Integrated framework for controller placement and attack-aware strategy in SDN-enabled wide area network

RB Sebopelo

orcid.org 0000-0002-1291-1173

Thesis accepted in fulfilment of the requirements for the degree Doctor of Philosophy in Computer and Information Sciences with Computer Science and Information Systems at the North-West University

Promoter: Prof B Isong

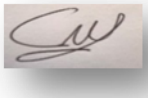
Co-promoter: Prof N Gasela

Graduation: November 2024

DECLARATION

I, Rodney Sebopelo, declare the ownership of this project report titled "*Integrated framework for controller placement problem and attack-aware strategy in SDN-enabled wide area networks"* carried out at the North-West University, Mahikeng Campus and the work has not been submitted/published before for any award of a degree to any other institution or tertiary education. The material used for this research as the source of information is acknowledged in the text.

Signature: Date: 22nd August 2024



Rodney Buang Sebopelo

APPROVAL

Promoter: **Prof B Isong**
Department of Computer Science
North-West university
Mafikeng Campus
South Africa

Signature:  Date: 25th August 2024

Co-supervisor: **Prof N Gasela**
Department of Computer Science
North-West University
Mafikeng Campus
South Africa

ACKNOWLEDGEMENT

I am very grateful to my supervisors, Professor Bassey Isong and Naison Gasela, for allowing me to go through this research process and providing the necessary support. I am very thankful for all the wisdom and knowledge you gave me; without your extensive experience, none of this would have been possible. I am thankful for your constructive criticism, support, and advice—all the new things you taught me about the research. I am thankful for the extensive patience you both exercised on me under your supervision to finish this degree; without your assistance, none of this would have been possible.

I also express my sincere thanks to the Department of Computer Science, North-West University, Mafikeng Campus, for their support during the process of this course.

Finally, I would also like to thank myself for my determination, persistence, and willingness to stand up to the challenges since the beginning of the research project. I am very thankful.

ABSTRACT

Software Defined Networks (SDNs) are a promising networking paradigm for managing and controlling large-scale or complex networks, such as wide area networks (WAN). However, SDN also presents new challenges including controller placement problems (CPP) and the vulnerability to cyberattacks. CPP finds the optimal locations and numbers of controllers in the network, considering various performance and reliability systems of measurement. Several controller placement solutions and attack-aware strategies have been proposed to achieve network performance and protect the SDN plans from malicious attacks respectively. Nevertheless, these solutions are disjointed and there is no robust model that integrates them to effectively identify anomaly-based attacks and make the routing and detection process more intelligent and autonomous for traffic analysis.

This study proposed and implemented an integrated framework for addressing both CPP objectives and attack-aware strategy in SDN-based WAN. The study carried out a comprehensive literature review to identify different approaches and empirically analyse them to select the optimum models and apply the design science research methodology to realize the framework. The framework consists of two main components: a machine learning (ML) based clustering model, enhanced DBSCAN model for the CPP, which minimizes both worst-case and average latency and maximizes the resilience of the control paths; and an attack-aware strategy for the SDN control plane, which leverages supervised ML-based approach, the k-nearest neighbour (kNN) to design an intrusion detection system (IDS). The integrated framework is evaluated via extensive simulations and experiments using the LAN dataset and various performance evaluation metrics including network latency, reliability, detection accuracy and precision.

The results obtained show that the integrated framework can improve the performance and security of SDN-enabled WANs. The enhanced DBSCAN model achieved the optimum number and location of controllers in a given WAN while the kNN-based IDS achieved a high detection accuracy of 100% compared to other models such as Random Forest and Naïve Bayes in the detection of network anomalous. About 8 anomaly attacks were identified in the network while considering the 1.096 threshold for traffic monitoring and identification based on the 95% percentile. Based on these results, it shows that a practical solution to CPP and other related problems of the SDN network should be addressed collectively rather than in isolation since it may depend heavily on solutions to other problems such as security, load balancing, resource requests or scheduling problems. Moreover, a hybrid approach of both supervised and unsupervised ML techniques is effective in developing a robust framework that effectively integrates CPP with IDS to enhance network performance and quality of service while simultaneously defending the network against all forms of malicious network attacks.

Keywords*: SDN, CPP, IDS, Security, Reliability, Integrated framework*

Table of Contents

DECLARATION i
ACKNOWLEDGEMENTS ii
ABSTRACT iii
Table of Contents iv
List of Figures ix
List of Tables xi
List of Acronyms xiii
Chapter 1: Introduction and Background 1
1.1 Chapter Outline 1
1.2 Introduction 1
1.3 Problem Statement 2
1.4 Research Motivation 4
1.5 Research Aim 4
1.6 Research Questions 4
1.7 Research Objectives 5
1.8 Research Methods of Investigation 5
1.9 Research Contribution to Knowledge 5
1.10 Research Outputs 6
1.11 Research Organisation 6
1.12 Chapter Summary 7
Chapter 2 : Literature Review: *SDN and controller placement problem* 8
2.1 Chapter Outline 8
2.2 SDN Overview 8
2.2.1 SDN Characteristics 9
2.2.2 Advantages of SDN 10
2.3 SDN Architecture 11
2.4 SDN Controller 14
2.4.1 Single Controller 14
2.4.2 Distributed Control Plane 16
2.4.3 SDN Controller Types 17
2.5 Controller Placement Problem 21
2.5.1 CPP Overview 21
2.5.2 CPP Influencing Factors 22
2.6 Controller Placement Problem Classification 24

2.7 Chapter Summary 34
Chapter 3: Literature Review: *Analysis of CPP clustering techniques* 35
3.1 Chapter Outline 35
3.2 Overview of Machine Learning 35
3.3 Clustering Techniques 36
3.3.1 Clustering Algorithms 39
3.3.1.1. Traditional Clustering Algorithms 39
3.3.1.2. Enhanced Clustering Algorithms 41
3.4 Performance Evaluation 45
3.4.1 Simulation Objectives 45
3.4.2 Topology and Tool 46
3.4.3 Model Parameters 47
3.4.4 Data Collection and Characteristics 47
3.4.5 Performance Metrics 48
3.4.6 Methodology 48
3.5 Results and Analysis 49
3.5.1 Network Size 49
3.5.2 Optimal Placement 50
3.5.3 Results 50
3.6 Discussions 58
3.7 Chapter Summary 61
Chapter 4: Literature Review: *Intrusion Detection* 62
4.1. Chapter Outline 62
4.2. Introduction 62
4.3. Intrusion Detection Systems 63
4.3.1 IDS Approaches 63
4.3.2 IDS Classification: Passive and Active Response 67
4.3.3 IDS Architecture 68
4.3.4 IDS Decision Making 71
4.4 Network Attacks 72
4.4.1 Denial of Service Attacks 72
4.4.2 Penetration Attacks 73
4.4.3 Scanning Attack 74
4.5 IDS and the SDN 75
4.6 Performance Evaluation 81
4.6.1 ML Objective 81

4.6.2 System Properties and Tools 82
4.6.3 Data Collection and Description 82
4.6.4 Machine Learning Models 83
4.6.5 Evaluation Metrics 85
4.6.6. Methodology 86
4.7 Results and Analysis 86
4.8 Discussion 87
4.9 Chapter Summary 89
Chapter 5: Research Methodology and Design 90
5.1 Chapter Outline 90
5.2 Chapter Introduction 90
5.3 Research Paradigms 92
5.3.1 Positivism 93
5.3.2 Pragmatism 93
5.3.3 Interpretivism 93
5.3.4 Socio-critic 94
5.3.5 Design Science Research Methodology 95
5.4 Qualitative vs. Quantitative 96
5.4.1 Qualitative 96
5.4.2 Quantitative 97
5.5 Research Design and Methods 98
5.5.1 Research Design 98
5.5.2 Research Methods 100
5.6 Data Collections and Analysis 100
5.6.1 Data Collection and Description 101
5.6.1.1 Qualitative and Quantitative Data 101
5.6.2 Data Analysis 101
5.7 Research Tools and Technologies 102
5.8 Research Evaluation, Validity and Reliability 103
5.8.1 Evaluation 103
5.8.2 Validity Threats 103
5.8.3 Reliability 104
5.9 Administration and Ethical Consideration 104
5.10 Chapter Summary 106
Chapter 6: Integrated Controllers Placement-IDS Framework 107
6.1. Chapter Outline 107

6.2 Chapter Introduction 107
6.3 Proposed Controllers Placement 107
6.3.1 Overview 107
6.3.2 Problem Formulation 108
6.3.3 Objective Parameters 108
6.3.4 Optimum Placement Algorithm 110
6.4. kNN Anomaly Flow Detection Technique 110
6.4.1 Intrusion detection architecture design 110
6.4.1.1 Traffic Collection Module 113
6.4.1.2 Anomaly Detection Module 114
6.4.1.3 Traffic Monitoring and Classification 114
6.4.1.4 Anomaly Identification 115
6.4.1.5 Mitigation Module 116
6.4.1.6 Report Module 117
6.4.2 The Intrusion Detection Algorithm 117
6.4.2.1 The kNN - IDS Technique 117
6.4.2.2 The kNN-IDS Pseudocode 118
6.4.3 Enforced Policy 118
6.4.3.1 The Policy Algorithm 118
6.5 Integrated Framework 119
6.6 Chapter Summary 123
Chapter 7: Implementation and Results 124
7.1 Chapter Outline 124
7.2 Chapter Introduction 124
7.3 System Properties 125
7.3.1 System Properties 125
7.3.2 Simulations Set-up 125
7.4 Implementation 127
7.4.1 Proposed Controllers' Placement 127
7.4.2 Proposed Intrusion Detection 129
7.5 Results and Analysis 132
7.5.1 Proposed Controller Placement 132
7.5.2 Detecting Network Attacks 139
7.5.3 Attack Mitigation 142
7.6 CPP Comparison and Evaluation 144
7.6.1 Proposed ODL Placement 144

7.6.2 Proposed Ryu's Placement 146
7.6.3 Proposed Floodlight Placement 149
7.7. SDN-based CPP Comparison 151
7.8 IDS Comparison and Evaluation 152
7.9 SDN-based IDS Comparison 154
7.10 Chapter Summary 156
Chapter 8: Summary, Conclusion and Recommendations 157
8.1 Chapter Outline 157
8.2 Summary 157
8.3 Conclusion 158
8.4 Recommendations 159
8.5 Future Works 160
8.6 Appendix A: SDN environment simulated flow statistics 161
8.7 References 167

List of Figures

Figure 1: SDN architecture [66] 12
Figure 2: K means algorithm 39
Figure 3. AP algorithm 40
Figure 4. DBSCAN algorithm 40
Figure 5: Mean-shift algorithm 41
Figure 6: DBSCAN enhancement algorithm 42
Figure 7: AF enhancement algorithm 43
Figure 8: K means enhancement algorithm 44
Figure 9: Mean shift enhanced clustering algorithm 44
Figure 10: Anomaly-based architecture 64
Figure 11: Signature-based architecture 65
Figure 12: Stateful protocol analysis-based methodology architecture 66
Figure 13: Hybrid based architecture 70
Figure 14: Network attack categories [478] 74
Figure 15: NSL-KDD data set classification model 86
Figure 16: ROC curves of the selected models 88
Figure 17: The flow of the proposed research work 92
Figure 18: Research design workflow 99
Figure 19: Optimum placement algorithm 110
Figure 20. Designed intrusion detection architecture 112
Figure 21: Proposed intrusion detection flow process 113
Figure 22: Anomaly detection and mitigation flow process 116
Figure 23: kNN anomaly detector algorithm 118
Figure 24: Policy-based IDS algorithm 119
Figure 25: Integrated controllers placement - IDS framework 120
Figure 26: CPP – IDS process 122
Figure 27: SDN environment 126
Figure 28: No. of required controller 133
Figure 29: Switch assignment 135
Figure 30: Processing latency 136
Figure 31: Controller load balance 137
Figure 32: Reliability results 139
Figure 33: Traffic monitoring and classification results 140

Figure 34: Traffic monitoring scatter plot.....141
Figure 35: Enforced action policy results 143
Figure 36: IDs performance models.....154

