses on using machine learning techniques to address the security issues posed by DoS/DDoS attacks in SDN. The aim of the research is to develop a scheme that can effectively detect and mitigate the impact of these attacks, thereby improving the security of SDN. The results presented in this research aim to contribute to the field by addressing the ongoing security challenges faced by SDN and offering practical solutions for improving its security.

With SDN, the controller is centrally located and heavily relied upon, making it a single point of failure and an easy target for DoS/DDoS attacks. A DDoS attack is a frequent network security risk with the goal of consuming system resources to the point where the target is unable to provide its services. A DDoS attack on the SDN controller has the potential to severely disrupt network functionality, possibly resulting in a total network failure (Jazi et al., 2017). DoS/DDoS mitigation and detection in the SDN is a difficult problem. Analyzing network traffic with machine learning, which often responds more quickly and effectively than traditional rule-based approaches, could be a proactive effort at early identification of threats (Chuang et al., 2022). Using machine learning, a computer is taught to form inferences from available data without being given any instructions. This is a complementary technique for assisting network operators in understanding the present situation. In a variety of fields, from medical research to data mining, machine learning has

been used (Kelleher et al., 2020; Murphy, 2018). As such, this study aims to improve our understanding of how to identify and defend against DoS/DDoS attacks in an SDN environment.

### **1.3 Problem Statement**

There has been a recent uptick in the use of the SDN architecture. As a result of the separation of the data and control planes, networks can be managed more efficiently and with greater programmability. By implementing these design principles, we can expect to see a network that is more flexible, faster, and with control functions entirely managed by software. However, there are several security challenges in the SDN architecture that must be addressed. If attackers are able to exploit these challenges, it could compromise the security of the network and the services it offers. The results presented in this research aim to contribute to the field by addressing the ongoing security challenges faced by SDN and offering practical solutions for improving its security (Jimenez et al., 2021).

DoS/DDoS attacks can be easily executed against a system employing SDN architecture by flooding the OpenFlow switch with packets that have random destinations. As a result, each packet is forwarded from the OpenFlow switch to the controller in search of a revised flow rule. The impact of this attack on various components of the SDN architecture can be significant, as it depletes essential resources such as bandwidth, computing power, and memory. This resource exhaustion may lead to decreased performance, increased latency, or packet loss. While a DoS/DDoS attack in an SDN environment may not completely disable the network or its components, it can cause data delivery issues due to the overwhelming demand on network resources (Gao et al., 2020; Imran et al., 2019).

DoS attacks can range from simple to complex, targeting various network layers and exploiting vulnerabilities at different points in the SDN architecture. Because of this, they are widely recognized as among the most challenging forms of cyberattacks (Alzubi et al., 2023),(Al-Mafrachi, 2017). An attacker can disrupt a network by flooding hosts with packets, which uses up bandwidth, or by delivering a huge number of tiny packets to several hosts at once, which uses up the memory and computational capacity of the controller and switch. However, legitimate network activities, such as an authorized user repeatedly scanning the entire network or sending a large volume of data to multiple nodes, can resemble the patterns of an attack. This similarity makes it challenging to distinguish between normal user behavior and malicious activity (Tian et al., 2019). Since both scenarios involve network activity, it becomes challenging for mitigation methods to differentiate between an attacker and a legitimate user. DoS attacks in SDN can lead to several problems such as network congestion, service interruptions, higher latency, packet loss, and exhausted resources, therefore complicating the differentiation between normal and malicious traffic (Gao et al., 2020).

Administrators can create high-level policies which are transformed into flow rules implemented on switches through SDN, which centralizes network administration by removing control functions from the switches and relocating them to a central controller. Nevertheless, this design is susceptible to DoS attack which seek to interrupt network services and hinder authorized users from accessing resources (Imran et al., 2019). These attacks frequently overflow the network with a large number of packets, therefore overpowering the controller's capacity to handle incoming data. During periods of high demand, the controller stores incoming requests in a queue, which delays the implementation of new flow rules and results in packets without matching forwarding entries being trapped

in the switches. Consequently, the congestion might result in the loss of packets and cause delays in decision-making, therefore, intensifying the impact of the attack (Tang et al., 2021).

DoS attacks target the CPU and memory resources of the controller, therefore impeding its capacity to efficiently operate the network. A threatening actor can inundate the network with a large volume of traffic, therefore complicating the differentiation between authentic and malicious flows. Upon encountering a packet without a corresponding flow rule, a switch temporarily stores the packet in its buffer and transmits a "Packet In" message to the controller to request additional instructions. Under an attack, the controller is inundated with these requests, therefore impeding its ability to handle valid traffic and hence deteriorating network performance (Gao et al., 2020; Imran et al., 2019).

Moreover, the existing bandwidth connecting the controller and switches can be depleted by malicious traffic, resulting in additional deterioration of the network state (Imran et al., 2019). DoS attacks can generate a substantial volume of table-miss packets, overwhelm the flow table of a switch with nonsensical rules, and deplete the processing resources of the controller as it handles these malicious requests. At maximum memory use, the performance of a switch will go down, resulting in a decrease in processing speed and packet forwarding. Owing to inadequate resources and bandwidth, the switch may encounter difficulties in receiving valid packet-out messages from the controller, resulting in delays and compounding congestion (Gao et al., 2020).

An OpenFlow-based SDN system operates by having the controller in the control plane establish flow rules on the data plane switches to govern network operations. These rules may be implemented either proactively or reactively. The proactive technique involves the pre-installation of flow rules on switches to effectively handle incoming packets. Within

the reactive strategy, packets that do not correspond to a flow entry are transmitted to the controller, which subsequently modifies the flow table and sends the packet back to the switch (Gao et al., 2020). Nevertheless, in the event of a DoS attack, the controller's response time exhibits a substantial escalation owing to the excessive volume of fraudulent requests and packet-in messages, therefore reducing the available processing capacity for genuine flow requests (Al-Mafrachi, 2017; Bera et al., 2016; Gao et al., 2020; Imran et al., 2019; Tian et al., 2019).

DoS attacks on SDN are a significant challenge due to their centralized control plane architecture. These attacks can compromise the entire network, leading to performance degradation, latency, packet loss, and potential network downtime. Early detection and mitigation are crucial for maintaining network availability and security. Existing strategies often have limitations, such as additional hardware, delays in routing decisions, and increased computational overhead (Mousavi & St-Hilaire, 2018). To enhance the resilience and reliability of SDN architectures, structured approaches to assess and mitigate the impact of DoS attacks on the SDN control plane are needed. By refining detection methods and optimizing mitigation strategies, SDN systems can be safeguarded against these threats.

Since existing solutions aren't enough to secure the network from these complicated threats, this flaw emphasizes the necessity for additional study into creating efficient ways to identify and mitigate DoS attack in SDN environment.

#### **1.4 Research Objectives**

This research focuses on enhancing the security of the SDN control plane against DOS/DDoS attacks. A two-phase detection strategy is proposed comprising a queuing based approach to manage network traffic flow, followed by the deployment of machine learning

algorithms specifically designed for accurately detection of DoS attacks and then mitigating those attacks in SDN environments. Building on prior research, the focus is on formalizing current techniques that represent the SDN control plane as a network of queues capable of processing both normal and malicious traffic. To effectively distinguish between legitimate and malicious traffic, advanced SDN controllers and switches require enhanced protective measures. The following are the main objectives (RO) of this research.

- i. To design an efficient scheme for DoS attack detection based on ML approach. This will involve the design and implementation of new algorithms that can effectively analyze network traffic and identify anomalous behavior indicative of an attack.
- ii. To design an efficient scheme for DoS attack mitigation. This will involve the design and implementation of new algorithms that can effectively mitigate DoS attacks.
- iii. To evaluate the effectiveness of the proposed scheme through extensive experiments and this will involve the testing of the proposed approaches under different traffic loads to assess their accuracy and efficiency.

## **1.5 Scope of Research**

This research scope is relevant and significant in today's world. The scope of this research work is to detect and mitigate DoS attacks to enhance the security of SDN system using machine learning approach. This thesis examines the current state of SDN-targeted DoS attacks, focusing on their detection and mitigation methods. It identifies gaps in existing approaches and explores opportunities for improving these methods using ML. The research explicitly examines the tools and methods used in this system. The work focuses on implementing the TPAAD system. This system uses packet filtration procedures based on flow analysis and machine learning classification techniques, including SVM and KNN algorithms. The research is carried out in SDN environment established using Mininet and

the RYU controller, with trials executed on VMware Ubuntu. The scope of this research is determined by the dataset utilized (Jazi et al., 2017), the specific SVM and KNN techniques employed, and the metrics assessed, including false positives, CPU utilization, control channel bandwidth consumption, packet delivery ratio, and flow requests to the controller.

It is important to mention here that scope of research is limited to analysis of DoS attacks only. However, this report does not provide an examination of the impacts of DDoS, nor does it discuss measures for prevention. The scope is limited to publicly available CICDoS2017 data set. The thesis does not encompass additional forms of attacks or wider SDN environments beyond those that are represented in the testing configuration.

## **1.6 Research Significance and Contribution**

This research provides an efficient scheme that enhances accuracy and efficiency over conventional methods for detecting and mitigating DoS attacks, hence making a significant contribution to the understanding of integrating machine learning techniques into SDN networks. Furthermore, it offers real-world benefits to network administrators, service providers, and cybersecurity specialists by providing a verified machine learning-based method that optimizes the security of SDN.

The research presented focuses on the investigation of DoS attacks against newly emerging networking infrastructure that are based on SDN. The research's most significant contributions are outlined below.

A comprehensive analysis of recent efforts to model and prevent DoS attacks in SDN environments and how they may be implemented. This includes looking at issues specifically relating to both the data and the control layers. In order to detect and mitigate DoS attacks, researchers have been concentrating on traffic analysis, anomaly detection, flow-based defences, and adaptive resource allocation at the data plane. The efforts in the control plane

involve distributed architectures, access control methods, secure communication channels, and specialised security applications on the controller. In addition, statistical methods have been applied in order to understand the influence that DoS attacks have on the behaviour of network and to design effective tactics for mitigating the effects of these attacks. This comprehensive approach seeks to increase the security and resilience of SDN environments, protecting against DoS attacks and ensuring the network's reliability and security.

Design of unique Two-Phase Authentication of Attack Detection scheme which will examine incoming packets first through packet filtration module. The packets are initially put through this module to weed out any suspicious activity, and then pass towards classification models to efficiently detect whether the abnormal traffic is actually an attack or not. In addition, detection module also lowers the number of false alarms which generate after detection and it also makes attack detection more scalable, easier to install as no extra devices needed, and less likely to cause packet loss.

Design of mitigation module which will mitigate DoS attacks is based on Banned IP queue, which exclusively stops flooding packet from specific IP or MAC. Likewise, if an attacker's behaviour returns to normal after it has been abnormal, the attacker will be unblocked when a certain amount of time has passed.

TPAAD scheme is intended to enhance network performance for the detection and mitigation of DoS attacks. The TPAAD scheme makes use of dynamic, adaptive, and intelligent strategies to improve the network's capacity for DoS attack detection and successful mitigation. TPAAD can identify malicious traffic and identify unusual patterns by combining machine learning techniques. As a result, DoS attacks can be quickly and precisely identified, minimising false positives and the impact on reliable network performance. Additionally, TPAAD uses queuing technique like rate restriction and dynamic

traffic rerouting to lessen the effects of DoS attacks while maintaining network performance. The TPAAD scheme strikes a balance between security and performance, allowing effective network operation even under DoS attack situations. It does this by dynamically modifying security measures based on the growing attack scenario.

## **1.7 Thesis Organization**

This research is organized into five separate chapters. A brief overview of the research is presented in Chapter 1. The rational, challenges, significance, and unique contributions of the study are discussed in Chapter 2, we offer a comprehensive review of the research that has been done to detect and mitigate DoS attacks in networks that are based on SDN with and without machine learning. In Chapter 3, we propose algorithms to detect and mitigate DoS attacks against SDN. The overall architecture of the proposed system, along with its working procedure, is discussed in this section. In Chapter 4, the results of the proposed algorithm for detecting and mitigating DoS attacks in SDN networks are validated by comparing network performance with different classification techniques. Chapter 5 provides a conclusion of research with findings and suggestions for further research.

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 Introduction**

In recent years, there has been a growing focus on the subject of SDN network security. Research has concentrated on both securing SDN's control and data planes and utilizing SDN-based network infrastructures to deploy various security features and services. Accordingly, this chapter introduces SDN-based architectures and protocols, followed by a discussion of relevant security challenges. It also offers a comprehensive analysis of the literature on the detection and mitigation of DoS/DDoS attacks in SDN, with and without the use of machine learning approaches. This literature review aims to establish the current state of knowledge, identify gaps and limitations, and highlight the advancements made in protecting SDNs against DoS/DDoS attacks by examining the work of scholars and practitioners in the field. Following this, several detection and mitigation strategies put out in the literature are reviewed, with an emphasis on the potential use of machine learning algorithms might be used to enhance improve security measures.

#### **2.2 Security in Traditional Network**

As the Internet has grown and expanded, it has enabled a digital society in which any device, wherever in the world, may be accessed and linked to any other device, anytime. These advancements in technology and network infrastructure are leading to a rise in the size and complexity of traditional network infrastructures. Such large-scale systems and networks cannot be supported by today's technology. Hence, new computing technologies have been established to match the difficulties of traditional networks with new computing resources (Parashar et al., 2019).

In traditional networking, network switches contain control and data planes. Data plane routing tables transfer packets between interfaces (Dargahi et al., 2017). The control plane creates data plane routing tables from the network map if a packet's information is absent from the routing table. When the data plane sends packet information that does not fit any rules, the control plane updates the routing table (Masoudi et al., 2016).

This decentralized SDN system is highly effective in handling network traffic and demonstrates a minimal failure rate as a result of its redundancy and fault-tolerance capabilities. In the event of a switch failure, the remaining switches have the capability to identify the problem and promptly redirect traffic in order to preserve optimal network stability and uninterrupted operation. The early design of IP (Internet Protocol) networks, which served as the foundation of the modern internet, prioritized the qualities of durability and ability to withstand disturbances, especially when these disruptions were widespread. This concept of resilience is particularly evident in decentralized systems such as SDN, where a breakdown of one switch does not disable the network; rather, other switches can identify the problem and redirect traffic to preserve network stability. Implementing this design philosophy has been crucial in guaranteeing uninterrupted network operation even in unfavourable circumstances (Khan et al., 2022; Rahouti et al., 2022). However, when networks expand in size, this decentralized architecture encounters difficulties. Within extensive networks consisting of hundreds or even thousands of switches, no individual switch possesses a comprehensive perspective of the network structure. This limitation emerges due to restrictions in memory, computational capacity, and access bandwidth. Although it is possible to manage individual switches, the expense and intricacy of expanding these solutions to enormous networks might become excessively demanding. As

a result, while the SDN design is reliable and effective in smaller networks, it needs more thought and resources to scale well in bigger settings (Banton, 2020).

In (Dacier et al., 2017) author explores the benefits and pitfalls associated with the security of SDNs. SDN's centralization makes it easier to address policy inconsistencies than a conventional network, however, a hacked SDN controller may devastate the network. The open-source architecture and programmability of SDN allow network operators to write and execute unique rules to meet specific network management needs. Attackers possess the capability to reconfigure the entire network. The study conducted by Xing et al. (2014) examines the security and enhancement aspects of SDN that how the programmability and centralization featured in SDN enable the deployment of Intrusion Detection Systems (IDS), hence improving the network's capacity to identify and address security risks. Unlike traditional networks, SDN facilitates collaborative anomaly detection. In this system, switches have the ability to transmit their detection judgments to the controller, therefore enabling the network to make collective decisions by majority voting. Such feature enables SDN to construct adaptable and robust DoS/DDoS mitigation solutions. Nevertheless, despite these technological improvements, the controller continues to be a crucial weakness, as it is always susceptible to being attacked by attackers (Wang et al., 2022). SDN control planes may quickly collect flow-based information from all switches for real-time data analysis, unlike packet-based logs in conventional networks. Using an SDN application for data processing and routing management on the same controller simplifies network analysis.

### **2.3 SDN Architecture**

The SDN paradigm is undergoing a dramatic transformation in the telecommunications industry, and it has been widely accepted as an enabling technology in efforts like 5G and the internet of things (Syed-Yusof et al., 2020). More than twenty years

have passed since the introduction of SDN, but the technology is still developing and expanding to meet the growing number of demands in the tech industry for even more dynamic, scalable, and reliable SDN networks (Bahare et al., 2023).

According to Figure. 2.1, an SDN architecture comprises three layers: data layer, control layer, and application layer. Each layer plays a distinct role in the overall functioning of the SDN. Every layer has a unique function that contributes to the SDN's overall operation.

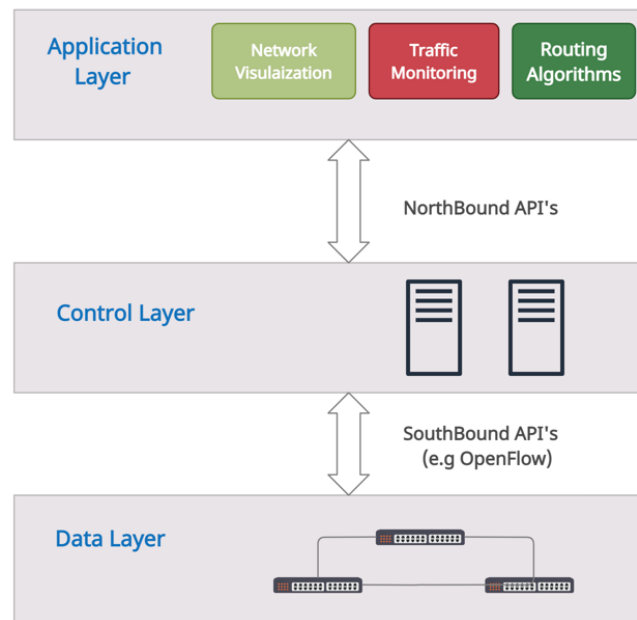


**Figure 2.1** SDN Architecture

There are multiple SDN switches at the data layer, all of which are linked together by wired or wireless media. Each switch is a very basic device with a forwarding table called the Flow Table that contains hundreds of rules needed to make forwarding decisions for network packets (Zhang et al., 2018). The Flow Table contains three fields for each rule item: an action, a counter, and a pattern. As a collection of values in the packet's header fields, the flow pattern is described by the pattern field. After receipt of a data packet, the

switch will look up the packet in its Flow Table for a matching rule. When the switch comes across a matching rule, it will increment the rule's counter and carry out the relevant action. If that's not the case, it will inform the controller for assistance or just drop the packet. It's important to remember that switch nodes don't come up with forwarding rule items on their own; rather, the controller pushes them down from the control layer (Han et al., 2018; Yan et al., 2015; Zhang et al., 2018).

The control layer acts as the SDN's central nervous system, directing and regulating traffic throughout the whole infrastructure. The SDN controller is the name for the network node that provides these capabilities and is often installed as a dedicated hardware appliance running specialised software. The SDN controller is able to see the full network architecture at the data forwarding layer (switches and connections) and interact with the switch through a standard south-bound API such as OpenFlow. The SDN controller hosts a variety of routing protocols, including Border Gateway Protocol (BGP) and Open Shortest Path First (OSPF), which are used to direct the data layer's forwarding activity. In SDN, OpenFlow was initially built with a single controller for simplicity's sake, however this creates a possible single point of failure. As a result, the scalability and availability of network resources are enhanced by practically all modern SDN architectural implementations, such as Floodlight (Nikitinskiy et al., 2015), NOX (Gude et al., 2008), and OpenDaylight (Arbettu et al., 2016; Rowshanrad et al., 2016). Each controller in the multi controller design is in charge of managing a subset of the total switches. As stated in (Kreutz et al., 2014; Shu et al., 2016), an SDN controller can talk to other controllers in the network using east-westbound APIs to keep the network's state consistent and to operate together.

When it comes to meeting the needs of their customers, network operators can pivot quickly thanks to the application layer. Smart application software has been developed to

run on top of SDN controllers to accommodate a wide range of application needs (Lara et al., 2013), including network virtualization, topology discovery, traffic monitoring, security improvement, load balancing, and more. With north-bound APIs like the REST API, the application layer can talk to the control layer. Since the control layer abstracts the network's physical resources for the application layer, network administrators may alter packets' data pathways through centralised software programming on the SDN controllers rather than manually configuring each of the data path's physical switches (Kreutz et al., 2014; Parashar et al., 2019; Zhao et al., 2019).

### **2.3.1 Open Flow Protocol**

In SDN, OpenFlow is a protocol that makes it possible for the control plane and the forwarding plane to communicate with one another (Keshari et al., 2021). In SDN, OpenFlow is a protocol that makes it possible for the control plane and the forwarding plane to communicate with one another (Keshari et al., 2021). The forwarding plane handles the actual data routing, while the control plane determines the traffic routing across the network. These two planes operate in tandem to ensure that the network runs smoothly. These two planes operate in tandem to ensure that the network runs smoothly (Liatifis et al., 2023).

With OpenFlow, the control plane is implemented in a separate controller that communicates with the forwarding plane via the OpenFlow protocol. This is possible because OpenFlow allows for multiple controllers. Because of this, centralised control of the network can be achieved, which makes the network simpler to administer and configure. OpenFlow can perform its function by breaking the process of packet forwarding into two distinct stages: the flow configuration stage and the data forwarding stage (Jimenez et al., 2021; Scott-Hayward et al., 2015) whole process is shown in Figure 2.2.

In the flow setup stage, the SDN controller is in charge of creating flow rules that specify the network's traffic routing. The flow rules provide the specific criteria, including source and destination IP addresses, port numbers, and transport protocols, that dictate the appropriate handling of data packets. When a packet is received by a switch during the data forwarding phase, the switch refers to its flow table to identify the appropriate flow rule that determines the manner in which the packet should be forwarded (Arif et al., 2020; Yang et al., 2020). The controller creates a new flow rule specifically for each packet that the switch forwards if there isn't a matching flow rule for it (Zhang et al., 2018).

SDN is an innovative architectural strategy for managing and controlling network traffic that allows for more adaptability and scalability. SDN facilitates this procedure by isolating the network's artificial intelligence and decision-making from the data-forwarding process by dissociating the control plane from the data (forwarding) plane. The control plane in conventional networks is included into individual network devices, such as routers and switches, to determine the routing protocols and destinations for traffic. In SDN architecture, the control plane is consolidated under an SDN controller accountable for managing the whole network. The centralization of this system enables more advanced and comprehensive decision-making, commonly known as the network's 'intelligence,' as the controller has a holistic perspective of the network. In contrast, the forwarding plane is located within the individual network devices and has the only responsibility of forwarding packets based on the predefined rules established by the controller. Implementing this division enables enhanced adaptability, dynamism, and effectiveness in network administration (Zhao et al., 2019). As a component of the control plane, the SDN controller independently controls network configurations and policies from a centralised location. This system utilizes the OpenFlow protocol, a universally accepted communication interface, to control and

configure the forwarding plane, encompassing routers and switches. Through the segregation of the control plane from the forwarding plane, SDN enables administrators to centrally establish and implement policies, hence enable dynamic control and programming of network behaviour. The flexibility of this system allows network operators to readily adjust their infrastructure to address changing traffic demands and evolving application requirements, hence enhancing agility and scalability in network management (Alsaeedi et al., 2019; Jia et al., 2019).

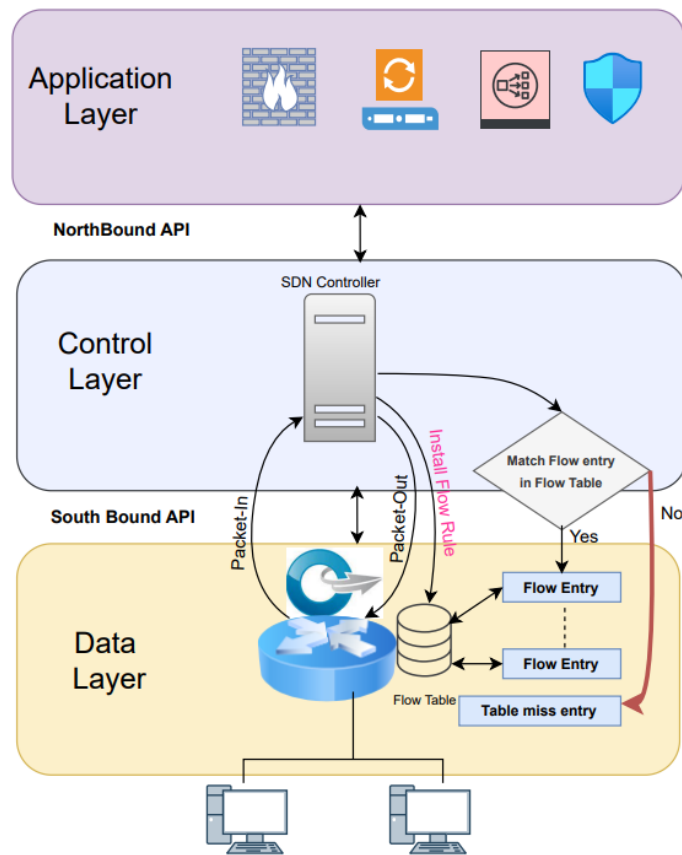


**Figure 2.2** SDN Controller to OpenFlow Switch Communication

SDN offers a novel architecture that decouples the data and control planes of a network. With this approach, the devices that really manage and forward the traffic are kept apart from the decision-making organizations that decide how to handle it (Ahmad et al., 2015; Masoudi et al., 2016; Sasaki et al., 2016). This is achieved by distinctly separating the

functions of the data plane, control plane, and application plane inside the SDN. The necessity to coordinate control activities across various dispersed systems has been eliminated because to this method's centralised approach to network management.

As the SDN architecture is built up in layers, a number of APIs have also been defined to communicate between those layers. The northbound interface (NBI) is one method via which apps can communicate with SDN administrators. Both user- and device-facing features are included in this interface. Like the northbound interface (NBI), the southbound interface (SBI) connects the SDN controllers and the switches. In order to facilitate data-path operations, such as packet encapsulation and decapsulation, as well as Quality-of-Service (QoS) support, the Southbound Interface (SBI) is designed to handle protocols at Layers 2, 3, and 4. The SBI supports several essential network protocols, such as Layer 2 protocols like Ethernet, Layer 3 protocols like IP (Internet Protocol), and Layer 4 protocols like TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). A detailed examination of the SBI is also presented in Figure 2.2. Here, the SBI transmits messages asynchronously to the SDN controller, indicating packet arrivals or changes in switch status as packets enter and exit the switches. In the meantime, the SDN controller updates the switch flow tables through the SBI to reflect the newly added or modified flows. OpenFlow is only one example of the variety of standardized protocols now available for use in SBI communications (Aziz et al., 2016; Nguyen et al., 2016; Singh et al., 2020).

Keep in mind that the ability to create flow rules is a crucial security measure, as it may be used to divert attacks from the main layers of SDN i.e controller or data plane as it includes switches and hosts. The chapter provides a more detailed explanation of how malicious users manipulate specific aspects of the SDN architecture in order avoid security measures. In particular, the SDN architecture is susceptible to a range of cyberattacks,

including infiltration, man-in-the-middle (MitM) attacks, DoS/DDoS attacks, and others that can target multiple SDN architecture layers (Adhikari et al., 2022; Li et al., 2017). In particular, DoS/DDoS attacks are a major cause for alarm because of the disaster they can impose on network availability and accessibility. User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) flooding attacks, amplification attacks, and database query attacks are all instances of such attacks (Kalkan et al., 2017). Different techniques have been developed, each providing different safeguards against DoS/DDoS attacks against SDN controllers. Various strategies for protecting the SDN controller have been explored, including the application of entropy-based techniques and classification algorithms to identify both high-volume and low-volume DDoS attacks by Banitalebi et al. (2021), a two-tier load balancing approach in SDN networks, encompassing both conventional load balancing among servers and load balancing among network devices by Belyaev et al., (2014), an attack detection trigger method that lessens the strain on controllers and switches and allows for a faster reaction to DDoS attacks. Furthermore, a neural network-based approach for detecting DDoS attacks has been developed to improve the precision of attack detection by Cui et al. (2016), flow table-based, statistical-based, and entropy-based methods for protecting the SDN controller have all been investigated in (Moazzeni et al., 2018; Tan et al., 2020; Wang, 2021; Zhang et al., 2016). However, these approaches only mitigate DoS/DDoS attacks on the control plane, thus it's vital to address data and application plane problems as well. These methods prioritize control plane DoS/DDoS minimization but often overlook data and application plane issues. Addressing these layers is necessary to defend the entire SDN infrastructure.

### **2.3.2 SDN Security in Broader Network Security**

Network security has always largely focused on protecting hardware-dependent, fixed systems. Management and protection of complex, distributed, and ever-changing network environments is gaining prominence in the current landscape of network security, making SDN security an essential component of this effort. To protect the perimeter, this entails putting security mechanisms including firewalls, IDS, and encryption at key points across the network into use. Still, the conventional concept of a network perimeter is changing as SDN divides the control plane from the data plane. By allowing the network to be controlled and programmed from a single location, SDN increases flexibility and efficiency in flow of network traffic control. The aforesaid change raises further security issues, mostly related to the SDN controller's more sensitivity. Should the SDN controller be compromised, the whole network may be seriously threatened by unauthorized access, data leaks, and DoS attacks. Consequently, in the whole network security architecture, the security of the SDN controller is of great relevance. Strong and thorough security policies must thus be put in place to protect this vital component from possible hazards (Chica et al., 2020; Pillai et al., 2024).

Apart from the specific shortcomings connected to SDN, its integration into the larger network security ecosystem has to be assessed in respect to forthcoming technologies including 5G networks and Network Function Virtualization (NFV). Together with SDN, these technologies provide more layers of possible vulnerabilities, hence complicating the security picture (Syed-Yusof et al., 2020). The natural programmability of SDN makes it possible to apply thorough and flexible policy execution. To properly fight the perpetually shifting threat environment, security strategies must, however, likewise be continuously adjusted. This is evident of a greater change in network security towards proactive and

intelligent security measures since the use of machine learning algorithms in SDN systems helps to quickly identify and handle such hazards. Still, this also raises questions about the dependability and comprehensibility of these algorithms as well as about the need of ongoing training and development to assure their effectiveness. Furthermore, as SDN develops it is imperative to make sure it fits regulatory and compliance requirements. This is making sure the dynamic rules made possible by SDN follow industry standards and legal frameworks. SDN security covers not only technical challenges but also more general strategic ones. It calls for thorough plans combining creative, flexible solutions with traditional security approaches to protect networks growing in complexity and interdependence (Chica et al., 2020; Pillai et al., 2024).

### **2.3.3 SDN Geographical and Industrial Perspective**

SDN evolves various industries by centralizing, flexible, and scaling network administration. SDN adoption depends on regional infrastructure and needs. Most SDN implementations are set up in Europe and North America, where large data centres and corporate networks are optimized. When used with cloud computing and IoT, it greatly increases responsiveness. However, SDN is helping the Asia-Pacific area install 5G networks faster and manage the expansion of its telecom's infrastructure more efficiently. At the same time, SDN is crucial in less developed countries like Latin America and Africa for overcoming infrastructure limitations. It streamlines network management at reduced cost and expands internet access to underserved areas (Syed-Yusof et al., 2020).

The applications of SDN are varied and can be found in a variety of industries. SDN ensures the safe and efficient management of healthcare networks, which include telemedicine, remote monitoring, and the transfer of massive volumes of data. SDN protects confidential financial data and reduces costs in the finance sector by improving security,

managing decentralized networks, and streamlining operations. SDN allows machines, sensors and different control systems to communicate instantly and continuously in the manufacturing industry for quality maintenance. SDN helps manage complex networks like 5G in telecommunications. For large-scale data sets, it optimizes the allocation of resources as well as efficiency of service (Sahran et al., 2023; Urrea et al., 2021).

## **2.4 Security Challenges in SDN**

Network security is being increasingly crucial in current society as it is now a fundamental determinant of the reliability of network technologies or applications. A network attack may cause system resources to run out, preventing authorized users from accessing pertinent services, or it may result in the theft of private information. An effective alternative is SDN, which improves network management, scalability, and adaptability by separating the control plane and forwarding plane. Modern SDN controller solutions such as Floodlight, NOX, etc. have effectively overcome certain challenges in traditional networks. However, the design and implementation of SDN provide novel obstacles, especially when networks expand in size (Chica et al., 2020). Performance optimization, control plane bottlenecks, and security vulnerabilities are becoming increasingly important, necessitating the implementation of advanced management techniques. There are numerous obstacles to overcome when designing the primary components of SDN. Some of these obstacles that the SDN must overcome include scalability, performance, and security concerns (Ahmad et al., 2015; Kreutz et al., 2014). SDN's are vulnerable to a variety of security threats, as shown in Figure 2.3. Based on these threats, SDN faces many challenges; some are listed below:

- One of the major security concerns with SDN is the network's dependency on a centralized controller at the control layer. If an attacker manages to get their hands on the controller, they will be able to reroute all network traffic and launch more network

attacks on board. For example an attacker who gains control of the SDN controller could redirect traffic to malicious servers, enabling data theft. Additionally, they could disable critical services by rerouting traffic away from essential network components, causing widespread disruption (Ahmad et al., 2015; Arif et al., 2020; Dacier et al., 2017).

- The use of SDN makes network segmentation possible at the control layer, which in turn permits the separation of various types of traffic, including data, voice, and video. However, this also introduces additional issues in terms of security, as the segments will need to be securely guarded in order to prevent unauthorized access. For example SDN divides a network into control layer segments, rendering each section vulnerable to unauthorized access. Insufficient access controls and encryption may allow attackers take advantage of deficiencies and acquire unauthorized access or penetration of various traffic segments, resulting in data breaches or sensitive information interception (Dargahi et al., 2017; Horvath et al., 2015).
- The data plane of a SDN is susceptible to eavesdropping attacks, in which an adversary intercepts and reads the traffic of a network. This can result in the unauthorized accessing of network resources or the theft of critical information. For example an attacker could intercept and understand unencrypted financial transactions in a company's internal systems, stealing secret financial data. The attacker could breach network security by intercepting device traffic and stealing login credentials (Iqbal et al., 2020).
- A side channel attack has the potential to target the network parts that are located on the data plane. These attacks use system implementation data rather than algorithm or software flaws. In network security, such attacks might target the data plane by measuring response times or other network component characteristics. For example, attackers could use packet processing timing delays to determine if a flow rule is

installed, revealing network traffic patterns. Then, a targeted DoS/DDoS attack on important network flows might be planned to use this knowledge (Ahmad et al., 2015; Kreutz et al., 2014).