List of Tables

Table 1: Comparison of SDN controllers [75, 158] 20
Table 2: CPP clustering strategy 25
Table 3: CPP game theory 26
Table 4: CPP ml, dl & others 27
Table 5: Homogeneity & completeness clustering techniques 42
Table 6: Adopted clustering enhancement algorithms 46
Table 7: Summarised performance metrics 48
Table 8 : Summarized results of affinity propagation enhancement algorithm 51
Table 9: Summarised results of the DBSCAN enhancement algorithm 52
Table 10: Summarised results of k-means enhancement algorithm 54
Table 11: Summarized results of mean shift enhancement algorithm 55
Table 12: Summarized results of affinity propagation algorithm 56
Table 13: Summarised results of the DBSCAN algorithm 57
Table 14: Summarized results of mean shift algorithm 58
Table 15: Summarised discussion results of the selected algorithms 59
Table 16: IDPS methods pros and cons 67
Table 17: Summarized IDS challenges and proposed solutions 80
Table 18: Attribute value type and attacks 83
Table 19: Normal and attack data set [543, 544] 83
Table 20: A summarized description of the chosen models 84
Table 21: Summarized evaluation metrics 85
Table 22: Selected algorithms performance analysis 87
Table 23 Summarised ROC results 87
Table 24:Design science research steps 96
Table 25: Extracted features 114
Table 26: Normal and anomaly effects on the traffic features 114
Table 27: List of implemented policies 117
Table 28: System properties 125
Table 29: Simulation parameters 126
Table 30: Deployed number of the controller 127
Table 31: Summarised controller-to-switch results 128
Table 32: Processing latency 128
Table 33: Load balancing in the network 129

Table 34: Summarised reliability results 129
Table 35: Traffic collection attribute details 130
Table 36: Injected DDoS Anomaly attack details 131
Table 37: DDoS anomaly attack mitigation policy 132
Table 38: Deployed no. of controller 133
Table 39: Summarised switch assignment results 134
Table 40: Network processing latency 135
Table 41: Summarised load balance results 137
Table 42: Summarised reliability results 138
Table 43: Traffic monitoring results 140
Table 44: Summarized DDoS anomaly attack details 141
Table 45: Enforced policy matching criteria 142
Table 46: Enforced policy results 143
Table 47: Deployed No. of controller 144
Table 48: Summarised switch assignment results 144
Table 49: Network processing latency 145
Table 50: Summarised load balance results 145
Table 51: Summarised reliability results 146
Table 52: Deployed no. of controller 147
Table 53: Summarised switch assignment results 147
Table 54: Network processing latency 147
Table 55: Summarised load balance results 148
Table 56: Summarised reliability results 148
Table 57: Deployed No. of controller 149
Table 58: Summarised switch assignment results 149
Table 59: Network processing latency 150
Table 60: Summarised load balance results 150
Table 61: Summarised reliability results 151
Table 62: Comparison with the existing CPP scheme 152
Table 63: NB anomaly detection results 153
Table 64: RF anomaly detection results 153
Table 65: Comparison with existing SDN-based IDS 155

List of Acronyms

API	Application programming interface
AI	Artificial intelligence
ARI	Adjusted rand index
AMI	Adjusted mutual information
AP	Affinity propagation
C2C	Controller to controller
CSV	Comma separate value
C2S	Controller to switch
CCPP	Capacitated controller placement problem
CA	Classification accuracy
CPP	Controller placement problem
DT	Decision tree
DDoS	Distributed denial of service
DSR	Design science research
DL	Deep learning
GBT	Gradient boosted tree
HIDS	Host-based intrusion detection system
IP	Internet protocol
IoT	Internet of things
IDS	Intrusion detection system
KNN	Knearest neighbour
LR	Logistic regression
ML	Machine learning
MCP	Multiple control path
NIDS	Network intrusion detection system
NB	Naïve Bayes
NB	Decision stump
OF	Open flow
ONOS	Open network operating system
OvS	OpenFlow switch
PC	Personal computer
QoS	Quality of service

QoE	Quality of experience
ROC	Receiver operating character
RF	Random forest
RAM	Random access memory
SDNs	Software-defined networks
SDWAN	Software-defined wide area network
TC	Traffic classification
TCP	Transmission control protocol
VM	Virtual machine
WAN	Wide area network

Chapter 1

Introduction and Background

1.1 Chapter Outline

In this chapter, we present the overview and background of this research. We presented the problem statement for this research. We further outlined the research aim, research questions, and objectives intended to answer the research questions of this study. Finally, we presented the research contribution of this study.

1.2 Introduction

Software-defined networking (SDN) is an emerging network technology with a decoupling architecture that separates the control plane from the forwarding plane. The design goal of the network is to address the challenges and network complexities faced by the internet protocol (IP) of traditional networks [1]. Operations that are performed at the control plane include routing, traffic engineering (TE), rules and policy management, and others. Accordingly, the forwarding or data plane deals with the forwarding of data packets in the network, etc. The tasks of network control and management are delegated to the controller, which is known as the “brain of the network." The controller, together with the forwarding elements such as switches, routers, sensors, etc., interacts in the network using important protocols such as OpenFlow (OF), which is one of the commonly deployed standard interfaces (southbound) used in the SDN. Currently, SDN has widespread applications in network control and management, such as in wireless sensor networks (WSN), cloud computing, VANETS [2], and so on.

Despite the benefits and challenges associated with the introduction of SDN, the technology is confronted with several challenges, such as security, reliability, scalability, and so on [3]. Security-wise, traffic classification (TC) and intrusion detection (ID) are of great concern, while reliability and scalability are challenged by controller placement in a large-scale network such as the wide-area network (WAN) [4], where controllers play a critical role. TC is crucial to optimizing the internet, the experience of the users, and intrusion detection. Classifying bandwidth and Internet service providers can be the best way to manage resources by prioritizing the flow of packets [5]. Two important approaches used for TC are port numbers (PNs) and deep packet inspection (DPI) in the network [6].

However, several studies have shown that signature-based techniques are not efficient for the SDN due to its architecture; thus, attention has shifted to flow-based anomaly detection. Moreover, the application of ML in the SDN has gained considerable attention in recent years. It has been employed to detect patterns within the applications that are based on the selected feature sets, classify the encrypted traffic at a lower computational cost, perform routing optimization, QoS and QoE prediction, manage resources, secure them, use CPP [7, 8], etc.

Furthermore, on a large scale in SDN environments such as WAN, multi-controller deployment is essential to ensuring QoS, effective performance, and other important requirements. Consequently, the number of controllers to be deployed, their allocation, and their location have to be carefully considered in the network [9]. To achieve this, the controller placement problem (CPP) strategy has emerged, and it deals with how to place the controller, how to allocate the switches associated with the controllers in the network, the number of controllers, the optimal location, and the relationship between the switches and controllers as well as between controllers [10]. Due to the important tasks of the controllers, such as their interaction with relevant switches and other controllers, CPP is, therefore, important, as poor location, allocation, and capability could negatively impact the performance and QoS of the network, as well as defending against DDoS attacks and others. To achieve CPP in the SDN, several techniques have been proposed and developed [11–13].

To achieve CPP in the network, ML techniques can be utilized to achieve the controller-switch association strategy of the SDN. It can also be used to address other challenges such as intrusion detection and communication between the controller and data planes, control plane delay between switches and controllers, controller placement vulnerabilities, attacks on the controllers that originated from untrusted applications, and to solve the challenges of the controller capacity matching subproblem [14, 15]. However, with several approaches that have been proposed and developed using ML to achieve CPP and intrusion detection/TC, respectively, or both, the SDN is still confronted with several challenges, such as security, controller load balancing, latency, reliability, etc., during the exchange of control messages.

1.3 Problem Statement

The innovation brought about by SDN's architecture makes it the ideal network technology to overcome the complexities and inherent challenges faced by traditional networks. However, SDN also faces several challenges that affect its security, reliability, scalability, QoS, etc. Some of these challenges include an efficient CPP scheme due to the scalability issues in large-scale SDN environments or WAN and ID due to inefficient TC. In SDN-based WAN, a single physical or logically centralized controller poses a single point of failure and does not support network scale-up, and other important performance requirements. SDN with multi-controller architecture

addresses these challenges by deploying multiple controllers in the network. However, the tasks of identifying the best location and an optimum number of controllers while still meeting important network performance requirements such as minimizing propagation and processing latencies based on the dynamic fluctuation of traffic workload, cost minimization, maximizing reliability and load balancing, etc., [16-18] bring about CPP. Without effective and efficient CPP, good network performance and QoS will not be achieved [2, 6, 19]. Consequently, controllers can easily experience a high workload, response time increases above normal value and when the load exceeds a certain threshold, the overall reliability and security could be compromised.

Moreover, though the nature of SDN architecture is advantageous to addressing the rigid and complex challenges in traditional networks, it also brings about security issues since it was not considered in the initial design [20]. In particular, without a robust built-in security model that effectively and efficiently identifies spurious flows, the whole network and other activities such as CPP, traffic prediction, etc. could suffer from a lack of trusted connection between layers, and controllers load balancing which makes it unable to defend against attacks such as distributed denial of services (DDoS) attacks, etc. during the exchange of control messages [21]. This could bring about economic loss, privacy issues, insufficient detection precision, and the speed to defend against DDoS and other attacks. Consequently, poor network performance, modification, and fabricating of bogus routing information, can cause the unavailability of valid services, etc., [22]. Moreover, attackers can create a tunnel and bypass the network [23] by short-circuiting the normal flow of routing messages.

Currently, several different CPP approaches exist but an efficient CPP algorithm is yet to be achieved [21, 22]. Also, several different security measures have been proposed and developed for the SDN such as the ID systems (IDS), however, a robust and efficient approach is yet to be achieved as well [2, 24]. This means to achieve a practical solution to CPP, other related problems of the network must be tackled collectively under a large integrated framework.

Therefore, this research is positioned to propose a robust and efficient solution to the SDN scalability issue through CPP that is both efficient and attack-aware. The objective is to achieve the best trade-off between network performance and security in the SDN-based WAN. Consequently, the approach involving ML techniques and the overall framework design implementation will be used to address the challenges of security (secure, efficiently monitor and manage), load balancing, latency, and reliability of the controllers during the exchange of control messages in the integrated framework thereby, improving the performance of the SDN in WAN setting.

1.4 Research Motivation

The importance of this research was to address the challenges faced by the SDN network since it faces challenges that include security, reliability, and load balancing because of the CPP and ID. It was relevant to design an effective and efficient built-in security model that could identify the flow-based anomaly attack and provide a trusted connection between the layers, making the controller defend against attacks such as DDoS. The solution could provide sufficient detection speed and defend against the DDoS attack during the placement of the controller in the network. Consequently, modifications, poor network performance, the fabrication of very large amounts of information, and the unavailability of valid services can be avoided in advance before any damage to the resources occurs.

1.5 Research Aim

This research aimed to provide a robust solution to software-defined network scalability through the controller placement problem that was both efficient and attack-aware through intrusion detection.

1.6 Research Questions

To address the above-stated research problem, the following research questions (RQs) were addressed:

RQ1: How do critical challenges such as security, scalability and reliability impact the SDN in WAN and what solutions exist?

To address this RQ1, challenges such as security issues and solutions in terms of IDS as well as scalability issues in terms of CPP strategies were reviewed and discussed.

RQ2: How can an effective practical solution to SDN CPP be addressed to achieve the best trade-off between network performance and security in WAN?

To answer RQ2, the following sub-questions were addressed:

RQ2.1 *how do the type, the number, and the placement of SDN controllers impact the performance of the controller in SDN-enabled WANs in terms of latency, load balancing and reliability?*

RQ2.2: *How efficiently can the SDN be secured via defending against network attacks?*

RQ2.3: *What impact does security have on the performance of the SDN-enabled WAN controller placement?*

1.7 Research Objectives

To answer the RQs and meet the aim of this research, the following research objectives (ROs) were performed:

RO1: Conducting a comprehensive literature review on SDN and related studies in terms of challenges such as scalability and security issues as well as existing solutions.

RO2: Designing and implementing a robust integrated framework that incorporates both security and CPP for SDN-enabled WAN.

RO2.1 proposed an efficient CPP scheme using an appropriate optimization algorithm for a practical solution to CPP to achieve an optimum number of controllers and their placement that meets performance requirements such as load balancing, latency, and reliability.

RO2.2 Designing an efficient IDS as a security model using ML to protect the SDN against malicious activities or attacks in the network.

RO2.3 We defined the metrics and formulas including the number of required controllers, load balancing, reliability, security, and classification accuracy. We then measured and analysed the effectiveness of the employed formulas and metrics on the implemented models as well as the impact of the security on the SDN-enabled WAN performance. We evaluated and compared the outcomes of our integrated SDN CPP – IDS framework with the previously proposed framework.

1.8 Research Methods of Investigation

In this research, we employed the design science research (DSR) methodology for the integrated SDN CPP-IDS presented in Chapter 5, Section 5.3.5. The employed methodology was used to provide solutions to the problem statement and the research gap about the nature of the SDN. Moreover, the design science research steps were used to seek and understand the process of solving the problem of the SDN environment. Furthermore, we employed the systematic research design and methods presented in Chapter 5, Section 5.5, to resolve issues, challenges, and problem statements on the SDN CPP-IDS. Both the DSR, design research, and method were collectively employed to address the problem statement about the nature of the SDN environment, controller placement, and intrusion detection presented in detail in Chapter 5.

1.9 Research Contribution to Knowledge

This research provided the following contributions to the SDN environment:

i. New knowledge and methodological advancement: It provided an integrated-based CPP-IDS that is both effective and attack-aware, thereby improving the performance of the SDN in

WAN settings. Adhere to the failure resilience in the network as well as provide security means for the number of placed controllers in the SDN environment.

ii. Conceptual framework and practical application: It provides the designed framework that provides the applied and implemented action policies that include dropping, exclusion, and pushing that serve as the countermeasures to enforce the event caused by the DDoS attacks in the network. It prioritized the DDoS event, and the countermeasures required to be applied to the network.

iii. Critical analysis and integration: Based on the research findings and contributions, the research provided a critical evaluation of the existing theory study based on an SDN environment, the methodology, and the paradigms aligned with the research study. The research identified the weaknesses based on the literature studies, inconsistencies, and gaps in the current studies or knowledge and proposed alternative perspectives on the SDN environment.

1.10 Research Outputs

This section presents the list of research outputs or publications achieved in this study

[29] R. Sebopelo, B. Isong, N. Gasela and A. M. Abu-Mahfouz, "A Review of Intrusion Detection Techniques in the SDN Environment," 2021 3rd International Multidisciplinary Information Technology and Engineering Conference (IMITEC), Windhoek, Namibia, 2021, pp. 1-9, doi: 10.1109/IMITEC52926.2021.9714581.

[30]Sebopelo, Rodney, and Bassey Isong. "An Integrated Framework for Controllers Placement and Security in Software-Defined Networks Ecosystem." *Journal of Information Systems and Informatics* 6.1 (2024): 464-494. https://journal-isi.org/index.php/isi/article/download/682/339.

1.11 Research Organisation

This section presents the organization of the chapters of this research.

i. *Chapter 1*: presents the overview and background of this research. It presents the problem statement, research aims, and objectives required to answer the research questions of this study. The research motivation, methods of investigation, contribution to knowledge, and outputs of this study are presented in detail.

ii. *Chapter 2*: presents a literature review on the SDN CPP, its architecture, as well as single and distributed CPP strategies. The chapter discusses the types of controllers and factors

influencing the CPP. Furthermore, the chapter presented the classification of the controller placement problem.

iii. *Chapter 3*: presents an analysis of the clustering strategy of SDN-based CPP in the WAN. Different ML and clustering strategies were presented, as well as traditional algorithms. The CPP performance analysis in WAN is presented in detail.

iv. *Chapter 4*: presents the SDN-based intrusion detection system. Various ML algorithms were employed and evaluated for the detection of intrusions in the network. Moreover, the methodology and performance analysis of the selected algorithms are presented in detail.

v. *Chapter 5*: presents the research methodology and design adopted in this research. The various paradigms, DSR, and qualitative and quantitative strategies are presented in detail. The tools and technologies employed in the research are presented in detail. The research's validity, reliability, and evaluation were discussed in detail.

vi. *Chapter 6*: presents the integrated framework CPP–IDS in the SDN environment. The problem formulation, parameters, equations, and algorithms are presented in detail. Moreover, how the parameters, algorithms, etc., are collectively used to achieve the CPP-IDS in the network is presented in detail.

vii. *Chapter 7*: presents the implementation of the integrated CPP-IDS in the network. The chapter presents the system properties, simulations set-up etc. employed in the research.

viii. *Chapter 8*: presents the summary of this research, and a brief discussion of the chapters is presented in detail. The research scope and limitations are presented in detail. Furthermore, the recommendations and future directions aligned with this research are presented in detail.

1.12 Chapter Summary

This chapter presents a brief presentation and the background of the study. It outlines the problem statement, research questions, and objectives of this study. Furthermore, the aim of this research was presented in detail. In addition, research motivation and contribution to knowledge were discussed in detail. The scope and limitations of this research were presented in detail. Additionally, it presents the aim of the research in detail and discusses the research motivation and contribution to knowledge.

Chapter 2

Literature Review: *SDN and Controller Placement Problem*

2.1 Chapter Outline

This chapter presents and discusses the literature review based on the SDN controller placement problem (CPP). It begins with an overview of SDN, and its architecture and then progresses to single and distributed controller placement strategies. Moreover, it discusses the different types of controllers and the factors that influence the CPP such as reliability, and latency. Lastly, we presented a CPP classification and summarizes the chapter.

2.2 SDN Overview

SDN is one of the most important paradigms shifts that the network industry and academics have not noticed in recent years [25, 26]. The main objective of SDN has been driven by the necessity to keep pace with the requirements of the network with the emerging trends of virtualized cloud computing, mobile computing, and the applications of big data [27, 28]. The primary goals of SDN include network innovations that result in fast and simple automation and the management of a large network. Although the principles behind the SDN are not new, the separation of the control and data planes was introduced by the leading standard, namely the Internet Engineering Task Force Forwarding and Control Element Separation Working Group (IETF ForCES), in the 2000s [29, 30].

SDN as a networking paradigm consists of various kinds of components, such as routers and switches, in the middleboxes that include firewalls, address translator networks, intrusion detection system applications [31, 32], etc. Moreover, the software nature of the network allows the implementation of network protocols and undergoes years of standardized and interoperability testing. SDN consolidates the forwarding traffic layer such that a single controller [33, 34] can control multiple elements of the data plane [35]. The SDN control plane plays a major role in exercising direct control over the state of the data plane network elements through a protocol that is well-defined as an application programming interface (API), such as the widely used OpenFlow [36, 37]. The OpenFlow switches consist of multiple flow tables that handle the packet rules, containing actions to drop, forward, and so on. Based on the rules and policies installed by the controller's applications, the OpenFlow switch can behave as a router, switch, and so on [38, 39]. SDN has gained important traction in industries, and many commercial companies switched to support the OpenFlow API, such as initial vendors that included HP, NEC, and others [40].

Since then, many different platforms for controllers have emerged. However, programmers have used these various platforms to create applications that include dynamic access control, the balanced load of the server, virtualization of the network, an efficient energy network, the virtual migration of the machine, and user mobility [41, 42]. Again, there are many of the world's largest technology information companies, such as cloud providers, vendor equipment, and so on, that have joined the idea of SDN industry consortia like the Open Network Foundation (ONF) [43, 44], etc.

Although the idea and excitement about the SDN have become more palpable in the past few years, many of the ideas still underlie the SDN, which has evolved in the past twenty years or more [45, 46]. The SDN capability still visits the ideas from the early telephony networks, which results in the separation of the control plane and the data planes to achieve the simplification and management of the network and the deployment of new services [47]. The main efforts in the creation of a programmable network infrastructure related to the thread are long hours of work on supporting programmable packet processing at high speeds [48].

2.2.1 SDN Characteristics

Due to the nature and dynamics of SDN, the network aims to achieve and provide the following objectives and functions: openness, centralization, decoupling, programmability, flow-based operations, and adaptability of network and switched models.

A. *Openness:* The components in the SDN include switches and routers. They provide an immense and limited ability for users to experiment with network protocols with real traffic. However, due to the dynamic change of the SDN, there are development components such as middleboxes [49] that can interact with the controller of the control plane and forwarding elements such as switches in the data plane layer [50]. Moreover, multiple platforms of controllers have been introduced as open source, namely: Open Daylight, Floodlight, Beacon [51], and so on.
B. *Centralization:* In SDN, different switches on the network are controlled and co-allocated by one logical place, which is the controller [52]. However, with a design mode, this is considered as splitting what from the how-to approach. Moreover, the architecture is capable of handling dynamic network challenges and situations [53], such as traffic that may dynamically change and may require the switch to dynamically join and leave the virtual network [54].
C. *Decoupling:* In SDN, the overall performance, reusability, and maintainability of the network can be improved by splitting the data plane from the control plane [55]. The rules and policies are handled by the controller. On the contrary, the information level of the network, such as firewall rules or flow, must be simple and closer to the attributes of the network [56, 57].

D. *Programmable:* The controller can be accessed and programmed by user-level applications or middleboxes. The programmability of the SDN controller is considered a major characteristic of open-source controller modules by the developers [58, 59]. The programmability can be extended further by writing down the applications and functionalities of the controller. It allows network administrators to write policies that can be monitored by OpenFlow within networks [60, 61].

E. *Flow-Based Management:* SDN shifts from IP-based to flow-based management and control. The flow level approach is possible in technical traditional networks such that routing can be taken based on the decisions of the IP addresses. Moreover, the records and policies in the forwarding devices and firewalls can lead to the impact of many applications on network traffic [62, 63].

F. *Dynamic:* The nature and dynamic changes of SDN bring the advantage of the software in comparison to the hardware due to the frequent changes and accommodation of a more dynamic and flexible network. The advantage of the software is that it can be programmed to respond to network activities about decisions on dynamic changes [64].

2.2.2 Advantages of SDN

SDN separates the control plane from the forwarding devices, such as routers and switches, to create a logically centralized network system. Moreover, SDN promotes data-intensive software and so on. SDN is controlled by a centralized unit called a central node for policy and security. Hyojoon Kim *et al.* [46] presented a network management approach to address frequent changes such as network support configuration and so on. To achieve this, the authors proposed enabling operations to implement a wide range of policies in a high-level language. Moreover, to reduce the hardware costs, software principles can be created with minimal available hardware and reduce the manual need for assistance; however, the setup expenses can be leveraged to improve the user network. Dimitri *et al.* [65] implemented the flow restoration approaches to run the extensive simulation in an emulated carrier-grade network.

In SDN, cloud computing is a unified infrastructure. However, it is easier to unify services through cloud abstraction [66, 67]. The network elements can make up a large data system. Flavio Meneses *et al.* [68] presented mobility framework management, where the procedures are abstractly handled by virtualized network entities. Furthermore, in security, it is easy to manage a single management network from a central point and provide a better security strategy. However, the console application can be used to disseminate information.

Moreover, network management can be a challenge for physical network devices. Again, it is possible to monitor safety measures within the organization. Lastly, SDN deals with access to the internet compared to previously due to automatic cloud responses. Mostafa Uddin *et al.* [70] presented SDN-based multi-protocol edge switching for IoT service automation to achieve automatic application. SDN has several advantages over the traditional network, which include management of the network, view of the global network, maintenance cost, error/update handling required time, utilization of the controller, controller's integrity, authenticity, and consistency.

2.3 SDN Architecture

In SDN, distributed control and transport network protocols [71] allow the information in digital packets to travel the world [72]. However, traditional IP networks are too complex and difficult to manage. The network environment should be able to endure the dynamic faults and the adaptation to load changes. The automatic reconfiguration and the response models are virtual and nonexistent in the current IP networks [73] and enforcing the required rules and policies in the dynamic environment is highly challenging. Again, the transition from IPV4 to IPV6 started decades ago and is still incomplete, while IPV4 represents a protocol update.

SDN separates the control plane from the forwarding plane. The goal of the network is to address the complex challenges faced by the traditional Internet protocol (IP). However, due to the separation, switches simply become the forwarding devices and the control logic is logically implemented by the controller by enforcing the rules and policies [74], or in terms of rules and policies over the so-called application programming interface (API) and OpenFlow [39]. OpenFlow switches can be made up of one or more tables of packets that handle the rules (flow tables). Moreover, the separation of the layers is key to the flexibility of breaking the network control into manageable pieces to create new abstractions [75, 76], network management simplification, innovation, and the evolution of the network.