- Attackers can implant malicious flows onto an SDN network via switch firmware defects or falsified control messages, causing packet loss, route redirection, and network congestion. An example, of this would be an attacker redirecting traffic to a hostile server, which would result in the data being intercepted (Li et al., 2017).
- SDN is susceptible to DoS attacks, in which an excessive amount of traffic can overwhelm and make the network unreachable. For example, a DoS attack can overload the controller excessive requests, therefore impeding its capacity to effectively administer the network (Ahmad et al., 2015; Rahouti et al., 2022).
- The application plane is the target of this particular app manipulation attack. An application vulnerability might be exploited, which could result in a malfunction, an interruption of service, or the interception of data. It is possible for an attacker to obtain access to an SDN application with high privileges and then carry out unauthorized actions. For example, attackers could stop security alarms in a network monitoring application to hide harmful activity. With high-level access to a traffic control tool, an attacker may divert traffic, disrupting services or stealing critical data (Valdovinos et al., 2021).
- A MitM attack occurs when an adversary inserts themselves into a communication channel between two parties for the purposes of eavesdropping, tampering with, or injecting data without either side being aware of the attack. Since SDN centralizes network administration and management, MitM attacks have the potential to do very severe damage. For example, an attacker might intercept SDN controller-switch

conversations and change flow rules to reroute traffic to harmful destinations. Additionally, the attacker may change data in transit to introduce fake information into the network, causing broad interruptions and data integrity issues (Sahu et al., 2023).

- There is a possibility that the APIs of a software component include vulnerabilities that might enable an attacker to undertake information disclosure without authorization. Exploitation of the API can take place at the northbound interface as well, which can result in the disruption of network flows. For example, an attacker could access sensitive configuration data via a northbound API flaw, disclosing important network characteristics. In addition, altering the API could change traffic flow regulations, disrupting network operations and perhaps triggering service failures (Singh et al., 2020; Yan et al., 2017).

The Figure 2.3 illustrates specific security threats that may occur in an SDN system. Fictitious traffic flowing between switches (1 in Figure 2.3) has the potential to overburden network resources, resulting in deterioration or disruptions. The presence of vulnerabilities in individual switches (2 in Figure 2.3), such as holes in firmware, might be leveraged to illicitly attain access or disrupt operations. The communication link between the SDN controller and switches (3 in Figure 2.3) is crucial; any attacks in this area can interrupt network control and affect the functionality of the switches. The SDN controller, specifically threat 4, is a very vulnerable target, as the exploitation of security vulnerabilities could enable attackers to exert control over the entire network. The presence of vulnerabilities in the interface between the controller and administrative applications (5 in Figure 2.3) can result in unauthorised modifications to network policies. Vulnerabilities on administrative stations (6 in Figure 2.3) might potentially provide attackers with administrative privileges to the SDN controller. Applications operating on the SDN controller, such as MAC learning

and intrusion detection systems (7 in Figure 2.3), are highly vulnerable to injected malicious software. Lastly, inter-controller communication (8 in Figure 2.3) is susceptible to attacks that may result in network management inconsistencies, performance degradation, or even network segmentation. Therefore, it is imperative to protect every level of the SDN architecture against these risks.

Using the security analysis and SDN architecture outlined above, we will explore how different challenges affects the three SDN layers.

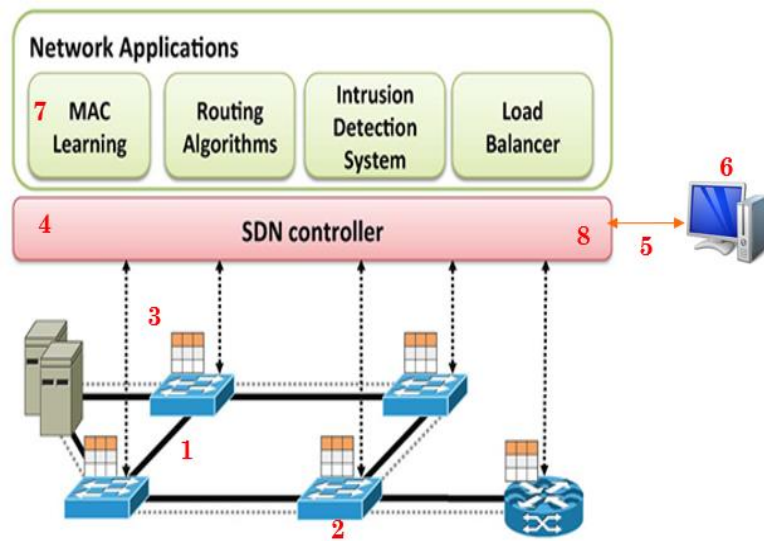


**Figure 2.3** Possible SDN Attack Points

#### 2.4.1 DoS/DDoS Attacks on the Control Plane

Since, the controller is responsible for processing requests from the data plane and creating flow entries for the network, the computing resources on the control plane are a primary target for attacks (Gao et al., 2018). In the event of a controller failure, the data plane will be unable to accept commands from the controller since it will be unable to "think" and respond to any inquiries (Jimenez et al., 2021; Rawat et al., 2016; Wang et al., 2021). The capacity for SDN to respond to changes in the network, such as the deployment of security rules in SDN applications to the switches, is lost if the control plane is not present.

Switches can still forward packets in the classic network mode by referencing previously created flow entries. A delay in the request procedure will also inject latency into the packet transfer, even if the controller is not completely affected (Srivastava et al., 2005). The SDN controller is often built on an open-source platform, making it simpler to integrate with other methods than the old network. Hence, complementary technologies or techniques, such as a cloud network and ML, are employed to improve SDN anomaly detection (Chen et al., 2021; Dong et al., 2016; Mohammed et al., 2018). Maintaining consistent backups of the controller's settings is crucial to facilitate quick recovery in case of a configuration reset. Should the controller be targeted, it is imperative to isolate the infiltrated sections in order to halt the propagation of malicious software. Following this, a rollback process can be executed to reinstate the impacted regions to their initial, protected condition (Sasaki et al., 2016).

Furthermore, researchers have established that these attacks can be intensified by taking advantage of weaknesses in the OpenFlow protocol, a widely used means of communication between the controller and the data plane. The controller may be further burdened by attacks that transmit a large number of well-constructed. OpenFlow messages, resulting in a breakdown of the control plane's capacity to effectively govern the network (Shin et al., 2013). Furthermore, DDoS attacks pose a significant threat in the SDN environment due to their ability to traverse several compromised devices, therefore complicating the task of mitigating the attack by just isolating the origin. The attack traffic might be varied, focusing on several entrance points and thoroughly depleting the controller's ability to manage valid network operations (Scott-Hayward et al., 2015). A straightforward method of combating DoS/DDoS attacks is to reduce the controller's processing rate, as this forces abnormal requests to wait in a queue before approaching the controller. However, this

strategy slows down the usual burst traffic (Cui et al., 2016; Kuerban et al., 2016; Zhang et al., 2016).

#### **2.4.2 DoS/DDoS Attacks on the Data Plane**

When it comes to SDN, the data plane is simply responsible for enforcing policies; it does not need to make any choices. But it does require processing power to apply flow entries provided by the controller and transmit packets. Overloading a switch with malicious traffic might cause it to discard genuine packets and ignore controller commands (Dargahi et al., 2017). Moreover, storage resources like the flow table are a target in DoS/DDoS attacks because a switch operation is defined by flow entries in the flow table and, if the flow table is full of entries, there will be no room for legitimate entry to be inserted, rendering the controller unable to enforce new strategies in the network (Gao et al., 2018). DoS/DDoS attacks disrupt the network's normal functioning, causing it to slow down or become unresponsive, similar to how the original system behaves under heavy load or failure conditions. If the bandwidth between the source and the destination is fully utilised, packet transmission cannot be completed, making the link between switches just as susceptible as the switch itself (Wang, 2021).

Many solutions have been developed to detect and mitigate DoS/DDoS attacks in order to lessen the impact on the data plane. Attacks can be slowed by splitting the available time into many slots and preparing processing queues, much to the strategy used in the control plane (Yan et al., 2017). One common strategy for reducing switch workload is to deploy additional devices, like the middle-box and to filter attack traffic. However, this requires the creation of new interfaces and protocols between the data plane device and the middle-box. It is possible for a switch to utilise proactive flow rules to transmit legitimate packets to their destination while sending any suspect traffic to a filter for additional

examination (Anwer et al., 2013). In order to make room for new entries in the flow table and so update the traffic management in the network, it is recommended that the switch delete idle entries in the flow table (AlGhadhban et al., 2022). The network should always use the safest pathways for packet transmission and reception, even against attacks. Every network node and route can be rated using precise statistical metrics including packet loss rate, latency, jitter, and traffic load. These metrics assist the network to identify the safest data transfer channels by detecting unusual patterns that may indicate compromised nodes or routes. This method protects the network from DoS/DDoS attacks, which exploit holes in less secure or busy paths (Imran et al., 2019; Wang et al., 2017).

### **2.4.3 DoS/DDoS Attacks on the Application Plane**

The SDN application layer, as shown in Figure 2.2, is made up of a number of user applications that handle tasks including traffic management, routing, network virtualization, security, and survivability. The Northbound Interface (NBI) is used to call for application services while the application devices are located on the SDN controller. In light of the crucial function and extensive use of these applications, it is important to establish strong security protocols to safeguard them against possible risks. In order to enable communication between the control plane and the application layer, the SDN stack's NBI makes use of a number of communication protocols, including the Java API, Frenetic, and the Representational State Transfer (REST) API. Nevertheless, these interfaces are deficient in adequate security measures, therefore increasing the susceptibility of the SDN architecture. At present, the NBI does not have established uniform security procedures to effectively deal with developing risks (Cox et al., 2017). By compromising an application at the application layer, an attacker may possibly obtain unauthorised access to the SDN controller, therefore

potentially changing network traffic, intercepting sensitive data, or causing disruptions to operations (Shu et al., 2016).

Application-layer security can be improved by integrating security-based applications that are vital for safeguarding the network from threats, like intrusion detection systems (IDS), firewalls, and antivirus software. These programmes can protect the SDN network from being disrupted by intruders. Most typical DoS/DDoS threats may be effectively detected and mitigated by implementing the idea of running an IDS and IPS on the SDN application plane. By monitoring network activity for signs of abuse or other unusual behaviour, an IDS may quickly identify malicious activity and notify system administrators. ML techniques can be executed at the application layer to aid in the detection of threats based on observations of traffic patterns made at the data plane (Jazi et al., 2017; Mazhar et al., 2015).

## **2.5 Machine Learning in SDN**

Machine learning (ML) is a subfield of artificial intelligence that gives computers the ability to learn from data and make judgments without being explicitly programmed. ML technology has progressed dramatically in recent years, and some researchers have begun incorporating ML approaches into SDN to enhance the effectiveness of network management and conformance, or to address issues that can't be readily addressed by more conventional means (Liu et al., 2019). ML is a very efficient technique for detecting attacks. It achieves this by using a training dataset obtained from a representative sample of a network's data to correctly identify and categorize possible threats.

The efficiency of a ML model is significantly influenced by the selection of features for the training dataset and the allocation of weights to those features during the training process. Weights quantify the extent to which each feature affects the ultimate predicted

outcome. Overfitting arises when a model is excessively fitted to the training data, capturing extraneous information and particular characteristics that do not extend to new data, leading to subpar performance on unfamiliar datasets (Revathi et al., 2022). ML's adaptability makes it a good choice when both the data and the task's complexity are in a constant state of flux. Several different types of DoS/DDoS attacks can be stopped by employing machine learning strategies that examine the packets to determine whether or not they include malicious code. ML techniques can be used to mitigate various forms of DoS/DDoS attacks by analyzing packets to identify the presence of malicious computer code. Specifically, Huang et al. (2014) introduced a ML-based detection system that use clustering methods to detect anomalous traffic patterns linked to DDoS attacks. To enhance the precision of DoS/DDoS attack detection, Polat et al. (2020) proposed a hybrid ML approach that integrates decision trees with support vector machines. In a similar vein, (Bi et al., 2019) devised a deep learning methodology that exploits neural networks to categorize and detect DoS/DDoS attack traffic in real-time.

### **2.5.1 Machine Learning Types**

In Supervised Learning (SL), a model is trained on a labelled dataset, with each input matched to the proper output, in order to effectively predict or classify new data. SL learns the connection between input properties and target outcomes using labelled data. Using this labelled dataset, the SL algorithm finds patterns and adjusts its parameters to reliably anticipate fresh data outcomes. SL detects DoS/DDoS threats through pre-processing, algorithm selection, model training, evaluation, model refinement, and prediction (Alpaydin, 2020). Data preparation from collection through labelling is the most important and laborious step. Instead of plugging the obtained data into the SL, duplicates must be removed, features must be extracted and changed to a format the SL technique can

comprehend, and classifiers must be applied to all data to create labelled data (Cai et al., 2018; Wang et al., 2021). The labelled dataset is usually separated into training and validation sets for SL model building. The training set is utilized to prepare the SL algorithm for learning the correlations between input features and the desired result. On the other hand, the validation set is employed to evaluate the performance of the model and determine its capacity to generalize to novel data. Adjusting hyperparameters like learning rate or tree depth after evaluation can improve model performance with SL algorithms (Merlin, 2018; Morente-Molinera et al., 2016). The best setup for achieving maximum accuracy and generalizability can be found using methods such as grid search or cross-validation. Evaluation of the model's ability to predict outcomes on unseen data is essential to prevent overfitting to the training data and to accurately identify DoS/DDoS attacks in real-world SDN settings. These SL approaches improve SDN security and administration by identifying and responding to harmful traffic patterns. The model improves the network's response time by minimizing possible attacks before they have a substantial impact by giving rapid insights (Angarita-Zapata et al., 2019).

Unsupervised Learning (UL) analyses and finds patterns in data without labelled datasets. UL uses clustering techniques to find hidden data structures or groupings, unlike SL, which trains on labelled instances (Ahmad et al., 2020). UL is used for clustering and dimensionality reduction, which simplify datasets while keeping important information. UL is important in market research for segmenting clients by purchasing behavior and in object identification for discovering data trends (Huang et al., 2014).

Semi-Supervised Learning (SSL) approach is used to combines labelled and unlabelled data to build a classifier. It is useful when there is insufficient labelled data. SSL uses the training procedure of supervised learning to classify unlabelled data and assigns a

confidence rating to each pseudo-labelled sample. The classifier is updated by adding the highest confidence samples to a new training set until all data is labelled. SSL is more practical than SL for detecting DoS/DDoS attacks in the real world due to its ability to obtain a solid forecast with minimal labelled data. However, if the unlabelled data is misclassified, the model may make an inaccurate prediction (Van Engelen et al., 2020).

Reinforcement Learning (RL) is a type of ML in which an agent learns to make decisions by interacting with surroundings and getting rewards or punishments. From action feedback, the agent adjusts its strategy to maximize cumulative rewards, enhancing its decision-making. In order to determine whether or not the computer has made a successful decision, RL looks at the state, the action, and the reward. In RL, an agent observes states, acts, and receives rewards. The RL algorithm lets the agent learn and adjust its behaviour to maximize cumulative rewards (Liu et al., 2019; Rahman et al., 2019). The environment changes the agent's state and applies a reward or punishment based on the activities performed by the agent. In the absence of a termination state for the agent, this loop will continue to iterate indefinitely. The agent learns the appropriate policy to maximize long-term benefits through the incentives provided by the environment (Sahu et al., 2023). The various types of ML are thoroughly illustrated with examples in Table 2.1.

**Table 2.1:** Machine Learning Types

| ML Type                 | Descriptions                                                                                                                                                                                        | Examples                                                                       |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| Supervised Learning     | The model is taught by providing both inputs and intended results in the form of labelled training data.                                                                                            | Regression, Classification by training a model.                                |
| Unsupervised Learning   | The model is able to learn from data that has not been labelled, discovering hidden patterns.                                                                                                       | Clustering by grouping similar behaviour.                                      |
| Semi Supervise Learning | Enhances learning accuracy by combining a significant amount of unlabelled input with a small amount of labelled data.                                                                              | By training model for anomaly detection by using labelled and unlabelled data. |
| Reinforcement Learning  | Involves an agent that picks up decision-making skills by interaction with the environment, punishment or reward for behaviours, and gradual strategy optimization to maximize accumulated rewards. | By training agents that can play games both for winning and losing situations. |

Hence, SDN is significantly improved by the integration ML techniques. SL technique employed to classify traffic and identify security concerns such as DoS/DDoS attacks by acquiring knowledge about patterns observed in annotated datasets. The application of UL in anomaly detection and clustering of network traffic enables the identification of anomalous patterns that may suggest security problems, without relying on labelled data. In order to enhance accuracy in network traffic categorization and security detection, particularly in cases when labelled data is scarce, SSL integrates both labelled and unlabelled data. To maximize dynamic network decisions, like traffic routing and resource allocation, RL interacts with the environment and adapts to real-time network modifications. Collectively, these ML methods enhance the capabilities of SDN in traffic management, network security, and performance improvement.

### 2.5.2 Supervised Learning Algorithms

SL approaches encompass a range of categories, such as Discriminant Analysis (DA), Decision Tree (DT), Generalized Linear Model (GLM), K-Nearest Neighbour (KNN), Naive Bayes, Feedforward Neural Network (FNN), and Support Vector Machine (SVM) (Liu et al., 2019; Santos et al., 2020). Within the set of SL algorithms, we have chosen KNN and SVM for our system to identify DoS attacks due to their well acknowledged accuracy in classification tasks. Both approaches have been extensively employed in intrusion detection systems because of their capacity to precisely categorize network traffic as either normal or malicious, rendering them appropriate for detection of DoS attacks.

As a SL approach, SVM are widely utilized for both regression and classification. SVM is an effective machine learning technique specifically designed for problems such as linear or nonlinear classification, regression, and outlier detection. In SVM, data points are encoded as coordinates in a space of dimensions  $n$ , where  $n$  is the number of features. Iteratively, the method determines a hyperplane that optimally divides the data points into various groups or classes. By maximizing the margin between the two classes, this hyperplane guarantees precise classification of fresh data points. This approach is very efficient when there is a distinct threshold of differentiation between the classes (Santos et al., 2020). In SVM, the data points which lie closest to the hyperplane are called support vectors, and they are crucial for figuring out the hyperplane's location and orientation. By determining the best hyperplane that separates two classes or categories in a dataset, SVM are very efficient in differentiating between them. In high-dimensional areas, SVM are effective in classifying patterns and images, although they may not always beat neural networks or decision trees. The use case and dataset determine SVM's ability to detect abnormalities and categorize network traffic in SDN (Hadem et al., 2021). The goal is to

classify a large number of images as belonging to one of two categories as  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$  where  $x_i \in R^d$ , here  $d$  is dimensional feature element space and  $y_i \in \{-1, 1\}$  here  $-1$  and  $1$  represents the class name, for  $i = 1 \dots n$ . SVM aims to identify the most efficient hyperplane that optimizes the margin between two specified classes. The data are essentially divided into two groups by this hyperplane. SVM is able to categorize complicated datasets that are not linearly separable because of the kernel function  $K$ , which can be either linear or non-linear and which converts the input data into a higher-dimensional space. Images containing feature vectors on one side of the hyperplane are categorized as  $-1$ , whereas those aligned on the opposite side are categorized as  $1$ . In SDN, SVM have been utilized to categorize network traffic and identify irregularities, therefore facilitating the differentiation between regular and malicious traffic patterns (Rimal et al., 2020).

The KNN method is a supervised machine learning technique that is used to solve both classification and regression issues; it is simple, widely applicable, and incredibly versatile. As new information is ingested, its classification is established by comparing it to its immediate  $K$  neighbors. To calculate the separation between two sets of information, we can use either the Manhattan, Minkowski, or Euclidean distance functions. In this study, samples are categorized according to how close they are to pre-existing data points in a certain class using the Euclidean distance measure. Euclidean distance is a useful metric for detecting similarities in network traffic patterns and categorizing them as benign or malicious in DoS attack detection. It is computed as the straight-line distance between two points in a multidimensional space. Manhattan and Minkowski are two more popular alternative distance measures that are employed based on the dimensionality and structure of the problem (Wang et al., 2021). In high-dimensional domains, Euclidean distance is

simple and effective, making it ideal for identifying small network traffic fluctuations in SDN systems and detecting abnormalities. Each instance of novel data required a separate calculation of its distance from the training set using the Euclidean function. After that, we choose the  $k$  dataset with the least distance to use as our classification set. Using the majority class of the  $k$ -nearest data sets/points in a multidimensional space, The KNN technique is a popular method for classification. The algorithm's success is heavily dependent on the value of  $k$ , which determines how many neighbors are considered while classifying. Particularly, (Liu et al., 2019) emphasize that selecting a suitable value for  $k$ , such as  $k=10$ , can greatly enhance the precision of the classification procedure. In SDN, KNN can classify and detect anomalies in network traffic in real time to identify security concerns like DoS attacks. The KNN algorithm improves the network's performance and security by assisting SDN controllers in making well-informed decisions on traffic routing and management based on features derived from network flows.

The purpose of DA is to divide data into two or more known categories based on the characteristics of the data. In DA, the basic idea is to design one or more functions based on the discriminant criterion and then use the training set to outline out the indeterminate coefficient. The chance that a piece of data belongs to each group is used to determine how the data should be categorized; then, the data are placed in the group that has the highest probability (Peng et al., 2008). DA may be broken down into two subcategories: canonical variations analysis (CVA) for more than two groups and discriminant function analysis (DFA) for just two groups. DA is utilized in several applications, such as intrusion detection feature extraction. Through the analysis of network traffic, DA enables the identification of particular attributes that can be utilized to detect possible security risks, such as unauthorized access or attacks (Paquin et al., 2013).

The GLM is employed in cases where a normal distribution is not expected from the data. It is a term used to describe a class of models in which the relationship between a dependent variable and the outcome it predicts is linear (Yu et al., 2018). The sigmoid function is used in logistic regression, a common GLM, to classify data into two categories. In terms of network security, GLM's are utilized for the purpose of vulnerability screening as well as the identification of attackers (Liu et al., 2019).

### **2.5.3 Comparative Analysis of Machine Learning Techniques in SDN Security**

In SDN security, supervised learning approaches like SVM, Decision Trees, and Random Forests are best at categorizing network traffic from labelled datasets. These strategies are especially efficient in contexts where accurately labelled data is accessible, allowing them to distinguish between normal and malicious traffic with great accuracy. Supervised learning's reliable performance makes it well-suited for identifying familiar dangers, serving as a strong initial protection in secure network environments. Nevertheless, these approaches have no table constraints, particularly their need on annotated data, which can be arduous to gather and sustain in ever-changing networks. Moreover, supervised models frequently encounter difficulties in identifying new or changing threats that were not included in the training data, resulting in possible vulnerabilities in the network's security.

However, unsupervised learning methods like K-means clustering and Principal Component Analysis (PCA) may find anomalies without labelled data, making them ideal for detecting new SDN attacks. These techniques have the ability to detect atypical patterns or behaviors in network traffic that could potentially signal a security risk, even if that risk has not been seen before. Nevertheless, the drawback of having this level of adaptability is an increased occurrence of false positives. Unsupervised algorithms have the potential to misidentify legal traffic as harmful since it deviates from established standards. This may

result in unwarranted mitigation measures, which could potentially impede lawful network activities. The decision to use either supervised or unsupervised learning in the SDN environment is contingent upon various factors, such as the presence of labelled data, the characteristics of the threats, and the level of acceptance for false positives in the network security approach.

## **2.6 Analysis on Current Approaches**

In response to the increasing volume and sophistication of DoS/DDoS attacks, several strategies have been developed to handle these difficulties within the context of SDN. This research takes a look at the various methods that are being utilized for the detection and mitigation of DoS/DDoS attacks in SDN systems. The following four approaches, each of which is outlined in detail, will serve as benchmarks for this research. Two approaches are for DoS attacks and two are on DDoS attacks, we are considering these DDoS approaches for Dos perspective.

### **2.6.1 SECOD Approach**

Wang et al. (2021) proposed SECOD algorithm, which is designed to be used within the SDN architecture for the management and protection of networks against DDoS attacks. SECOD is deployed at the SDN architecture's application layer and operates on the same device as the controller. The controller takes the pre-defined SECOD rules and inserts them as flow entries into the OpenFlow switches. In the data plane, these policies either permit or forbid access. SECOD continuously analyzes the network for abnormalities and irregularities against predefined thresholds. When these limits are surpassed, security risks are detected. These limits reflect typical measurements of network behavior, including response times or traffic volume. A rapid increase in benign traffic or a DDoS could be the cause of an excessive amount of 'packet\_in' messages, which are generated when switches

send unknown packets to the controller for processing. Because the amount of benign traffic varies, the threshold is continually modified to conform to changes in the network environment. However, when demand exceeds the threshold, harmful traffic is automatically discarded without algorithms. SECOD can precisely determine the source of a DDoS attack, whether it is originating from a host or a switch, and thereafter implement suitable countermeasures. The mitigation mechanism remains active in the background until the attack dissipates and the threshold is reestablished. Once the attack ends, SECOD resumes its regular period of observation.

The author proposes incorporating the Monitor Function into the SECOD algorithm to identify atypical increases in network traffic, potentially an indication of a DDoS attack. The packet\_in count per IP address and switch port is calculated by the function. DDoS detection is activated if the controller is overloaded, which this method determines by adding up the total amount of packet\_in messages received by each switch. The Monitor Function considers the threshold,  $w$ , and each switch port's current and historical workload. The method moves into the Threshold Update stage if one or more packet\_in messages from a port exceed a threshold but the switch's total stays below  $\omega$ . Iteratively, the algorithm modifies port thresholds in response to workload statistics collected in the past. The packet\_in message rate for a switch is not explicitly stated by the author; however, a pre-established limit,  $\omega$ , is provided, enabling the switch to effectively handle incoming flows. Specifying  $\omega$  prevents DDoS attacks from overloading the switch and preventing valid traffic from being processed. Before network involvement, the application and customer needs determine the  $\omega$  value. It shows the switch's maximum data throughput without compromising performance. The administrator can adjust  $\omega$  to 500 to double the switch's throughput if it can only handle 500 of its 1000 packet\_in messages per minute. Since  $\omega$  is

smaller, the switch can process more flows without compromising performance. Ultimately, the  $\omega$  value is utilized by the SECOD algorithm to identify switch overload and commence DDoS mitigation strategies, all the while preserving the efficiency of the network. The application's and customer's specific requirements dictate the value of  $\omega$  prior to network involvement.

Threshold values assist distinguish between typical and abnormal traffic, which is crucial in DDoS detection and defence. The network might not be able to identify potential threats if the threshold value is set too high. If the threshold settings are too low, however, the network may reject perfectly legal traffic, leading to interruptions in service. SECOD can adjust to new traffic patterns over time since it uses past data to adjust its threshold settings. For instance, if traffic suddenly spikes, the threshold values might be modified to avoid erroneous DDoS detection. It is also crucial to reset the threshold settings after a DDoS attacks, as the statistics collected during the attack may not be indicative of the network's usual behaviour. With the thresholds reset to their pre-set settings, SECOD may begin normal traffic data collection and make appropriate adjustments.

SECOD relies on its DDoS Detection module to detect and stop DDoS attacks. The module detects a network attack when each switch's packet\_in messages exceed  $\omega$ . The algorithm then sends flow entries to all possibly malicious switches and goes to the DDoS detection module. The switches are directed to discard a new flow if it does not instantly match an existing flow in the flow table. In the event that the controller does not receive any packet\_in signals, the attack is being launched by the hosts. On the other hand, if the packet\_in messages keeps coming to the controller, it could mean that the attack is coming from a hacked switch. DDoS detection output determines host/switch mitigation. If a host is attacking, Host mitigation occurs. This phase involves blocking the attacker's IP address or

rate limiting. If the attack is originating from the compromised switch, the switch mitigation step entails isolating the switch or blocking the attacker IP addresses.

The authors proposed a function called Host Mitigation, which seeks to detect and prevent harmful traffic coming from certain hosts. To identify which hosts are generating an excessive number of `packet_in` messages, the function use the per-IP address counter. If only a single IP address is flagged as suspicious, the controller will add a flow entry to switch  $j$  to prevent any data from being sent to or received from that address. Because of this, the malicious communication is blocked before it can reach its destination. If there are too many `packet_in` messages, the controller will request that the switch delete all data from the port. This reduces the probability of a DDoS attack from a single server flooding the switch or controller with traffic. By filtering data at the switch level, the controller may reduce network load and prevent risky data from reaching its destination. The packet-blocking flow item vanishes after the attack with idle timeout. This minimises additional interruption to genuine traffic and speeds network restoration. Here the author designed SECOD Switch Mitigation technique to handle situations where a controller discovers a compromised switch. The controller then ignores any more `packet_in` signals from the hacked switch and keeps track of its total requests. This allows SECOD to detect the switch's DDoS attack's termination and re-establish network operation. The authors in (Wang et al., 2021) recommended this because a hacked switch may not use the controller's preferred method of communication. Because of this, the controller may become too busy to effectively handle the switch and may even crash under the weight of the switch's requests. SECOD can halt the hacked switch from having any further effect on the network by telling the controller to ignore `packet_in` messages from the switch. This allows the controller to focus on restoring normal functioning as soon as feasible. The sum of a switch's `packet_in` messages (count), is one of

the metrics monitored by the Switch Mitigation method. If this count is less than  $w$ , the blocking flow entry is unblocked, and the switch is no longer under DDoS attack. This prevents the switch from being stopped for too long and gets the network back to normal as soon as feasible. The overall workflow of SECOD is shown below in Figure 2.4.

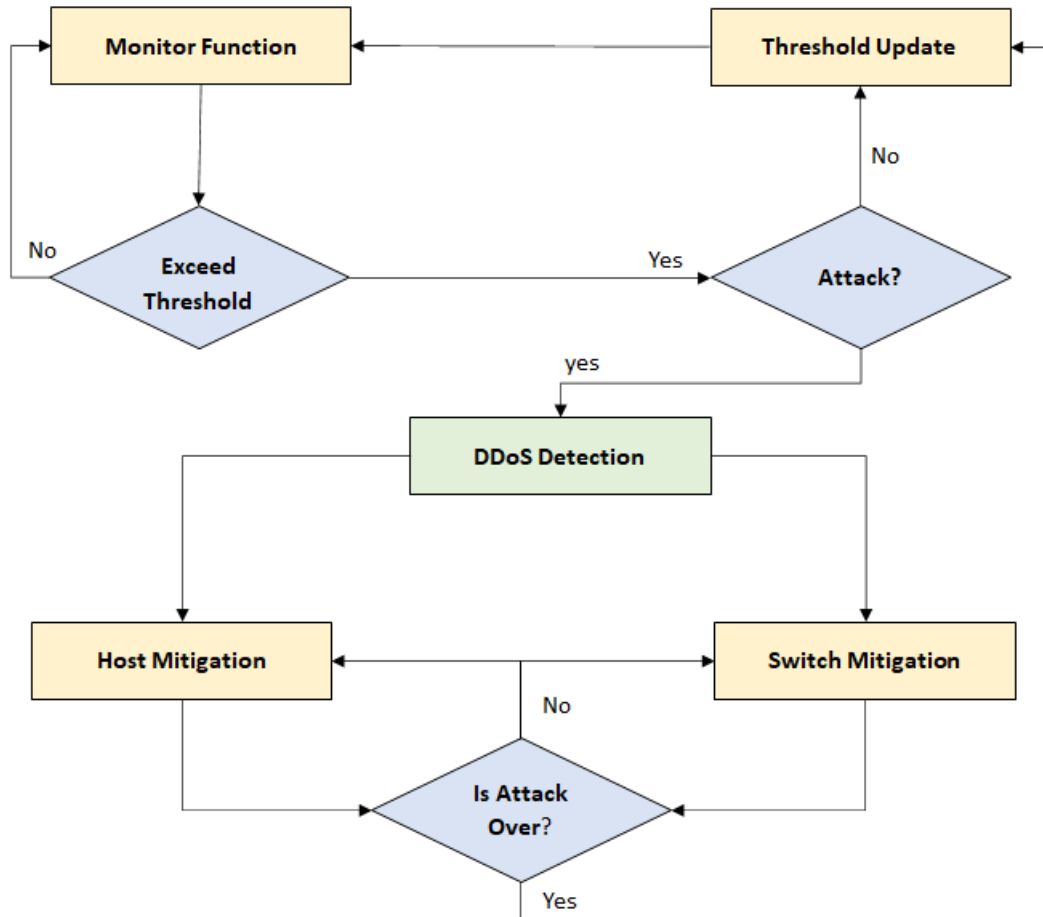


**Figure 2.4** Workflow of SECOD algorithm (Wang et al., 2021)

#### 2.6.1.1 SECOD Approach

The SECOD algorithm integrates a dynamic threshold adjustment mechanism, enabling it to refine its thresholds through the analysis of recorded data and adjust to evolving traffic patterns. This technique can recognize if a DDoS attack is coming from a host or a switch and react appropriately. SECOD consistently monitors the network, regularly assessing its condition against predetermined performance criteria. Implementing

real-time monitoring of all ports and IP addresses allows for the timely identification of DDoS attacks.

To improve the accuracy of DDoS detection and mitigation, the algorithm dynamically adjusts the threshold values based on the network's historical traffic data. The SECOD algorithm relies heavily on the  $w$  value to determine whether the switch is overloaded and to initiate the appropriate mitigation procedures to fend off a DDoS attack without negatively impacting performance.

#### 2.6.1.2 Weaknesses

The author's work does indeed present details on the upper limit of `packet_in` messages that a switch is capable of processing. The configuration allows the administrator to set the threshold  $\omega$  to 500 `packet_in` messages per minute, which is equivalent to 50% of the switch's capacity, provided that the switch's maximum throughput is 1000 `packet_in` messages per minute. Another point raised by the article is that 100 `packet_in` messages per minute is the limit for a single switch port. If the network has a detection threshold that is too high, DDoS attacks may go undetected since the attack traffic will be masked as normal network activity. In contrast, if the threshold values are set too low, the network may reject perfectly lawful traffic, resulting in service disruptions. Statistics gathered during a DDoS assault may not be representative of the network's typical activity, thus it's important to reset the threshold settings afterwards.

Once all switches have received an enough amount of `packet_in` messages, the DDoS detection module of SECOD is triggered. Indeed, SECOD predominantly focuses on application-layer DDoS attacks, since it functions at the application layer of the SDN architecture. Consequently, its effectiveness is restricted when dealing with DDoS

attacks that specifically aim at lower network layers, such as the transport or data link layers, which necessitate distinct detection algorithms.