The SDN architecture is shown in Figure 1 and is divided into 3 parts: the data plane, the control plane, and the application plane, each part is explained in detail below.

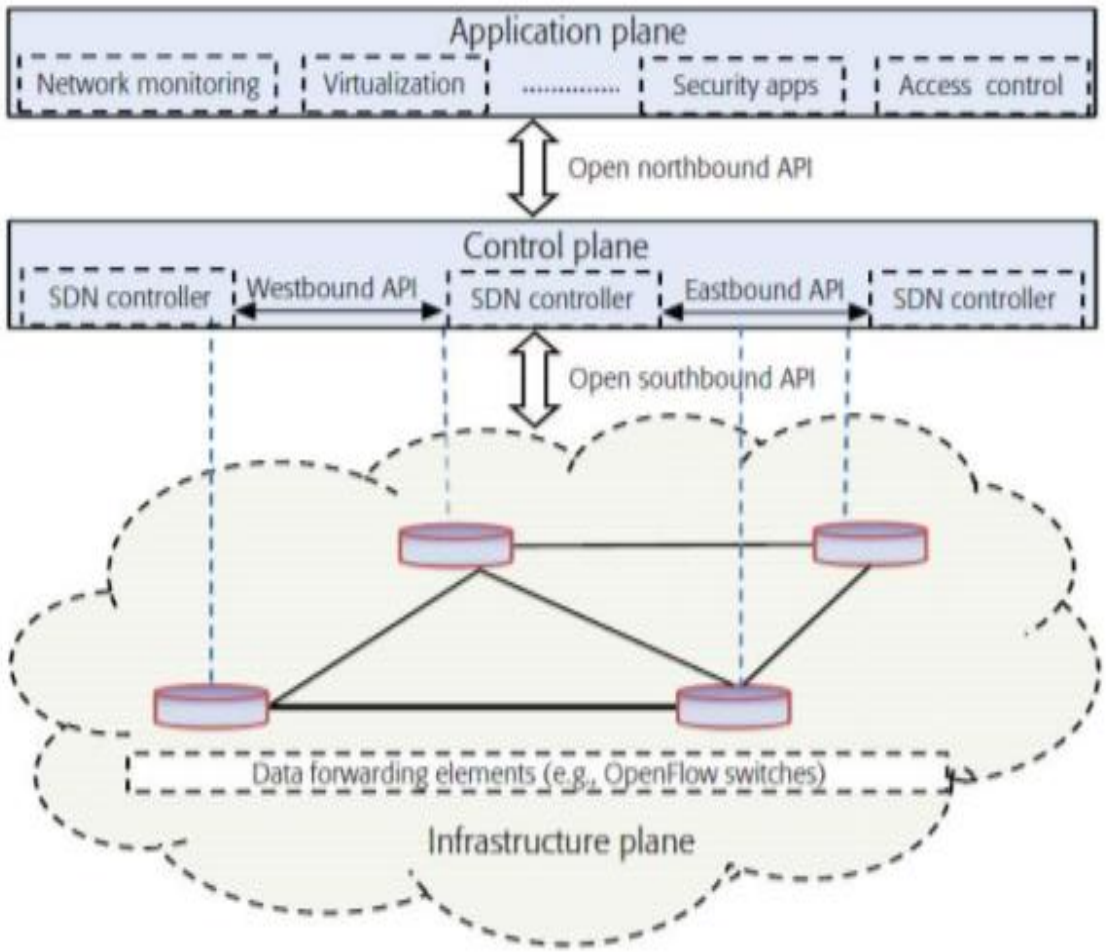


Figure 1: SDN architecture [66].

A. Data plane

The infrastructure layer consists of the network devices connected, such as routers, middleboxes, switches, and so on. Each device contains the rules, actions, and other statistics [77]. However, these rules are populated by the controller in the control layer network in the form of a proactive or reactive approach. The packets from the host are received by the forwarding devices and are forwarded according to the rules and policies specified by the controller. However, when the packets do not match the specified rules, the rule is sent to the controller for decision [78, 79]. Shaghaghi *et al.* [80] presented and identified the challenges in securing the data plane. The authors formalize the problem space by identifying the potential attack scenario while highlighting the possible vulnerabilities and establishing the set of requirements and challenges to protect the SDN network.

B. Control plane

This layer is found at the centre of SDN architecture. The layer consists of the main component called the controller, which is referred to as the intelligence of the SDN network. The tasks of the

controller provide a programmatic interface that helps the network administrators to include other new features without vendor dependence and allows the automation of configuration, management, and so on [81–83]. Panda *et al.* [84] focused on handling unplanned updates to the networks. Moreover, contrary to the prevailing wisdom, the distributed SDN control plane needs to be slightly more complicated than the single image controllers.

C. Application plane

The layer of applications above the control plane is called the application layer. The SDN applications use the controller's API to configure the traffic flows for the packets to be routed to the best possible path in the network, such as load balance of traffic, link failure, redirection of traffic, security, and so on. This level of application is event-driven, which means the response can range from the simple default of the action to a complex dynamic reaction [85–87]. Furthermore, SDN interfaces provide communication between the data plane and the control plane called the southbound API, the control plane and application plane called the northbound API, and the east-west interface protocol, and each part is explained below.

a) *Southbound API:* The communication between the data plane and the control plane takes place through an API such as OpenFlow. Moreover, OpenFlow is not only the southbound API; there are other well-known APIs such as ForCES, OPFlex, and presence protocol (XMPP) [88, 89], which is a communication protocol that allows the exchange of messages between clients for a centralized hybrid SDN.

b) *The Northbound API* is a low-level interface that provides access to SDN applications and network devices. However, these applications can be programmed using API-based programming and domain-specific languages. This level of API gives and supports the application developers with an abstract view of the topology of the network [90–92]. However, there is no standard northbound API, and the most popular northbound APIs are the REST API and OSGi.

c) *The east-west interface protocol* provides and manages the communication among multiple SDN controllers in the control plane. However, the communication between the interfaces is the most challenging and is considered inter-controller communication or controller-to-controller interface (CCI) [93, 94].

2.4 SDN Controller

The flow-based routing in SDN enables the network to support the rules and policies of the controller. However, the issues and limitations of a single controller still raise some issues and challenges, etc. Consequently, researchers have proposed various strategies to overcome the problems of offloading, addition, and so on of a single controller [95]. S. Ahvar *et al.* [96] proposed a model that addresses the challenges faced by the control plane, such that a logically centralized controller leads to a single point of failure in a software-defined wide area network (SDWAN). Accordingly, the authors proposed a multiple-controller architecture for the SDN. This section, therefore, discusses the related issues, limitations, and challenges faced by the SDN single controller and the proposed solutions.

2.4.1 Single controller

In a small-scale SDN, the network allows a single controller to meet the requirements of the SDN network and address issues such as latency. Consequently, some researchers have advocated for the use of physical controllers to improve performance. An example of this is the widely known NOX, which is a centralized controller for the single-threaded OpenFlow system [97].
D. Staessens *et al.* [65] discussed that NOX can encounter an abnormal state of automatic interface for the administration and system control advancement applications that reflect the change and sorting out of a product issue. The authors discussed that this controller fails to provide the required capacities to the QoS. It fails due to the provision on the transporter review supplier internet such as QoS cautious virtual network, ends to end arrange QoS appraisal and control components joint efforts for other network space. Moreover, M. Joselli *et al.* [98] discussed that NOX can empower administration applications to develop projects together over a high state of reflections of the system assets as an inverse to the dispersed low-level calculation location. Again, Parulkar *et al.* [99] discussed that NOX can control organized activities and arrange switches. In this case, the switches that assist the OpenFlow switches' reflection can also be used.

David Erickson *et al.* [100] discussed that the beacon can explore new areas of the OpenFlow design space and was identified as focusing on being friendly with the developers to provide high-ability performance, such as to commence and stop the existing and new applications at run time. Again, the authors mentioned that the controller shows surprisingly high performance and the ability to scale the linear processing cores by handling 12.8 million packets in messages per second with 12 cores using Java software. Again, Tony Manuel *et al.* [101] also discussed that the

controller can control 100 vSwitch, 20 physical switches, a provisional data centre, code packages that can begin or halt during the runtime, and other hindrances of non-dependent groups.
On the other hand, Alexander Shalimov *et al.* [102] discussed that the maestro's controller scalability is limited to 8 cores and does not run with more than 8 threads. According to the authors, the controller has the capability of distributing incoming packets using algorithms such as round-robin algorithms, etc. to show better results. However, due to an unbalanced load, the controller can crash when the messages are sent without delay. Syed *et al.* [103] also discussed that the controller can receive OpenFlow packets from the sockets via the main thread and put them in the shared raw packet queue. The controller can also use two-packet batching with the number of bytes that is not static to depend on the present workload. Moreover, the authors discussed that Maestro is the only publicly available controller that can use task batching for the worker threads to pull a batch of tasks and multiply the flow process request in a single execution.

Andreas *et al.* [104] presented McNettle as an extensible SDN control system with the capability to control event processing throughput scales, which consist of the number of system CPU cores and can support the control algorithms required for globally visible state changes occurring at flow arrival rates. The authors mentioned that the controller can handle the message once for each switch in the network. The message handlers can include a function that can be applied whenever a packet misses the message that is sent by a switch in the network.
Furthermore, the choice of the controller is based on various requirements of the user, such as the programming language and so on. However, multiple controllers focus on specific domains, such as RouteFlow [105, 106], which can be achieved and optimized for routing inter-domains, such as the POX controller, which is widely known for research and education due to its easy understanding and the prototype. Mishra *et al.* [107] stated multiple disadvantages of the controller, such as throughput and time response, that are not efficient and effective. Lynn *et al.* [108] discussed that OpenDaylight can support high modularity with efficiency based on the Java programming language and support multithreading.

As the scale of the network grows, the deployment of a single controller cannot meet the requirements of the network due to challenges such as:

i. *Control Traffic Overhead*: In the SDN network, the flow is set up on an N-switch path. To achieve the one-way cost of the network, the load is about 94+144N in bytes. As a result, the control traffic overhead destined towards a central controller will directly be proportional to the number of switches due to the increase [109, 110].

ii. *Flow Setup Latency*: In an SDN network, the flow setup response time for the controller far from the switches will experience high latency. However, the network with a larger scale diameter consisting of several clusters of nodes experiences challenges that include long flow setup response latencies, irrespective of the placement of the controller [111, 112].

iii. *Controller Capacity*: The nature of SDN networks causes the controller to handle a limited number of flow setups per second, such as the NOX, which can handle flow [111, 114]. However, the flow setup times are bounded by the power of each controller; as a result, this will cause the flow setup times to increase as the scale of the network increases.

iv. *Flow Statistical Delay*: In SDN, schedulers are one of the primary sources of use to improve static load balancing, reorder policies, and use switches that can improve the processing model that is widely distributed in Hadoop [115]. The objective of the schedulers requires time and access to flow statistics to perform optimally. Moreover, the centralized controller can frequently query the switches that increase the flow of traffic control in the SDN network [116].

v. *Availability*: In SDN, network availability can be affected by reliability and security. However, due to the centralized controller, this results in a single point of failure. It is more prone to malicious activities such as security attacks and degrades the whole network's performance [117].

2.4.2 Distributed Control Plane

To overcome the issues and challenges of the single controller as discussed above, distributed controllers should be employed in a physically distributed and logically centralized manner. However, there are still challenges, such as the synchronization of controllers. Moreover, the local controller’s view can synchronize and update the information of other controllers and achieve multi-controller deployment among the controllers [118]. F. J. Ros *et al.* [119] reported that the network can consist of 20 nodes and two OpenDaylight controllers, and the synchronization of the aggregate traffic can be about 1 Mbps. Additionally, the researchers have explored the different designs of the control plane that is distributed. Multiple techniques have been employed to synchronize distributed control planes in SDN that differ from the topology of the controllers, the storage of network information, and the exchange of information.

Furthermore, the deployment of distributed and logically centralized controllers should meet the requirements of the network. However, some components should be taken into consideration when designing the network, such as:

i. *Topology*: In SDN, the distributed control plane can be divided into categories based on the arrangement of the controllers, such as flat, hierarchical, and hybrid. In a flat structure, the controllers are organized at one level, and communication is based and takes place among the controllers to the entire global view. In a hierarchical model, controllers are arranged in a tree-like structure [120, 121]. As a result, the controllers are not at the same level of structure. Lastly, the hybrid is the same as the hierarchical, except for the fact that multiple controllers are employed at the higher level to provide communication with one another.

ii. *Data Storage*: In SDN, the controllers require storage that stores the status of each device under the domain. This can be achieved using the relational and non-relational database characteristics of a global or standalone database system [122, 123]. The framework for the global data store can be organized by building the aggregate of the network state on the individual controllers, and the commonly known data stores are MongoDB, Cassandra, WheelFS, Hazelcast, and others.

iii. *Synchronization*: In SDN communication, logically centralized controllers need to exchange information frequently to manage the global view of the physical layer. The subscribed model can provide asynchronous means of communication in the form of a distributed system [124–126]. For example, the queue can be represented as an intervening entity in the topic or channel that allows the sender to interact with the receiver. The primary goal is that the sender can publish the message to the queue as compared to the receiver and subscribe to the message directly from the queue.

2.4.3 SDN Controller Types

This section presents the comparison of the SDN controllers and is discussed as follows:

i. ***NOX:*** In SDN, it is considered the original OpenFlow controller. This controller serves as a platform for network control that can provide a programmatic level of interface for control applications and management of the network [127–129]. Moreover, system-wide abstractions can turn networking into a software problem. Natasha Gude *et al.* [130] discussed that NOX can deal with the payloads of the packet. The authors mentioned that commercial companies use deep-packet inspection, proxies, and so on to enhance the performance of the model. While NOX can only inspect a few hundred bytes in a flow, model-based application management can incorporate processing payload technologies by proving the direction with the flows of middleboxes.

ii. ***Beacon:*** In SDN, Beacon is considered the fast and cross-platform modular Java-based OpenFlow controller that can be able to support both threaded operation and event-based.

Tony Manuel *et al.* [131] mentioned that Beacon can generate and manage the traffic of OpenFlow packets and incoming and outgoing packets. The authors mentioned that the controller can run on several platforms, ranging from lofty dual-core Linux servers to smartphones. Moreover, the cross-platform supports are advantageous for the controller and the custom for the extensible UI framework, and so on.

iii. ***Open Daylight****:* In SDN, Open Daylight is considered an open-source software project that has the support of IBM, Cisco, several other major networks, and so on. The software can be implemented in the Java programming language [132]. However, the flexibility of the software can be deployed on any hardware and other operating system platforms that can support Java [133]. Jan Medved *et al.* [134] discussed that the controller enables the support of multiple southbound protocol plugins and services from diverse sets and applications. Again, it brings the applications of the network closer together and allows network researchers to focus on SDN APIs rather than the protocols that are used for communication with the network devices.

iv. ***Open network operating system*** **()**: The controller can provide a global view of the network distributed to multiple servers. Moreover, follows in the footsteps of the distributed SDN controller called NOS [135, 136] and the improvements in performance such as latency. Rohmat Tulloh *et al.* [137] discussed that the controller feature can use a reactive forwarding application that can be used to allow and reject data traffic on SDN networks. The authors mentioned that for testing performance, the number of clients that access the server can increase. Moreover, the addition of the client can be done by the traffic generator that can provide requests to the server.

v. ***Floodlight*** is considered an open-source community of developers, such as big switch networks, that can use the OpenFlow protocol to control the traffic flows in the SDN environment. The controller provides an advantage for the developers with the compatibility of the open stack as a software tool that can be used to assist, build, and manage cloud computing platforms such as private and public clouds. Moreover, the software can run as part of the network backend for the open stack using the Neuron plugin, which can expose the networking as a service model with the REST API offered by Floodlight [138–140]. Hamza Mokhtar *et al.* [141] discussed that the controller provides easy adaptation of software and development of applications. Moreover, the authors discussed that floodlights could provide the capability to control the module and support the distribution of controllers and cooperation mechanisms.

vi. ***McNettle*** is considered an extensible SDN control system that can be able to control event processing throughput scales and the number of CPU system cores that support control

algorithms that are required for globally visible state changes occurring at flow arrival rates. Andreas Voellmy *et al.* [104] discussed that the controller has the capability to server 5000 switches using a single controller with 46 cores, which can achieve a throughput of 14 million flows per second and latency under 200 for light loads that consist of up to 5000 nodes. The authors mentioned that McNettle provides the execution on the shared-memory multicore servers, which can preserve a simple and natural programming model for developers of the controller. Moreover, it can include libraries of the concurrent data structures used to track the state of the network executed efficiently on multicore architectures.

vii. ***Terma*** is an SDN controller framework; this controller contains the requirements to build SDN controllers with the two programming languages (Ruby and C). The objective of Terma is to provide various samples of the software that is developed on top of Terma that can be run as SDN controller's samples [142]. This controller supports the development and support of C and Ruby languages, and the developers have the freedom to select the language based on performance. Marcial Fernandez *et al.* [143] discussed that the controller can provide several sample applications to permit execution, and this makes it easy to extend a new feature. Moreover, the authors mentioned that the controller is enhanced with libraries and other functional modules that can work together as an interface to OpenFlow switches.

viii. ***Mul*** is considered an OpenFlow controller that can provide support and handle multithreading operations in SDN. The controller can also support the multilevel northbound interfaces for the attached applications in the SDN network. The controller supports the C language and is used for the northbound API [144, 145]. Moreover, it can enhance the scalability and performance of SDN networks. Luciani Mamushiane *et al.* [146] discussed comparative performance-related controllers. The authors discussed the proposed metrics to achieve latency, etc. To achieve the objectives of the research, the recorded results proved the capability of the proposed controller to achieve the best latency.

ix. ***Ryu*** is a component software-based software-defined networking framework. It can provide the software components with a well-defined API that makes it easy for developers to create new networks and applications. Moreover, Ryu can logically manage APIs in a programmable network at the upper layer in various applications [51, 147]. Saleh Asadollahi *et al.* [148] discussed that the controller has the capability of creating OpenFlow packets and event management related to incoming and outgoing data packets. The author also discussed how the controller could work hand in hand with protocols such as XFlow, etc.

x. ***Juniper Contrail*** supports end-to-end dynamic configuration and optimization applications for any cloud infrastructure. The controller has support for both SBI and NBI [149–151], is integrated with cloud services called Contrail Cloud Service, and so on. It can support network function virtualization, specification of the hardware, and so on. Mark Shtern *et al.* [152] proposed a model that provides services to the network by providing a higher level of concept, and so on. The authors discussed the complex operations that can be provided by service chains.

xi. ***Rosemary****:* This controller implements the application of the network containment and resiliency-based strategy on the notion of spawning the applications independently within a micro-NOS [92, 153]. The controller can distinguish itself by the blend of process containment to prevent common failures of network applications in the SDN Stack. Seungwon Shin *et al.* [154] discussed that the objective of this controller is to prevent applications from performing operations that can be corrupt. Moreover, the authors mentioned that the controller has resilience in the presence of malicious network applications that would otherwise crash other OpenFlow controllers, and so on.

xii. ***POX*** is considered a Python-based open-source OpenFlow/SDN controller. The controller is used for the faster development and prototyping of network applications. The modeling structure of the POX can assist in designing and building SDN applications that can use the OpenFlow interface, and so on [155–157]. POX controllers provide an efficient way to implement the OpenFlow protocol, which is the de facto communication protocol between the controllers and the switches. Sukhveer Kaur *et al.* [143] discussed that the controller can be used to run different applications like a hub, switch, load balancer, and so on and is inherited from the NOX controller. Moreover, the controller can provide great strength and can be used with real hardware, in testbeds, with a Mininet emulator, etc.

Table 1: Comparison of SDN controllers [75, 158].

Controller	Programming Language	Instances
NOX [159]	C++	Single
POX [160]	Python	Single
Ryu [76]	Python	Single
Beacon [131]	Java	Single
[161]	Java	Multiple
Open Daylight [162]	Java	Multiple
McNettle [75]	Haskell	Single
Terma [105]	Ruby, c	Multiple
Mul [163]	C	Multiple
Open Contrail [164]	C, Python, Java	Multiple

2.5 Controller Placement Problem

This section presents the fundamentals of the CPP with the primary goal of achieving the research problems identified in the SDN. Moreover, the mathematical modelling for the placement of the controller is also discussed in detail.

2.5.1 CPP Overview

The SDN is a new paradigm shift intended to address the problems faced in traditional networks. As discussed above, the core idea of the SDN is the separation of the control plane and the data plane. The decoupled network can be programmed to provide a wide range of services, such as simplifying network management, increasing efficiency, and so on [165, 166]. Moreover, with the introduction of the multiple controller architecture to overcome the challenges posed by the single controller counterpart, a new challenge has surfaced in terms of controller location and allocation. Since the location and deployment of the controller are critical to the performance of the network, CPP is a challenge in the recent study of SDN. For a given network topology, the primary objectives of the CPP are [113, 167, 168]. Moreover, to achieve the research of the CPP in the network, the general formulation of the CPP can be achieved, such that the SDN network can be formulated and modelled.

G = (V, E, S) 2.1

V is considered as the collection of switches, physical links E between switches and controllers, and S is the collection of controllers. In general, $n = |V|$ denotes the switches and $k = |S|$ denotes the controller number. However, the studies of the CPP exploit the objectives to solve 1) the value of the K or the controller; and 2) $S \rightarrow V$ which denotes the relationship between the switch and the controller in the network.

Furthermore, the main goal of the CPP is to achieve latency and so on. Latency can be further classified into four subsections such as latency of the switch to controller b) the worst-case latency, average latency and others. The CPP problem considers the propagation latency and processing latency [169, 170], such that the propagation latency is considered the measured distance of two switches, etc. Moreover, the failure of the network can induce interruptions between the components of the network, such as controllers and switches, which can lead to packet loss and degrade the performance of the network. Reliability should be taken into consideration when deploying controllers. Thus, the approach to improving reliability can be divided into three categories [52, 171]. Lastly, the cost of the network should no longer be ignored due to the rapid

development of the Internet, which includes construction and maintenance costs. Thus, the cost can be classified into two aspects: a) deployment cost and b) energy consumption [144, 172].

2.5.2 CPP Influencing Factors

Since CPP is the recent hotpot of the SDN, the deployment and the number of controllers in the SDN can result in an issue and impact the network performance. However, the CPP consists of various factors that influence the placement of the controller, such as latency, reliability, load balance, cost, etc.

i. SDN Latency

All traffic functions are conducted through the exchange of messages between the controllers and switches; however, SDN experiences critical latency. Network latency includes transmission latency, propagation, queuing of the messages, and so on [173, 174]. However, if a network has not been obstructed, latency is insignificant. Moreover, transmitted latency is more dependent on the packet and rate. Again, CPP challenges focus on the propagation and processing of the latency [170]. The propagation latency is the transmission distance of data packets between the two nodes in the network; on the other hand, the processing latency can be affected by the processing performance and the controller's load. In SDN, the latency can be analyzed from a different perspective, such as the domain, which includes the worst case between the controller and switches, and the average latency of the switch and controller, which is the overall performance that should be possibly small [175–177].

a) SC-avg Latency

Since the control logic of the network is decoupled from the switches and all the functions are handled and carried out through the exchange of messages between the controller and routers, latency is very important and critical in SDN CPP [169]. The SC-avg latency is considered the average of the transmission of the packet between the controller and switches to achieve the performance of latency as shown by Equation 2.2.

$$L_{SC-avg} = \frac{1}{n} \sum_{v \in V} \min_{s \in S} d(v, s) \quad \text{.....2.2}$$

Moreover, to achieve an SC average latency, the existing algorithms can be divided into categories such as data centres to find the optimal solution using *k* centres for the dynamic traffic in WAN. Heuristic algorithms [178, 179] can reduce the search process and ensure optimal placement.

b) SC-worst Latency

involves the transmission value of the packet latency from the controller to the switch, which considers the optimal solution, and so on. The formula is shown in Equation 2.3.

$$L_{SC-worst} = \max_{v \in V} \min_{s \in S} d(v, s) \quad \text{..... 2.3}$$

Wang *et al.* [180] proposed the k-means approach to avoid the random impact position. Moreover, in WANs, the heuristics PSO and Firefly were proposed to speed up the resources and constraints. Sahoo *et al.* [181] proposed to divide the network into multiple subdomains using meta-heuristic approaches.

c) CC-avg Latency

On the other hand, communication between the controllers is important to achieve a global view of the network's state. To achieve this communication, the formula is shown in Equation 2.3.

$$L_{CC-avg} = \frac{1}{k} \sum_{s \in V} \min_{s' \in S} \mathrm{d}(s, s') \quad \text{..... 2.4}$$

Ksentini *et al.* [182] used the method to achieve CPP and optimize the distance. Again, Zhang *et al.* [170] used the method to transform the CPP challenges in space with the addition of the constraint parameter. Ishigaki *et al.* [171] considered the CPP a minimum issue by using the greedy algorithm; however, the authors ignored the bandwidth challenges.

d) Processing latency

In SDN, the processing latency can affect the processing speed, and so on. For example, when the load on the controller exceeds, the processing delay can increase. Thus, processing latency can balance the controller load, and the formula is shown in Equation 2.4.

$$L_{processing} = min \left(\max_{s \in S} l(s) - \min_{s' \in S} l(s') \right) \quad \text{.....2.5}$$

Yao *et al.* [183] discussed the issues of the load of the controller using the k-center approach for the deployment of the controller. The results obtained from the simulation were compared with dynamic scheduling approaches. It was identified that the approach can decrease the required controller, and so on.