### **2.6.2 Flood Defender Approach**

The author's (Gao et al., 2020) suggested FloodDefender as a defense mechanism against attacks directed at SDNs in OpenFlow networks. SDN controllers are susceptible to attacks that take advantage of the controller's flaws or alter the flow rules because of their reliance on centralized administration of network resources. To prevent attacks from reaching their intended targets, FloodDefender acts as a barrier between the controller platform and other controller apps. The FloodDefender is divided into a detector and a protector. Whenever an attack is detected, the detection module will notify the mitigation module. When an attack is detected, the mitigation module enters an active state where it filters packet\_in messages, supervises the installation of flow rules, and prohibits table-miss packets from entering the victim switch over the input port.

To overcome the shortcomings of pre-existing SDN security solutions, the author advocated developing FloodDefender. For instance, in an SDN setting, traditional security solutions like firewalls and IDS cannot function properly since they rely on packet inspection at the network edge. SDN security solutions that are already available, such as FlowVisor, FlowChecker, and FlowGuard, offer multiple defense mechanisms in addition to threat detection. FlowGuard not only identifies flow-based conflicts in SDN but also assists in resolving these conflicts, therefore offering an extra level of security. Typically, in many SDN systems, the detection and mitigation modules collaborate actively while continuously monitoring the network. Once a DoS attack is detected, the mitigation module immediately responds by implementing many measures to halt the attack and protect the network. Initially, the detection module monitors the network's health while the mitigation processes

run. The following steps and measures are taken to mitigate attacks when the detection module identifies SDN-targeted DoS attacks:

1. The module detects non-attacked switches in the victim's immediate vicinity that link to the controller.
2. Table-miss engineering sets up safeguarding rules on the victim switch to reroute certain table-miss packets to neighbour switches; flow rule management logically divides the flow table into flow table region and cache area.
3. The controller records packet\_in messages from neighbour switches that have received detoured table-miss packets and applies a coarse filter to the stored messages to remove attack flow.
4. The control applications take the filtered packets, process them, and then dispatch the appropriate action messages and flow rules. To the switch that reports packet\_in, action messages will be delivered. However, the flow rule management component will act as an intermediary and intercept flow rules.
5. The flow rule management component will then determine the monitoring rules based on the processing rules that it has intercepted.
6. The cache area of the victim switch will be updated with the processing rules intercepted during the attack. Instead of installing monitoring rules in the cache region, the victim switch might transfer a rule to the flow table area if it is validated as lawful by the packet filter subcomponent. The cache will be cleared out to make room in the flow table.

#### 2.6.2.1 Detection Module

Using an OpenFlow-based network, the authors describe FloodDefender, a system for detecting and mitigating network attacks. When an attack is identified, the mitigation module is activated and FloodDefender is in a proactive state. The detection module does

this by continually monitoring the status of the network. Information for dynamically modifying protection rules is also provided by the detection module. Due to high message rates and utilisation in reactive networks, established solutions such as FloodGuard might generate false alarms. FloodDefender introduces flow entrance frequency as a characteristic to recognise attack traffic, which improves the accuracy of detection. The total number of flows and their average frequency are calculated for each flow entry type. These categories are further subdivided into flows with low, medium, and high frequencies. With this method, malicious traffic can be pinpointed with greater accuracy, and false alarms that degrade network speed are eliminated.

In this article (Imran et al., 2019), the author's look at how FloodDefender, an attack detection and mitigation solution for OpenFlow-based networks, makes use of frequency attributes to spot brand-new flows and tell them apart from normal network start-ups from DoS attacks. The author's research shows that this is not the case for newly formed flows (table-miss packets), but the packet count field of the flow rules offers a convenient means of obtaining the frequency of matched flows. A heuristic determines the new flow frequency by treating each table-miss packet as a 1-frequency flow and clustering it based on the destination IP and MAC address. Once we implement the associated flow rule, we determine the flow's frequency and delete all records with a frequency of 1 from this set. The author's research demonstrates how low-frequency flows, in particular, might have drastically different values under attack and in network starting conditions. FloodDefender's state changes are anomaly-triggered depending on the frequency characteristics, coupled with other factors like packet\_in message rate, controller memory, buffer memory, and CPU, to analyze the health of the network.

#### 2.6.2.2 Mitigation Module

To defend OpenFlow networks from SDN-targeted DoS attacks, FloodDefender employs a combination of mitigation module functions: packet filtering, flow rule management, and table-miss engineering.

The table-miss engineering module of the FloodDefender system, which lessens SDN-targeted DoS attacks, is discussed by the author. In order to minimize bandwidth, use on the victim switch, the module establishes flow rules to reroute packets that include table misses to neighboring switches. The creation of protective policies, however, faces a number of difficulties, such as the misidentification of diverted traffic and packet bouncing. By leveraging the Type of Service (ToS) field and other reserved fields in packet headers to signal redirected traffic while preserving the original INPORT (input port) data, the author suggests a solution to the INPORT loss problem. To avoid packet bouncing between the neighbor and victim switches, the author also advises implementing protection rules just to non-detoured traffic. The author also goes through how to build security rules that are based on MAC addresses and IP DSCP to improve the efficiency of processing good flows and save bandwidth between the victim switch and the controller.

The author talks about packet filtering, a mitigation module feature used in SDN-focused DoS attacks. For conserving computing resources, the packet filter is made to identify and filter out attack traffic. The two-phase filter and packet-in buffer make up the packet filter component. In the packet filter component, the author proposes a method for efficiently storing and indexing packet\_in messages. A B+ tree is used to store the packet-in messages after they have been categorised based on protocols. Five seconds are suggested by the author for gathering packet-in messages. To properly distinguish between unfiltered and filtered flows, the two-phase filter uses two filtering techniques: a frequency-based filter

and a traffic-based filter. In order to identify attack packets, the traffic-based filter extracts features such as traffic rate imbalance. Monitoring flow rules are used to track reverse traffic in order to isolate the asymmetries. The author recommends against the use of asymmetric features in their suggested system's traffic-based filter. While the author acknowledges that asymmetric features can be useful for spotting attack traffic, they also have the potential to produce inaccurate results in some circumstances, a phenomenon he refers to the asymmetric feature problem.

Since not all flows have substantial reversible traffic from which asymmetric features can be extracted, this issue arises. Asymmetric characteristics may not adequately reflect a flow and lead to inaccurate classification if, for instance, the flow is unidirectional, or the response packets are discarded by the network. To address this problem, the author recommends prioritising flows with asymmetric traits for monitoring and taking into account additional aspects during categorization.

FloodDefender is a comprehensive controller platform software comprising two basic modules: detection and mitigation as illustrated in Figure 2.5. It safeguards OpenFlow networks from SDN-targeted assaults by implementation of tablemiss engineering, packet filtering, and flow rule management. The FloodDefender engine functions in three distinct states: alert, active, and block. In the absence of any detected threats, the system monitors the network condition and transmits packet\_in messages. Severe attacks result in the blocking of table-miss packets. It reverts to alert mode upon detection of attacks. FloodDefender architecture is shown below in Figure 2.5.

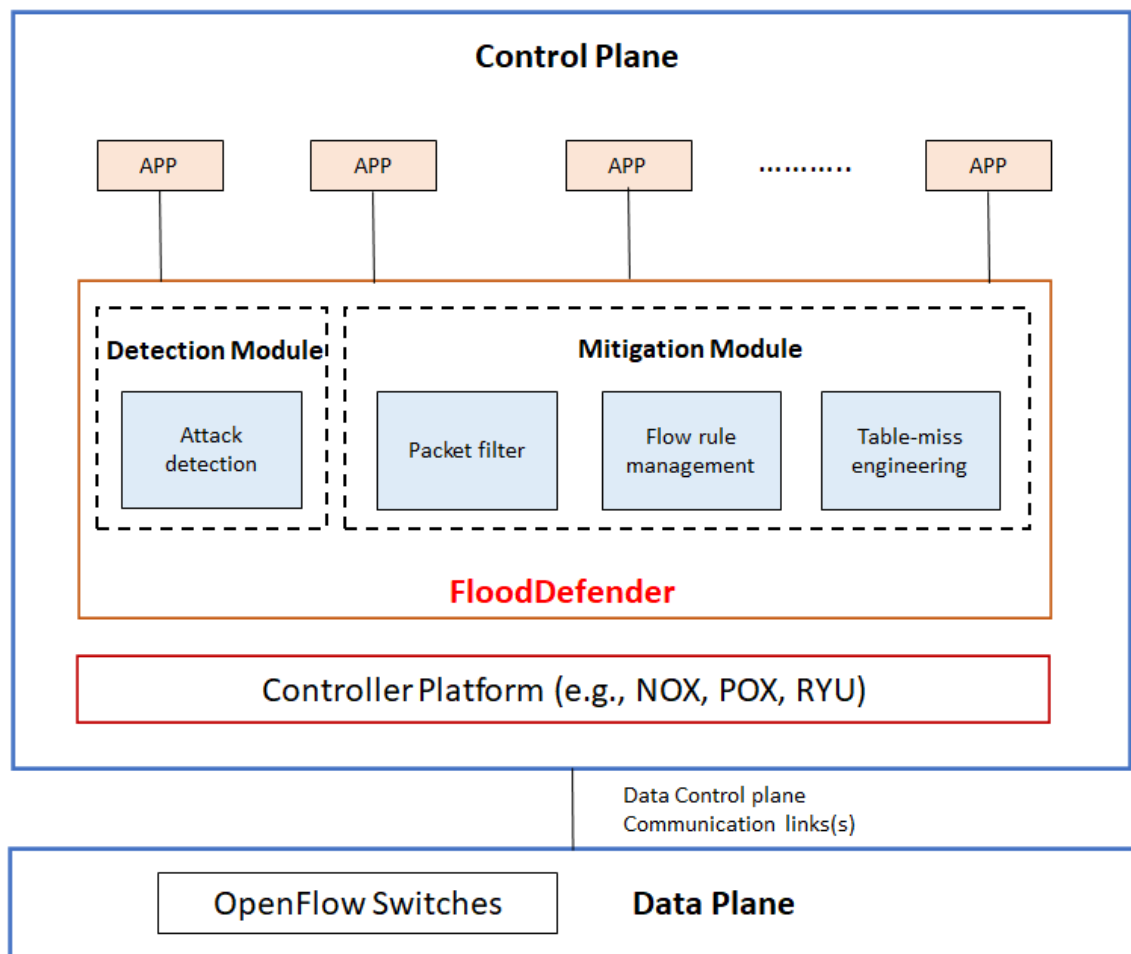


**Figure 2.5** FloodDefender Architecture (Gao et al., 2018)

### 2.6.2.3 Strengths

The FloodDefender system employs a hybrid strategy that combines detection and mitigation modules to secure OpenFlow-based networks against DDoS attacks.

The attack detection module constantly checks the network's condition and uses the frequency of flow entries as a crucial measure to detect perhaps harmful traffic. As a result, detection is more precise, and false alerts, which slow down the network, are eliminated. Functions such as packet filtering, flow rule management, and table-miss engineering are included in the mitigation module.

By configuring flow rules to forward packets containing table misses to neighboring switches, the table-miss engineering module helps to reduce the victim switch's bandwidth consumption. table-miss traffic is split off and sent to other neighbor switches in the network using protective rules, which are low-priority wildcard flow entries, and the load is spread out by dynamically adjusting the match fields of protection rules. When rerouting traffic, the ToS field is used to indicate the new path while the original INPORT data remains unaltered, solving the INPORT Loss Problem.

#### 2.6.2.4 Weaknesses

FloodDefender was developed for OpenFlow-based networks and may not function with other network architectures, therefore restricting its wider usefulness.

To function properly, the system needs a sizable amount of computational resources, which may not be present in all network setups. For example, the FloodDefender may exert significant computational demands on smaller networks and overwhelm controllers in large-scale networks with substantial traffic volumes, therefore resulting in performance bottlenecks. The usefulness of the system may be compromised in some use cases due to the limited range of INPORT values caused by the number of bits in the reserved field. Misidentification of detoured traffic and packet bouncing are two examples of problems that might arise during the development of protective rules and reduce the system's ability to function effectively.

There will be longer delays for previously innocuous flows, and the controller will need more memory. When the two-stage filtering process is complete, the packet\_in buffer flushes all B+ trees that have accumulated packet\_in messages over the course of the time period. The user has complete control over how long is spent waiting for packet\_in messages.

A longer period can save more computational resources, but it also means the controller will need to use more memory while waiting for newly benign flows.

The frequency-based filter has limited efficiency: The packet filtering function makes use of a two-phase filter, which performs two separate filtering tasks. To roughly eliminate attack data in the beginning of its operation, the frequency-based filter makes use of the frequency characteristic. This filter may not be very accurate because its threshold is not fixed and it cannot effectively block complex attacks.

### **2.6.3 DAISY Approach**

The author's (Imran et al., 2019) suggested a statistical method called DAISY to lessen the impact of DoS attacks. The method relies on a predefined flow rule rather than a total port, IP, or MAC address block. The authors argue that their method is more effective in mitigating DoS attacks while also minimizing the impact on legitimate traffic.

This system takes two approaches to spotting and preventing malicious traffic. If the system will first label the traffic as suspicious and place a temporary block on it if it identifies any activity that seems out of the ordinary for a certain flow. Because of this, the system can distinguish between safe and risky traffic and lessen the burden on the former. If the attack persists, the system will label the traffic as malicious and put up a longer block. By constantly adjusting the time interval in response to the hostile traffic's activity, the system may counteract new kinds of attacks as they emerge. DAISY eliminates the blocked flow rules from the system when flooding ceases. This prevents unnecessary or irrelevant flow rules from affecting the network's performance.

There are two primary parts of the proposed system: parameters and functions. We need both the parameters and the functions to configure the system for DoS protection.

#### **2.6.3.1 DAISY's Parameter's Insights**

For the DAISY framework's initial configuration, the specified parameters in the suggested system are essential. The iteration interval, the first essential parameter, is the duration between DAISY's repeated data scans to identify DoS attacks. Given the processing capacity of the controller and the desired speed at which the network aims to detect and prevent fraudulent traffic, the iteration interval can range from 2 seconds to 10 seconds.

The Flow Request Limit (FRL) specifies how many requests for flow rules an ordinary host can make to the controller in one cycle. DAISY will begin to suspect a host as malicious if it sends more flow requests than the FRL. Depending on the number of hosts in the SDN, the FRL can be set to either 50 for networks with 15 or less hosts or 100 for larger networks.

The number of times a host can cross the FRL before triggering a warning is the third parameter to consider. When this threshold is crossed, a warning flow rule is set up to temporarily halt suspicious traffic. From 2 to 6, the Warning threshold can be adjusted.

The Blocking Threshold is the fourth parameter, and it specifies the maximum number of times a host can exceed the FRL in a given time frame. A flow rule is set up to permanently stop the traffic if it reaches this level, indicating that it is likely malicious. The blocking threshold has a value that is twice as high as the warning threshold.

Fifthly, the Idle and Hard timeouts determine whether a flow rule is deleted from the switch after an idle time span during which no packets passed through it. Inactive flow rules are also deleted from the switch when the Hard timeout has elapsed. These settings will replace OF's defaults.

In general, these factors contribute to the process of adjusting the DAISY system to perform successfully in detecting and mitigating DoS attacks. Table 2.2 provides an overview of the simulation parameters that were utilised for the DAISY approach. These

parameters must be set in accordance with the network's size and complexity for optimal performance.

**Table 2.2:** DAISY's Parameters

| Used Parameter     | Value   |
|--------------------|---------|
| Flow Request Limit | 50      |
| Iteration Interval | 5 sec   |
| Warning Threshold  | 3       |
| Blocking threshold | 6       |
| Idle Timeout       | 15 secs |
| Hard Timeout       | 15 secs |

#### 2.6.3.2 Performance Optimization of DAISY Functions

The authors proposed four primary functions in DAISY system which are information gathering, threat detection, attack prevention, and risk mitigation.

The data collecting procedure parses incoming packet\_in messages for relevant information and stores it in a Table for later use. The data gathered consists of the switch-ID, in-Port, Ethernet type, MAC and IP addresses of the sender and receiver, protocol, and sender and receiver port numbers. These details are subsequently utilised by the system's subsequent operations. Threat detection is the core function of DAISY system, which monitors incoming traffic for irregularities by comparing the amount of flow requests from a certain switch-ID, in-Port, MAC, IP, or port to the FRL limit and this will be done in data collection. The elements of this combination are passed as parameters to the attack prevention procedure if the FRL threshold is exceeded.

DAISY's primary purpose is to detect threats in order to stop them from launching attacks on the network. To begin, the algorithm determines if the FRL has been exceeded by the amount of flow requests coming from a certain switch-ID, in-Port, MAC, IP, or port

during the current iteration. The components of this combination are passed as parameters to the attack prevention procedure if they are found to be above the FRL. When a threat is detected, the detection process continues to the next level, where it checks for IP spoofing and flooding. This allows the detection mechanism to zero in on the fine-grained characteristics of spoofing and flooding, hence reducing the legitimate packet drop.

The recommended system's detection algorithm is too sophisticated to execute when the controller gets packet-in signals from  $s$  switches and  $h$  hosts. When  $h$  is inflated due to spoofing or flooding, the software will investigate the attack in depth. Responses to threats are managed by DAISY's two primary functions, the threat detection and prevention system. After an attack, attack prevention checks the Malicious Flows database for suspicious traffic. The function creates a database record for the threat-detected flow combination. The in-Port type increases the flow combination's threat value by 0.5 or 1.0. The function temporarily prevents suspicious traffic if the new risk value exceeds the warning level. The function blocks harmful traffic longer if the threat exceeds the blocking threshold. The blocking rule's duration is the square of the threat value multiplied by the warning rule's length, which is selected by adding a random number to the system's hard timeout value. Malicious Flows records are updated using a UNIX timestamp.

Threat Value reduction is in accountable for decreasing the threat values and timespan of inactive blocking flow rules. A blocking flow rule is deleted from the associated switch if it has been dormant for the idle timeout period, but it is still stored in DAISY's malicious flow database. After their time in the malicious flow table has elapsed, the Threat Value Reduction function will reduce the threat value and duration of these rules to half. DAISY iterations keep repeating this procedure, clearing the table of obsolete blocking flow entries. Incoming packet header information is kept by the data gathering module in a

statistics table, which is purged at the end of each threat detection cycle. A flow rule restricts harmful traffic if the threshold is surpassed, and the attack prevention technique increases the threat value of the malicious flow rule. The risk Value Reduction technique gradually reduces the threat levels for inactive blocking flow rules if there is no current danger. The effectiveness and efficiency of DAISY are preserved by automatically responding to new threats and reducing false positives. Working procedure of DAISY is shown in Figure 2.6.

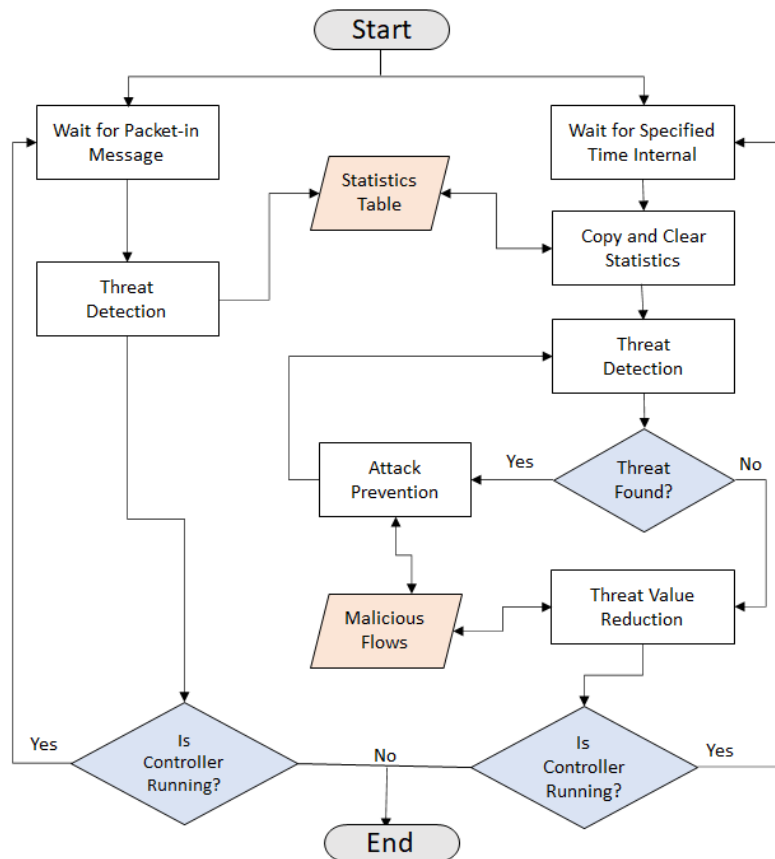


**Figure 2.6** Working of DAISY (Imran et al., 2019)

### 2.6.3.3 Strengths

This system does not add any extra time to the route construction process since it only replicates the required information from packet\_in messages and does not disrupt the routing process. This ensures that the implementation of DAISY will not impede the

performance of your system. It offers a quick reaction to flooding threats by immediately stopping unwanted traffic. The combination of a warning threshold and an iteration interval allows for this to be accomplished. Within 15 seconds, the controller has the first flow rule in place to prevent the flooding traffic.

It only restricts certain flows according to MAC or IP addresses and ports. This prevents any throttling that may occur if other ports on the same host or on different hosts were to be blocked. In contrast, some methods may arbitrarily halt all port activity such as rate limiting, or port shut down. When the timer for the flow rule expires, DAISY restores normal traffic flow automatically. This prevents the prolonged blocking of necessary traffic.

Unlike some other methods, DAISY does not rely on statistics from OF switches to prevent DoS attacks. The controller's path-finding data are all that's needed, thus no extra traffic is generated for checking and detecting. In order to identify attacks without depending on a signature database, DAISY employs an anomaly-based detection methodology. By detecting atypical patterns in network traffic instead of comparing them to established attack signatures, this approach helps to preserve the controller's resources.

#### 2.6.3.4 Weaknesses

This approach is specifically developed to detect and reduce DoS attacks that especially aim at elements of the SDN architecture, including the controller, control channel, and OF switches. However, it does not provide immunity to servers functioning inside the SDN environment. Hence, in the event that the server becomes the subject of an attack, DAISY will be unable of averting or reducing the consequences.

It is built to identify and reduce traffic flooding, but it cannot identify and eliminate the potential for forwarding loops caused by conflicting flow rules. Unfortunately, this could mean that DAISY is unable to detect and fix problems caused by incompatible forwarding

policies. Without any new route requests, this system cannot tell if a data transfer between hosts is garbage, thus the controller won't have to deal with it. This means that DAISY will be helpless in the face of attacks wherein malicious actors send garbage data between two hosts. It takes a lot of computing power to run the system, and that's especially true during DDoS attacks. This might cause delays or even system failures.

#### **2.6.4 Scalable Memory Based Approach**

The authors (Revathi et al., 2022) suggested a method for detecting and protecting against DDoS attacks in SDN infrastructures. The author presents an SDN mitigation architecture for attack detection and a discrete, scalable memory-based SVM technique for DDoS threat identification. The suggested system utilizes the Spark standardization method for initial data processing, followed by semantic multilinear component analysis for feature extraction. The DSM-SVM method, utilised by the classifier, is known for its precision when predicting attacks on a given target. To counteract the danger, it is necessary to first pinpoint the source of the attack before dropping the harmful traffic generated by bots and absorbing the remainder.

The suggested approach is evaluated using the KDD dataset, the flow diagram of which is shown in Figure 2.7. Within an SDN threat detection and mitigation environment, the learned network model is implemented. All testing is performed on an Ubuntu virtual machine that is generated within the VMware platform. The suggested classifier model is trained and assessed using the Weka software, while the SDN infrastructure is set up with Mininet and a RYU controller. The suggested approach exhibits exceptional performance, attaining a precision of 99.7 percent in both the training and testing phases on the KDD dataset, hence surpassing these (Altamemi et al., 2022; Roopa et al., 2020) methodologies.

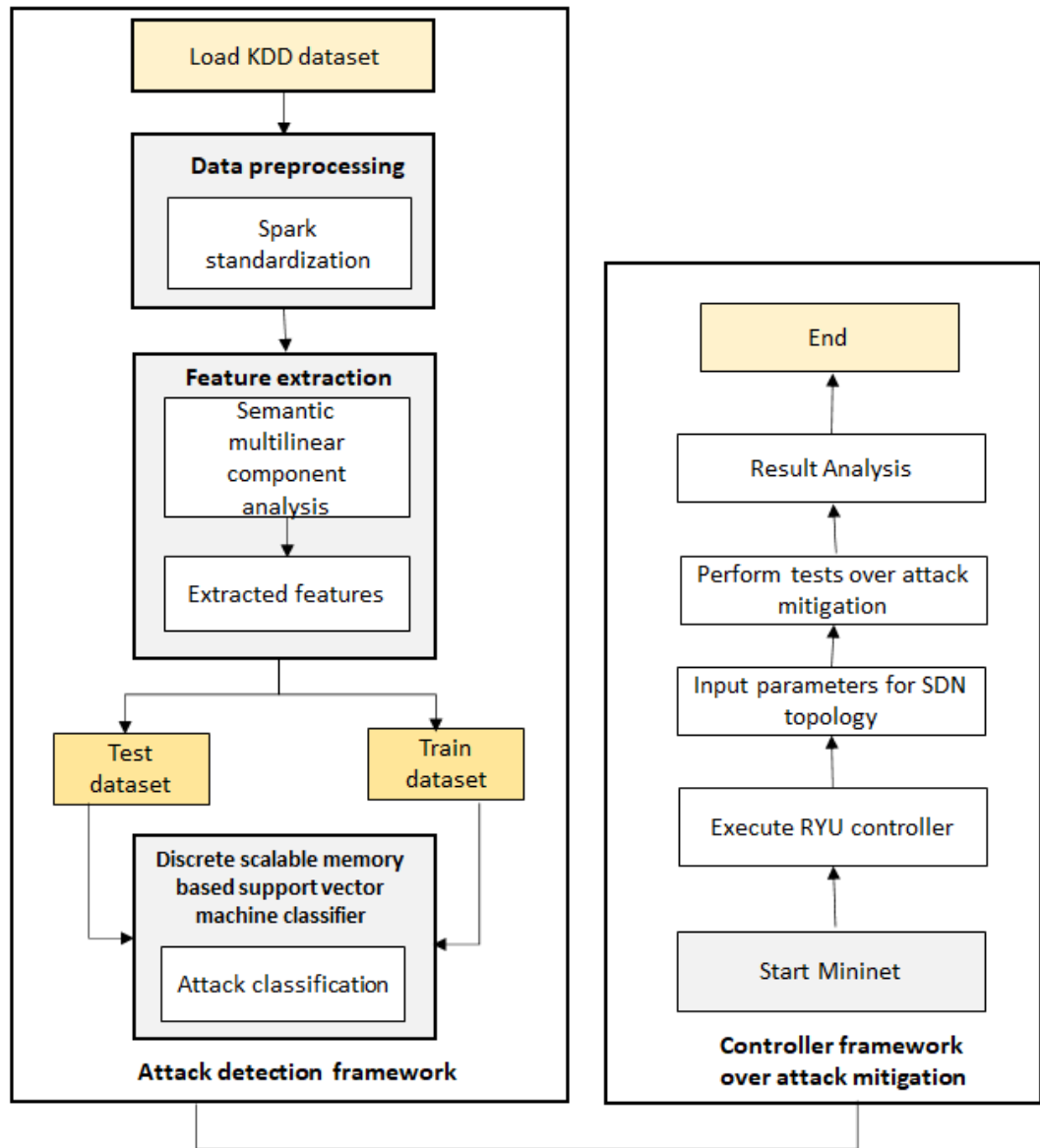


**Figure 2.7** Flow Diagram of (Revathi et al., 2022)

Together, pre-processing methods, feature extraction algorithms, and a unique classifier algorithm make up the author-suggested method for detecting and mitigating DDoS attacks in SDN networks. The suggested approach detects and mitigates DDoS attacks with remarkable precision and efficiency in an SDN setting.

#### 2.6.4.1 Strength

The suggested solution achieves 99.7% accuracy in both training and testing on the KDD dataset, offering a high degree of accuracy in recognising and mitigating DDOS attacks in SDN networks.

A thorough and efficient method for recognising DDoS attacks uses the Spark standardisation method for initial data processing, semantic multilinear component analysis for feature extraction, followed by the DSM-SVM method for classification. The suggested mitigation architecture is made to locate the attack's origin before filtering out dangerous bot-generated traffic and absorbing the remaining traffic.

Using Weka for classifier training and assessment and Mininet with a RYU controller to set up an SD Network, the suggested technique is tested in a virtual environment that simulates real-world SDN infrastructures.

#### 2.6.4.2 Weaknesses

The proposed approach is evaluated only on the KDD dataset, which may not fully simulate real-world DDoS attacks. Using a single dataset may limit the approach's generalizability. Testing the approach on more datasets would provide a more robust assessment of its effectiveness against different threats. The effectiveness of the proposed solution in a real-world SDN system is currently unknown due to the absence of testing in that environment. This method demands a lot of computational power for real-time monitoring, feature extraction, and classification, which may restrict its scalability and efficiency in large networks with high traffic. This may lead to systemic delays or performance limitations in the detection and mitigation of attacks.

Although the scalable memory based algorithm's accuracy is vital to the suggested approach, its performance can differ based on data characteristics and network conditions, which can affect its effectiveness in some situations.

## **2.7 Related Works**

This section gives an in-depth review of several methods used in recent years to defend SDN against DoS/DDoS attacks.

Mousavi et al. (2018) introduces an entropy-based method for preventing DoS attacks in SDN with centralized management. DoS attacks can be detected by monitoring for entropy changes at an IP address. Variation in the incoming data packet request is one way to detect DoS attempts. Entropy rises, as uncertainty rises and vice versa. Both the window's size and the threshold's setting control how much entropy is produced. The entropy of a set of data measures how unpredictable or random the data is. In order to detect suspicious activity that could indicate a system attack, computing the entropy of an incoming packet is a useful tool for packet analysis. The barrier point will always be set at fifty; however, the height of the barrier will vary depending on the dimensions of the window. Using the packet's IP address, the entropy of an incoming packet can be calculated. When the entropy value of such an attack reaches a specified requirement or threshold, it is considered a DoS attack. On the other hand, looking at the temporal range can improve precision and yield better outcomes. Here the temporal range is the specific time period in which network traffic or events are examined. Through the analysis of data within defined time intervals, the system is able to identify recurring patterns or irregularities that may arise and persist over time. However, this approach can only be used on centralized SDNs and on a single host at a time.

To prevent damage from DDoS attacks and maintain a safe and effective SDN networking environment, Safaa (Mahrach et al., 2020) suggested a lightweight and feasible mitigation technique. To distinguish between valid requests and malicious ones in SYN flood attacks, the authors employ the SYN cookie paradigm. The efficiency of this approach in combating SYN flood attacks is enhanced by its independence from the storage of the TCP connection's state. Nevertheless, several essential prerequisites must be fulfilled before this approach may be efficiently implemented.

The tool LineSwitch was developed by Ambrosin et al. with the purpose of monitoring and proxying every incoming controller connection. Its main objective is to safeguard against possibly unwanted Internet Protocol addresses (Mohammadi et al., 2017). This technique is designed to only protect against TCP SYN flood attacks, which involve the deployment of synchronized floods to launch a DoS attack. LineSwitch incorporates an integrated blacklist to prevent the transmission of extra packets by scrutinizing all incoming TCP connections from each IP address and intercepting those that are detected as malicious. This approach efficiently mitigates attacks such as TCP SYN floods, which frequently employ arbitrary IP addresses. By conducting simulated traffic testing, the authors have shown that LineSwitch effectively mitigates packet loss in the presence of TCP SYN floods.

The TCP handshake method is utilized by Avant-Guard (Shin et al., 2013) to authenticate the origin of the attack, hence reducing the impact of DoS attacks on OF switches. Once validation is complete, the controller will set up a flow rule to begin the data transmission. However, in order to fully work, this technique only enables the processing of TCP data and needs updates to the OF switches. To avoid both plane saturation, FloodGuard (Wang et al., 2015) employs a proactive flow rule analyzer and packet migration. When FloodGuard detects an overloading attack, it will send any packets that failed to be processed

because of an OpenFlow table miss to the data plane cache. The Analyzer module helps to monitor the values of sensitive variables in running programs or applications, transform these into path conditions, and finally install the flow rules as proactive ones in OF switches. Then packet-in message notification will be sent towards the controller by data plane cache. However, both the above methods focus on protecting against DDoS attacks directed against SDN. When compared to attack prevention, the importance of detection is often overlooked. Furthermore, both methods employ extra hardware, like in Avant-Guard (TCP-Proxy) and in Flood Guard (data plane cache), that is incompatible with the standard OpenFlow protocol.

In order to protect the controller against DoS attacks, Kandoi et al. (2015) set a rate limit on the number of packets sent in its direction. They offered flow aggregation as a way to prevent flow table overflow and provided an appropriate timeout value for flow rules. This method mitigates DoS attacks, but if the rate restriction is too low or flow aggregation is too severe, it can lose valid traffic. SDN setups must carefully tune thresholds to prevent deleting acceptable data with malicious packets to mitigate attacks and retain network functionality. Belyaev et al. (2014) introduced a load-balancing strategy to enhance the overall survival of SDN by dividing the congested lines with other routes between switches. This was done in order to maximize the amount of traffic that may pass through the network. There must exist a different path that can reduce the weight in order to implement this method. In order to fully know the connected host, Oktian et al. (2014) developed a program that stores their MAC and IP addresses in a binding table. When a person leaves or joins a host, the information about that host is updated accordingly. In order to provide DoS prevention, the application gathers three essential OpenFlow messages: `packet_in`, `flow_removed`, and `port_status`. These messages serve as essential sources of information

regarding arriving packets, changes in the flow table, and port statuses. The programme uses different properties to detect abnormalities and respond to DoS attacks by analysing each message independently and together. SDN controllers can avoid DoS attacks by regulating traffic flow and decreasing overload with rate limiting, flow aggregation, and timeout values. Rate-limiting limits the number of packets transmitted to the controller in a specific time period to avoid overloading it with packet\_in messages. This protects the controller against malicious traffic, allowing it to process valid traffic. Flow aggregation prevents the controller's flow table from becoming overwhelmed by merging comparable flows into one entry. Flow rule timeouts clear up space and ensure resources for future lawful flows by automatically eliminating idle or expired flow entries. These methods manage traffic and optimize SDN resource consumption to prevent DoS attacks.

In order to defend the network from DoS attacks, Shoeb et al. (2016) used a trust-based method that included deploying an IP-based trust value for every new packet, depending on the communication history of the packets. This approach is for DoS attack protection, and it enhances network performance by prioritizing valid traffic, minimizing false positives, and enhancing detection and countermeasures by categorizing packets based on trust levels. To mitigate the consequences of DoS attacks on the controller and the control channel, Imran et al. (2019) suggested a parallel flow installation (PFI) paradigm. Upon receiving the flow request, this model will install flow rules on all switches between the source and the destination. Imran's PFI paradigm improves network performance by accelerating data flow, reducing latency, and enhancing efficiency by preconfiguring switches, making it suitable for SDN systems.

In order to mitigate DDoS attacks, Yang et al. (2020) designed SD-Anti-DDoS, a system consisting of four major components: attack trigger, detection, trace back, and

defense. The system employs a backpropagation neural network to detect the attack, track its source, and block it by establishing flow restrictions. However, this mechanism will effect performance while using different OF protocols. Packet-in messages entering from several switches are queued to prevent DoS attacks via multilayer queuing in (Zhang et al., 2016). Requests are retrieved from the queues using a weighted round-robin algorithm. Depending on the number of ports, a switch's queue can be broken down into separate queues. It is also possible to combine the queues of several switches into a single queue. Peng et al. (2018) demonstrated DDoS attack detection system for SDN that relies on the centralised management. Pre-processing and anomaly detection modules are what make up this component. The flow feature vectors are standardised and normalised by the pre-processing function, and the anomaly detection method then labels those whether they are legitimate or normal.

A method utilizing statistical analysis and ML was presented by (Banitalebi et al., 2021). for the detection of malicious flows. This approach integrates KNN into a cohesive methodology for detecting attacks by examining the imbalanced characteristics and rapid fluctuations in network traffic. Furthermore, Tan et al. (2020) employed this hybrid KNN method to enhance the precision of detection. Studies conducted by Ahmed (2021); Sahoo et al. (2020) used a SVM to identify DDoS attacks on SDN.

Proposals have been made to use artificial neural networks (ANN) to detect DDoS attacks, both frequent and rare, by imitating the neural networks found in human brains for data analysis. In order to improve the accuracy of detecting DDoS attacks (Singh et al., 2020) employ a dynamic multilayer perceptron (MLP) with a feedback mechanism. However, this MLP strategy depends on a subset of attributes that may fail to distinguish legal and malicious communication patterns, exposing a drawback. The researcher (Ye et al., 2018)

investigated DoS/DDoS detection in an SDN context by employing a SVM built on MLP for detection. They evaluated six-tuple feature values, which included source IP, destination IP, source port, destination port, protocol type, and packet size. These characteristics facilitated more accurate classification of network traffic, thereby enhancing the identification of DoS/DDoS attacks in an SDN implementation. By precisely categorizing attacks using partial data, their approach improves the accuracy of detection and minimizes the production of false positives. To mitigate a DoS/DDoS attack, locating its source is essential. SVM-based traffic categorization assists in identifying hostile flows, allowing for a more focused response. Among the DDoS classifiers here, the SVM classifier is well-liked because of its high accuracy and low false-positive rate (Tuan et al., 2020; Xu et al., 2017). Study demonstrates that SVM is an effective solution for early detection of DDoS attacks utilizing entropy to ascertain the degree of unpredictability in flow data that are provided in (Kumar et al., 2018; Sahoo et al., 2018). Since static threshold settings in traditional detection approaches fail to account for network traffic fluctuations, Chuang et al. (2022) propose a solution with principal component analysis (PCA), PCA examines gathered traffic data to develop novel models that more effectively capture the fluctuations, therefore enabling more precise forecasting and identification of DDoS attacks. According to Alshamrani et al. (2017), the current safeguards against DDoS attacks are insufficient. Hence, they looked at how improper conduct and fresh flow attacks affect SDN. In order to deal with the unexpected changes in traffic that happened in the SDN architecture at the time of the attack, they regularly collected data from transmission devices on a data plane and used ML classification algorithms to that data (Sahoo et al., 2020). The analysis began by examining the packet-in communications that were sent back and forth between the transmission devices and the controller throughout the attack. Chuang et al. (2022) used three

ML methods such as SVM, J48, and NB to classify traffic. The decision tree technique J48 is used for classification, the probabilistic classifier NB is based on Bayes' theory, and the SL model SVM is well-known for its performance in binary classification problems.

All of the aforementioned research contributes to identifying and blocking DoS/DDoS attacks, however each offers only a partial answer. Although, they handle certain areas of the problem, none give a comprehensive framework that unifies different approaches to comprehensively solve DoS/DDoS attack detection and mitigation. Each method gives useful information but leaves opportunity for additional improvement in areas such as performance. We find that DoS/DDoS attacks in SDN can be identified and mitigated by different methods such as, inclusion of hardware requirements, packets drop rate, the blocking of malicious traffic at the port and switch levels, false alerts, detection accuracy and computational overhead. ML based systems require a lot of computational resources, which might slow network performance. In order to guarantee that ML-based solutions work well and efficiently in real-time with limited resources, it is essential to address these problems. In end, I will compare my work with these benchmarks (Imran et al., 2019; Tang et al., 2021; Wang, 2021) mentioned above. An overview of the relevant studies is presented in Table 2.3.

The literature provides a comprehensive analysis of various methods to detect and mitigate DoS attacks in SDN. However, it highlights the need for further research to understand the trade-offs between detection precision and network stability. FloodDefender, a protocol that can be expanded, can potentially reject harmless packets, affecting network performance. The entropy-based method, while efficient for centralized SDNs, is limited in dealing with complex, dispersed settings. Techniques like Avant-Guard and FloodGuard rely on hardware or protocol upgrades, which may not be compatible with common OpenFlow

protocols. Machine learning algorithms like SVM and KNN can identify DoS attacks but may face challenges with insufficient data, leading to potential threat categorization errors. To address these inconsistencies and deficiencies in future research, it is crucial to develop more resilient and widely applicable solutions for SDN security.

In general, SDN designs are highly vulnerable to extensive DoS/DDoS attacks. In this chapter, a general overview of this field is presented, along with specific strategies to strengthen each SDN layer. In general, experts have put out numerous strategies to combat the problems with availability caused by these attacks. Early detection of network attacks is a vital necessity for ensuring effective defence with detection accuracy. This can be achieved by utilizing machine learning algorithms and integrating them into attack detection and mitigation systems. Another crucial requirement is to reduce resource depletion and congestion at the SDN controller and along the essential control paths. Furthermore, existing methods frequently include constraints such as the need for extra hardware, delays in establishing routes, and higher computing burden. These constraints emphasize the necessity for more extensive and efficient methods that can concurrently identify and mitigate attacks. Moreover, there is an increasing requirement to provide more structured approaches to assess and measure the influence of DoS/DDoS attacks on the SDN control plane, as well as the corresponding measures to mitigate them with better performance. There is also a growing need to codify ways to classify and quantify DoS/DDoS attacks on the control layer of SDN and refine criteria for unblocking ports and hosts to prevent repeated attacks.

This thesis specifically examines the security of the SDN control plane when subjected to DoS/DDoS attacks. The analysis is conducted utilizing a tunnelling technique. The primary objective is to build upon the current research in this specific subtopic and

provide more structured approaches that depict the SDN control plane as a network of queues handling both normal and attack traffic with improved accuracy and performance. One important goal is to incorporate additional measures for more sophisticated SDN controllers and switches that utilize classification techniques to distinguish between normal and malicious flows.

## **2.8 Summary**

This chapter examined previous research on the detection and mitigation of DoS/DDoS attacks in SDN, both with and without the application of machine learning. In addition, a review of current studies and research literature was conducted.

This chapter investigates the application of SL methods in the identification of DDoS attacks within SDN's. ML is emphasized as being essential for the detection and prevention of DoS/DDoS attacks in SDNs. In this context, ANN, SVM, and KNN are frequently employed. The availability of labeled data determines whether these algorithms can be trained using supervised, unsupervised, or semi-supervised learning methods. Depending on the type of learning method, these algorithms can be learned. The significance of machine learning in the real-time detection and mitigation of DoS/DDoS attacks is emphasized in the chapter, which in turn reduces their impact and improves network security and reliability. The discussion concludes with the contributions of a variety of researchers to the detection of DoS/DDoS attacks through the use of machine learning techniques.

**Table 2.3:** Related Work Overview

| Reference/<br>Year      | Description                                                                                                                                                                                                                                           | Advantages                                                                                                                                                                                                              | Limitations                                                                                                                                                                                                                                                                                                                           |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Ali et al.<br>(2015)    | To allow efficient network verification, threat detection, remediation, and the maintenance of SDN network consistency, the authors emphasize the need for using a variety of methodologies, including data mining, machine learning, and automation. | An investigation is being conducted on network-level threats, including network verification and the application of automated methods to verify network consistency in SDNs.                                            | Author's mostly addresses broad SDN security concerns, although it does not include thorough examination of practical implementation challenges. In addition, it fails to consider the possible benefits of centralized control, such as scalability and controller failure, and neglects to tackle concerns associated with secrecy. |
| Kreutz et al.<br>(2014) | The authors suggest using virtualization technologies with purpose-built tools and applications. These technologies speed up bottom-up slicing, debugging, and troubleshooting, which may help find and fix complex system and program issues faster. | This article employed a tiered approach to meticulously analyze the state of the art in terms of concepts, ideas, and components of SDN, encompassing a wide spectrum of present solutions as well as future prospects. | The adoption of SDN, its expansion to carrier transport networks, the realization of the cloud computing model of "network as a service," or SDEs not covered.                                                                                                                                                                        |
| Dabbagh et al. (2015)   | Techniques for probing, countermeasures, reactive caching for attacks using forwarding planes, and replication for DoS attacks using control plane attacks                                                                                            | SDN solves many security issues, but it is vulnerable to new risks and attacks on the forwarding plane, control plane, and linkages between them. Several security                                                      | Exclusively centered on OpenFlow, By focusing primarily on these factors, the article may neglect other critical network security issues and lose opportunities to                                                                                                                                                                    |

|                          |                                                                                                                                                                                                                                                                   |                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                             |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          |                                                                                                                                                                                                                                                                   | vulnerabilities were addressed via preventative and mitigating methods.                                                                                                                                                                                                 | examine a wider range of attack routes and mitigation measures.                                                                                                                                                                                             |
| Kandoi et al. (2015)     | Two types of DoS attacks are particular to OpenFlow SDN networks. Authors imitate and analyze these attacks using Mininet. The switch's functionality depends on a flow rule's timeout and control plane bandwidth. They may enable DoS attacks if misconfigured. | This study evaluates two DoS attacks. To avoid DoS attacks, timeout values and control plane bandwidth must be tailored to network needs. Network operators may also use mitigating measures.                                                                           | Larger real-world topologies are not studied for mitigation and performance enhancement. Monitoring controller packet processing and network operator traffic patterns avoids rate limitation. Switch implementations do not consider SDN network security. |
| Alshamrani et al. (2017) | This research suggested an SDN-based approach to determine a host's attacker probability and a detection model. Experiments validate this strategy.                                                                                                               | Cloud center internal network attacks are analyzed using Bayesian prediction probability and attacker identification. SDN manages internal networks and ARP packets. SDNs separate attack elements from regular sections to detect intruders. We remove mistakes.       | This approach will consider it a network fault if the attack frequency is minimal and the attack feature is absorbed in the regular feature.                                                                                                                |
| Dong et al. (2016)       | This study proposes a unique way to use SDN with real-time network anomaly detection to give operators a picture of the network's security condition and the ability to rapidly react to changes.                                                                 | SDN-adaptive security event monitoring. Cheap distributed monitoring sensors listen to all traffic, and SDN reroutes suspect traffic to a comprehensive IDS system. New network management technologies enable security administrators to discover breaches accurately. | The article exclusively discusses man in the middle attacks using administrator or human competence. No program or computer can interrupt.                                                                                                                  |

|                             |                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                     |                                                                                                                                                                                                                                        |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Scott-Hayward et al. (2015) | Author acknowledged the significance of network security in the context of SDN and provided an overview of the developments, difficulties, and requirements for a comprehensive security architecture of sensitive security data.                                                                                             | Both the academic and commercial developments in the field of SDN security are included in this article.                                                                                                                            | Authors propose a centralized data repository to enhance network visibility, but also suggest resolving policy conflicts directly for real-time control. This could reduce security risks and improve the SDN framework's performance. |
| Yanbing et al. (2016)       | By separating security operations from security controls, the suggested security architecture presents a programmable and simplified framework. By lowering software module sizes, this approach streamlines the general system architecture and thereby lessens the level of difficulty and quantity of developing software. | The suggested security architecture decouples security operations from security controls in a simple and programmable framework, reducing software module sizes, software development intensity, and system security extensibility. | Depending on the intended validation, authors do not describe the layer protocol architecture, layer functionality, or layer interface standards.                                                                                      |
| Shu et al. (2016)           | For a DoS attack to overload the flow table and flow buffer, use FloodGuard, Virtual source Address Validation Edge, and FLOWVisor                                                                                                                                                                                            | Security risks that SDN networks are subject to, as well as potential defences. The data forwarding layer, the control layer, and the application layer were used to analyse the security problems.                                 | A key use for SDN will be network virtualization and middleboxes based on cloud computing, which will introduce additional security risks not covered here.                                                                            |
| Zhang et al. (2016)         | The author suggested MLFQ, a straightforward yet efficient DoS mitigation technique.                                                                                                                                                                                                                                          | Authors offer a queueing-based countermeasure for controller resource saturation. MLFQ dynamically expands and aggregates queues based on controller                                                                                | MLFQ may reduce DoS attacks depending on network environment and assault. SDN, network topologies, and traffic patterns may impact MLFQ performance.                                                                                   |

|                      |                                                                                                                                                                                                                                           |                                                                                                                                                                                                                                                                                                             |                                                                                                                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      |                                                                                                                                                                                                                                           | load to solve the problem of too many queues.                                                                                                                                                                                                                                                               | The research report should evaluate the method's applicability and scalability.                                                                                                                                                                       |
| Lara et al. (2016)   | OpenFlow-based security framework that lets network security operators write and apply human-readable security rules.                                                                                                                     | OpenSec transforms security policies into OpenFlow messages to implement them. Second, they explained how the system automatically responds to policy-defined security alerts. Third, they evaluate the framework's scalability on GENI using campus network datasets.                                      | Protecting the OpenFlow framework and ensuring the execution of policies lacked here.                                                                                                                                                                 |
| Koning et al. (2016) | VNET, a touch-table-driven interactive SARNET prototype, was created in this study. SARNET allows SDNs to autonomously respond to security attacks.                                                                                       | The paper's VNET prototype and SC15 presentation showed the qualities needed to autonomously defend against cyberattacks. We showed that concurrent network and service information enhances DDoS assault analysis. This experiment determined automated response efficacy by subtracting cost from income. | This research does not include additional aspects that affect countermeasure efficacy and cost. In this prototype, the SDN platform API restricted them from adding and deleting network components.                                                  |
| You et al. (2017)    | SDN centralises network communication for DDoS detection. This strains OpenFlow controllers. This study does not gather flow table data, isolates three Packet In characteristics for controller-switch communication, and conducts real- | This research proved that DDoS attacks enhance ipsource-address entropy and reduce destination-port entropy. Compared to flow entry inspection-based DDoS attack detection, distributed data collection reduces controller processing demand.                                                               | Due to the simulated setting, experimental data in this article is hard to verify. Real network traffic is needed for reliable data. This work did not track the assault source, which would limit DDoS attacks and develop a full defence mechanism. |

|                    |                                                                                                                                                                                                                                                                                                      |                                                                                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                              |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                    | time attack detection to reduce controller processing overhead.                                                                                                                                                                                                                                      |                                                                                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                              |
| Yan et al. (2015)  | Author mainly discusses DDoS detection and mitigation techniques like SYNcookie, Fast flux, DNS, and swarm intelligence                                                                                                                                                                              | The research acknowledges the growing prevalence and complexity of cloud computing DDoS attacks. Defensive multi-site implementation, cross-layer traffic analysis, defensive point communication, DDoS-tolerant system design, and SDN-based cloud DDoS prevention are addressed. | SDN is both a potential technique for preventing DDoS attacks in cloud computing systems and a vulnerable target. DoS cannot use this tool.                                                                                                                                                  |
| Dong et al. (2016) | This paper presents a novel idea how to exploit the software defined networking (SDN) and the real-time network anomaly detection to an efficient whole that offers for an operator both, a view of the security state of the network and the ability to immediately react to the changed situation. | SDN-adaptive security event monitoring. Cheap distributed monitoring sensors listen to all traffic, and SDN reroutes suspect traffic to a comprehensive IDS system. New network management technologies enable security administrators discover breaches quickly and accurately.   | Cheap distributed monitoring devices have poor scalability and resource use. High network traffic can affect performance and efficacy. Rerouting suspicious traffic to a comprehensive IDS system is ignored in the study. IDS systems can be hacked if redirected traffic is not protected. |
| Mot et al. (2024)  | Author proposed the host tracking service (HTS) that is included in an SDN controller allows the whereabouts of all hosts in an SDN network to be tracked and monitored.                                                                                                                             | SDN controllers incorporate HTS to centralise host tracking. Network administrators may view hosts' existence and behaviour, increasing resource allocation, security, and management. HTS shows author's creation and SDN                                                         | SDN controller's host tracking service can be poisoned. This article examined compromised host attacks on host tracking services and found that many are dangerous                                                                                                                           |

|                     |                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                            |                                                                                                                                                                                                                |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     |                                                                                                                                                                                                                                         | contribution. Host tracking enhances SDN for network operators and administrators.                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                |
| Ma et al. (2016)    | This work suggested a Bayes-based method and detection model to calculate a host's attacker likelihood. Experiments verified this approach.                                                                                             | Cloud centre internal network attacks are predicted and detected using a Bayes-based algorithm. SDN regulates internal network connectivity and handles ARP messages. We discovered SDN-based network attackers by differentiating attack and standard SDN technologies. We lowered misjudgement likelihood with few experiment omissions. | If the attack frequency is minimal and the attack feature is absorbed into the regular feature, this approach will not detect the attacker and consider the attack a network mistake.                          |
| Li et al. (2017)    | This study proposes a Bloom filter-based lightweight countermeasure. To track covert packet alterations, authors create a prototype. Bloom filter monitoring system is efficient and resource-efficient, according to their evaluation. | This article addressed OpenFlow channel IoT-Fog MitM threats. The proposed SDN architecture illustrates an attack. Three attacks are shown. They recommend the Bloom filter for MitM attacks. Bloom filter monitor prototype expands OpenFlow. Bloom filter is small and effective, per assessment.                                        | IoT-Fog MitM attacks on OpenFlow channels are the authors' primary emphasis, which may limit the findings' generalizability and application to a wider variety of network infrastructures or attack scenarios. |
| Zhang et al. (2016) | FTGuard collects traffic characteristics from the network operating system's basic service. Attack traffic differs from normal traffic in packet quantity and frequency of new flows.                                                   | This overflow attack requires FTGuard, a behavior-based priority-aware protection. They distinguish flows with different priority to soft-isolate users of different judgements. Traffic improvements that improve safety are rewarded. Evaluation scores dynamically prioritise user flows.                                               | This research did not explore new traffic aspects, but it did provide more strategy factors to construct more accurate assessment criteria. More trials should assess controller load.                         |

|                          |                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                               |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Yu et al. (2017)         | Author proposed TARN which employs numerous short-lived, seemingly random IP addresses to reduce communication session vulnerability. By breaking the static host-IP binding, TARN prevents attackers from studying source and destination addresses to discover communication sessions. | By eliminating a critical weakness in the current addressing scheme - the fixed binding between hosts and IP addresses - which has been leveraged by cyberattacks to readily pinpoint their targets - the TARN system is highly resistant to DNS/IP filtering and traffic analysis.                                                           | This study does not mention doing tests to demonstrate the value of different designs, completing two-way IP randomization, undertaking an in-depth security analysis of the SDN-based TARN solution, or deploying TARN as the future of SDX. |
| Al-Zewairi et al. (2017) | For the purpose of identifying and preventing IP and MAC spoofing attacks on the network, the authors suggest an experimental software-defined security controller based on the Open vSwitch Controller.                                                                                 | Experimental Open vSwitch Controller-based SDSec controller. SDN inhibits IP/MAC spoofing. Secures SDN.IP and MAC spoofing plague networks. This personalised strategy permits focused analysis and specific resistance to various threats.                                                                                                   | Not discussed the separation of SDSec controller from the SDN controller and implementing defense policies against other types of attacks.                                                                                                    |
| Wang et al. (2017)       | The author designed and implemented SGuard, a security application on top of the NOX controller with two modules—access control and classification—to minimise DoS attacks in OpenFlow networks.                                                                                         | IP/MAC spoofing-DoS attacks. NOX-controlled SGuard protects Open-Flow networks. SGuard will collect MAC, IP, and Port information from all hosts and add it to a Hashtable for access control when the network is set up. Traffic flows are classified by the classifier. After results, NOX controller will limit attackers with flow rules. | To get more compelling findings, SGuard should be implemented with bigger experiment topologies.                                                                                                                                              |
| Dargahi et al. (2017)    | Author used The following strategies, including access control, load                                                                                                                                                                                                                     | This study reviews the research on stateful data plane methods and highlights                                                                                                                                                                                                                                                                 | Stateful SDN data plane requires varied platforms, programming languages, and                                                                                                                                                                 |

|                          |                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                            |                                                                                                                                                                                                       |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          | balancing, mitigation, and monitoring, are employed in this study for various goals.                                                                                                                                                         | important security concerns caused by their fundamental qualities.                                                                                                                                                                                                                                                                                         | applications, hence many different techniques have been proposed in the literature. None of them have built-in security.                                                                              |
| Al-Zewairi et al. (2017) | DoS attacks can lower network capacity, which would stop real activity. A way out is performance-awareness. SFlow is used in software-defined networking to continuously watch over networks. sFlow and OpenFlow defend against DoS attacks. | This study uses sFlow-RT to achieve performance conscious networking. REST flow API converts continuous data from sFlow agent to actionable metrics. Applications utilise API to monitor network changes and deliver commands to SDN OpenFlow controller.                                                                                                  | This research only addresses DoS attack detection and mitigation utilising performance-aware software-defined networks. Protecting the single controller will assist with bandwidth fatigue.          |
| Chin et al. (2018)       | This article employed two strategies.<br>1) Rule-based strategy utilising the Snort tool<br>2) Anomaly-based methodology with the BRO tool.                                                                                                  | This study provided rule-based and anomaly-based SDN DoS attack mitigation strategies. This solution mitigates DoS attacks using SNORT and BRO IDS. They created the SDN setup for DoS attack origination, detection, and mitigation. They detected and mitigated DoS attacks using SNORT and BRO attack signatures and scripting log file packet history. | These methods are only applicable against DoS techniques weren't used to counteract DDoS attacks. As a result, stopping a single source of the attack basically makes it impossible to halt it.       |
| Wang et al. (2017)       | SDN secure Controller (SECO) is an innovative and simple controller-based detect and defence technique that improves SDN security against DoS attacks.                                                                                       | SECO algorithm is useful to identify and fight SDN controller DoS attacks. The controller collects global network information; therefore the suggested approach uses switch port statistics                                                                                                                                                                | This simplified approach might result in false positives or false negatives in attack detection by omitting other crucial signs or attack paths that are not only reliant on switch port information. |

|                          |                                                                                                                                                                                                                                                                             |                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                                                                                                             |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          |                                                                                                                                                                                                                                                                             | performance. Attack source placement allows SECO to prevent, detect, and stop the attacker using these statistics.                                                                                                                                                                                |                                                                                                                                                                                                                                                             |
| Mohammad i et al. (2017) | In this research, we provide SLICOTS, a practical and effective defence against TCP SYN flooding attack in SDN. To recognize and stop assaults, SLICOTS makes use of SDN's dynamic programmability.                                                                         | The lightweight SDN SYN flooding security improvement SLICOTS was explored. SLICOTS monitors all network TCP connections to prevent SYN flooding. After TCP connection validation, SLICOTS creates permanent forwarding rules between client and server. SLICOTS stops half-open TCP connections. | Another SYN flooding exploit involves the attacker completing TCP three-way handshaking but without sending any data packets. This work cannot handle this attack since OPERETTA migrates TCP connections.                                                  |
| Gao et al. (2018)        | Authors proposed FlowKeeper, a standard framework to design an attack-resistant data plane. FlowKeeper ensures data plane port control and minimises control plane strain by screening unlawful traffic.                                                                    | First, the authors examine three data plane attacks DoS, topological poisoning, and side-channel and their defences. FlowKeeper, a typical design, defends against these three threats. FlowKeeper prevents DoS and topology poisoning attacks.                                                   | The study excludes side-channel attacks and solely provides FlowKeeper to guard against DoS attacks and topology poisoning attacks. The comprehensiveness and overall efficacy of the suggested solution may be constrained by this omission.               |
| Dacier et al. (2017)     | This article uses SDN to identify possibly harmful frames and defend business WLANs with reactive and proactive defence measures. Propose a RADIator-based business WLAN security architecture. RADIator creates controller-based wireless networks with centralised packet | This study used RADIator SDWN architecture to create a novel business WLAN security technique. They exposed IEEE 802.11 MAC and PHY layer vulnerabilities. Based on this, they realized wireless communication. Their approach detects dangerous frames and provides a virtual WLAN barrier.      | Machine learning applications that identify abnormalities across wireless network layers will not operate with this design. The method reduces communication overhead and improves scalability by dynamically shifting jobs from the controller to the APs. |

|                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                     | processing and configurable Access Points (APs) using SDN.                                                                                                                                                                                                                                                                                                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| SECOD<br>(Wang et al., 2021)        | SECOD is a detection approach designed for managing DDoS attacks but has limitations. It lacks clarity on the maximum packet_in messages a switch can handle, making it less effective under high traffic loads. The detection thresholds are set too high, risking undetected attacks, and it struggles with low-volume attacks. While effective against application-layer DDoS attacks, its efficiency in addressing lower-layer attacks is limited. | The approach lacks a clear indication of the maximum number of packet_in messages a switch can transmit per second, which may impact its ability to handle high traffic volumes effectively. Additionally, detection thresholds are set too high, increasing the risk of DDoS attacks going undetected. The system also exhibits limitations in detecting low-volume attacks, which can evade its detection mechanisms. Furthermore, while it is primarily effective against application-layer DDoS attacks, its efficiency in addressing attacks targeting lower network layers remains limited, reducing its overall versatility. | The system faces several challenges, including the absence of clear indications regarding the maximum number of packet_in messages a switch can transmit per second, which may hinder its ability to manage high traffic efficiently. High detection thresholds further exacerbate this issue, potentially allowing DDoS attacks to go undetected. Additionally, the system struggles with identifying low-volume attacks, limiting its ability to address more subtle threats. While it demonstrates effectiveness against application-layer DDoS attacks, its efficiency in mitigating attacks targeting lower network layers is limited, reducing its overall adaptability and coverage against diverse attack vectors. |
| FloodDefender<br>(Gao et al., 2020) | FloodDefender is a network defense mechanism for mitigating DoS attacks in OpenFlow-based environments. While effective, its reliance on OpenFlow limits versatility, and its high computational and memory demands make it less suitable for resource-constrained systems. Challenges like detoured traffic misidentification, packet                                                                                                                 | The hybrid approach combines detection and mitigation modules to effectively counter DoS attacks. It incorporates precise attack detection mechanisms based on frequency, ensuring accurate identification of threats. Additionally, the system provides comprehensive mitigation functions to address these attacks effectively. A key feature of the approach is the table-miss engineering                                                                                                                                                                                                                                       | The approach relies heavily on OpenFlow-based networks, which may limit its applicability in diverse network environments. It also demands high computational resources, posing challenges for resource-constrained systems. Potential misidentification of detoured traffic and issues such as packet bouncing can reduce the system's overall effectiveness. Furthermore, longer delays                                                                                                                                                                                                                                                                                                                                  |

|                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                   | bouncing, and delays in legitimate flows can impact performance. Additionally, the lack of a fixed threshold in its frequency-based filter affects detection accuracy, posing reliability concerns.                                                                                                                                                                                                                                                                                                                   | module, which intelligently distributes table-miss packets to neighboring switches. This reduces bandwidth consumption and prevents congestion at the victim switch, thereby enhancing overall network performance and resilience against DoS attacks.                                                                                                                                                                                                                          | may occur for innocuous flows, impacting user experience. The filtering process increases memory usage for the controller, adding to resource strain. Additionally, the lack of a fixed threshold in the frequency-based filter affects the accuracy of detection, potentially compromising the reliability of the approach.                                                                                                                                                                                                                                                                                                   |
| DSM-SVM<br>(Revathi et al., 2022) | DSM-SVM is a classification algorithm designed to enhance the detection and mitigation of DDoS attacks. While it ensures high accuracy in identifying attack patterns, the technique's reliance on significant computational resources can pose challenges for large-scale network deployments with resource constraints. Its effectiveness, although proven on the KDD dataset, may vary when applied to diverse datasets or complex real-world scenarios, raising questions about its reliability and adaptability. | The approach demonstrates a high accuracy rate of 99.7% during both training and testing phases on the KDD dataset, showcasing its reliability and effectiveness. It employs the DSM-SVM method for classification, enabling precise detection and mitigation of DDoS attacks. A key focus of the system is on identifying the origin of attacks and filtering out bot-generated traffic, ensuring enhanced network security and minimizing the impact of malicious activities. | The suggested technique may require substantial computing resources, which can present challenges for implementation in large-scale networks with resource constraints. While the DSM-SVM algorithm is integral to ensuring high accuracy, its effectiveness could be limited in certain situations or when applied to diverse datasets, raising concerns about its reliability in varying scenarios. Additionally, the solution's testing has been confined to the KDD dataset, which may not fully represent the complexities of real-world network environments, potentially affecting its generalizability and robustness. |
| DAISY<br>(Imran et al., 2020)     | DAISY is a lightweight system designed for detecting and mitigating DoS attacks in SDN environments. Its effectiveness relies on high-quality and comprehensive data collection, which may be affected by network                                                                                                                                                                                                                                                                                                     | DAISY employs a statistical approach to manage DoS attacks in SDN environments by installing specific flow rules rather than completely blocking ports, IPs, or MAC addresses. This lightweight solution effectively detects and mitigates attacks                                                                                                                                                                                                                              | The effectiveness of DAISY's threat detection and mitigation heavily depends on the quality and completeness of the collected data, which can be influenced by network conditions or sophisticated attack strategies. While it is designed as a                                                                                                                                                                                                                                                                                                                                                                                |

|                    |                                                                                                                                                                                                                                                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                    | conditions or advanced attack strategies. While DAISY minimizes resource overhead compared to other solutions, it can still introduce processing demands, particularly in large-scale deployments with substantial traffic volumes. The system includes dynamic threat management capabilities, allowing it to adjust to changing attack scenarios. | while minimizing resource overhead. It features an efficient data collection and threat detection system, ensuring accurate identification of malicious activity. Additionally, DAISY dynamically manages threat entries and adjusts block durations based on threat values, allowing it to adapt seamlessly to evolving attack scenarios and maintain network stability. | lightweight solution, DAISY may still introduce processing overhead, particularly in large-scale SDN deployments with high traffic volumes. Although it offers mechanisms for dynamic threat management, it may face challenges in quickly adapting to emerging or unknown attack patterns without manual intervention or updates. Furthermore, the system demands significant computational power, which can pose scalability issues in resource-constrained environments. |
| Abid et al. (2024) | Two SOM-based DDoS attack detection methods were suggested.<br>1. Hybrid machine learning<br>2. Som distributedcenter classification method after som training.                                                                                                                                                                                     | This paper presented an SDN-based DDoS detection system. SDN and five typical traits make the system lightweight and DDoS-resistant. The suggested approaches outperform typical detection methods for many purposes. This work mimics network traffic.                                                                                                                   | The aspects stated in the paper could vary depending on the host type that has to be protected and the sort of attack that might occur. It is proposed to research techniques that aim to automate feature selection.                                                                                                                                                                                                                                                       |
| Nisa et al. (2024) | The TPAAD technique is for detecting and then for mitigating DoS attacks on SDN systems. It identifies malicious network traffic and redirects it to a mitigation mechanism. The technique uses SVM and KNN for efficient identification of DoS attacks.                                                                                            | <ul style="list-style-type: none"> <li>- Reduces false positives</li> <li>- Minimizes CPU utilization and control channel bandwidth consumption</li> <li>- Improves packet delivery ratio</li> </ul>                                                                                                                                                                      | <ul style="list-style-type: none"> <li>- Vulnerability to evolving attack methods not addressed</li> <li>- Limited testing environment</li> <li>- Scalability and performance in larger, real-world networks not tested</li> </ul>                                                                                                                                                                                                                                          |

## **CHAPTER 3**

### **METHODOLOGY**

#### **3.1 Introduction**

The growing popularity of SDN raises new security concerns. One such difficulty is the possibility of DoS attacks on SDN systems, with OpenFlow switches being a prime target. In response to a substantial flood of packets intended for unidentified addresses, the switch initiates a query to the controller in order to obtain new flow rules. This can cause network bottlenecks by placing excessive demand on SDN resources like as switch memory, control channel bandwidth, and CPU utilization, leading to decreased network speeds, more packet loss, and reduced detection accuracy.

A system called Two-Phase Authentication of Attack Detection (TPAAD) is proposed as a solution to address the issues mentioned above. The system's primary objective is to detect and counteract DoS attacks with as little system load as possible. Modules for detection and mitigation are built into the SDN network's control layer as part of the TPAAD system's design. The detection component watches the network for potential threats using a two-step authentication procedure as shown in Figure 3.1. Firstly, packet filtering analyzes packet flows to identify any unusual activity. The proposed solution implements ML based classifier models, such as, SVM and KNN, for the purpose of detecting and countering harmful attacks. These models work especially well because they lessen the requirement for a large amount of data storage and processing power on the control plane. There are a number of stages in the TPAAD process. When an SDN switch receives a packet, its first task is to check whether the packet matches an existing flow entry in its flow table. For packet filtration, the switch looks at the packet's source, destination, and receiving port then compares it to a flow database. The packet is processed in accordance with the entry it finds

in the flow table. In the event of a table miss, the switch will notify the SDN controller through an OpenFlow packet\_in message. Next, the controller will send the messages to the OpenFlow Mod messages (OpenFlow Flow Modification) switch, which are used to modify the flow table by adding, updating, or removing rules. With these messages, the controller can determine suitable forwarding rules for each packet according to its destination address. The controller also filters out unwanted data. ML models are then trained on the CICDoS 2017 dataset to accurately identify and detect attacks. However, data cleaning or pre-processing is performed on the dataset in order to correct mistakes, missing values, and unsuitable features before the models are trained. Overfitting is mitigated and training time is shortened by using filtering approaches to pick the best characteristics for attack detection. DoS attacks are detected using a dataset and a malicious queue in conjunction with SVM and KNN models. Once an attack has been identified, the mitigation module is triggered as shown in Figure 3.1. Through the use of tunnelling, attack traffic is managed effectively while the controller's computational bandwidth is protected. Upon receiving a packet, the module examines its header information to determine whether or not it poses a threat based on packet rate and potentially malicious content. Mechanisms that temporarily and permanently prohibit access are used to reduce the impact of current attacks and stop further attacks from the same source. The thresholds at which the block on the blacklisted source IP is lifted are also checked by the module.