Placing the controller in any of the available positions in the network not only degrades the network performance but increases the overhead delay of the services of the network. As a result, the overall performance of the SDN-based infrastructure can decrease. The main idea behind the CPP is to

place the optimum number of controllers in the locations to achieve performance that is better for the SDN-based infrastructure.

ii. SDN Reliability

In SDN, the degradation of network performance and failure can cause communication interruptions of network elements such as controllers and switches. As a result, severe packet loss is experienced, which slows the performance of the network. However, due to these challenges, it is important to achieve and consider the reliability of the network of SDNs due to the deployment of controllers [183–185]. Moreover, the path that is completed consists of the number of nodes and . corresponding controllers in the control plane [186, 187]. Again, for node and switch failure, it is advised that switches are linked to respective controllers [181, 182]. In general, the methods of achieving and improving the reliability of SDN include multiple control paths, multiple controllers, and minimization of the control path.

iii. Cost

In consideration of the economy and environmental protection, the cost cannot be left out of internet development [188]. It includes the construction of the current maintenance costs of the operation, such as deployment costs and energy consumption. On the other hand, the devices in the network can still account for 90% of the load during busy hours [182, 186, 187]. Ruiz-Rivera *et al.* [189] proposed to reduce the active links to save energy, and the results obtained from the simulation demonstrated that rerouting could cause controller overloading.

iv. Load Balance

Load balancing is considered one of the most important ways to effectively allocate the resources of the network to improve network performance and QoS. Compared to traditional network architectures, the SDN has higher efficiency and performance in different aspects but can also solve the problem of load imbalance [190–192]. Moreover, due to the effectiveness of the controller that controls the whole network in managing and allocating all the traffic in the network, The controller can also collect the load of the network in real time and dynamically adjust the load of the servers and links based on load balancing strategies, which can maximize the utilization of the resources of the network and improve its performance [193, 194].

2.6 Controller Placement Problem Classification

This section presents some of the related papers on the SDN CPP classification. We first presented some of the employed techniques, including CPP clustering, GT, ML, DL, etc. Secondly, we briefly

summarize some of the techniques and methods used to achieve CPP (latency, reliability, load balancing, etc.). Lastly, we presented a brief discussion about the drawbacks of all the listed approaches and their justification.

i. CPP Clustering

This subsection presents the CPP-reviewed papers based on clustering strategies. The reviewed papers were compared based on the methods, algorithms, placement metrics, evaluation, and limitations.

Table 2: CPP clustering strategy

Author	**Method & Algorithm**	**Placement Metric (s)**	**Evaluation**	**Limitation**
Guodong Wang *et al.* [180]	Optimized k means	Latency minimization, load balancing, resistance/reliability	Maximum latency between the centroid and its associated nodes	Decrease the complexity of the controller placement
Jiang Liu *et al.* [205]	Clustering algorithms	Reliability	An optimal and sub-optimal solution	Multiple objectives optimization
Lange *et al.*[178]	Simulated Annealing	Switch to controller latency, reliability, and load balancing	Allows analysis and results in a trade-off between time and accuracy	Large problem instances cannot be computed (enormous memory requirements)
Guodong *et al.* [180]	Optimized K-means	load balancing, resistance/reliability awareness	Network topologies	Energy saving
Liu *et al.* [186]	Simulated Annealing	Reliability	Allows for finding close optimal results	Consequent control traffic
Wang *et al.* [169]	Improved k means	End-to-end latency, queuing latency	Reduce maximum latency between controllers	Extensive simulations need to be conducted under multiple real topologies
Lei *et al.* [16]	Heuristic	Propagation, transmission delay	Numerical demonstration	Control plane minimization
Lange *et al.* [178]	Pareto simulated annealing	Resilience, Failure Tolerance, Latency	User's requirements for accuracy	large problem instances that cannot be computed

ii. CPP - Game Theory

In this subsection, we presented the reviewed papers based on CPP game theory. The reviewed papers were evaluated based on the used methods and algorithms, placement metrics, evaluation, and limitations.

Table 3: CPP game theory

Author	Method & Algorithm	Placement Metric (s)	Evaluation	Limitation
Hemant *et al.* [219]	Non-zero-sum game	Load balance, latency	Optimization engine	Quality of service
Ksentini *et al.* [182]	Bargaining game	Minimizing latency	Network topology	Communication minimization overhead
Marwa *et al.* [18]	Game theory, a mathematical model	Latency, throughput	Topological network Smart Node	Cost minimization
Bala *et al.* [220]	Cooperative game theory	Latency, load balance	Network subnetworks	Network partition
Erika *et al.* [52]	Not –a zero-sum game	Latency, load balance	Routing, offloading	Communication
Adlen Ksentini *et al.* [203]	Bargaining game	Cost, latency	Virtual mobile network	Flow installation
Guozhen *et al.* [217]	Game-theory	Latency, bandwidth	Formulation switch migration decisions	Available resource
Guowei *et al.* [218]	Non-cooperative game	Scalability, load balancing, resource utilization	Switch Migration	Pareto optimality
Ksentini *et al.*.[203]	Game Theory Nash Bargaining game	Intel controller latency and load balancing	Reduce the SGW relocation, which is costly for mobile operators	Flow installation latency is still a challenge
Ksentini *et al.* [182]	Bargaining game	Latency, communication overhead, load balancing	Achieves the Pareto optimal efficiency	Game-theoretic approaches should be evaluated for performance

iii. CPP - ML, DL and others

This subsection presents the CPP ML-reviewed papers. Moreover, other CPP-based algorithms were presented in detail. The literature review was evaluated based on the CPP objectives, including the switch to the controller, worst-case latency, load balancing, and others.

Table 4: CPP ML, DL & others

Author	Method & Algorithm	Placement Metric (s)	Evaluation	Limitation
Heller *et al.* [113]	K centre	Switch to controller & worst-case latency	k-centres can consider up to n − 1 controller failures	Fully distributed control is more resilient than decoupled control planes
Maryam Tanha *et al.* [195]	Capacitated k centre	Switch to controller latency Reliability	Minimize the total incurred cost incremental design	Design of heuristic algorithm, testing random topologies
Yao *et al.* [183]	Linear relaxation	Switch to controller latency and load balancing	A strategy that reduces maximum-load	Dynamic controller provisioning of the radius is still a challenge
Jimenez *et al.*.[196]	k–critical	Load balancing	Discover the minimum number of controllers and location	Tree topology, a mechanism for load migration among the controller
Sanner *et al.* [199]	NSGA	Intel controller latency and load balancing	NSGA maximize the cluster's average connectivity	Adoption of an optional crossover step
Wendong *et al.* [201]	l–w greedy	Switch to controller latency	Improve reliability without introducing unacceptable latencies	Optimized reliability is insufficient to meet the existing response time
Hock *et al.*.[202]	POCO	Switch to controller latency, Reliability and load balancing	Offer network operators a range of options	Placement metrics for other SDN scenarios, such as SDN in business networks
He *et al.* [204]	Mixed-integer programming	Average flow setup time	For low densities, an adaptive flow controller	The cost of controller migration, depending on its load, and the cost of switch reassignment
Liao *et al.* [185]	Particle swarm optimization	load balancing, latency	MOGA generate a Pareto frontier with a larger diversity	More evaluations on network topologies and scales
Ashutosh *et al.* [209]	Heuristic	Scalability, reliability	Varna-based optimization (VBO)	Deployment of numerous controllers
Pantelimon *et al.* [210]	Critical analysis of multi-CPP	Processing capacity	Single point of failure	Limited controller processing capacity
Yuanxiang *et al.* [213]	Deep neural networks	Latency, load balance	CIFAR- 10 benchmarks	Response time, policy
Rong *et al.* [216]	DL algorithms	Transmission and processing delay	Kuhn Munkres model	Controller capacity matching

It is true from the CPP classification that each paper proposed has applied a unique approach due to the placement of the controller and so on; however, finding the optimal placement of the controllers can be a challenge when the performance accuracy of the algorithms is not efficient. Moreover, there are various approaches adopted and applied to achieve reliability, latency, load balance, and so on.

iv. CPP – Latency

Heller et al. [113] discussed that how many and where to deploy controllers depends on the desired reaction bounds such as the choice of the metrics and the chosen topology. The authors examined the fundamental limits of the control plane propagation latency on internet2 production deployment and expanded the scope to over 100 publicly available WAN topologies. Moreover, the authors

mentioned that one controller location is sufficient to meet the existing reaction time requirements of fault tolerance requirements. Similarly, Yao et al. [183] discussed the factors of the controller's load as the challenges of the CPP. The authors introduced the CCPP together with an effective algorithm to achieve the objective. The results obtained proved that the new approach can reduce the number of controllers' placement and reduce the overload of the busiest controllers. Moreover, it can achieve a smaller radius for the dynamic controller provision or the dynamic scheduling strategy. In the same vein, Jimenez *et al.* [221] discussed the key challenges faced by providers that define scalable network control and exploit the benefits of the SDN when used in conjunction with management-efficient strategies. The authors discussed control layer models that are not concerned with controller scalability. The results obtained demonstrated the effectiveness of the solution compared to other controller placement solutions.

In a similar study, Bari *et al.* [197] discussed the traditional implementation within SDN that relies on a logically centralized controller in the control plane. However, the authors discussed that deployment of the controller in a WAN using a centralized approach brings some limitations due to performance. To achieve the limitation, the authors proposed the deployment of multiple controllers in the network. The framework adjusted the number of controllers and delegated each to the OpenFlow switches based on the dynamics of the networks to achieve minimal flow setup time and overhead communication. The results obtained showed that the solution can minimize the flow setup while incurring very low communication overhead. Also, Ishtaique et al. [198] discussed that CPP assumes fixed traffic solutions that can't adapt to the placement due to the load. To achieve this, the authors addressed the dynamic CPP, consisting of determining the location of the controller modules to bind the communication latencies, determining the number of controllers per module, etc. Moreover, the authors proposed the LiDy+ algorithm combined with the controller placement module for dynamic flow management. The results proved that LiDy+ was not only achieving a smaller number of controllers but also incurring less energy and maintenance costs.

v. CPP – Load & Reliability

Jean-Michel *et al.* [199] discussed the inherent complexity of optimal placement and the lack of flexibility of the heuristic. The authors proposed the generic algorithms designed from the NSGA II framework to deal with the CPP. The algorithm provided multi-objectives and multi-constraint solutions computed in parallel.

Again, it maximizes the average cluster connectivity and balances the load of the controller of the clusters to improve reliability. Hemamalini *et al.* [212] also proposed a framework for dynamic controller deployment in SDN networks using the ML approach. The authors mentioned the challenges that include the estimation of the required number of controllers and optimal locations.

To achieve CPP, the ML, together with the mathematical formulations and heuristics approach was used to achieve CPP. The number of controllers required was given as an input parameter. Again, traffic engineering based on the deployment of the controller was integrated. ML techniques were used, and the ANN-based model was created for the analysis and prediction of traffic. Yiwen *et al.* [208] discussed deep reinforcement learning techniques for CPP in SDN. The authors discussed the challenges that include network characters as well as flow fluctuations in SDN. To achieve CPP in the SDN, a deep-network (DQN)-enabled, dynamic flow data-driven approach for controller placement problems (D4CPP) was proposed. The D4CPP integrated the historical data learning network into the controller deployment and real-time switch controller mapping decisions. The task of the D4CPP was to take flow fluctuations, etc., into full consideration and reach an optimized balance among the metrics.