The proposed TPAAD system uses the Ryu Controller as its primary SDN controller, centralising detection, and control of DoS attacks. OpenFlow switches communicate and receive instructions from the controller through control channels. In order to detect abnormalities and attack patterns, the controller examines information from the switches and the network traffic. When an attack is identified, the mitigation module is activated and

protective measures are taken, such as rate limitation, packet filtering, traffic redirection, and dynamic flow entry modifications. A number of metrics, including CPU utilisation, control channel bandwidth, packet delivery ratio, and reaction time, are used to assess the performance of the TPAAD system. These metrics measure how well a system operates both in typical network conditions and while under attack from a DoS attack. Overall, the TPAAD system offers a reliable method for identifying and mitigating DoS/DDoS attacks in SDN-controlled infrastructure.

### **3.2 Model Overview**

A DoS attack can be readily carried out against a system that employs the SDN architecture by the attacker by flooding the OpenFlow switch with packets that have random destinations. As a result, each packet is sent from the OpenFlow switch to the controller in search of a revised flow rule. This attack will have consequences for several parts of the SDN architecture due to the consumption of their resources. The system involves key resources such as switch storage, control channel bandwidth, and CPU utilization. Excessive resource usage can lead to decreased performance, increased delays, packet loss, and reduced detection accuracy. When using SDN, a DoS attack may not completely disable the network or any of its components, but it may cause data delivery issues due to an overflow of network resources.

In order to better understand the proposed architecture, it is helpful to first examine how a network managed by SDN operates. In specifically, the IP addresses of incoming data packets are verified against the local flow table to verify if there is an existing entry, or match, with a flow forwarding rule. This is known as a "hit" on the table, and the packet will be processed or transmitted in accordance with the rule set. If no match is detected (a table miss), the switch will submit a search request to the SDN controller, maybe through an

OpenFlow packet\_in message. When a packet's destination is identified, the SDN controller performs a lookup and, if necessary, sets forwarding rules on the relevant network switches along the packet's path by sending OpenFlow Flow Mod messages. The SDN controller can configure rules to filter out undesirable traffic. It manages and organizes incoming traffic requests in a manner similar to a queuing system within the control plane architecture as shown in Figure 3.1.

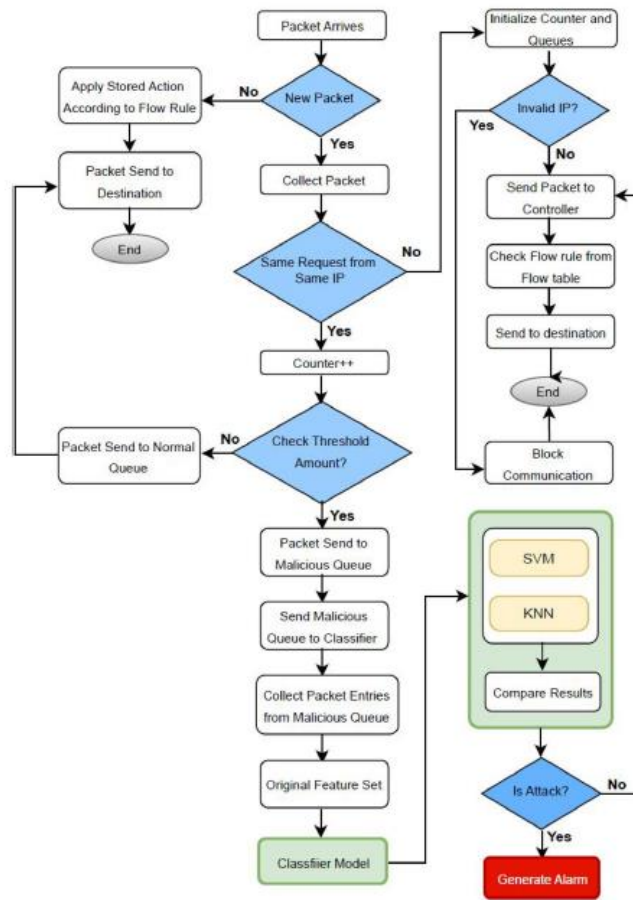


**Figure 3.1** Flow Chart of TPAAD

### 3.3 Proposed System Architecture

As illustrated in Figure 3.1, the proposed solution consists of detection and mitigation modules. Within the detection module, TPAAD effectively detects and precisely identifies DoS attacks. Upon detection of an attack, the system activates a mitigation module that is

incorporated into the control layer. This module simulates and mitigates the effects of DoS attacks on networks controlled by SDN.

In order to mitigate the impact of SDN-targeted DoS attacks, we have devised the TPAAD Scheme. This system is designed to precisely identify such attacks while efficiently utilizing network resources, such as bandwidth, compute, and flow table space. To further mitigate the impact of DoS attacks, we have also developed a mitigation module. Detailed information regarding the development process of TPAAD is provided here below. This includes the proposed system architecture, the flow of its processes, and the integration of the mitigation module.

### **3.3.1 Two Phase Authentication of Attack Detection (TPAAD)**

In the beginning, the detection module will proceed to keep an eye on the current state of the network for any signs of an incoming attacks using TPAAD. And the activation of the mitigation module is dependent on the occurrence of attacks as shown in Figure 3.1. By employing a two-step authentication procedure as shown in Figure 3.1, the Two-Phase Authentication System in the detection module seeks to improve the accuracy of attack detection. This approach assists in reducing false positives and enhancing the attack detection mechanism's general dependability.

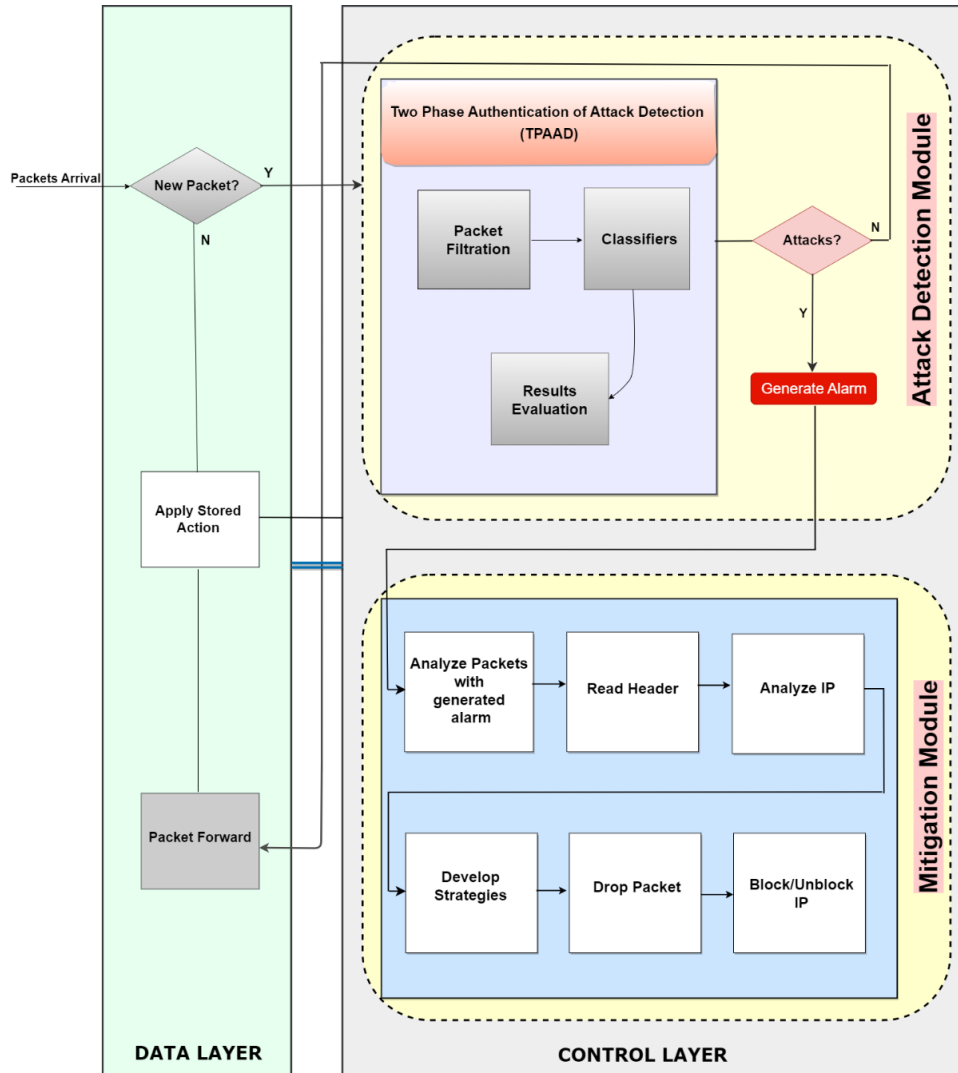


**Figure 3.2** Propose TPAAD Solution

A single First-In First- Out (FIFO) buffer is used by default, with no attempt made to distinguish between normal and malicious requests forwarded from the data plane switches.

To address this, an effective way is introduced here for the configuration of SDN controller and to make more work efficient by connecting detection module with SDN RYU controller, where detection program will be run to detect DoS/DDoS attacks as shown in Figure 3.2. The detection module uses a TPAAD approach. Firstly, it is based on packet filtration, which will identify abnormal traffic by analyzing packet flows, and then to pass

this abnormal traffic flows to ML based models for effective detection and management of malicious attacks. These ML based models SVM and KNN are more efficient at differentiating between malicious and benign packet flows, reducing the burden on the control plane's processing resources and preserving available communication bandwidth.

#### 3.3.1.1 Working Procedure of TPAAD

The TPAAD scheme is SDN based solution designed to identify flooding DoS attacks based on stored controller statistics. It encompasses fundamental routing functionalities that perform in a similar way to traditional routers in packet processing. The attack detection module in TPAAD primarily focuses on packet-in messages, which it determines based on the quantity of such messages received at a specific time and the controller's ability to process them. The attack detection model filters and analyses network data packets through a sequence of phases, each containing specialized functions which are mentioned below:

#### 3.3.1.2 Packet's Filtering

A data packet is sent from the sender to the SDN switch once the sender logs into the network using *sender.login-func()*. The first thing a switch does upon receiving a packet is checked by *S\_switch.receive\_func(Pkt)*. The SDN switch performs a packet verification process to determine the uniqueness of the received packet and to identify any duplicates. The SDN switch looks at each received packet to see whether it is unique in a preliminary sense. This involves examining the packet's attributes to spot and filter out any duplicates. By comparing incoming packets with those already observed, the system avoids processing the same packet more than once. This step is essential for maintaining network efficiency and preventing the waste of resources on duplicate packets. Through the utilization of this function, it is ensured that every data packet is processed just once, which prevents

duplication of packets from utilizing network resources that are not essential. So this mechanism employs a function *chk\_duplicate(Pkt)* that does a comparison between the arriving packet and previously received packets in order to identify its distinctiveness.

For packet duplication, first it needs to extract important features from incoming packets. In Mininet, if an SDN switch (Open vSwitch) does not have a corresponding flow entry, it will communicate with the SDN controller by sending a "Packet-In" message. This message is then utilized by the SDN controller to extract vital features of the packet, such as the source and destination IP addresses, source and destination ports, packet sequence no, MAC address and time stamp. The packet feature extraction process is performed using the Python-based Ryu controller parser, enabling the extraction of protocol layers. Packet features are obtained by utilizing the '*packet.Packet*' class to parse incoming packets, allowing the Ryu controller to process them.

A *Hash\_Packet(Pkt)* function is applied to aggregate and analyze the extracted features in order to create a unique packet identifier. By quickly comparing the hashed key of a received packet with a stored set of hashes, the controller after checking threshold amount will send these duplicated packets towards malicious queue for further operations.

#### 3.3.1.3 Flow Table Process

When a packet is received by an SDN switch, the switch must determine what action to do with the packet. The rules in its flow table determine what happens to packets that meet particular requirements, such as IP addresses, MAC addresses, and port numbers, and this choice is based on those rules. The flow entries are installed into the switch by the Ryu controller, which also handles those cases where no matching flow is discovered in the flow table (It's called "table miss") as shown in Figure 2.2.

The flow table function effectively performs by systematically assessing each flow entry in a sequential manner to ascertain its correspondence with the incoming packet. The initial stage of this procedure involves the switch gathering essential characteristics from the packet, including source and destination IP addresses, MAC addresses, ports, and other pertinent data. The previously extracted attributes are subsequently utilized to conduct a comparison with the match fields that have been defined in each flow entry inside the flow table.

The switch then compares the retrieved key characteristics to the match fields in each flow entry in a systematic manner, beginning with the most important entry. By employing this approach, the evaluation process prioritizes the assessment of the most particular or critical regulations. In the event that a match is detected, indicating that the attributes of the packet agree to the match fields of a flow entry, the switch proceeds to carry out the action specified by that entry as it will be forwarded towards destination. The aforementioned action may encompass the routing of the packet to a designated port, the elimination of the packet, or the execution of additional network operations in accordance with the prescribed flow rule. In the event that no match is detected inside any of the flow entries, a table miss event is triggered, thereby initiating an alternative handling procedure and that packet will be handed over to SDN controller for subsequent decision-making. This process will be done by using equation 3.1

$$PktForward(Pkt, flow_{entry}) = \begin{cases} True \\ False \end{cases} \forall x, Pkt.Feature(x) = flow_{entry}.Match_{Field}$$

Equation 3.1

Here  $Pkt.Feature$  denotes the  $xth$  attribute derived from the incoming packet, such as the source IP, destination IP, source MAC, destination MAC etc and  $Match_{Field}$  denotes the  $xth$  match specified in the flow entry of the flow table of switch.

#### 3.3.1.4 Rate limiting Threshold Check

The detection of malicious packets is crucial in SDN, as it plays a vital role in maintaining network security and optimizing performance. Two key indicators for identifying potentially malicious traffic are frequency, which refers to the rate at which packets are received, and time, which denotes the interval between successive packets from the same sender. By analyzing these parameters, the SDN controller can detect unusual patterns that may signify a DoS attack, network scanning, or other forms of malicious activity.

The frequency of flow entries helps identify malicious traffic by characterizing the number of packets in each flow as they enter the switch for this equation 3.2 used here.

$$Fq_{pkt} = \frac{TN_{pkt}}{T_a - T_{a-1}} \quad \text{Equation 3.2}$$

Here the packet frequency from a specific source is denoted by  $Fq_{pkt}$ .  $TN_{pkt}$  is the total number of packets transmitted by the source in a certain time frame.  $T_a - T_{a-1}$  measures the duration between the arrival of the  $a$ th and  $(a - 1)$ th packets from the same source. Both the size of the SDN and usual traffic patterns are taken into consideration while determining this frequency threshold level.

Attack traffic is typically designed to have a lower frequency compared to normal flows, as the goal is to flood the target with malicious traffic. Setting frequency and latency criteria allows for suspicious packet detection in equation 3.3. If a packet is determined to originate from the same IP address and exhibits abnormal characteristics, it is flagged and added to a malicious queue. These packets are then processed through a classifier model, where the best assessment results are used to identify potential attacks.

$$Abnormal_{pkt} = (Fq_{pkt} > ThFq_{pkt}) \vee ((T_a - T_{a-1}) < ThT_{pkt}) \quad \text{Equation 3.3}$$

Here  $Abnormal_{pkt}$  is the malicious packet if it meets the above criteria as mentioned in equation 3.3,  $Fq_{pkt}$  is the sending source packet frequency,  $ThFq_{pkt}$  is the threshold frequency for a specific defined network,  $T_a$  and  $T_{a-1}$  are receiving packets timestamps and at last  $Th_{pkt}$  is the threshold time.

Frequency threshold can be adjusted to 40-50 packets per second for networks with up to 10-15 hosts, and 90-100 packets per second for networks with more beyond that. To identify abnormal traffic as shown in equation 3.3, TPAAD continuously runs upon gathered data, with an iteration period between successive iterations. In accordance with the requirements of the network, this time period can range anywhere from 3 seconds to 12 seconds, and this refers to the speed at which the controller desires to identify and stop harmful traffic within its computational capacity.

### 3.3.1.5 Queuing Abnormal Packets

To prevent the needless loss of valid traffic owing to transient mismatches or the absence of specified flow rules, this scenario necessitates particular management. The steps include buffering the packet, fixing the table miss through processing or flow rule updates, and finally, deciding what to do with the packet.

When it comes to preserving the stability and performance of a network, one of the most important aspects is how an SDN switch handles table misses. In the event that a table miss happens, and the packet will be considered as abnormal packet and will be sent towards the malicious queue rather than dropped, which prevents the loss of possibly valid traffic that would otherwise be unneeded this can be done using formula mentioned in equation 3.4.

$$Malicious_{Queue} = Malicious_{Queue} \in Abnormal_{pkt} \quad \text{Equation 3.4}$$

Here  $Malicious_{Queue}$  denotes the queue where abnormal packets will be stored until it's status will be resolved by the controller. So, to avoid dropping packets that might

be part of a valid flow, queuing process gives the controller more time to assess the condition and make a well-informed choice by means of machine learning approaches.

The Ryu controller then resolves the miss by explicitly classifying the data packet by machine learning. This ensures that authentic packets are treated in the appropriate manner by identifying either it is abnormal or not. By utilizing this dynamic approach to resolving table misses, the SDN network is able to adjust to new traffic patterns while simultaneously preserving high levels of performance and protection of the network.

#### 3.3.1.6 Implementation of TPAAD System

The technique known as Packet Filtration in TPAAD is specifically developed to actively monitor incoming network packets, promptly detect possible anomalies, and segregate abnormal packets into a queue specifically intended for malevolent purposes. This is particularly advantageous in identifying diverse network attacks such as DoS floods, or other forms of malicious conduct. The working of algorithm 3.1 is described below.

Initially, the method initiates an infinite loop (Step 1) of algorithm 3.1, which persistently monitors for new packets. The uninterrupted operation of this continuous loop is crucial due to the constant flow of network traffic, which is necessary for the adequate detection of possible assaults. Prior to each iteration (Step 2), it initializes many variables, such as a counter to monitor the quantity of packets originating from the same IP address, timers to document the timing and frequency of incoming packets, a threshold value to establish boundaries for anomalous behavior, and two queues: one for regular packets and another for potentially harmful ones. An initialised prohibited list is configured to contain IP addresses that are deemed invalid or suspicious. Upon the arrival of a new packet (Step 3), the algorithm 3.1 initially verifies if the packet is originating from an IP address that has already been detected in the ongoing session (Step 4). If the packet originates from the same

IP address, the counter is increased (Step 5) to monitor the number of packets received from that source, and the data on the time and frequency of the arriving packets is recorded (Step 6). Implementing this crucial stage enables the algorithm to observe the traffic patterns originating from certain IP addresses. This capability is valuable in identifying anomalous activities like flooding attacks, characterized by the rapid transmission of a substantial volume of packets from a single IP address.

If the packet originates from an unidentified or distinct IP address, the algorithm proceeds to the subsequent set of criteria (Step 7). In Step 8, the system verifies if the packet originates from an invalid IP, which may indicate either a malicious traffic source or an IP that does not match the anticipated structure. In the event that the IP is deemed invalid, the algorithm disables any subsequent communication from that specific address (Step 9) and includes the IP in a list of prohibited IPs (Step 10) to deter future interactions from this particular source. Furthermore, the packet and its origin IP are recorded (Step 11), therefore generating a record for subsequent audit or reporting. This stage is essential for ensuring long-term network security, since it facilitates the preservation of a record of malicious origins. Provided that the packet originates from a legitimate IP address and shows no indications of being anomalous, it is promptly forwarded to a standard queue (Step 13), which will then transmit the packet to the controller for additional processing (Step 14). The controller guarantees accurate routing of valid traffic to its designated destination without any interruption. This enables the program to effectively differentiate between regular network data and any adversarial attacks.

Nevertheless, if a specific limit of time and frequency is surpassed (Step 17), for example, if a single IP is transmitting packets excessively within a limited period, the packet is identified as anomalous. Under these circumstances, the algorithm 3.1 assigns the packet

to the malicious queue (Step 18) and then returns this queue for additional scrutiny (Step 19). This threshold-based filtering facilitates the identification of abnormalities that may not be readily apparent, such as DoS attacks, in which a substantial number of packets are transmitted within a brief timeframe to inundate the system.

If the time and frequency criteria are not exceeded, hence showing the absence of any aberrant behavior, the packet undergoes normal traffic processing (Step 21), is transferred to the normal queue, and then delivered to its destination by the controller (Step 22). By maintaining vigilance for anomalous activity, this system guarantees the uninterrupted flow of regular traffic.

The loop, specified in Step 24, guarantees the perpetual continuation of this operation, therefore enabling the algorithm to consistently filter, detect, and process packets in real-time. This architecture renders it well-suited for settings that necessitate continuous surveillance, such as data centers, cloud platforms, and corporate networks, where any delay in identifying anomalous traffic could result in security breaches or interruptions in operation.

In brief, this method effectively manages incoming packets by detecting abnormal patterns using IP address, time, and frequency values. Normal processing is applied to legitimate traffic, while packets originating from suspect or invalid sources are segregated for additional examination. A malicious queue and a forbidden list are implemented to guarantee prompt and suitable responses from the system to potential attacks.

---

**Algorithm 3.1:** Packet Filtration in Attack Detection (Anomaly Detection in TPAAD)

---

**Input:** Incoming new packet, switch Id, IP address

**Output:** Abnormal packets in the malicious queue

Initialize counter, time interval, frequency, threshold amount, malicious queue, normal queue, Banned list, time threshold = predefined minimum inter-arrival time between packets

frequency threshold=predefined maximum allowed packet rate (packets/second)

**while true do**

Collect packet from buffer

**If** packet from same IP **then**

Increment counter

Record time and frequency

**else**

**If** invalid IP **then**

Block communication

Add that IP to Banned list

Log packet and source IP

**else**

Send packet to normal queue and then to controller for action

Packet send to destination

**end if**

**end if**

**If** time interval<time threshold OR frequency> frequency threshold amount **then**

Send this abnormal packet to malicious queue

Continue it for next iteration

**else**

Send packet to normal queue and then to controller for action

Packet send to destination

**end if**

**end while**

---

### 3.3.1.7 Data Set

The CICDoS2017 data set was used (Barona & Baburaj, 2023) for both training (70%) and testing (30%) the classification models. The dataset addresses many of the problems and restrictions of previous efforts. In order to establish the CICDoS2017 data set, a total of 79 network traffic characteristics were obtained and calculated for both benign and DoS data flows. The dataset provides the results of the network traffic analysis together with labelled flows that are based on the following like time stamp, source and destination IPs, source and destination ports, protocols, and attack. The procedures that were employed to

train the classification models are illustrated in Figure 3.3. In order to prepare the data for shuffling, we eliminated outliers and any values that were infinite or blank. Then, to perform binary classification, the category labels were converted to integer format, where 1 indicates safety and 0 indicates attack. Misclassification was eliminated by indexing the string values and then standardizing the data.

The dataset has (6,393,767) records and occupies huge amount of memory (1.4 gigabyte). The dataset is well-balanced, with a total of 6,398,925 normal records and a percentage of (50.02%), and a total of 6,395,702 fraudulent records and a percentage of (49.89%). Overall classification record is shown in the below mentioned Table 3.1.

**Table 3.1:** Classification Record

| Number of Classes | 2                   |
|-------------------|---------------------|
| Normal Records    | 2, 153, 72 (50.02%) |
| Malicious Records | 2, 153, 70 (49.98%) |
| Total Features    | 79                  |
| Features Selected | 20                  |

### 3.3.1.8 Data Preparation

Feature engineering, data wrangling, data cleaning, and pre-processing are all part of data preparation, which is often regarded as the most challenging stage of ML are illustrated in Figure 3.3. In order to use ML algorithms to discover insights or generate predictions, data preparation is necessary. Nevertheless, the most important part of the data preparation procedure is dealing with the incomplete or missing information in the dataset. Data preparation refers to the process of transforming raw data into a usable format. Label encoding is used to transform the continuous class label into a discrete form (0,1), where 0 represents a safe class and 1 represents a DoS attack.

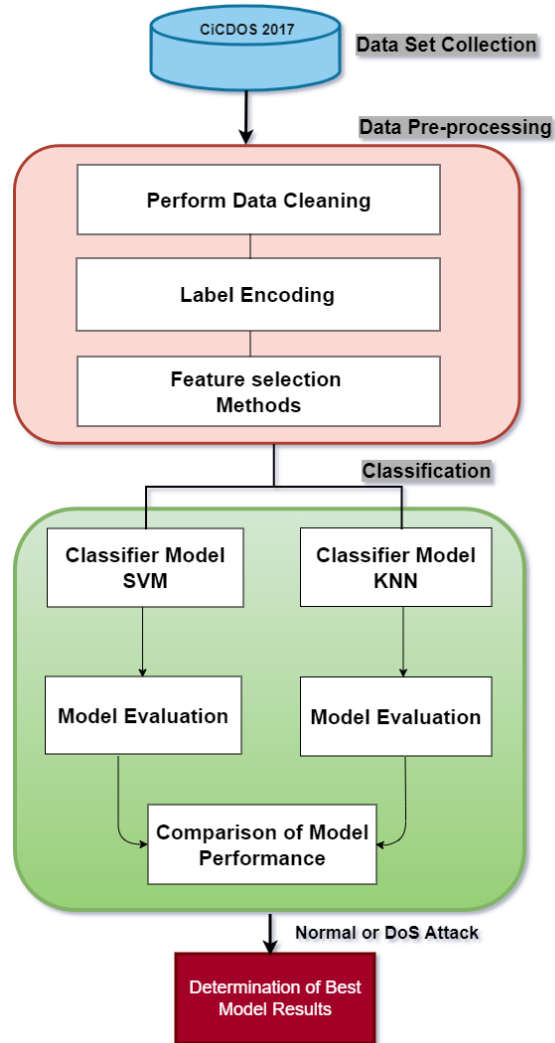


**Figure 3.3** Proposed Methodology for Classification

In order to get the dataset ready for ML models such as SVM and KNN, data cleaning is an essential first step. This procedure is carried out to guarantee that the models are trained on accurate data by eliminating mistakes and noise from the dataset. As a first step as illustrated in Figure 3.3, It is necessary to address missing values, which can occur due to logging mistakes or packet loss. The dataset contains multiple properties (columns) with no values, which can have a detrimental impact on the ML model. For example, several of the entries had empty values for the Fwd Byts/b Avg, Fwd Pkts/b Avg, Fwd Blk Rate Avg, Bwd

Byts/b Avg, and Bwd Pkts/b Rate Avg characteristics. Imputation methods like median or mean imputation can be used for features with many missing values. It might be more efficient to remove the missing rows entirely if the percentage of missing data is small. To further ensure that the ML model's performance is not impacted, it is necessary to detect and eliminate duplicate records. These records can arise in network traffic data as a result of packet revisions or logging artifacts.

Data wrangling, which involves combining and transforming the raw data into a cohesive structure that machine learning algorithms can utilize. Here, the dataset, which will be divided up by attack type or by day, into one cohesive whole. Only day data which includes both normal and attacked traffic used here. It is critical to standardize IP address forms, normalize timestamps, and convert qualitative variables to numerical representations. In order to handle the massive amount of network traffic data, methods such as data sampling are used here. This way, important information is preserved such as by combining traffic data over short time periods and by sampling subsets for training purposes.

#### 3.3.1.9 Feature engineering with Feature Selection method

When trying to maximize DoS attack detection using machine learning models on the dataset, feature engineering is an essential step. Finding the best way to differentiate between legitimate and malicious traffic requires extracting, transforming, and selecting information. The wrapper feature selection method's forward feature selection approach is utilized to enhance the accuracy and overall performance of detection models by identifying and extracting the most relevant subsets of features from the dataset.

There are 79 features that are initially included in the dataset. These features are obtained from packet headers and network traffic metrics. For the SVM and KNN models, 20 of these features were chosen as the most informative using the forward feature selection

method. An empty set of features is used as the starting point for the forward feature selection process, which then adds the most important characteristics one at a time in an iterative, sequential manner. At each stage, the model is trained using the available characteristics and its performance is assessed using measures like F1-score, recall, accuracy, and precision. In order to complete the feature set, the feature that brings about the most significant enhancement to the performance of the model is included.

By using forward feature selection in particular, the wrapper technique trains the model on different subsets of features and then chooses the ones that produce the best accuracy while still guaranteeing good performance across other metrics. This process repeats until model performance does not improve, so 20 feature were selected here. For this case, the wrapper method is a useful for feature engineering since it assures that the selected features contribute considerably to the effectiveness of the model in detecting denial of service attacks. This is accomplished by directly measuring the impact that each feature has on the performance of the model.

#### 3.3.1.10 Machine Learning Classifiers

Classifiers that are based on machine learning are extremely important for the detection of DoS attacks because they make it possible to automatically and accurately identify harmful traffic patterns within a wireless network. DoS attacks are specifically engineered to inundate network resources, hence posing challenges for networks in effectively adapting to the ever-changing strategies employed by fraudulent users.

#### 3.3.1.11 Classification Using SVM

The main goal of the SVM-based DoS detection system is to correctly detect and categorize network-based DoS attacks by utilizing the optimal feature set that was generated through the wrapper feature selection technique above. In order to avoid the SDN controller

from processing illegal requests, it is necessary to identify malicious traffic and restrict it from consuming all of the network's resources. The SVM model is trained to detect patterns in traffic that suggest a DoS attack is about to happen or is already underway, since these attacks usually lead to a drain on network resources like bandwidth and CPU utilization. The optimized features set, which are based on network measurements and packet header attributes, play a crucial role in detecting these patterns and guaranteeing that the SVM model can accurately differentiate between benign and malicious traffic packets.

The strategy of maximizing the margin was employed in the system, referring to the separation between the hyperplane and the support vectors, which are the nearest data points from each class. This margin is maximized to enhance the model's ability to generalize to new data, enabling the SVM to effectively differentiate between normal and malicious traffic across various network scenarios. The utilization of this methodology enables the SVM to establish a resilient decision boundary that effectively reduces the classification error for both the training and subsequent data, hence enhancing the overall dependability of our attack detection system. Training the SVM model using a reduced set of features is the next step after identifying the optimum set of features. During training of proposed system, a decision boundary is built to distinguish between attack traffic and regular traffic in the multidimensional space defined by the features that were chosen. The SVM is able to function by locating the hyperplane that maximizes the margin between the two classes. The margin is defined as the distance between the hyperplane and the data points from each class that are closest to it. In order to categorize incoming traffic using the feature values obtained from the optimized set, the support vector machine's decision function is employed, as mentioned in equation 3.5.

$$I(k) = a^T k + m \quad \text{Equation 3.5}$$

Here the variable  $k$  denotes the feature vector of the incoming packet, which is derived from the optimized feature set during feature selection process. Another variable  $a$  is the weight vector that determines the orientation of the hyperplane within the respective feature space.  $T$  variable denotes the transpose of the weight vector. The bias term, denoted as  $m$ , is responsible for moving the hyperplane far from the origin.

The SVM locates this hyperplane in the  $k$ -dimensional feature space for data that can be separated along a straight line. In cases where linear separation is not feasible, SVM will use a kernel approach to transform the input data into a higher-dimensional space. And for this data transformation, we use common kernel functions as the polynomial kernel function. In this transformed space, the decision function for the SVM is mentioned in equation 3.6

$$I(k) = \text{sign} \left( \sum_{p=1}^n \alpha_p y_p H(k_p, k) + m \right) \quad \text{Equation 3.6}$$

Here the Lagrange multipliers produced from the dual optimization problem are denoted as  $\alpha_p$ . The class labels are represented as  $y_p$ . The kernel function denoted here as  $H(k_p, k)$ .

The training dataset is used to train the SVM model. The features  $k_p$  and labels  $y_p$  in this dataset are labelled with 1 for DoS attacks and 0 for normal traffic, respectively. Optimizing SVM entails modifying the weight vector  $a$  and bias  $m$  until the hyperplane optimizes the margin between the two classes and correctly classifies the training examples. The final model proves it can correctly classify network traffic as either benign or harmful, showing that it can detect DoS attacks. In order to accomplish this categorization, we use newly collected data about network traffic and check the sign of the decision function  $I(k)$ .

An alternative methodology employed here involves the utilization of the optimal features identified by the SDN controller in the packet filtration process to generate the feature vector of a packet. Once it identified, the feature vector is then supplied into the SVM

model for classification. The SVM model assesses the feature vector in relation to the decision boundary that it has acquired through the training process. In the event that the decision function in equation 3.5 such as if  $I(k) > 0$  , signifying the classification of a packet as a component of a DoS attack, the packet is promptly assigned to the malicious queue. This queue has been purposefully built to manage traffic that is considered suspicious, enabling the system to perform further actions such as recording the event, generating an alert for mitigation system, and ordering the network switches to either drop or reroute the packets originating from the identified malicious source.

On the other hand, the packet is forwarded to the normal queue if the decision-making function  $I(k) \leq 0$  by designating the packet as normal. The normal queue is employed to manage routine network traffic that does not display any indications of being associated with a DoS attack. The packets within this queue undergo processing by the controller and are then transmitted to their designated destinations without requiring any additional action. The implementation of traffic division guarantees that the processing of valid requests remains unaffected and uninterrupted by the management of potentially malicious packets

#### 3.3.1.12 Classification using KNN

Following the construction of the feature vector  $k_p$  for an incoming packet, the feature vector is then entered into the KNN classifier model. Next, for our proposed attack detection system, Euclidean distance function was used here. Here this technique is employed to determine the k nearest neighbours ( $n$ ) of  $k_p$  inside the pre-processed dataset. The calculation of the distance between the feature vector  $k_p$  and each data point  $k_q$  in the training set is performed using the Euclidean distance as mentioned in equation 3.7

$$d(k_p, k_q) = \sqrt{\sum_{n=1}^s (k_{pn} - k_{qn})^2} \quad \text{Equation 3.7}$$

Here, the variable  $s$  denotes the number of features contained in the vector. The technique finds the  $k$  nearest data points (represented  $n$  here) that are the furthest away from  $k_p$  and uses a majority vote to decide the incoming packet's class. The value of categorization is used to determine the number of surrounding KNNs, denoted by the letter  $n$ . In the process of categorization, we used the value of  $n$  as 5.

The system will consider a packet is malicious if its neighbours consider it part of a DoS attack. It directs the packet to the malicious queue, initiating actions like recording data, alarm generation, and transmission to a mitigation module. If the packet is regular traffic, it's directed to the normal queue. The division of traffic into malicious and normal queues guarantees the timely handling of potentially hazardous data without interfering with lawful network activities.

The evaluation of the performance of SVM and KNN models in detecting DoS attacks within a SDN environment encompasses various metrics.

#### 3.3.1.13 Implementation of Classifiers for TPAAD

In order to identify DoS attacks, the classifiers for attack detection in TPAAD described in algorithm 3.2 processes packets from the malicious queue, extracts pertinent information, and feeds those features into ML classifiers for attack detection. Anomaly detection and classifiers are combined in this method to improve harmful traffic identification.

Step 1 of the algorithm 3.2 involves obtaining a dataset that includes details about the packets, their characteristics, and maybe annotated examples of recognized attack activity. This dataset functions as the fundamental basis for training or assessing the classifiers that will be employed to identify possible assaults. Next, the dataset is subjected to pre-processing (Step 2). Pre-processing encompasses several tasks such as data cleansing,

missing value handling, feature normalization, and categorical variable encoding to facilitate the use of ML algorithms SVM and KNN. Improperly structured data can significantly impair the performance of the classifiers, making this step crucial.

Following the pre-processing of the dataset, the algorithm proceeds into an infinite loop (Step 3), during which it consistently examines the arriving packets from the malicious queue. In this iterative process, the algorithm utilizes feature selection techniques to guarantee the inclusion of the most pertinent features in the classification mechanism. The algorithm sequentially processes each feature set (Step 4) for every individual packet or block of packets in the malicious queue. At Step 5, the objective is to derive the most informative characteristics from the packet headers and network data. The features may encompass the source IP, destination IP, port numbers, packet size, frequency, or time intervals associated with packets. Through the execution of mathematical computations and the extraction of these characteristics, the system is capable of generating significant data inputs for the classifiers. Once the pertinent features are extracted from the packets, they are inputted into the classifier models (Step 7). The system presumably use several machine learning models, such as decision trees, support vector machines, or neural networks, to categorize the data as either normal or representative of an assault. The classifiers are trained to identify patterns that are indicative of attacks, such as abnormally high traffic volume, irregular intervals between packets, or the presence of particular packet headers often linked to DoS attacks.

Next, the algorithm 3.2 assesses the outcomes produced by the classifier models and selects the most optimal results (Step 8). This may include comparing predictions from several models or choosing the most precise classifier based on performance measures such as accuracy, precision, recall, or F1-score. Upon detection of an attack by the chosen model

(Step 9), the system promptly initiates action. The initial action is to include the IP address accountable for the attack into a prohibited list (Step 10) in order to impede any more network communication from that IP. This guarantees the automatic blocking of following packets originating from the same source. Concurrently, the system records the specifics of the attack, such as the packet and the originating IP (Step 11), for future consultation or use in forensic investigation. Furthermore, it produces a notification (Step 12), notifying the system administrators or other pertinent entities of the detection of a possible DoS attack.

Upon generation of the alarm, the packet is transmitted to the mitigation module (Step 13), which may employ several techniques such as rate-limiting, IP blocking, or implementing more sophisticated countermeasures to render the attack ineffective. Implementing this measure is essential for safeguarding the system against the detrimental consequences of the identified attack.

When the classifier fail to identify any indications of an attack (Step 14), the packet is deemed to be normal. Once received, it is placed in the standard queue and then transmitted to the controller for regular processing (Step 15). Subsequently, the packet is directed to its designated endpoint (Step 16) without any additional intervention, therefore guaranteeing the uninterrupted flow of valid communication.

The algorithm 3.2 iterates through this procedure indefinitely (Step 17), continuously scrutinizing the incoming packets, categorizing them, and identifying any possible threats. By iteratively executing this loop, the system may maintain a state of alertness and adaptability to any developing risks, therefore guaranteeing the security of the network.

In brief, this algorithm 3.2 is specifically developed to analyze harmful packets by extracting pertinent characteristics, categorizing them using machine learning models, and identifying possible DoS attacks. Upon detection of an attack, the system prohibits the IP

address, records the occurrence, produces an alert, and transmits the packet to a vulnerability management module. In the absence of any detected assault, the packet is regarded as normal and sent to its intended destination. The inherent continuity of this loop guarantees constantly vigilant monitoring of the system for any possible attacks.

---

**Algorithm 3.2:** Attack Detection Module (Classifiers in TPAAD)

---

**Input:** Malicious queue packets with features, Data set

**Output:** DoS Attack Detected

Provide data set

Pre-process data set

**while** true **do**   // Apply Feature selection method

    Feature list = wrapper ()

    Input feature list to classifier models

    Choose best results

**If** Model detects attack **then**

        Add that IP to Banned list

        Log packet and source IP

        Generate Alarm

        Send packet to mitigation module

**else**

        Send packet to normal queue and than to controller for action

        Packet send to destination

**end while**

---

### 3.3.2 Measure Performance using Confusion Matrix

A confusion matrix is a useful matrix used here for evaluating how well a ML model performed on a test set of data. It is a way to show how many times the model's predictions were either correct or incorrect. It is widely used to evaluate classification models that predict category labels for input instances.

- True Positive (TP): Number of packets correctly predicted as attack.

- True Negative (TN): Number of packets correctly predicted as normal
- False positive (FP): Number of packets incorrectly predicted as attack
- False Negative (FN): Number of packets incorrectly predicted as normal

For every prediction in a two-condition scenario, the output might be True Positive, True Negative, False Positive, or False Negative, as illustrated in Table 3.2 of the Confusion Matrix used to evaluate ML model's output. Both TP and TN indicate accuracy in prediction, but FN indicates that the predictor incorrectly interpreted a positive state as negative, and FP indicates that the predictor incorrectly interpreted a negative state as positive. The chance of malicious traffic being ignored in the network (FN) and the rate of false alarms (FP) are related to the definition of positive and negative in ML model as anomalous and normal states in the network, respectively (Murphy, 2018).

**Table 3.2:** Confusion Matrix for Two Class Scenario

| Confusion Matrix |          | Predicted Condition |                     |
|------------------|----------|---------------------|---------------------|
|                  |          | Positive            | Negative            |
| Actual Class     | Positive | True Positive (TP)  | False Negative (FN) |
|                  | Negative | False Positive (FP) | True Negative (TN)  |

The following are the five primary measures used to assess the efficiency of ML models:

- The overall quality of an ML model is measured by its Accuracy (A) in equation 3.8, which shows how many predictions (positive and negative) were spot-on.

$$A = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{Equation 3.8}$$

- Precision (P) in equation 3.9, also known as Confidence, is solely concerned with positive class predictions. Whenever a machine learning model returns a positive state, this represents the chance of an accurate prediction being made.

$$P = \frac{TP}{TP+FP} \quad \text{Equation 3.9}$$

- False alarm rate in equation 3.10 for the said system is computed as:

$$FAR = \frac{FP}{TN+FP} \quad \text{Equation 3.10}$$

- Recall(R) in equation 3.11 or Sensitivity is the only class that cares about authentic positivity. It shows how well trained the ML model is to recognize a positive condition when it occurs.

$$R = \frac{TP}{TP+FN} \quad \text{Equation 3.11}$$

- The harmonic mean of Recall and Precision is F-Measure (F) in equation 3.12, often known as F-Score. The higher the F-Measure number, the better the recall and accuracy are likely to be.

$$F = 2 * \frac{R*P}{R+P} \quad \text{Equation 3.12}$$

### 3.3.3 Mitigation Module

When the system detects an attack, it triggers the activation of a mitigation module that is built into the control layer system. This module analyzes and lessens the impact of DoS attacks on SDN-controlled networks. This technique seeks to efficiently manage attack traffic while conserving the controller's computational bandwidth. The following is a detailed explanation of the processes involved with the respective used parameters:

#### 3.3.3.1 Packets Arrival

The tunnel within the mitigation module receives packets from the malicious queue, which contains potentially malicious packets designated as such. In contrast, packages from

the standard queue are delivered to their intended destinations. Let  $P_{malicious}$  stand in for the collection of possibly harmful packets that were received from the malicious queue, and  $P_{regular}$  for the collection of harmless packets. The following is a description of the packets arrival process in equation 3.13:

$$Pkt_{tunnel} = \{p \in P_{malicious}\} \cup P_{regular} \quad \text{Equation 3.13}$$

where  $Pkt_{tunnel}$  denotes the collection of packets that are transmitted through the tunnel of the mitigation module. It comprises of ordinary packets ( $P_{regular}$ ) that are immediately sent to their intended receivers and possibly dangerous packets ( $P_{malicious}$ ) that are removed from the malicious queue.

### 3.3.3.2 Header Analysis

The process of analyzing the header fields of arriving packets in SDN for the purpose of detecting DoS attacks begins with the examination of the header fields utilizing a variety of methodologies and algorithms. One popular method is anomaly detection, which uses statistical analysis to find suspicious traffic patterns.

In order to determine what constitutes typical traffic patterns, statistical analysis necessitates the computation of statistical measures for a variety of header fields. These measurements include the mean and the standard deviation. The following are some examples of formulae used to calculate certain statistical measures:

$$Mean(\mu) = \sum \frac{K}{n} \quad \text{Equation 3.14}$$

$$Standard\ Deviation\ (\sigma) = \sqrt{(\sum (K - \mu)^2 / n)} \quad \text{Equation 3.15}$$

In equation 3.14,  $K$  stands for the values that were found in a particular header field in the packets that were examined,  $\sum$  stands for the summation,  $n$  stands for the total number of packets,  $\mu$  stands for the mean, and in equation 3.15 ( $\sigma$ ) stands for the standard deviation. Indicative of the presence of DoS attacks is a divergence from the regular patterns that have

been established, which may be detected as a possible anomaly by conducting an analysis of the statistical features of the header fields.

So the controller analyses the metadata of packets received through the tunnel. Analyse header fields including source and destination addresses, source port, destination port, VLAN priority, and other pertinent information for further processing.

#### 3.3.3.3 Threat Values

Several packets are identified as having threatening values based on an analysis of their packet headers. These values indicate whether or not the transmissions are potentially malignant. For instance, abnormally high packet rates or doubtful payload characteristics may signal an attack.

#### 3.3.3.4 Dropping and Blocking Packets

The dropping and blocking of suspicious packets prevent them from reaching their destinations. The IP address associated with these messages is also temporarily blocked. This transient blocking mitigates the impact of the attack by preventing the same source from continuously overwhelming the network.

#### 3.3.3.5 Permanent Blocking

If malicious activity persists for an extended period of time from a particular source IP address, the mitigation module permanently blocks that source IP. This provides a long-term measure of defence against the attack, as no further packets will be received from that particular source.

#### 3.3.3.6 Threshold Check

The controller can unlock the source IP if the packet deluge from the blocked source IP falls below a specified threshold in the queue. This allows normal traffic to resume from the previously blocked source, assuming the threat has subsided.

By utilizing this mitigation module and the corresponding tunnelling technique, the SDN controller can effectively manage attack traffic without exhausting its computational resources. The aforementioned measures ensure that potentially malicious packets are discarded, that sources of ongoing attacks are temporarily or permanently blocked, and that normal traffic is permitted to continue once the threat has diminished. This strategy protects the SDN network against DoS attacks while preserving the availability and performance of legitimate network traffic.

#### 3.3.3.7 Implementation of Mitigation Module

The purpose of the Attack Mitigation Module described in algorithm 3.3 is to counter DoS attacks by analysing packets that have generated alerts and reducing the destructive effects of the attack. An essential objective of this module is to examine packets linked to the identified attack and prevent harmful traffic while permitting legitimate traffic to proceed.

The procedure commences with an examination of the packet that initiated the alarm (Step 1) of algorithm 3.3. This stage entails meticulous examination of the packet, including its header, which includes vital pieces of information such as the source and destination IP addresses, ports, and protocol. In Step 2, the system establishes a threshold time and configures the flow request counter to the value of 0. The threshold time is a metric used to establish the duration for which the system should monitor packets originating from a specific source before proceeding with any further actions. On the other hand, the flow request counter is responsible for keeping a record of the number of packets received from the identified source. Next, the program proceeds to examine the IP address of the packet that is being received (Step 3). Conducting this analysis is crucial in order to ascertain the validity of the IP address, identify any prior bans, or detect any consistent patterns of

malicious activity. On the basis of this analysis, the system initiates the development of methods to manage the attack (Step 4), such as determining whether to restrict communication from a specific IP address or implement other measures to reduce the impact of the attack.

The system examines the flow request count (Step 6) if the source IP address is identified in the prohibited list (Step 5), suggesting that it has already been reported for questionable or malicious activities. If the number of flow requests from this IP exceeds 4, which suggests a continuous high volume of traffic that may be associated with a Denial of Service (DoS) attack, the system promptly initiates measures to block the IP (Step 7) and discard the packet (Step 8). In addition, a new flow rule is generated and included into the system (Step 9) to prevent any more communication from this IP address, therefore guaranteeing that any future packets originating from this source are automatically rejected.

If the number of flow requests is below 4, the system proceeds to increment the flow request counter (Step 11), therefore enabling it to persist in monitoring the traffic volume originating from the internet address. Once again, the package is dropped at Step 12, therefore impeding its arrival at its intended destination. Concurrently, the system generates a notification to the IP address (Step 13), maybe by means of logging or notifying network administrators to oversee the IP's activities. This warning serves as a more neutral reaction, enabling the system to distinguish between possible false positives and genuine malicious activity. In Step 15, the system checks if the IP address is not banned. Then, in Step 16, it increments the flow request counter and in Step 17, it drops the packet. This process is basically repeated to watch for suspicious activity, monitor traffic, and discard potentially harmful packets. Furthermore, the method has a system to periodically evaluate the situation (Step 19). If no further packets are received from the same IP within the specified time limit,

the system automatically concludes that the threat has diminished and proceeds to unblock the IP address (Step 20). Subsequently, the IP is eliminated from the prohibited list (Step 21), and the prohibited list is consequently revised (Step 22). This stage is essential to guarantee that genuine users, who may have been briefly identified as suspicious, can continue regular communication after the threat of attack is considered to have passed.

Nevertheless, in the event that packets originating from the same IP persistently arrive over the designated time limit (Step 23), the system upholds its defensive stance by obstructing the IP (Step 24), hence strengthening the previously implemented strategy. In order to minimize the impact of a DoS attack, this measure guarantees the prevention of continuous attack traffic from accessing the network.

---

**Algorithm 3.3: Attack Mitigation Module**

---

**Input:** Packet with Generated Alarm

**Output:** Mitigates DoS/DDoS Attack

---

Investigate packet

Read header, initialize threshold time, flow req=0, threshold time

Analyse IP

Develop strategies for attack handling

**If** Src IP from Banned List **do**

**if** flow request > 4 **do**

        Block IP

        Drop packet

    Add new flow rule to block communication

**else**

        Increment Flow req

        Drop Packet

        Warn IP

**end if**

**else**

    Increment Flow req

    Drop packet

**end if**

**If** No more packets coming from same IP at threshold time **then**

    Unblock IP

    Remove IP from Ban List

    Update Ban list

**else**

    Block IP

**end if**

---

The algorithm 3.3 offers a comprehensive and systematic approach to counteracting DoS attacks by meticulous monitoring and analysis of traffic originating from dubious IP addresses. It employs thresholds, flow request counts, and a prohibited list to dynamically regulate IP addresses, guaranteeing the blocking of malicious IP's while allowing normal traffic the opportunity to recover if incorrectly identified. The algorithm 3.3 has been specifically developed to adapt to variations in traffic patterns, providing both proactive and responsive strategies to safeguard the network against persistent attacks.

### **3.4 Simulation Setup**

Each VM in a Mininet represents a combination of a host computer and an OpenFlow switch. In the SDN context, these VM act as a simulation of network nodes and switches. They are responsible for running the required software components in order to take part in the network and communicate with the Ryu Controller. The VM's that make up Mininet are linked together by means of OpenFlow switches. It is the responsibility of OpenFlow switches to forward packets in accordance with the instructions received from the Ryu Controller. In the SDN architecture, they make possible the centralised control and administration of network traffic.

The Ryu Controller is the primary SDN controller used in the model of the network, and its name comes from its function. It is a piece of software that was developed in Python and utilises the OpenFlow protocol in order to connect with OpenFlow switches. The Ryu Controller is in charge of directing and maintaining the network by providing instructions to the switches on how to deal with incoming packets of data as shown in Figure 3.4.

The connection between the Ryu Controller and the OpenFlow switches is established through the use of the control channel. It makes it possible for the controller to communicate with the switches, both to transmit and receive commands. The OpenFlow

protocol is utilised in the implementation of the control channel. This protocol makes it easier for the controller and switches to communicate with one another and exchange messages.

The Ryu Controller incorporates a module for the detection of attacks, which is responsible for determining whether or not an attack is a DoS. The network traffic and data that have been acquired from the switches are analysed by the attack detection module. This is done in order to find trends and abnormalities that are suggestive of attack traffic. In order to identify potentially harmful actions, it could make use of a variety of methods, such as traffic filtering, and other attack identification machine learning techniques for classification. The Ryu Controller will activate the mitigation module as soon as the attack detection module determines that a DoS attack has been launched against the system.

The mitigation module will respond appropriately to any identified attacks and take the necessary steps to reduce their impact by sending those identified packets to mitigation as shown in Figure 3.4. It is able to safeguard the network resources and keep operations running normally by implementing countermeasures such as rate limitation, packet filtering, traffic redirection, or dynamically modifying flow entries. The network paradigm provides centralised control and detection of DoS attacks by utilising the control channel and the capabilities of the Ryu Controller. The Ryu Controller is provided with network updates by the OpenFlow switches. With this information, it is able to do an analysis of the traffic and take preventative steps to protect against attacks.

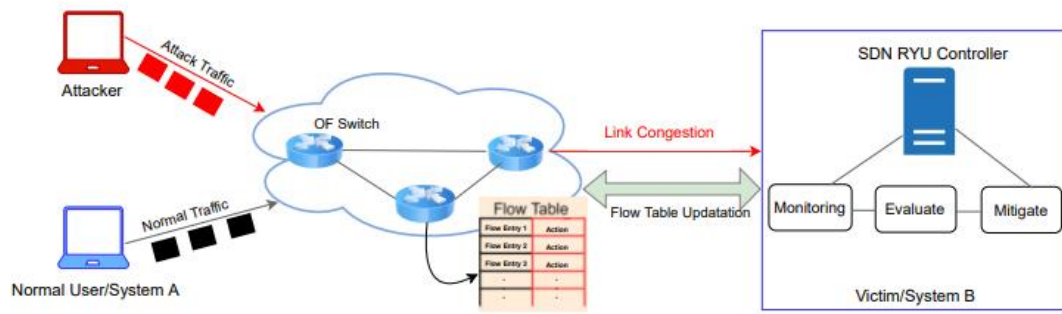


**Figure 3.4** Proposed System Visualtization Diagram

### 3.4.1 Tools used

To detect DoS attacks, the following tools were utilized:

1. **Virtual Machines:** VM's offer a practical means of establishing and replicating network environments for the purposes of testing and assessment. Ubuntu, the Ryu controller, and the other necessary components for the SDN network are all be operated on VM's. VMs are sand operated with software such as VirtualBox or VMware.
2. **Ubuntu:** Ubuntu is an open-source operating system that is frequently used for SDN installations. It offers a dependable and modifiable environment for the operation of SDN network components like the Ryu controller and OpenFlow switches.
3. **Ryu Controller:** Ryu is a free and open-source Python SDN controller. It facilitates centralized administration of SDN. Ryu allows the controller and switches to communicate according to its support for several network protocols. Installation and operation of Ryu on an integrated development environment (IDE) such as PyCharm makes it easier to design and test software.
4. **OpenFlow/SDN Switch:** SDN Switch is a software function (can also be hardware but we used software function for our experiments) in a

SDN environment. The instructions that are received from the SDN controller are used by these switches to determine how to forward packets. By utilising the OpenFlow protocol, the Ryu controller has the capability of both programming and controlling the switches.

5. **Protocols for the Network:** The SDN environment allows for the use of a wide variety of network protocols, which can help to make communication easier and enable network administration. When it comes to the detection of attacks in SDN, OpenFlow used here as it is utilised here to gather traffic data, monitor network behaviour, and identify abnormalities that are indicative of DoS attacks.
6. **Visualization Tool:** For visualization, Matplotlib (a Python library) has been used. It is a scientific visualization tool designed for generating graphs and other visual representations of data.

### 3.5 Evaluation Metrics

The following are the evaluation metrics (along with confusion matrix) used to effectively measure and secure the performance of the proposed TPAAD solution.

- CPU Usage is the measurement of how much the system's CPU is being used during both regular network operation and conditions of a DoS attack. A lower percentage of total CPU utilization implies that resources are being managed effectively and that the system can handle greater traffic demands.
- Control Channel Bandwidth refers to the total amount of bandwidth that is utilized by the control channel of the SDN both while it is functioning normally and when it is under attack by a DoS. A more efficient use of bandwidth shows effective communication between the SDN controller and the switches, which in turn enables more effective administration and control of the network.

- The packet delivery ratio is the percentage of data packets (packet-in messages) that were successfully delivered in comparison to the total number of packets that were sent under both normal conditions and during DoS attacks. This ratio may be calculated both before and after an attack. If the percentage of successfully delivered packets is larger, it indicates that the network is more dependable and resistant to attack traffic.
- Response Time is the amount of time it takes for the system to reply to requests made by the network under normal circumstances as well as during DoS attacks. A lower response time is indicative of greater system efficiency, which in turn leads to faster detection and defence against threats.
- The amount of flow requests that are created by the SDN switches and submitted to the controller for processing during normal operation as well as during DoS attacks is referred to as the flow requests submitted to the controller. A lower flow request count shows that flow management within the network has been optimised, resulting in a reduction in controller overhead and an improvement in overall system performance.

### **3.6 Summary**

This chapter discusses how an attacker may easily launch a DoS attack on a system using the SDN architecture by sending an overwhelming number of packets to the system's Open Flow switch. Hence, the Open Flow switch forwards each packet to the controller to request an updated flow rule. The attack's usage of resources from many elements of the SDN architecture will have far-reaching effects. The system consists of the switch's storage space, the control channel's bandwidth, and the CPU load. When too many resources are being utilized, performance, latency, lost packets, and detection accuracy may all suffer.

This type of attack can be recognized using an ML classifier, with SVM and KNN utilized for accurate detection. The purpose of TPAAD is to develop more efficient method of detecting DoS attacks and the workflow of TPAAD is also discussed here with the feature set obtained from the CICDoS2017 used data set which is also available at <https://www.kaggle.com/datasets/dhoogla/cicdos2017>.

## **CHAPTER 4**

### **RESULTS ANALYSIS**

#### **4.1 Introduction**

In this chapter, we will evaluate the effectiveness of the proposed system in detecting threats and implementing mitigation strategies within an SDN environment. This system incorporates the Ryu controller, which is a free and open-source software for SDN controller that helps increase network responsiveness and streamline traffic management. The analysis is performed with the CICDoS 2017 dataset and includes a number of steps, such as packet filtering, classification using TPAAD classifiers, and attack defense systems. The findings from the proposed system are compared to those from other well-established systems in order to determine its performance. The purpose of this comparative analysis is to assess the effectiveness and possible benefits of the suggested solution in ensuring the security of the network.

#### **4.2 Experimental Approach**

This section provides a detailed explanation of the experimental approach for detecting and mitigating DoS attacks in SDN environment.

##### **4.2.1 Hardware and Operating System Configuration**

The research experiments were conducted using a PC running on the Ubuntu 18.04.2 LTS operating system. The personal computer functions at a frequency of 2.4GHz and is furnished with a dual-core Intel Core i7 8th Generation CPU. The research required a dependable operational environment, and the technical characteristics of this hardware arrangement provided both the necessary processing capability and the environment. To

ensure that network simulations and SDN tests were efficient and reproducible, Ubuntu 18.04.2 LTS was chosen due to its reliability and consistency as a Linux-based platform.

#### **4.2.2 Virtual Machine Environment Setup**

Using a virtualization tool like VMware or VirtualBox to run an Ubuntu instance increases the chances of a successful Ubuntu installation. This VM environment was used to deploy the SDN and switches. A VM provided a clean, controlled environment for the tests by isolating it from the host system. Additionally, virtualization made it possible to replicate the setup for additional validation and testing.

The experiments in this research were conducted using the Mininet network emulator, which is widely recognized for its efficiency in SDN research. For the installation of Mininet and other VM environmental setup, the following commands were executed in the ubuntu terminal.

1. To install mininet, following commands are executed.

```
sudo apt-get update
```

```
sudo apt-get install mininet
```

2. In order to install the Ryu controller, following command are executed.

```
sudo apt-get install python3-pip
```

```
sudo pip3 install ryu
```

3. Set up the Ryu Controller

Create a unique script for Ryu controller that includes modules for detection and mitigation specifically for this purpose. To identify suspicious packets and manage mitigation with a blacklist of IP addresses, this script employs a machine learning-based detection technique. For configuration, perform the following step by step procedure.

```
ryu-manager ryu.app.simple_switch_13 ryu.app.my_detection_script.py
```

4. By executing the following command, verify the installation mentioned earlier by running Ryu controller.

```
ryu-manager
```

5. Scapy for Traffic generation

Scapy(“Scapy A Python Tool For Security Testing,” 2015) is a python based tool that allows packet manipulation and we used it here to simulate both normal and abnormal traffic. For this use the following command.

```
sudo pip3 install scapy
```

6. Python Library for ML Installation

The *scikit-learn* library is used for both SVM and KNN classifiers. The following command is executed for its installation.

```
pip3 install scikit-learn
```

7. Launch a Custom Network Topology in Mininet

As shown in Figure 4.1, a 2-depth tree-type network topology was built in Mininet using command. This topology consists of 3 switches and 6 hosts.

```
sudo mn --topo tree, depth=2, fanout=3 --controller=remote --mac --switch ovsk --link=tc
```

In this way, a virtual network is created using Open Flow switches, and a remote controller is assigned the responsibility of managing network operations.

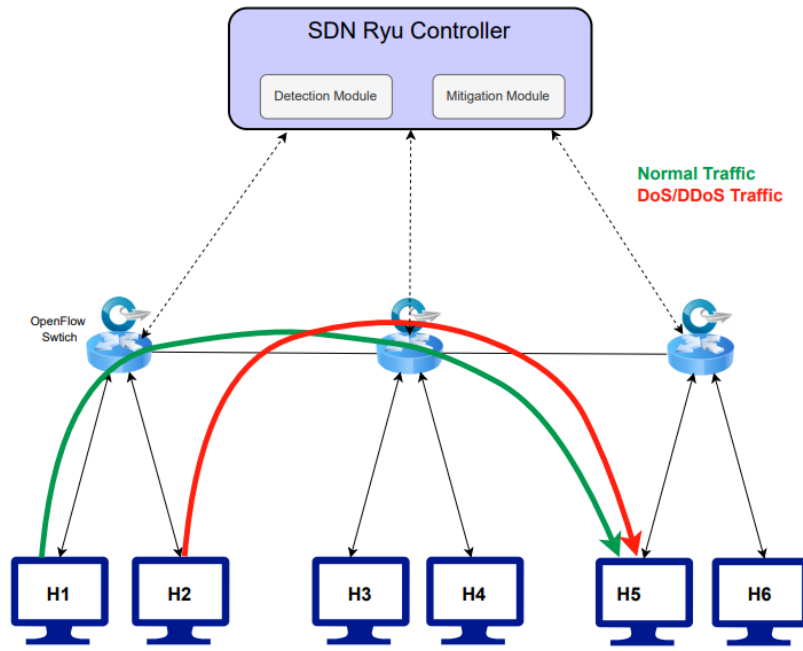


**Figure 4.1** Network Model for TPAAD

#### 4.2.3 Experimental Setup

For this research, a Python script was created to establish a new TCP connection to a different host on the network every second, generating random network traffic. Each connection initiated by the script transmits 100 data packets to the target host. This traffic generating software operates on a single host per end switch in the Mininet network configuration. Scapy (“Scapy A Python Tool For Security Testing,” 2015), an exceptional tool for manipulating packets, was employed to initiate various DoS attacks. In order to overload the SDN controller, the control channel, and the network switches' resources, the attacker hosts continuously created flooding traffic. To replicate real-world attack variations, each scenario included different kinds and quantities of attacking hosts. Every scenario involved here a remote connection between the SDN controller, which operated the Ryu controller, and the network topology, which existed on a distinct VM. Using VM, a number of experiments with different scenarios were carried out, each of which simulated a different

kind of DoS attack and simulation parameters used for the proposed system are mentioned in Table 4.1. The frequency threshold was adjusted to 40-50 packets per second for networks with up to 10-15 hosts, and 90-100 packets per second for networks with more beyond that for different scenarios. TPAAD continuously runs upon gathered data to identify abnormal traffic by using equation 3.3, with an iteration period between successive iterations. In accordance with the requirements of the network, this time period can range anywhere from 3 seconds to 12 seconds, and this refers to the speed at which the controller desires to identify and stop harmful traffic within its computational capacity.

The experiments were conducted with and without the implementation of the TPAAD system. At end the TPAAD system's performance and efficiency in detecting and preventing attacks were assessed during the evaluation process.

**Table 4.1:** Simulation Parameters for Proposed System

| Parameter           | Value     |
|---------------------|-----------|
| Size of data packet | 1000 Byte |
| Bandwidth           | 10 Mbps   |
| Flow Request Limit  | 100       |
| Iteration interval  | 5 sec     |
| Blocking threshold  | 4         |
| Monitoring Period   | 5 sec     |

### 4.3 Performance Analysis and Comparison with Existing Techniques

The performance of proposed system is compared with the benchmarks (Gao et al., 2020; Imran et al., 2019; Revathi et al., 2022; Wang et al., 2022) and are described below.

#### 4.3.1 CPU Utilization

CPU utilization is the proportion of time that the CPU spends carrying out activities or processing commands. It is an important statistic to consider when evaluating the

performance and efficiency of a system, particularly in the context of network security, where the system must handle a variety of tasks, such as processing and analyzing packets.

The comparison of CPU utilization during normal traffic that TPAAD and benchmarked systems offers is shown in the Figure 4.2. During normal traffic the average CPU utilization of (Gao et al., 2020; Imran et al., 2019; Revathi et al., 2022; Wang et al., 2022) and the proposed system are almost equal as shown in Table 4.2 and Figure 4.2.

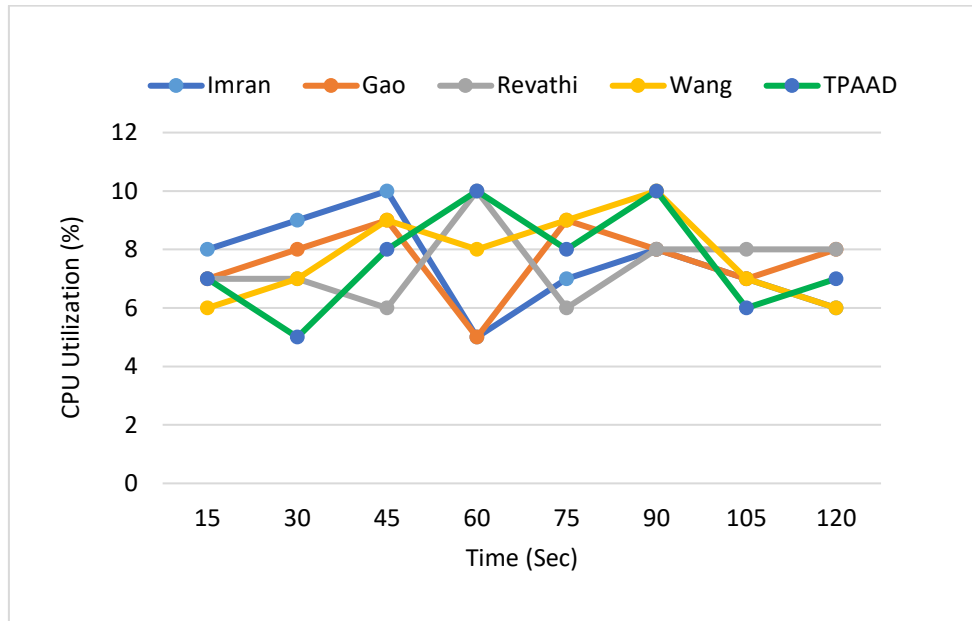


**Figure 4.2** CPU utilization with normal traffic

On the other hand, during DoS attack the average CPU utilization of the proposed system against benchmarked systems are compared and shown in Figure 4.3 and Table 4.2. During DoS attack, the controller's CPU resources are diverted to the endless installation of meaningless flow rules for flow tables, leaving less processing power for actual flow requests coming from the source. When an attack occurs, the percentage of time that the CPU is being used immediately increases to its high capacity, which is somewhere around 9

percent and takes less than 2 seconds to reach. Then it goes down slowly because the packet filter starts to pass this in malicious queue.

**Table 4.2:** CPU Utilization difference in % between Normal and DoS Attack Traffic

| Methodology                         | Normal Traffic (%) | With DoS Attacks (%) |
|-------------------------------------|--------------------|----------------------|
| FLOODDEFENDER<br>(Gao et al., 2020) | 7.63               | 14.87                |
| SECOD<br>(Wang et al., 2021)        | 7.75               | 11.50                |
| DSM-SVM<br>(Revathi et al., 2022)   | 7.52               | 9.75                 |
| TPAAD                               | 7.62               | 9.13                 |
| DAISY<br>(Imran et al., 2020)       | 7.50               | 8.50                 |

Following a delay of around 2 seconds, the level of CPU use never changes. However, at point, the packet in buffer is capable of storing the packet in messages in such an effective manner while only consuming approximately 1 % of the CPU's utilization. After approximately 8 seconds, there is a brief increase in CPU utilization as shown in Figure 4.3, which peaks at approximately 4% before dropping down precipitously in 2 second. This is due to the two-phase filtering that occurs in the packet filter. During DoS attack the average CPU utilization of Imran (Imran et al., 2020), Gao (Gao et al., 2020), Revathi (Revathi et al., 2022), Wang (Wang et al., 2021), and our proposed system are almost equal aa shown in Figure 4.3 and Table 4.2. The finding demonstrates that our system is able to conserve the computational power of the control layer in an effective manner. In addition to this it also demonstrates that the latency of the packet filter is relatively low. The level of CPU utilization during DoS attack benchmarks, along with a comparison, is shown in Figure 4.3.