Ashutosh *et al.* [209] proposed heuristic approaches for reliable SDN-CPP. The authors address the challenges that include the scalability and reliability issues of the network based on one controller. To achieve CPP, the technique adopted varna-based optimization (VBO) to minimize the total average latency of SDN. Again, the approach was compared and evaluated with particle swarm optimization (PSO), teacher learning-based optimization, and Jaya algorithms. Shu *et al.* [69] discussed the challenges that include improving the performance of SDN networks and so on. To achieve the CPP, the authors introduced a single link failure (CPSLF) and multiple failures (CPMLF). Moreover, the greedy placement algorithm (GPA) was adopted to address multi-link failures, and Monte Carlo simulation was used for computational overhead. The real network topologies were used, and a heuristic algorithm was good for performance. Again, the authors mentioned that in the future, the failure rates of the nodes and controllers should be considered, and they proposed solutions for the CPP and network. Pantelimon *et al.* [210] discussed the critical analysis of multi-CPP in large SDN networks. To achieve CPP, the authors mentioned an approach that attempts a critical study of various placement solutions. Again, the authors identified the guidelines and evaluation of the CPP performances.

vi. CPP – Reliability & Load balancing

Ashutosh *et al.* [211] also suggested varna-based optimization (VBO), a novel method for capacitated controller placement problems in SDN. The authors mentioned the challenges that include the capacity of controllers and the dynamic load on switches as the main factors. To achieve CPP, the authors proposed a novel optimization algorithm called VBO to minimize the total average latency of the SDN, TLBO, and Jaya algorithms. The results from the simulations prove that the TLBO outperformed the PSO, and the VBO outperformed the TLBO and Jaya algorithms.

Amin *et al.* [11] discussed the approach to controller performance in SDNs. The authors addressed the challenges that include the deployment to establish the new lower bound on the maximum throughput for SDN controllers. To achieve CPP, the authors proposed an approach that achieves controller throughput, maximum throughput, and others. Moreover, they discussed that single-controller micro-benchmarks should be taken into consideration in the future to understand the performance and implications of SDN. Yuanxiang *et al.* [213] proposed optimization placement using deep neural networks in SDN. The authors mentioned the challenges, which include the increased amount of computation, resources, and so on. To achieve CPP, a learning algorithm based on proximal policy optimization was used to find the optimal placement of the DNN training. The Markov model decision process with multiple stages was also used to determine the placement and performance of the controllers. The results obtained from the simulations demonstrated that the training time for placements was recommended for use by Spotlight. Similarly, Mohamed *et al.* [214] proposed a framework for managing CPP for grid-aware SDN. The authors mentioned the challenges that include security in crucial fields for the smart grid. To achieve placement, ML algorithms were introduced with various metrics to affect the placement problem in terms of grid operation and performance. Further, the UK GSP network capacity map was used. The results obtained from the simulation showed the magnitude of the decision concerning standard characteristics within the network.

Elliott *et al.* [215] also proposed a framework that manages the CPP grid aware. The authors mentioned the challenges that include determining a suitable number of controllers' locations for optimization and communication delay. To achieve CPP, the algorithms were used to tackle the CPP with consideration of communication delay, control plane utilization, and so on. The new algorithm was integrated with a genetic algorithm (GA) and gradient descent (GD) optimization. The results showed that the algorithm can effectively strike a trade-off between control plane utilization and network response time.

In a similar study, Habib *et al.* [15] proposed a framework that uses a learning automaton-based CPP algorithm for SDN. The authors highlighted the challenges of fault tolerance, placement of the controller, and others. To achieve CPP, the simple heuristic algorithm was used to leverage a learning algorithm (LA). The algorithm can place the controllers based on a predefined propagation latency between the controllers and switches by minimizing the overall propagation latency. The available topology and algorithms were used for the simulation, and results showed that the results outperformed the state-of-the-art algorithms in terms of propagation latency.

vii. CPP – Load Balancing & Switch Assignment

Guodong *et al.* [180] discussed the implementation of DL that partitions the network to achieve CPP in SDN. The authors mentioned latency minimization between controllers and switches as a significant factor that affects the network. To achieve CPP, the authors proposed an optimized K-means algorithm with optimized K-means, and then the entire network was divided progressively from 1 to K subnetworks to calculate the shortest path for the optimal placement. Also, Rong *et al.* [216] discussed the framework for control plane delay minimization based on capacitance in SDN. The authors discussed the approach for determining the number of controllers and so on. To achieve control plane delay minimization, DL algorithms were used and tested for controller-switch association and controller capacity matching algorithms to check the controller-switch association subproblem with the application of the Kuhn Munkres model to achieve control plane delay minimization-based capacitated CPP.

Guang *et al.* [183] discussed the approach based on the capacitated controller placement problem (CCPP) in SDNs. This study dealt with the challenges faced by the controllers in the network. The challenges encountered include propagation latency, load on the controllers, and so on. For this approach, the authors proposed ML techniques as part of the solutions to improve performance. Lei *et al.* [16] discussed the framework for control plane delay minimization based on the SDN controller placement scheme. The controller placement problems and a delay between the switch and others were identified as the challenges of SDN. The ML/DL techniques and heuristic methods were incorporated together to improve the performance of the system. To achieve CPP, the authors proposed controller placement algorithms that connect the variables and the switches for communication. The results obtained showed a numerical demonstration of the effectiveness of the proposed techniques. The authors thumbed up the use of ML/DL algorithms based on SDN, which should be considered for future use.

Guozhen *et al.* [217] proposed the dynamic switch migration towards a scalable SDN control plane. The authors presented the challenges that include the migration of switches from controllers and others. To achieve switch migration, the authors proposed the formulated approach for decisions as a centralized resource utilization maximization problem with constraints on the multiple resource dimensions. However, the results obtained from the simulation proved that the mechanism can be utilized as an available resource in the control plane. Guowei *et al.* [218] proposed dynamic switch migration using a non-cooperative game. The authors mentioned the challenges that include control plane scalability and others. To achieve CPP, a switch migration scheme using non-cooperative game theory was adopted for the improvement of the control plane.

Erika *et al.* [52] discussed the challenges of the CPP in SDNs. The authors highlighted the challenges, which include latency, network congestion, and others. To achieve CPP, the authors introduced the non-zero-sum game for decision-based controller overload. The inner control modules of SDN can communicate with the peers for routing updates, payoff values, and offloading messages. The optimal value k was obtained dynamically and optimally for controllers. Ksentini *et al.* [203] discussed the approach of the SDN in 5G. For this approach, the authors devised a new algorithm for the gateway CPP in SDN-based virtual mobile networks. To achieve CPP, the proposed algorithm was introduced to find a fair trade-off between reducing the relocation, which was costly for the mobile operators, and the flow installation latency. Hemant *et al.* [219] discussed the optimal controller placement in SDN using a non-zero-sum game-based distributed technique. The authors presented the challenges that include the optimization engine in each SDN controller, QoS, and others. To achieve CPP, the authors mentioned the optimization in each controller and computed the payoff function for comparison with the payoff value of the neighbours for decisions such that new controllers can be added, deleted, or offloaded between controllers.

viii. CPP – Deployed Controller and Latency

Ksentini *et al.* [182] discussed the approach using the bargaining game for optimal controller placement in large SDN networks. The authors discussed the problem of minimizing the latency in SDN and the communication minimization overhead between the controllers and others. To achieve CPP, the authors proposed bargaining games to ensure a fair trade-off between the controllers.
The results obtained from the simulations showed that the proposed solutions can ensure a better trade-off compared to other mono-objective solutions. Marwa *et al.* [18] proposed game theory-based networks. The authors mentioned the challenges that include a rapid increase in traffic flow in the mobile network and others. To achieve CPP, the authors indicated the roles of the smart node within the model and took advantage of game theory to study the optimal placement of the intelligent entity in the network. Habib Mostafaei *et al.* [15] mentioned the challenges that include fault tolerance and others. To achieve CPP, the heuristic algorithm was used to leverage a learning automaton (LA) approach. The algorithms used can place the controllers based on a predefined propagation latency between the controllers and switches, minimizing the overall propagation latency. The topology, together with the algorithm, was used for the simulation, and outperformed results were recorded for efficiency and effectiveness.

Elliott Wen *et al.* [215] discussed the framework for managing controller placement problems for grid-aware SDN CPP. The authors mentioned the challenges that include determining a suitable number of controllers deployed for optimization and communication delay. To achieve CPP, the algorithms were used to tackle the CPP with consideration of communication delay, control plane

utilization, and the controller workload distribution simultaneously. The results obtained from network scenarios (small-scale and large-scale) showed the performance of an algorithm for effectiveness and good response time. Mohamed Samir *et al.* [214] discussed the framework for managing CPP grid SDN. The authors mentioned the challenges that include the security of the smart grid. To achieve the placement of controllers, the proposed techniques were used to determine the number of controllers that manage and control the large network in SDN. Again, ML algorithms were introduced with various metrics to affect the placement problem in terms of grid operation and performance. The results obtained from the simulation showed the magnitude of the decision concerning standard characteristics and the effect of optimally selecting the controller location within the network. Bala *et al.* [220] discussed the approach of cooperative game theory-based network partitioning for controller placement in SDN. The authors mentioned the challenges that include the partition of the network into subnetworks and multiple domains and the deployment of the controller in the domains. To achieve CPP in the network, the authors proposed the addition of a cooperative game k-means algorithm. This approach divided the network into sub-regional simulations as an asset of cooperative games.

ix. Discussion and Justification

From the reviewed papers, it is evident that several CPP strategies have been applied in SDN. Most of these strategies are driven by machine learning (ML), deep learning (DL), mathematical formulations, bargaining games, non-cooperative games, and so on. Despite the variety of implemented CPP strategies and approaches, an efficient CPP algorithm has yet to be achieved. Additionally, various security mechanisms, such as IDS, have been proposed and implemented for SDN, but a robust and efficient strategy is still lacking. Moreover, some of the reviewed research demonstrates individual strategic implementations, which are insufficient to address the issues and challenges of the SDN environment comprehensively. Recent research indicates that the challenges of SDN-CPP should be tackled collectively, as they are interdependent with solutions to other problems, such as security challenges and load balancing. To achieve a practical solution to the SDN environment, the network challenges must be tackled collectively within a large integrated framework.

Therefore, in this research, we aim to address the challenges in an SDN environment by proposing an efficient solution for SDN scalability through CPP and attack awareness. Our objective is to achieve the best trade-off between network performance and security in SDN-based WANs. We have incorporated ML techniques into our integrated framework design to tackle challenges related to security, load balancing, processing latency, reliability, and the number of deployed controllers

during control message exchanges. This approach is intended to enhance the performance of SDN-enabled WAN settings.

2.7 Chapter Summary

This chapter presented general information based on the SDN CPP literature reviews. The advantages, characteristics, and layers of the SDN were explained in detail. SDN single- and multiple-controller strategies were fully discussed. Some of the widely used controllers, such as POX, NOX, Beacon, OpenDaylight, etc., and their performance were explained. Some of the factors that influence the CPP, such as cost, reliability, and so on, were further explained. CPP classification approaches and techniques, algorithms, metrics and factors, evaluation, limitations, and so on were fully explained in detail. The approaches that were adopted for this research are also discussed in detail.

Chapter 3

Literature Review: *Analysis of CPP Clustering Techniques*

3.1 Chapter Outline

This chapter presents the analysis of the clustering approaches for SDN-based CPP in the WAN. It begins with an overview of ML and various clustering strategies and then, a discussion on the traditional clustering algorithms, enhanced clustering algorithms, etc. Moreover, the results of the performance analysis are presented to evaluate the algorithms for SDN CPP selection in WAN settings. Lastly, we present the chapter summary.