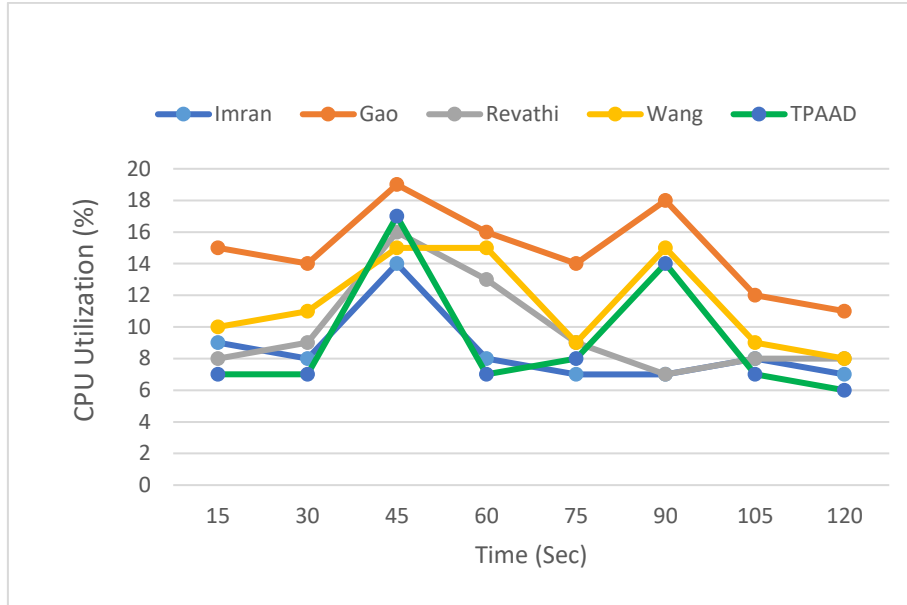


**Figure 4.3** CPU utilization with DoS attack

The enhanced performance is attributed to the implementation of TPAAD technique. The packet filter efficiently counteracts the DoS attack in Figure 4.3 (highlighted green) by promptly redirecting harmful traffic into the malevolent queue, therefore enabling more optimal CPU utilization. By limiting the execution of superfluous flow rules induced by the attack, the filtration decreases the load on the controller's CPU, therefore enabling more efficient utilization of the available computing resources.

#### 4.3.2 Control Channel Bandwidth

In a communication system, the control channel bandwidth is used to describe the fraction of available bandwidth or the amount of capacity that is set aside for the transmission of control information and the administration of network resources. Monitoring the utilisation of control channel bandwidth is essential for determining the efficiency and performance of a system, particularly in the context of network management and control. This is because monitoring the utilization of control channel bandwidth requires a lot of bandwidth.

The comparison of the percentage of control channel bandwidth during normal traffic that TPAAD and benchmarked systems is shown in Figure 4.4. During normal traffic the average control channel bandwidth of (Gao et al., 2020; Imran et al., 2019; Revathi et al., 2022; Wang et al., 2022) and the proposed system is almost equal as shown in Table 4.3 and Figure 4.4. Figure 4.4 exhibits slight variations over time, suggesting that the demand for bandwidth remains consistent during various time intervals, namely at 15, 30, and 45 seconds. The stability of the TPAAD system guarantees its ability to handle regular traffic without significant use of control channel resources, which is crucial for sustaining seamless communication between the switches and the controller. The essentially same performance seen throughout the benchmarked systems under normal traffic conditions validates that the proposed approach does not impose any extra burden, therefore ensuring effective allocation of resources and utilization of bandwidth.

The comparison intends to evaluate the performance and efficiency of various systems during routine operation by analyzing the control channel bandwidth under typical traffic situations. This makes it possible to evaluate each system fairly in terms of its capacity to use the control channel for delivering control messages without interference from actual or future DoS attacks.

**Table 4.3:** Control Channel Bandwidth diff. in % between Normal and DoS Attack Traffic

| Methodology                         | Normal Traffic (%) | With DoS Attacks (%) |
|-------------------------------------|--------------------|----------------------|
| DSM-SVM<br>(Revathi et al., 2022)   | 8.13               | 15.54                |
| DAISY<br>(Imran et al., 2020)       | 8.63               | 15.38                |
| SECOD<br>(Wang et al., 2021)        | 8.34               | 15.25                |
| FLOODDEFENDER<br>(Gao et al., 2020) | 8.25               | 15.13                |
| TPAAD                               | 8.13               | 14.13                |

The data shown in Table 4.2 illustrates that the benchmarked systems and the proposed system both maintain comparable control channel bandwidth percentages. This shows that each system is efficient at managing the control channel resources and using a predictable amount of bandwidth for the delivery of control messages.

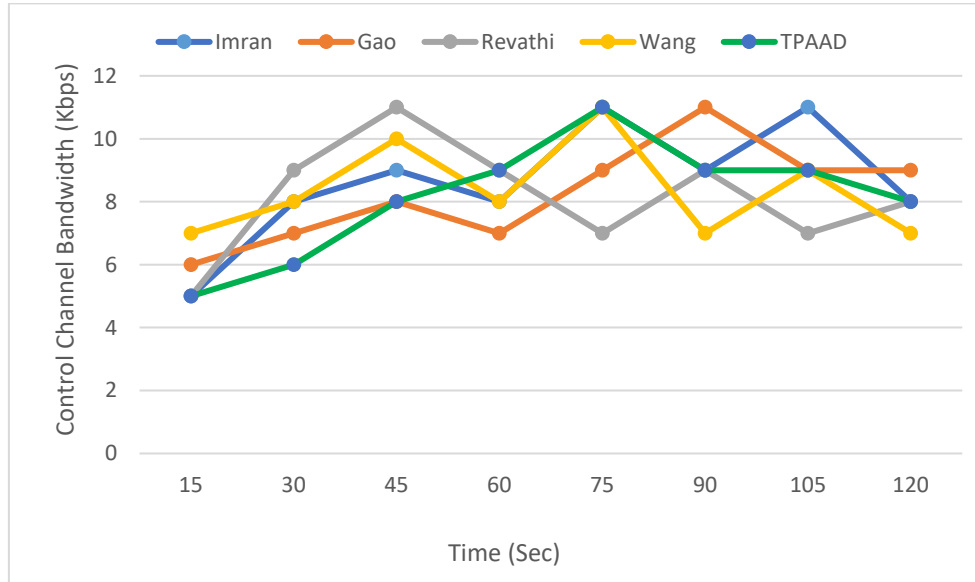


**Figure 4.4** Control Channel Bandwidth without DoS Attack

On the other hand, separating attack traffic from regular traffic can be challenging because the two are often mixed together. OpenFlow specifies that a switch will store a new packet in its Flow Buffer before sending a Packet In message to the controller for further direction if the switch has no prior knowledge of what to do with the packet. Therefore, to check bandwidth send multiple packets from same IP in 5s and see that when packets flood came in bandwidth increases by maximum of 30 kbps but it suddenly falls down and again increases after 30s as shown in Figure 4.5. During the peak of the DoS attack, benchmarked systems such as (Imran et al., 2019; Revathi et al., 2022) observed significant increases in bandwidth consumption (up to 30 kbps), followed by rapid declines and subsequent increases after 30 seconds. This indicates that the control channels in these systems become

flooded with attack traffic, leading to fluctuations in bandwidth usage. In contrast, the proposed system shows a more consistent trend in bandwidth utilization. While the bandwidth increases during the attack, it is managed more effectively, avoiding the significant oscillations seen in the benchmarks.

The controller may run out of resources needed to process normal traffic if DoS attack occurs and a large number of packet\_in messages are generated by the flooded traffic in a short period of time. Additionally, the attacking traffic may use up all available bandwidth between the controller and the switches, drastically slowing down the entire network. Requests for new flows must pass through the control channel, which can become congested during DoS attacks. To address this, the proposed solution not only blocks attacked traffic at switches but also implements an additional safety measure. If the same flow request from a blacklisted IP is detected four times, the packet is dropped, and the IP is blocked at the switches. This approach ensures that attacked traffic is prevented from passing through switches while adding an extra layer of protection. TPAAD method decreases the amount of attacked traffic that must pass through the control channel by using the packet filtering technique, reducing congestion and preserving control channel capacity. This decrease is also a direct result of the system's ability to block repetitive flow requests after the fourth occurrence from the same IP, thus conserving bandwidth and processing resources.

In comparison to the benchmarks, the control channel bandwidth is significantly decreased by the proposed solution as shown in Table 4.3. The benchmarked systems see bandwidth increases of 15.38%, 15.13%, 15.54%, and 15.25%, whereas the suggested TPAAD reduces control channel capacity by 14.13% as shown in Figure 4.5 (coloured in green). The percentages show how the suggested solution affects control channel bandwidth

in comparison with the other methods. The proposed approach exhibits exceptional management of bandwidth use during attack scenarios. While the initial processing may need more time and bandwidth, it effectively decreases long-term bandwidth usage by prompt filtering of harmful traffic at control layer. In addition to maintaining effective network operations and successfully mitigating DoS attacks in SDN, TPAAD ensures constant performance in both normal and attack scenarios.

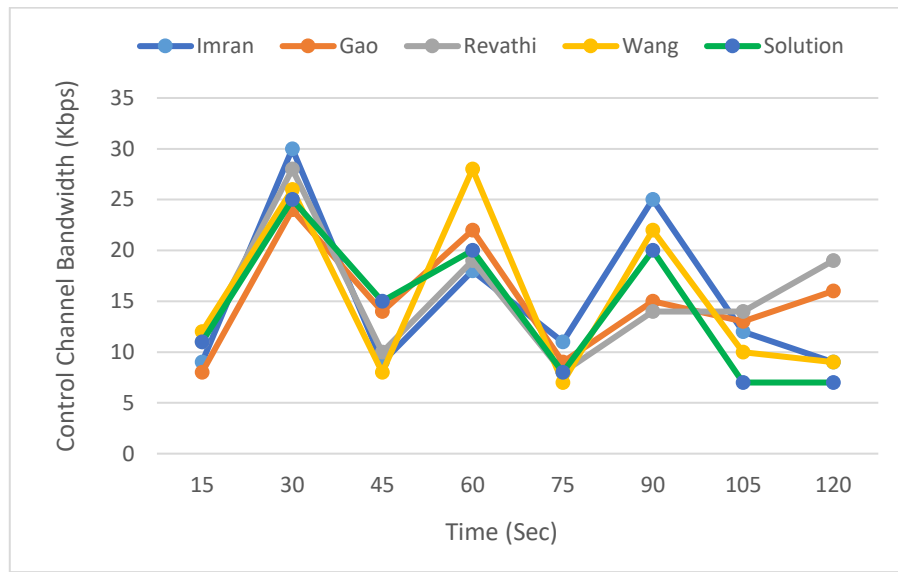


**Figure 4.5** Control Channel Bandwidth with DoS Attack

### 4.3.3 Packet-In messages

To handle incoming messages from the SDN, switches in an OpenFlow network employ flow rules. In the switches, these flow rules are kept in flow tables. A switch determines the best course of action when it receives a packet by comparing the header fields to the entries in its flow table. In an SDN, each packet's header is checked against the entries in the flow table during normal traffic flow. The outcome of this matching procedure will determine how the packet should be handled correctly. A related flow table action is triggered if a match is discovered in the flow table, informing the switch on how to handle

the packet. A table miss, which denotes that there is no entry that matches the packet's header, happens if there is no match in the flow table.

Figure 4.6 depicts the rate of packets transferred to the controller during normal traffic. The graph shows that for all benchmarks taken into consideration, the flow rate is between 200 and 400 packets per minute. Under typical traffic conditions, this range shows the usual rate of packet transfer from switches to the controller. The Packet In messages produced by normal traffic without a DoS attack are caused by flow table matches or table misses. While non-matching packets result in Packet In messages being delivered to the controller for further instructions, matching packets cause flow table actions to be triggered.

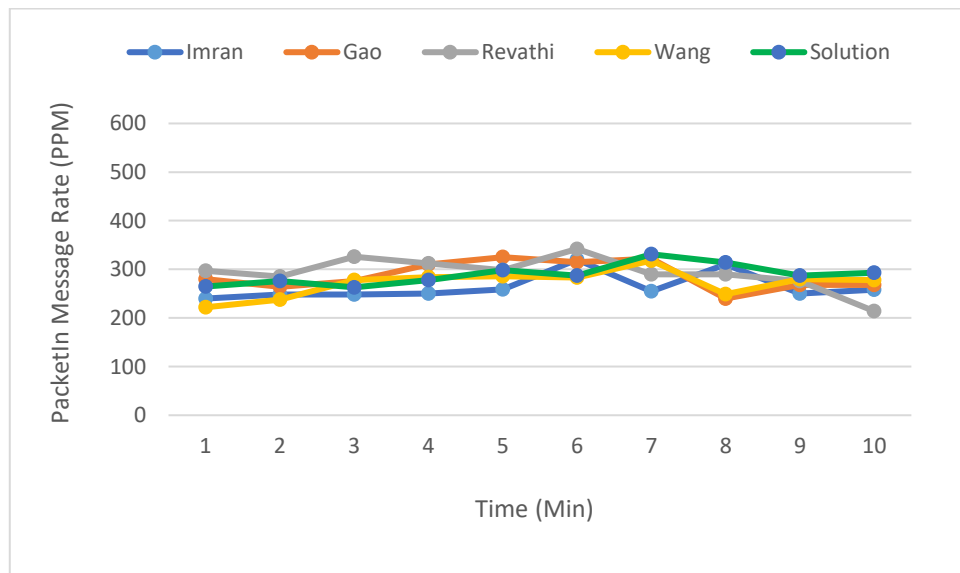


**Figure 4.6** Packet-In messages without DoS attack

While DoS attacks causes a large rise in the number of flows entering a switch, necessitating the insertion of additional rules to the switch's flow table. By comparing the arriving packets to the data in its database, the controller plays a critical part in this process. The controller first determines if the questioned packet is on the packet filtering module's banned list. The controller orders the switch to discard the packet if it is detected in the

prohibited list. By blocking known malicious traffic or traffic from sources that have been blacklisted, this packet filtering module assists in minimising the effect of the DoS attack. If there is no match in the current flow table, the controller creates a new flow table entry if the packet is not on the banned list. As a result, the controller is able to manage incoming packets from brand-new flows that were created as a result of the DoS attack.

The suggested method has a reduced packet drop ratio in comparison to previous benchmark systems, as seen by a smaller number of packet-in messages reaching the controller. Reduced processing of redundant flow requests enables more efficient handling of real packets, thereby enhancing the total packet delivery ratio. This indicates that the system has the capability to handle a greater amount of legitimate traffic during a DoS attack, therefore avoiding the misuse of bandwidth on harmful traffic.

In contrast, with the benchmarked systems as shown in Figure 4.7, the elevated packet-in message rates suggest an increased number of flow requests reaching the controller, thereby resulting in a higher rate of packet drop. Insufficient efficacy of these systems in filtering attack traffic results in an increased volume of flow requests being transmitted to the controller, thereby heightening the probability of resource depletion and a greater number of missed packets.

The average flow rate of requests made in one minute following a DoS attack is shown in Figure 4.7 and table 4.4 for both the benchmarked systems and the proposed solution. The benchmarks' values are 5565, 5209, 5874, and 5272 in the Figure 4.7, which shows a rise in flow rate as a result of the attack's surge in flows. The proposed approach, however, shows a lower average flow rate of 5197.

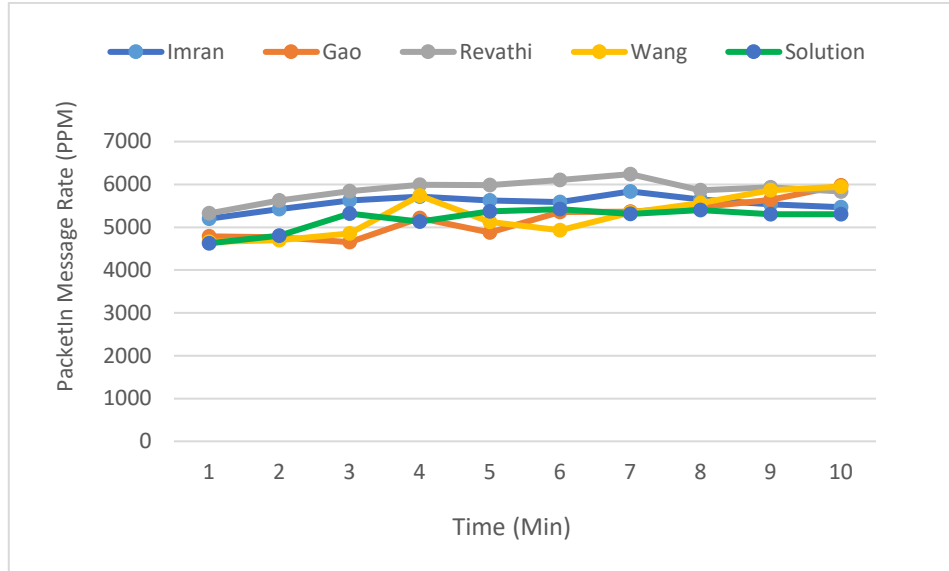


**Figure 4.7** Packet-In messages with DoS Attack

The inclusion of a packet filtration module, which successfully filters and discards harmful packets found in the disallowed list, is responsible for the recommended solution's reduced flow rate. The proposed method reduces the number of new flow requests that need to access the flow table, resulting in a lower average flow rate during a DoS attack. Table 4.4 shows the comparison of benchmark's flow rate requests for normal traffic and with DoS attacks.

**Table 4.4:** Packet Delivery Ratio Difference (%)

| Methodology                         | Normal Traffic (%) | With DoS Attacks (%) |
|-------------------------------------|--------------------|----------------------|
| DSM-SVM<br>(Revathi et al., 2022)   | 293                | 5874                 |
| DAISY<br>(Imran et al., 2020)       | 263                | 5565                 |
| SECOD<br>(Wang et al., 2021)        | 271                | 5272                 |
| FLOODDEFENDER<br>(Gao et al., 2020) | 286                | 5209                 |
| TPAAD                               | 289                | 5197                 |

#### 4.3.4 Evaluation of Latency

The proposed approach and benchmarks such as (Gao et al., 2020; Imran et al., 2019; Revathi et al., 2022; Wang et al., 2022) are represented in Figure 4.8, which presents a comparison of the average response times during normal traffic across several systems. In comparison to the benchmarked systems, which have average reaction times of 4.50%, 4.75%, 5.75%, and 5.0% for the different hosts (H1, H2, H3, and H4), the TPAAD exhibits a superior response time of 4.25%. The enhanced performance can be linked to the effective queue management system included in the proposed solution, which segregates normal packets and malicious packets into different queues. This guarantees the uninterrupted processing of regular traffic, free from any delays induced by malicious data, therefore leading to reduced latency during routine operations.

Based on Figure 4.9, it is evident that the response times for H1 are very consistent throughout all systems. However, the TPAAD demonstrates notable enhancements for H3 and H4, with response times that are much lower than the benchmark Figures. These findings indicate that the suggested queue mechanism is more efficient in situations with higher traffic or hosts, since it reduces the processing time for valid packets by removing potentially harmful ones at an early stage.

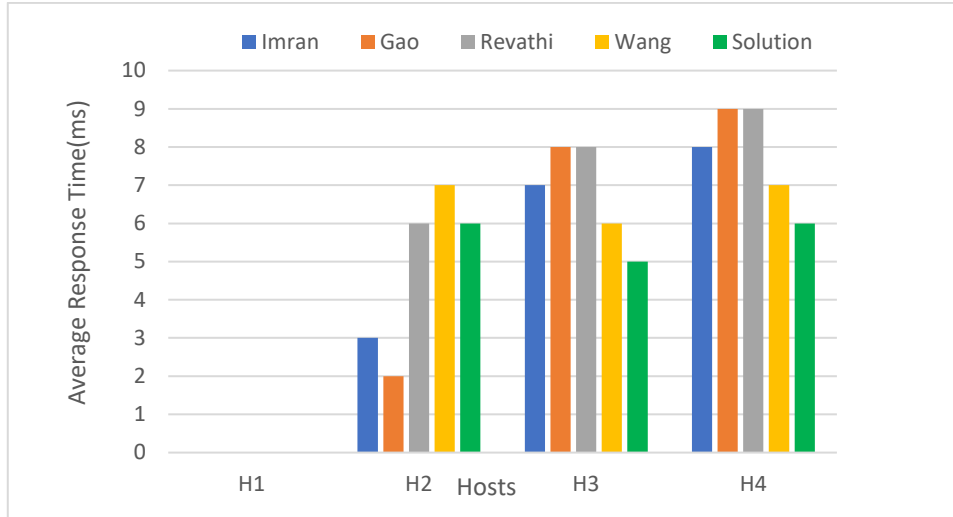


**Figure 4.8** Response time without DoS Attack

While during DoS attacks the average response time of the proposed solution is approximately (6.50%) similar to few of the benchmarked systems because of two phase authentication filter where packets are placed in two separate queues after filtration, and it rejects malicious traffic and frees up the controller to serve legitimate hosts. The average response time for the benchmarked systems as depicted in Figure 4.9 and Table 4.4 are 6.0%, 6.5%, 6.5% and 5.5% respectively.

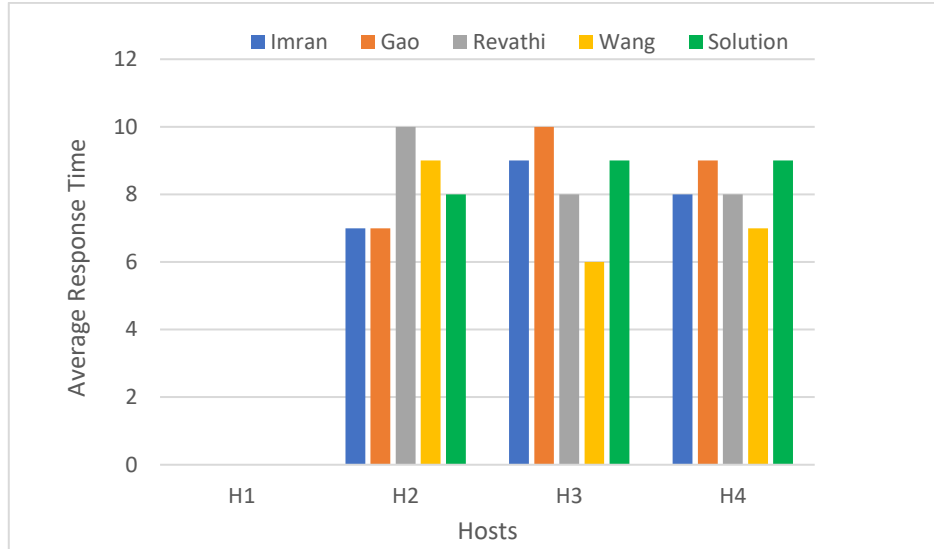


**Figure 4.9** Response time with DoS Attack

Overall, the TPAAD system operates effectively in normal traffic conditions, as seen in Figure 4.8, where its queue structure decreases the amount of delay experienced by genuine sites. This is a positive development in terms of the implications for latency. Nevertheless, when considering DoS attacks, as shown in Figure 4.9, the average response time tends to align more closely with other benchmarks. This suggests that the system's supplementary security measures, although successful in filtering hostile traffic, impose some processing burden. The proposed system is unable to surpass other benchmarked systems in the event of strong attack scenarios due to this overhead. As a result, the average response time during a DoS attack is comparable to other systems, although it is not considerably better than other systems.

Table 4.5 shows the comparison of the response time for benchmark systems under normal traffic and with DoS attacks.

**Table 4.5:** Response Time Diff in % between Normal and DoS Attack Traffic

| Benchmark's   | Normal Traffic (%) | With DoS Attacks (%) |
|---------------|--------------------|----------------------|
| FLOODDEFENDER | 4.75               | 6.5                  |
| DSM-SVM       | 5.75               | 6.5                  |
| TPAAD         | 4.25               | 6.5                  |
| DAISY         | 4.50               | 6.0                  |
| SECOD         | 5.00               | 5.5                  |

#### 4.3.5 Attack Identification False Rate

In Figure 4.10, we can see how well the TPAAD detects attacks. As the frequency of attacks increases, so does the proportion of false positives. This is due to the fact that a higher attack rate will result in a greater occurrence of the same flow within a given time window. So, in order to filter out attack traffic, packet filtration will employ a higher threshold, and move the attack traffic to malicious queue.

After packet filtering, the system employs SVM and KNN to reliably distinguish between attack traffic and legitimate communication, regardless of how nuanced or changing the features of the attack traffic may be. As a result, there are fewer false positives since the classifiers thoroughly evaluate packets that might have been wrongly reported as harmful by the initial filtration before classifying them as such. In this way, the system's recall and precision are enhanced by this multi-step process that guarantees that only packets that are indeed dangerous are blocked.

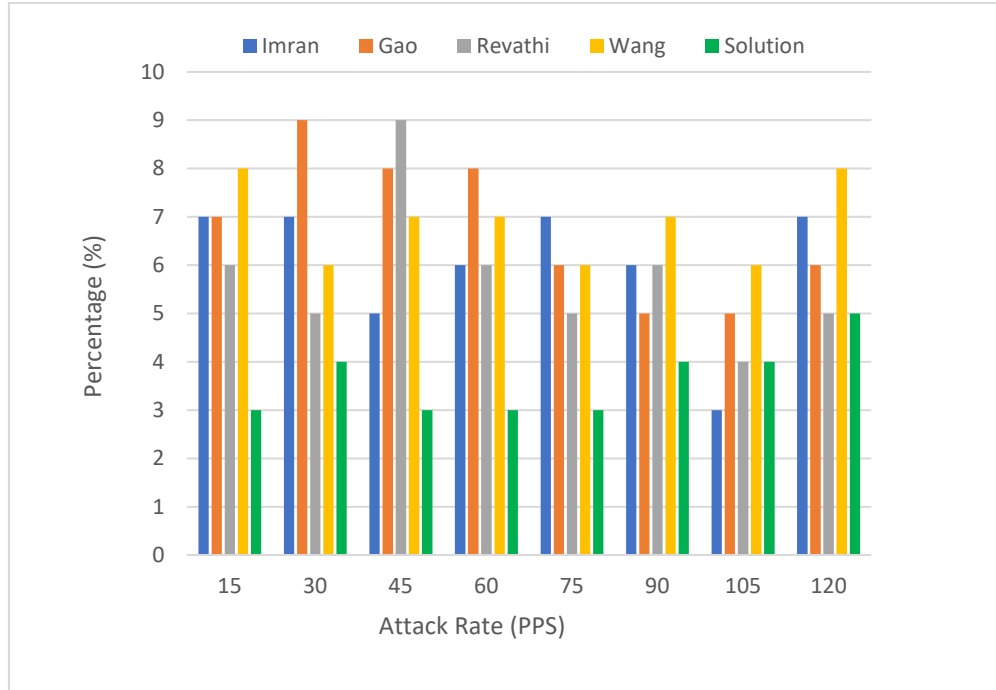


**Figure 4.10** Attack identification and false-positive rate

As the attack rate rises, a greater number of attack packets are falsely labelled as normal flow, but the fraction of false positives remains constant, making for a more consistent recall rate. These results, achieved using the SVM classifier, demonstrate that the TPAAD filtering method SVM has a false-positive rate of less than 5% and can accurately identify over 99% of attack traffic as shown in Table 4.5.

The TPAAD exhibits notably reduced false alarm generation in comparison to the benchmarks, especially in situations where other systems might mistakenly label benign traffic as harmful. In SDN systems, where it is essential to keep the controller's availability and efficiency at a high level, this is of the utmost importance.

**Table 4.6:** Detection Accuracy

| <b>Methodology</b> | <b>With DoS Attacks (%)</b> |
|--------------------|-----------------------------|
| TPAAD              | 99.56                       |
| DSM-SVM            | 99.12                       |
| DAISY              | 99.05                       |
| SECOD              | 98.64                       |
| FLOODDEFENDER      | 97.15                       |

#### **4.3.6 Detection Accuracy**

Using TPAAD, we evaluated our system with the dataset and found that some of the benchmarks generated false alarms during the evaluation. On the other hand, proposed solution only blocks the IP which continuously generating malicious packets. So, TPAAD system provides higher detection accuracy as 99.56% with SVM than the other benchmarked system i.e DAISY (Imran et al., 2020) has 99.05%, FloodDefender (Gao et al., 2020) has 97.15%, DSM-SVM (Revathi et al., 2022) has 99.12% and SECOD (Wang et al., 2021) has 98.64% as depicted in Figure 4.11 and Table 4.6.

The integration of SVM and KNN into the TPAAD system substantially enhances accuracy by ensuring that the decisions of the packet filtration module are verified by two reliable classification algorithms. By using these two classification strategies, the likelihood of incorrectly labelling normal packets as malicious is reduced, leading to a lower false-positive rate. In the end, after evaluation, SVM was chosen because its accuracy was better than KNN. The performance of used ML models is mentioned in Table 4.7 mentioned below.

**Table 4.7:** Performance of ML Models

| ML Algorithms | Accuracy | Sensitivity | Precision | F1-Score |
|---------------|----------|-------------|-----------|----------|
| SVM           | 99.23%   | 98.78%      | 99%       | 98%      |
| KNN           | 96.99%   | 97.13%      | 95%       | 95%      |

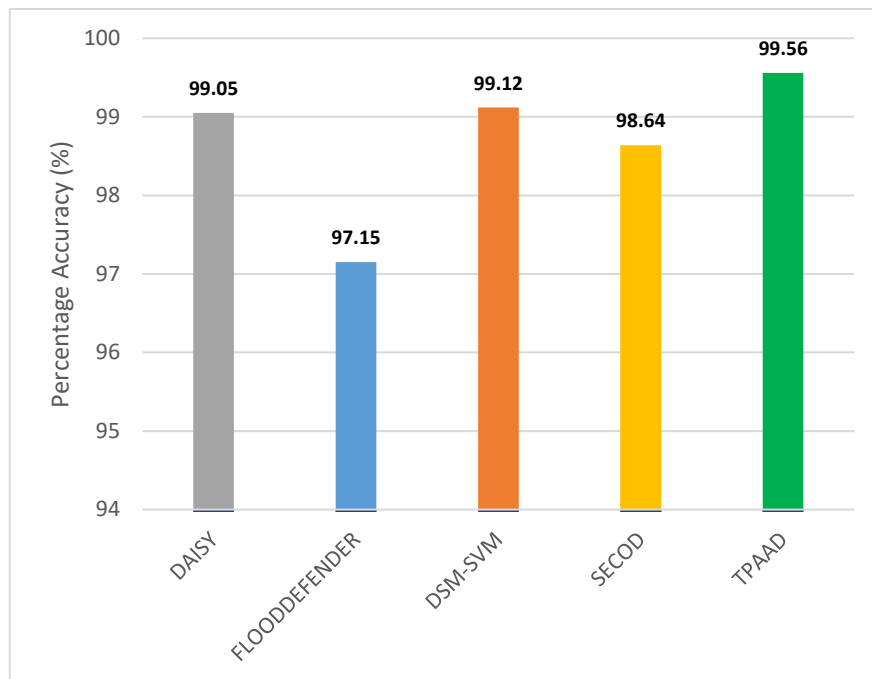


**Figure 4.11** Attack Detection Accuracy

#### 4.4 Summary

Because SDN is an open-source platform, network administrators now have greater flexibility in deciding where and how to deploy ML in the SDN for DoS detection. The ML model's analysis of the CICDoS2017 data set keeps the controller updated with real-time predictions of the network state. SDN's centralized control and administration streamlines data gathering for ML training and analysis, allowing for immediate implementation of new

rules in the network upon prediction. Since then, we have combined SDN with ML to manage and handle unexpected responses to network behavior. In this chapter, we investigate the potential of TPAAD for detecting DoS attacks using SVM and KNN. We have tried several training set types to determine the effect on performance. We have validated detection metrics such as CPU utilization, control channel bandwidth, packet-in message rate, latency evaluation, attack detection rate, and detection accuracy for each SL approach to facilitate a direct comparison of their respective performances. The results show a significant difference in the performance of TPAAD when it comes to generating false alarms for attack detection using two different queues as mentioned in attack detection. Results show that the proposed TPAAD solution offers a much better detection system with respect to false alarm generation and detection accuracy than the benchmarked algorithms.

## CHAPTER 5

### CONCLUSION AND RECOMMENDATIONS

#### 5.1 Conclusion

In this research, we investigate the issue of DoS/DDoS attacks on SDN-based network architectures, which is becoming increasingly critical as more businesses and network service providers adopt centralized control setups to reduce costs and expedite service delivery. This research showed that large-scale DoS attacks on SDN-based network topologies can be effectively mitigated by using TPAAD. Through the use of ML classifiers (SVM and KNN) in conjunction with a queue-based approach to distinguish between legitimate and malicious traffic, the system was able to minimize false positives and greatly increase detection accuracy. The SVM was ultimately chosen because to its exceptional accuracy, attaining a detection rate of 99.56%, surpassing benchmarks systems. The system effectively filtered attack traffic, alleviating the burden on the SDN control plane and assuring network reliability during attack situations, while sustaining a low false-positive rate.

The research is organized as follows: Chapter 1 discusses SDN's DoS/DDoS vulnerabilities and the study's logic. Chapter 2 reviews the literature and highlights failures in current detection methods and the need for ML in SDN. In Chapter 3, TPAAD design and technique emphasize packet filtration, queue management, and SVM and KNN classifiers. Chapter 4 describes the experimental setup, using CICDoS2017 to evaluate TPAAD's accuracy and decreased false positives compared to benchmarks. The main findings in Chapter 5 illustrate TPAAD's performance and recommend future study to improve its use and scalability.

## 5.2 Research Contributions

Final chapter concludes our examination of the detection and mitigation of DoS attacks in SDN systems. This research has effectively addressed the problems caused by DoS/DDoS attacks within SDN designs, which has made a substantial contribution to network security.

Major accomplishment of this research is the development of a more advanced SDN DoS attack detection system. SDN controllers' network visibility and control features were used to develop detection algorithms. These methods assess typical network behavior, atypical activity, and resource utilization to successfully detect DoS attacks. SDN's global network view and fine-grained management enable clever detection algorithms to dynamically divert traffic away from attacked network segments to less busy channels. This preventative measure lessens the effect of attacks on the network with respect to performance and guarantees continuous service delivery to genuine users. In this way the proposed TPAAD detection system is able to successfully differentiate between legitimate traffic and malicious activity because to its high accuracy and low false positive rates, both of which are made possible by the centralised view of the network.

Finally, the research addressed the major issue of preventing DoS attacks in SDN environments. This research developed a dynamic and flexible mitigation system that filters traffic by queues using programmable switches' adaptation. Our system continuously monitors traffic and analyzes attacks, allowing fast defense policy changes. This prevents harmful attacks and permits only legitimate traffic to pass through. Simulations demonstrate that our mitigation mechanism prevents broad range of DoS attacks without putting.

### 5.3 Future Work

As part of future research efforts, we can extend our work in the following areas. We can continue our work in the following areas as part of our future research efforts. This study focuses on flooding attacks specifically in terms of DoS attack detection, although there are a lot of other kinds of DoS attacks in SDN that are still investigated. To make the attack scenarios more realistic, fake addresses are included to increase the difficulty of detection. The reason behind this is that there are situations where the counter-threshold method might not be enough.

In order to build a SL model for ML-based detection, we will examine additional characteristics of DoS attacks. This is because current countermeasures are unable to identify all DoS attacks. In addition, testing TPAAD in a real-world network setting could be part of future studies.

## REFERENCES

- Abid, Y. A., Wu, J., Xu, G., Fu, S., & Waqas, M. (2024). Multi-level deep neural network for distributed denial-of-service attack detection and classification in software-defined networking supported internet of things networks. *IEEE Internet of Things Journal*, 11(14), 24715 - 24725. <https://doi.org/10.1109/JIOT.2024.3376578>
- Adhikari, T., Kule, M., & Khan, A. K. (2022). An ECDH and AES based encryption approach for prevention of MiTM in SDN southbound communication interface. In *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Kharagpur, India.
- Ahmad, A., Harjula, E., Ylianttila, M., & Ahmad, I. (2020). Evaluation of machine learning techniques for security in SDN. *2020 IEEE Globecom Workshops (GC Wkshps)*, Taipei, Taiwan.
- Ahmad, I., Namal, S., Ylianttila, M., & Gurtov, A. (2015). Security in software defined networks: A survey. *IEEE Communications Surveys & Tutorials*, 17(4), 2317-2346.
- Ahmad, Z., Khan, A. S., Aqeel, S., Julaihi, A. A., Tarmizi, S., Annuar, N., & Habeeb, M. S. (2022). S-ADS: Spectrogram image-based anomaly detection system for IoT networks. *2022 Applied Informatics International Conference (AiIC)*, Serdang, Malaysia.
- Ahmed, S. (2021). A study of ML algorithms for DDoS detection. *International Journal for Research in Applied Science and Engineering Technology*, 2(1), 1-14.
- Al-Mafrachi, B. H. A. (2017). Detection of DDoS attacks against the SDN controller using statistical approaches, *Wright State University*.

- Al-Zewairi, M., Suleiman, D., & Almajali, S. (2017). An experimental software defined security controller for software defined network. *2017 Fourth International Conference on Software Defined Systems (SDS), Valencia, Spain*.
- Al Mtawa, Y., Haque, A., & Lutfiyya, H. (2021). Migrating from legacy to software defined networks: A network reliability perspective. *IEEE Transactions on Reliability*, 70(4), 1525-1541.
- Alashhab, Z. R., Anbar, M., Singh, M. M., Hasbullah, I. H., Jain, P., & Al-Amiedy, T. A. (2022). Distributed denial of service attacks against cloud computing environment: survey, issues, challenges and coherent taxonomy. *Applied Sciences*, 12(23), 12441.
- AlGhadhban, A., & Showail, A. (2022). SALMA: A novel middlebox infrastructure system based on integrated subnets. *Systems*, 10(5), 165.
- Ali, S. T., Sivaraman, V., Radford, A., & Jha, S. (2015). A survey of securing networks using software defined networking. *IEEE Transactions on Reliability*, 64(3), 1086-1097.
- Alpaydin, E. (2020). *Introduction to machine learning*. MIT press.
- Alsaeedi, M., Mohamad, M. M., & Al-Roubaiey, A. A. (2019). Toward adaptive and scalable OpenFlow-SDN flow control: A survey. *IEEE access*, 7, 107346-107379.
- Alshamrani, A., Chowdhary, A., Pisharody, S., Lu, D., & Huang, D. (2017). A defense system for defeating DDoS attacks in SDN based networks. *Proceedings of the 15th ACM International Symposium on Mobility Management and Wireless Access, New York, United States*.
- Alsmadi, I., & Xu, D. (2015). Security of software defined networks: A survey. *Computers & Security*, 53, 79-108.

- Altamemi, A. J., Abdulhassan, A., & Obeis, N. T. (2022). DDoS attack detection in software defined networking controller using machine learning techniques. *Bulletin of Electrical Engineering and Informatics*, 11(5), 2836-2844.
- Angarita-Zapata, J. S., Masegosa, A. D., & Triguero, I. (2019). A taxonomy of traffic forecasting regression problems from a supervised learning perspective. *IEEE Access*, 7, 68185-68205.
- Anwer, B., Benson, T., Feamster, N., Levin, D., & Rexford, J. (2013). A slick control plane for network middleboxes. *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, Hong Kong, China*.
- Arbettu, R. K., Khondoker, R., Bayarou, K., & Weber, F. (2016). Security analysis of OpenDaylight, ONOS, Rosemary and Ryu SDN controllers. *2016, 17th International Telecommunications Network Strategy and Planning Symposium (Networks), Montreal, QC, Canada*.
- Arif, M., Wang, G., Geman, O., Balas, V. E., Tao, P., Brezulianu, A., & Chen, J. (2020). SDN-based vanets, security attacks, applications, and challenges. *Applied Sciences*, 10(9), 3217.
- Aziz, M., Fazely, H. A., Landi, G., Gallico, D., Christodoulopoulos, K., & Wieder, P. (2016). SDN-enabled application-aware networking for data center networks. *2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS), Monte Carlo, Monaco*.
- Bahare, M. K., Gavras, A., Gramaglia, M., Cosmas, J., Li, X., Bulakci, Ö., Rahman, A., Kostopoulos, A., Mesodiakaki, A., & Tsolkas, D. (2023). The 6G Architecture Landscape-European perspective.

- Banitalebi Dehkordi, A., Soltanaghaei, M., & Boroujeni, F. Z. (2021). The DDoS attacks detection through machine learning and statistical methods in SDN. *The Journal of Supercomputing*, 77(3), 2383-2415.
- Banton, M. D. (2020). *A Deep Learning-based Approach to Identifying and Mitigating Network Attacks Within SDN Environments Using Non-standard Data Sources*. Liverpool John Moores University (United Kingdom).
- Belyaev, M., & Gaivoronski, S. (2014). Towards load balancing in SDN-networks during DDoS-attacks. In 2014 *International Science and Technology Conference (modern networking technologies)(MoNeTeC)*, Moscow, Russia.
- Bera, P., Saha, A., & Setua, S. K. (2016). Denial of service attack in software defined network. In 2016 *5th International Conference on Computer Science and Network Technology (ICCSNT)*, Changchun, China.
- Bi, W. L., Hosny, A., Schabath, M. B., Giger, M. L., Birkbak, N. J., Mehrtash, A., Allison, T., Arnaout, O., Abbosh, C., & Dunn, I. F. (2019). Artificial intelligence in cancer imaging: clinical challenges and applications. *CA: a cancer journal for clinicians*, 69(2), 127-157.
- Cai, J., Luo, J., Wang, S., & Yang, S. (2018). Feature selection in machine learning: A new perspective. *Neurocomputing*, 300, 70-79.
- Chen, K.-Y., Liu, S., Xu, Y., Siddhau, I. K., Zhou, S., Guo, Z., & Chao, H. J. (2021). SDNShield: NFV-based defense framework against DDoS attacks on SDN control plane. *IEEE/ACM Transactions on Networking*, 30(1), 1-17.
- Chica, J. C. C., Imbachi, J. C., & Vega, J. F. B. (2020). Security in SDN: A comprehensive survey. *Journal of Network and Computer Applications*, 159, 102595.

- Chin, T., Xiong, K., & Rahouti, M. (2018). SDN-based kernel modular countermeasure for intrusion detection. *Security and Privacy in Communication Networks. Proceedings in 13th International Conference, SecureComm 2017, Niagara Falls, ON, Canada, October 22–25, 2017.*
- Chuang, H.-M., Liu, F., & Tsai, C.-H. (2022). Early detection of abnormal attacks in software-defined networking using machine learning approaches. *Symmetry*, *14*(6), 1178. <https://doi.org/doi.org/10.3390/sym14061178>
- Cox, J. H., Chung, J., Donovan, S., Ivey, J., Clark, R. J., Riley, G., & Owen, H. L. (2017). Advancing software-defined networks: A survey. *IEEE access*, *5*, 25487-25526.
- Cui, H., Karame, G. O., Klaedtke, F., & Bifulco, R. (2016). On the fingerprinting of software-defined networks. *IEEE Transactions on Information Forensics and Security*, *11*(10), 2160-2173.
- Dabbagh, M., Hamdaoui, B., Guizani, M., & Rayes, A. (2015). Software-defined networking security: pros and cons. *IEEE Communications Magazine*, *53*(6), 73-79.
- Dacier, M. C., König, H., Cwalinski, R., Kargl, F., & Dietrich, S. (2017). Security challenges and opportunities of software-defined networking. *IEEE Security & Privacy*, *15*(2), 96-100.
- Dargahi, T., Caponi, A., Ambrosin, M., Bianchi, G., & Conti, M. (2017). A survey on the security of stateful SDN data planes. *IEEE communications surveys & tutorials*, *19*(3), 1701-1725.
- Dong, P., Du, X., Zhang, H., & Xu, T. (2016). A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows. *In 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia.*

- Flauzac, O., González, C., Hachani, A., & Nolot, F. (2015). SDN based architecture for IoT and improvement of the security. *In 2015 IEEE 29th international conference on advanced information networking and applications workshops, Gwangju, Korea (South)*.
- Gao, S., Li, Z., Xiao, B., & Wei, G. (2018). Security threats in the data plane of software-defined networks. *IEEE network*, 32(4), 108-113.
- Gao, S., Peng, Z., Xiao, B., Hu, A., Song, Y., & Ren, K. (2020). Detection and mitigation of DoS attacks in software defined networks. *IEEE/ACM Transactions on Networking*, 28(3), 1419-1433.
- Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., & Shenker, S. (2008). NOX: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, 38(3), 105-110.
- Hadem, P., Saikia, D. K., & Moulik, S. (2021). An SDN-based intrusion detection system using SVM with selective logging for IP traceback. *Computer Networks*, 191, 108015. <https://doi.org/doi.org/10.1016/j.comnet.2021.108015>
- Han, Z., Li, X., Huang, K., & Feng, Z. (2018). A software defined network-based security assessment framework for cloudIoT. *IEEE Internet of Things Journal*, 5(3), 1424-1434. <https://doi.org/doi.org/10.1109/JIOT.2018.2801944>
- Horvath, R., Nedbal, D., & Stieninger, M. (2015). A literature review on challenges and effects of software defined networking. *Procedia Computer Science*, 64, 552-561. <https://doi.org/doi.org/10.1016/j.procs.2015.08.563>
- Huang, G., Song, S., Gupta, J. N., & Wu, C. (2014). Semi-supervised and unsupervised extreme learning machines. *IEEE transactions on cybernetics*, 44(12), 2405-2417. <https://doi.org/doi.org/10.1109/TCYB.2014.2307349>

- Imran, M., Durad, M. H., Khan, F. A., & Abbas, H. (2019). DAISY: A detection and mitigation system against denial-of-service attacks in software-defined networks. *IEEE Systems Journal*, 14(2), 1933-1944. <https://doi.org/doi.org/10.1109/JSYST.2019.2927223>
- Imran, M., Durad, M. H., Khan, F. A., & Derhab, A. (2019). Reducing the effects of DoS attacks in software defined networks using parallel flow installation. *Human-centric Computing and Information Sciences*, 9, 1-19. <https://doi.org/doi.org/10.1186/s13673-019-0176-7>
- Iqbal, W., Abbas, H., Daneshmand, M., Rauf, B., & Bangash, Y. A. (2020). An in-depth analysis of IoT security requirements, challenges, and their countermeasures via software-defined security. *IEEE Internet of Things Journal*, 7(10), 10250-10276. <https://doi.org/doi.org/10.1109/JIOT.2020.2997651>
- Jazi, H. H., Gonzalez, H., Stakhanova, N., & Ghorbani, A. A. (2017). Detecting HTTP-based application layer DoS attacks on web servers in the presence of sampling. *Computer Networks*, 121, 25-36. <https://doi.org/doi.org/10.1016/j.comnet.2017.03.018>
- Jia, Y., Xu, L., Yang, Y., & Zhang, X. (2019). Lightweight automatic discovery protocol for openflow-based software defined networking. *IEEE Communications Letters*, 24(2), 312-315. <https://doi.org/doi.org/10.1109/LCOMM.2019.2956033>
- Jimenez, M. B., Fernandez, D., Rivadeneira, J. E., Bellido, L., & Cardenas, A. (2021). A survey of the main security issues and solutions for the SDN architecture. *IEEE access*, 9, 122016-122038. <https://doi.org/doi.org/10.1109/ACCESS.2021.3109564>
- Kalkan, K., Gur, G., & Alagoz, F. (2017). Defense mechanisms against DDoS attacks in SDN environment. *IEEE Communications Magazine*, 55(9), 175-179. <https://doi.org/doi.org/10.1109/MCOM.2017.1600970>

- Kandoi, R., & Antikainen, M. (2015). Denial-of-service attacks in OpenFlow SDN networks. *In 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, Canada.*
- Kelleher, J. D., Mac Namee, B., & D'arcy, A. (2020). *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*. MIT press.
- Keshari, S. K., Kansal, V., & Kumar, S. (2021). A systematic review of quality of services (QoS) in software defined networking (SDN). *Wireless Personal Communications, 116*(3), 2593-2614. <https://doi.org/doi.org/10.1007/s11277-020-07812-2>
- Khan, A. S., Sattar, M. A., Nisar, K., Ibrahim, A. A. A., Annuar, N. B., Abdullah, J. b., & Karim Memon, S. (2022). A survey on 6G enabled light weight authentication protocol for UAVs, security, open research issues and future directions. *Applied Sciences, 13*(1), 277. <https://doi.org/doi.org/10.3390/app13010277>
- Koning, R., de Graaff, B., de Laat, C., Meijer, R., & Grosso, P. (2016). Interactive analysis of sdn-driven defence against distributed denial of service attacks. *In 2016 IEEE NetSoft Conference and Workshops (NetSoft), Seoul, Korea (South).*
- Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2014). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE, 103*(1), 14-76. <https://doi.org/doi.org/10.1109/JPROC.2014.2371999>
- Kuerban, M., Tian, Y., Yang, Q., Jia, Y., Huebert, B., & Poss, D. (2016). FlowSec: DOS attack mitigation strategy on SDN controller. *In 2016 IEEE International Conference on Networking, Architecture and Storage (NAS), Long Beach, CA, USA.*
- Kumar, P., Tripathi, M., Nehra, A., Conti, M., & Lal, C. (2018). SAFETY: Early detection and mitigation of TCP SYN flood utilizing entropy in SDN. *IEEE Transactions on Network and Service Management, 15*(4), 1545-1559.

- Lara, A., Kolasani, A., & Ramamurthy, B. (2013). Network innovation using openflow: A survey. *IEEE Communications Surveys & Tutorials*, 16(1), 493-512.
- Lara, A., & Ramamurthy, B. (2016). OpenSec: Policy-based security using software-defined networking. *IEEE Transactions on Network and Service Management*, 13(1), 30-42.
- Li, C., Qin, Z., Novak, E., & Li, Q. (2017). Securing SDN infrastructure of IoT–fog networks from MitM attacks. *IEEE Internet of Things Journal*, 4(5), 1156-1164.  
<https://doi.org/doi.org/10.1109/JIOT.2017.2685596>
- Liatifis, A., Sarigiannidis, P., Argyriou, V., & Lagkas, T. (2023). Advancing sdn from openflow to p4: A survey. *ACM Computing Surveys*, 55(9), 1-37.  
<https://doi.org/doi.org/10.1145/3556973>
- Liu, J., & Xu, Q. (2019). Machine learning in software defined network. In *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China*.
- Ma, H., Ding, H., Yang, Y., Mi, Z., Yang, J. Y., & Xiong, Z. (2016). Bayes-based ARP attack detection algorithm for cloud centers. *Tsinghua science and technology*, 21(1), 17-28.
- Mahrach, S., & Haqiq, A. (2020). DDoS flooding attack mitigation in software defined networks. *International Journal of Advanced Computer Science and Applications*, 11(1). <https://doi.org/doi.org/10.14569/ijacsa.2020.0110185>
- Masoudi, R., & Ghaffari, A. (2016). Software defined networks: A survey. *Journal of Network and Computer Applications*, 67, 1-25.  
<https://doi.org/doi.org/10.1016/j.jnca.2016.03.016>
- Mazhar, M. M., Jamil, M. A., Mazhar, A., Ellahi, A., Jamil, M. S., & Mahmood, T. (2015). Conceptualization of software defined network layers over internet of things for

- future smart cities applications. *In 2015 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE), Orlando, FL, USA.*
- Merlin, J. R. D. (2018). *Machine-learning and statistical methods for DDoS attack detection and defense system in software defined networks* (Doctoral dissertation, Toronto Metropolitan University).
- Mirkovic, J., & Reiher, P. (2004). A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2), 39-53.  
<https://doi.org/doi.org/10.1145/997150.997156>
- Moazzeni, S., Khayyambashi, M. R., Movahhedinia, N., & Callegati, F. (2018). On reliability improvement of software-defined networks. *Computer Networks*, 133, 195-211. <https://doi.org/doi.org/10.1016/j.comnet.2018.01.023>
- Mohammadi, R., Javidan, R., & Conti, M. (2017). Slicots: An sdn-based lightweight countermeasure for tcp syn flooding attacks. *IEEE Transactions on Network and Service Management*, 14(2), 487-497.  
<https://doi.org/doi.org/10.1109/TNSM.2017.2701549>
- Mohammed, S. S., Hussain, R., Senko, O., Bimaganbetov, B., Lee, J., Hussain, F., Kerrache, C. A., Barka, E., & Bhuiyan, M. Z. A. (2018). A new machine learning-based collaborative DDoS mitigation mechanism in software-defined network. *In 2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Limassol, Cyprus.*
- Morente-Molinera, J. A., Mezei, J., Carlsson, C., & Herrera-Viedma, E. (2016). Improving supervised learning classification methods using multigranular linguistic modeling and fuzzy entropy. *IEEE transactions on fuzzy systems*, 25(5), 1078-1089.  
<https://doi.org/doi.org/10.1109/TFUZZ.2016.2594275>

- Mot, T., Sorn, D., Tiv, D., Heav, T., Ron, S., & Khon, K. (2024). Cloud-Based Remote Control of Household Appliances: Enhancing Convenience and Energy Efficiency. *In 2024 16th International Conference on Knowledge and Smart Technology (KST), Krabi, Thailand.*
- Mousavi, S. M., & St-Hilaire, M. (2018). Early detection of DDoS attacks against software defined network controllers. *Journal of Network and Systems Management*, 26, 573-591. <https://doi.org/doi.org/10.1007/s10922-017-9432-1>
- Murphy, K. P. (2018). Machine learning: A probabilistic perspective (adaptive computation and machine learning series). *The MIT Press: London, UK.*
- Nguyen, T.-H., & Yoo, M. (2016). Attacks on host tracker in SDN controller: Investigation and prevention. *In 2016 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea (South).*
- Nikitinskiy, M. A., & Alekseev, I. V. (2015). Analyzing the possibility of applying asymmetric transport protocols in terms of software defined networks. *Automatic Control and Computer Sciences*, 49, 94-102. <https://doi.org/doi.org/10.3103/S0146411615020042>
- Nisa, N., Khan, A. S., Ahmad, Z., & Abdullah, J. (2024). TPAAD: Two-phase authentication system for denial of service attack detection and mitigation using machine learning in software-defined network. *International Journal of Network Management*, 34(3), e2258.
- Oktian, Y. E., Lee, S., & Lee, H. (2014). Mitigating denial of service (dos) attacks in openflow networks. *In 2014 International Conference on Information and Communication Technology Convergence (ICTC), Busan, Korea (South).*

- Paquin, F., Rivnay, J., Salleo, A., Stingelin, N., & Silva, C. (2013). Multi-phase semicrystalline microstructures drive exciton dissociation in neat plastic semiconductors. *arXiv preprint arXiv:1310.8002*.  
<https://doi.org/doi.org/10.1039/C5TC02043C>
- Parashar, M., Poonia, A., & Satish, K. (2019). A survey of attacks and their mitigations in software defined networks. *In 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India*
- Peng, H., Sun, Z., Zhao, X., Tan, S., & Sun, Z. (2018). A detection method for anomaly flow in software defined network. *IEEE Access*, 6, 27809-27817.
- Peng, J., Zhang, P., & Riedel, N. (2008). Discriminant learning analysis. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(6), 1614-1625.  
<https://doi.org/doi.org/10.1109/TSMCB.2008.2002852>
- Pillai, S. E. V. S., & Polimetla, K. (2024). Integrating Network Security into Software Defined Networking (SDN) Architectures. 2024 International Conference on Integrated Circuits and Communication Systems (ICICACS), Raichur, India.
- Polat, H., Polat, O., & Cetin, A. (2020). Detecting DDoS attacks in software-defined networks through feature selection methods and machine learning models. *Sustainability*, 12(3), 1035. <https://doi.org/doi.org/10.3390/su12031035>
- Rafique, W., Qi, L., Yaqoob, I., Imran, M., Rasool, R. U., & Dou, W. (2020). Complementing IoT services through software defined networking and edge computing: A comprehensive survey. *IEEE communications surveys & tutorials*, 22(3), 1761-1804. <https://doi.org/doi.org/10.1109/COMST.2020.2997475>

- Rahman, O., Quraishi, M. A. G., & Lung, C.-H. (2019). DDoS attacks detection and mitigation in SDN using machine learning. *In 2019 IEEE world congress on services (SERVICES), Milan, Italy.*
- Rahouti, M., Xiong, K., Xin, Y., Jagatheesaperumal, S. K., Ayyash, M., & Shaheed, M. (2022). SDN security review: Threat taxonomy, implications, and open challenges. *IEEE access*, 10, 45820-45854. <https://doi.org/doi.org/10.1109/ACCESS.2022.3168972>
- Rawat, D. B., & Reddy, S. R. (2016). Software defined networking architecture, security and energy efficiency: A survey. *IEEE communications surveys & tutorials*, 19(1), 325-346. <https://doi.org/doi.org/10.1109/COMST.2016.2618874>
- Revathi, M., Ramalingam, V., & Amutha, B. (2022). A machine learning based detection and mitigation of the DDOS attack by using SDN controller framework. *Wireless Personal Communications*, 1-25. <https://doi.org/doi.org/10.1007/s11277-021-09071-1>
- Rimal, A. N., & Praveen, R. (2020). DDOS attack detection using machine learning. *J. Emerging Technol. and Innovative Res.*, 7(6), 185-188.
- Roopa Devi, E., & Suganthe, R. (2020). Enhanced transductive support vector machine classification with grey wolf optimizer cuckoo search optimization for intrusion detection system. *Concurrency and Computation: Practice and Experience*, 32(4), e4999. <https://doi.org/doi.org/10.1002/cpe.4999>
- Rowshanrad, S., Abdi, V., & Keshtgari, M. (2016). Performance evaluation of SDN controllers: Floodlight and OpenDaylight. *IIUM Engineering Journal*, 17(2), 47-57. <https://doi.org/doi.org/10.31436/iiumej.v17i2.615>

- Sahoo, K. S., Puthal, D., Tiwary, M., Rodrigues, J. J., Sahoo, B., & Dash, R. (2018). An early detection of low rate DDoS attack to SDN based data center networks using information distance metrics. *Future Generation Computer Systems*, 89, 685-697.
- Sahoo, K. S., Tripathy, B. K., Naik, K., Ramasubbareddy, S., Balusamy, B., Khari, M., & Burgos, D. (2020). An evolutionary SVM model for DDOS attack detection in software defined networks. *IEEE Access*, 8, 132502-132513.
- Sahran, F., Altarturi, H. H., & Anuar, N. B. (2023). Exploring the landscape of AI-SDN: A comprehensive bibliometric analysis and future perspectives. *Electronics*, 13(1), 26. <https://doi.org/doi.org/10.3390/electronics13010026>
- Sahu, S. K., Mokhade, A., & Bokde, N. D. (2023). An overview of machine learning, deep learning, and reinforcement learning-based techniques in quantitative finance: recent progress and challenges. *Applied Sciences*, 13(3), 1956. <https://doi.org/doi.org/10.3390/app13031956>
- Santos, R., Souza, D., Santo, W., Ribeiro, A., & Moreno, E. (2020). Machine learning algorithms to detect DDoS attacks in SDN. *Concurrency and Computation: Practice and Experience*, 32(16), e5402. <https://doi.org/doi.org/10.1002/cpe.5402>
- Sasaki, T., Perrig, A., & Asoni, D. E. (2016). Control-plane isolation and recovery for a secure SDN architecture. In *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, Seoul, Korea (South).
- Scott-Hayward, S., Natarajan, S., & Sezer, S. (2015). A survey of security in software defined networks. *IEEE communications surveys & tutorials*, 18(1), 623-654. <https://doi.org/doi.org/10.1109/COMST.2015.2453114>

- Shariff, M. A. (2021). Multiple Controllers in SDN - A Comprehensive Survey. *International Journal of Advanced Research in Computer and Communication Engineering*, 10(10), 121-123. <https://doi.org/10.17148/IJARCCE.2021.101023>
- Shin, S., Yegneswaran, V., Porras, P., & Gu, G. (2013). Avant-guard: Scalable and vigilant switch flow management in software-defined networks. *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, Berlin, Germany*.
- Shoeb, A., & Chithralekha, T. (2016). Resource management of switches and Controller during saturation time to avoid DDoS in SDN. In *2016 IEEE International Conference on Engineering and Technology (ICETECH), Coimbatore, India*.
- Shu, Z., Wan, J., Li, D., Lin, J., Vasilakos, A. V., & Imran, M. (2016). Security in software-defined networking: Threats and countermeasures. *Mobile Networks and Applications*, 21, 764-776. <https://doi.org/doi.org/10.1007/s11036-016-0676-x>
- Singh, M. P., & Bhandari, A. (2020). New-flow based DDoS attacks in SDN: Taxonomy, rationales, and research challenges. *Computer Communications*, 154, 509-527. <https://doi.org/doi.org/10.1016/j.comcom.2020.02.085>
- Singh, R., & Adhikari, R. (2020). Age-structured impact of social distancing on the COVID-19 epidemic in India. *arXiv preprint arXiv:2003.12055*.
- Srivastava, V., & Motani, M. (2005). Cross-layer design: a survey and the road ahead. *IEEE communications magazine*, 43(12), 112-119. <https://doi.org/doi.org/10.1109/MCOM.2005.1561928>
- Syed-Yusof, S. K., Numan, P. E., Yusof, K. M., Din, J. B., Marsono, M. N. B., & Onumanyi, A. (2020). Software-defined networking (SDN) and 5G network: The role of

- controller placement for scalable control plane. *In 2020 IEEE International RF and microwave conference (RFM), Kuala Lumpur, Malaysia.*
- Tan, L., Pan, Y., Wu, J., Zhou, J., Jiang, H., & Deng, Y. (2020). A new framework for DDoS attack detection and defense in SDN environment. *IEEE Access*, 8, 161908-161919. <https://doi.org/doi.org/10.1109/ACCESS.2020.3021435>
- Tang, D., Yan, Y., Zhang, S., Chen, J., & Qin, Z. (2021). Performance and features: mitigating the low-rate TCP-targeted DoS attack via SDN. *IEEE Journal on Selected Areas in Communications*, 40(1), 428-444. <https://doi.org/doi.org/10.1109/JSAC.2021.3126053>
- Tian, Y., Tran, V., & Kuerban, M. (2019). DoS attack mitigation strategies on SDN controller. *In 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA.*
- Tuan, N. N., Hung, P. H., Nghia, N. D., Tho, N. V., Phan, T. V., & Thanh, N. H. (2020). A DDoS attack mitigation scheme in ISP networks using machine learning based on SDN. *Electronics*, 9(3), 413.
- Urrea, C., & Benítez, D. (2021). Software-defined networking solutions, architecture and controllers for the industrial internet of things: A review. *Sensors*, 21(19), 6585. <https://doi.org/doi.org/10.3390/s21196585>
- Valdovinos, I. A., Pérez-Díaz, J. A., Choo, K.-K. R., & Botero, J. F. (2021). Emerging DDoS attack detection and mitigation strategies in software-defined networks: Taxonomy, challenges and future directions. *Journal of Network and Computer Applications*, 187, 103093. <https://doi.org/doi.org/10.1016/j.jnca.2021.103093>
- Van Engelen, J. E., & Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine learning*, 109(2), 373-440.

- Wang, H., Xu, L., & Gu, G. (2015). Floodguard: A dos attack prevention extension in software-defined networks. *In 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Rio de Janeiro, Brazil.*
- Wang, J., & Wang, L. (2022). SDN-Defend: a lightweight online attack detection and mitigation system for DDoS attacks in SDN. *Sensors*, 22(21), 8287. <https://doi.org/doi.org/10.3390/s22218287>
- Wang, S. (2021). *DDoS attacks in software defined networks: detection and protection* RMIT University.
- Wang, S., Balarezo, J. F., Kandeepan, S., Al-Hourani, A., Chavez, K. G., & Rubinstein, B. (2021). Machine learning in network anomaly detection: A survey. *IEEE Access*, 9, 152379-152396. <https://doi.org/doi.org/10.1109/ACCESS.2021.3126834>
- Wang, S., Chavez, K. G., & Kandeepan, S. (2017). SECO: SDN sEcurE COntroller algorithm for detecting and defending denial of service attacks. *In 2017 5th International Conference on Information and Communication Technology (ICoIC7), Melaka, Malaysia.*
- Wang, S., Gomez, K., Sithampanathan, K., Asghar, M. R., Russello, G., & Zanna, P. (2021). Mitigating ddos attacks in sdn-based iot networks leveraging secure control and data plane algorithm. *Applied Sciences*, 11(3), 929.
- Wang, T., & Chen, H. (2017). SGuard: A lightweight SDN safe-guard architecture for DoS attacks. *China Communications*, 14(6), 113-125. <https://doi.org/doi.org/10.1109/CC.2017.7961368>
- Xie, F., Li, L., Zhao, S., Wang, J., Paterson, K. B., White, S. J., & Warrington, K. L. (2019). Aging and Pattern Complexity Effects on the Visual Span: Evidence from Chinese Character Recognition. *Vision*, 3(1), 11.

- Xing, T., Xiong, Z., Huang, D., & Medhi, D. (2014). SDNIPS: Enabling software-defined networking based intrusion prevention system in clouds. 10th international conference on network and service management (CNSM) and workshop, Rio de Janeiro, Brazil.
- Xu, T., Gao, D., Dong, P., Zhang, H., Foh, C. H., & Chao, H.-C. (2017). Defending against new-flow attack in sdn-based internet of things. *IEEE Access*, 5, 3431-3443. <https://doi.org/doi.org/10.1109/ACCESS.2017.2666270>
- Yan, Q., Gong, Q., & Yu, F. R. (2017). Effective software-defined networking controller scheduling method to mitigate DDoS attacks. *Electronics Letters*, 53(7), 469-471. <https://doi.org/doi.org/10.1049/el.2016.2234>
- Yan, Q., Yu, F. R., Gong, Q., & Li, J. (2015). Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges. *IEEE Communications Surveys & Tutorials*, 18(1), 602-622. <https://doi.org/doi.org/10.1109/COMST.2015.2487361>
- Yanbing, L., Xingyu, L., Yi, J., & Yunpeng, X. (2016). SDSA: A framework of a software-defined security architecture. *China Communications*, 13(2), 178-188.
- Yang, S., Cui, L., Chen, Z., & Xiao, W. (2020). An efficient approach to robust SDN controller placement for security. *IEEE Transactions on Network and Service Management*, 17(3), 1669-1682. <https://doi.org/doi.org/10.1109/TNSM.2020.2994837>
- Ye, J., Cheng, X., Zhu, J., Feng, L., & Song, L. (2018). A DDoS attack detection method based on SVM in software defined network. *Security and Communication Networks*, 2018(1), 9804061.

- You, X., Feng, Y., & Sakurai, K. (2017). Packet in message based DDoS attack detection in SDN network using OpenFlow. 2017 Fifth International Symposium on Computing and Networking (CANDAR), Aomori, Japan.
- Yu, L., Wang, L., Shao, Y., Guo, L., & Cui, B. (2018, 15-17 january). GLM+: An efficient system for generalized linear models. *IEEE International Conference on Big Data and Smart Computing (BigComp), Shanghai, China.*
- Yu, L., Wang, Q., Barrineau, G., Oakley, J., Brooks, R. R., & Wang, K.-C. (2017). TARN: A SDN-based traffic analysis resistant network architecture. *In 2017 12th international conference on malicious and unwanted software (MALWARE), Fajardo, PR, USA.*
- Zhang, P., Wang, H., Hu, C., & Lin, C. (2016). On denial of service attacks in software defined networks. *IEEE Network*, 30(6), 28-33.
- Zhang, Q. Y., Wang, X. W., Huang, M., Li, K. Q., & Das, S. K. (2018). Software defined networking meets information centric networking: A survey. *IEEE Access*, 6, 39547-39563. <https://doi.org/doi.org/10.1109/ACCESS.2018.2855135>
- Zhang, X.-Y., Yin, F., Zhang, Y.-M., Liu, C.-L., & Bengio, Y. (2018). Drawing and recognizing chinese characters with recurrent neural network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4), 849-862. <https://doi.org/10.1109/TPAMI.2017.2695539>
- Zhao, Y., Li, Y., Zhang, X., Geng, G., Zhang, W., & Sun, Y. (2019). A survey of networking applications applying the software defined networking concept based on machine learning. *IEEE Access*, 7, 95397-95417. <https://doi.org/doi.org/10.1109/ACCESS.2019.2928564>