3.2 Overview of Machine Learning

ML is the subfield of AI that gives computers the ability and performance to adapt and learn logical concepts [222-226]. Mannila *et al.* [227] discussed ML as a powerful collection of techniques. In the same vein, Alsheikh *et al.* [228] elucidated by improving performance and achieving prediction how to collect and analyze electronic data such as digitized human-labeled sets of training and other types of information. Taiwo *et al.* [230], on the other hand, stated that for an ML algorithm to run efficiently, it requires learning that includes gaining comprehensive knowledge to acquire training skills, experience for the samples, and learning using the practice or samples.

Furthermore, Michalski *et al.* [231] discussed the categorization of ML based on task-oriented, cognitive simulation, and theoretical analysis since learning with ML depends on the data used, which are based on the combined fundamental concepts. ML has been widely used for different purposes, such as classification and regression analysis, and a variety of applications that include bioinformatics, speech recognition, etc. ML algorithms and techniques come from different fields, such as mathematics, computer science, statistics, etc. Alsheikh *et al.* [232] also reported that there are proposed techniques of ML that exist that are used individually or in combination to solve problems, and these techniques are supervised, unsupervised, and reinforcement learning.

Supervised learning involves learning mappings between the input variables and an output variable, such that the application of this mapping predicts the outputs for the unseen data. It is one of the most important technological methods in ML and is central to the processing of multimedia data [233-236]. The training data is labelled, and the primary objective is to learn a function that generalizes the unseen data. Moreover, the form “supervised” means that the learner can be provided with the necessary labelled data. However, in supervised learning, some algorithms can be used for a variety of applications [237, 238], and these algorithms include support vector machines,

linear regression, logistic regression, naive Bayes, linear discriminant analysis, decision trees, k-nearest neighbour, and others.

On the other hand, unsupervised learning is considered to involve the samples of the data set offered to the system as observations without considering any label or the responsibility of the learner to achieve the necessary actions on the data set. However, this type of learning also has a variety of algorithms that can be utilized to achieve a specific task, and these algorithms include neural networks [239, 240], apriori, k-means clustering, hierarchal clustering, principal component analysis, and so on. Lastly, reinforcement learning (RL) is a type of learning that can make decisions based on the actions of the output results. A variety of algorithms for RL include deep deterministic policy gradients (DDPG) and so on.

Moreover, recent research has shown that SDN-based CPP practical solutions include controller load balancing problems, request/resource scheduling problems, and others. However, this research is designed to adapt and utilize both supervised and unsupervised learning methods to achieve the SDN controller's placement in the WAN [144, 241, 242]. Moreover, both approaches are integrated to achieve some practical solutions to clustering strategies for both IDS and CPP in SDN WAN. The density-based clustering strategy is also adopted to evaluate and analyze the number of clusters in a high-density area. The approach is based on the idea that the data points in each cluster are treated as a contiguous region of high point density, which is separated from others such that the clusters of contiguous regions are of low point density. Again, the method is used to detect concentrated, separated, and similar data points within empty or sparse cluster areas [243-245]. The collected datasets used to achieve the objectives of this research are considered and represented as the size of the data packets clustered within the SDN WAN. These data packets can communicate with each other during traffic flow. However, considering all the connections of the links of the bandwidth [246-248] remain constant, switches can be able to incur a load that is fixed, and the control path for the security in the network is considered perfectly solved [246-248].

3.3 Clustering Techniques

Clustering is considered the most useful approach in several techniques, such as exploratory pattern analysis, grouping, decision-making, and so on. Clustering is described as the use of methods and techniques to group unlabelled data. The method of clustering is taken as a division of related data sets into multiple groups. However, clustering methods are generally used in the network topology to form several clusters and to determine their centres cluster at the same as well as the controller locations etc. Although there are many related studies considered for CPP, there are several

challenges that should still be considered. This includes some parameters such as cluster number and threshold density [105, 196, 249].

Recently, clustering has been considered an effective approach to the placement of controllers by different researchers. Several authors reported that by using the clustering or partitioning approach in the network, the results from the experiment showed better performance in terms of time consumption, effectiveness, and efficiency of the SDN controller placement [175, 250–253]. Xiao *et al.* [254] tackled the CPP based on two questions: how the WAN topology can be partitioned into several subdomains, and where to place the controller in each SDN domain. By using the spectral clustering placement algorithm, the approach could partition the large network into several subdomains of SDN to achieve the problem of the CPP for WAN. Similarly, Liao J. *et al.* [179] discussed the approach of splitting the SDN WAN into several sub-networks using density-based switch clustering; one controller was deployed in each sub-network. The approach was proposed such that the range of each sub-network can be decided by finding the density of the topology. Again, the size of each sub-network was decided based on the capacity of the deployed controller. The simulation results showed that the algorithm is faster, outperforms the traditional network, and can improve the performance of the control plane. He. M. *et al.* [204] also used the concept of the clustering approach to model the end-to-end flow setup time. They analysed the CPP for the dynamic traffic flows based on a combined CPP model: the location of the controller and the assignment of the switch to the controller that is optimized for the minimum average flow setup time-based on the different traffic conditions inside the network.

Similarly, G. Wang *et al.* [180] used a clustering algorithm to divide the network into subnets to reduce the complexity based on the optimized k-means algorithm, which shortens the maximum waiting time between the centroids of the subnet and switches. The optimized k means efficiency was to make sure that each partition could reduce the maximum delay between the centroids of other nodes.

Accordingly, Wang *et al.* [169] also discussed the approach to decreasing the end-to-end latency and processing latency by adopting two steps of dividing the network into sub-domains to reduce the end-to-end latency between the controller and switch. For this approach, the authors proposed the clustering-based network partition algorithm (CNPA) for the partition of the network. They placed the multiple controllers in a subnet with many switches to further reduce processing latency.

A. Hierarchical Method

Hierarchical clustering is used to build a hierarchy of clusters. The algorithms of hierarchical clustering can build the clusters gradually and based on the strategies for hierarchical clustering and can be divided into two types, such as the data objects, which are not divided or partitioned into a particular cluster in a single step. A series of partitions can take place by running from the single cluster that contains all the data objects to other clusters that contain every single object. Some more basic hierarchical models include the Lance-Williams formula and the conceptual clustering idea [255-257]. Alhazmi, K. *et al.* [258] proposed a framework that combines the hierarchical clustering and betweenness centrality models to reduce the latency of the switch to the controller and apply the balance of the load by using the centrality scores of the nodes.

B. Partition Method

The partition method generally results in a set of N clusters, and each of the data objects can belong to one of the clusters. Each clustered data object can be represented by a centroid or the representation of a cluster, and this can be taken as the summary of all the data objects that can be contained in a cluster. However, the formation of the description will be determined by the type of clustered objects [259, 260]. Also, due to the situation where real-valued data is available, the arithmetic mean of the attribute vectors of the objects within a cluster can provide the appropriate representative. Wang, G. *et al.* [261] provided a simulation of the theory of matrix in SDN WAN such that the large SDN network was partitioned. The results obtained from the simulation showed that the throughput and latency performance were improved.

C. Density-based

The density-based method of clustering is capable of discovering clusters of arbitrary shapes. As a result, it provides the advantage of the natural protection of the outliers. The method has the advantage of grouping the objects according to specific density functions. This type of method can also be defined as the number of objects in a certain neighbourhood of the data objects. In this clustering method, a given cluster can continue to grow as long as the number of data objects in the neighbourhood exceeds some parameters [262-265]. Liao J., Sun et al. [179] proposed an algorithm to divide the network into various subnet clusters based on the density of the switches. The switches were connected and had connections with the other clusters. Moreover, each cluster had its controller based on the controller capacity.

3.3.1 Clustering Algorithms

This subsection presents an overview of different clustering algorithms and analyses the differences between them. In addition, some of the proposed enhanced-clustering algorithms are further explained, analysed, and adopted for implementation to achieve the objectives of this research.

3.3.1.1. Traditional Clustering Algorithms

This subsection discusses some of the traditional algorithms selected to achieve the objectives of this research. These algorithms are widely used and are the most known type of machine learning algorithm. Moreover, they can deal with large, unstructured, and unclassified datasets. Also, the choice of the selected algorithms is based on the widely used types of algorithms in machine learning to achieve clustering objectives.

A. K-Means Clustering Algorithm

The K-means clustering algorithm is an unsupervised and non-deterministic iterative method in data science, machine learning, and so on and can provide fast computation response times on data sets. Each data instance is taken to the nearest centre. The Euclidean distance is generally applied to data instances to determine the distance between each data instance and cluster centres [266-268]. The k-means algorithm formula is shown in Equation 3.1, and the algorithm is explained in Figure 2.

$$E = \sum_{i=1}^{k} \sum_{x \in c_i} | x - x_i |^2 \quad \text{..... 3.1}$$

k-means Algorithm
Input: data set similar and a constant k **Output**: k centroid **Steps:**1: Specify the number of assigned k clusters 2: Randomly initialize k centroids 3: repeat processes 1 and 2 4: expectation: assign each point to its closest centroid 5: maximization: Compute the new centroid (mean) of each cluster 6: Until the centroid positions remain constant

Figure 2: k means algorithm

B. Affinity Propagation Clustering Algorithm

Affinity propagation (AP) is considered one of the most used clustering techniques. Frey *et al.* [269] proposed the algorithm. Due to its performance, the algorithm first selects the data points and passes messages between them, namely responsibility and availability. Input data points of the algorithm also consider the similarities between pairs of data points defined such that *S = (*s, cluster exemplars of all the data points) and the relationship of each data point that corresponds to the

cluster's exemplar defined as the output. Moreover, *s(i,j)* is the similarity function of the similarity matrix. The affinity propagation algorithm is explained in Figure 3.

AP Algorithm
Input: similarities between the data points **Output**: obtained a high-quality set of exemplars **Steps:** 1: Similarity matrix(S) gives information about the similarity between any instances 2: Responsibility r (i, k) quantifies how well-suited the element k is to be an exemplar for element, i, considering the nearest contender k's to be an exemplar of i 3: Availability a (i, k) quantifies how appropriate is for i to choose k as its exemplar 4: Calculate the criterion matrix after the updating is terminated. 5: Assign element i to exemplar k which is not highly responsible but highly available to i

Figure 3: AP algorithm

C. DBSCAN Clustering Algorithm

The density-based clustering technique finds the clusters based on the data points in a region. The objective of a density cluster is that for every instance of data point in a cluster, the neighbourhood of the given radius (Eps) should have the minimum number of instances (MinPts). DBSCAN clustering algorithms are considered the most widely known density-based clustering algorithms [270, 271]. The DBSCAN algorithm begins with the assumption that all the stars are noise stars. Then the algorithm steps through each of the stars such that, for a given choice of ϵ and, it can check whether it is the core of the star. However, once the core of the star is identified, the algorithm can be able to organize the stars in the form of groups for the consideration of each of the stars in a sequence. Once the star is the core and is in the ϵ-neighbourhood of another core star of the group label, the star can be allocated and given the label that is of the same group. Otherwise, if the star is not core, the algorithm will further determine whether it is the border star such that it is assigned to the group of its neighbouring core point that belongs to or the noise star [272-275]. The DBSCAN algorithm is explained in Figure 4.

DBSCAN Algorithm
Input: N objects to be clustered and global parameters Eps, MinPts **Output**: Clusters of objects **Steps:** 1: Arbitrary select a point P. 2: Retrieve all points density-reachable from P wrt Eps and MinPts. 3: If P is a core point, a cluster is formed. 4: If P is a border point, no points are density reachable from P and DBSCAN visits the next point of the database. 5: Continue the process until all the points have been processed.

Figure 4: DBSCAN algorithm

D. Mean-shift Clustering Algorithm

The mean shift is considered a non-parametric clustering algorithm that does not require a prior definition of the cluster number or a restriction of the shape of the clusters, such that the estimation of the density forms the basis of the mean shift. Each data point is referred to as a feature vector, and the algorithm can execute the gradient ascend on the local estimation density such that convergence is reached [276, 277]. Moreover, during the running of the algorithm, the stationary points represent the modes of the density and the mathematical formulation is explained in detail as follows: Given the set of ***n*** data points in a d-dimensional space, the value of i = 1,..., n is such that the estimation of the multivariate density with the kernel *K(x)* calculated at the point ***x*** can be defined as the criterion.

$$f'(x) = \frac{1}{nh^2}\sum_{i=1}^{n} K(\frac{x - x_i}{h}) \quad \text{.....3.2}$$

where. *h* is defined as the bandwidth of the kernel size. The mean Shift algorithm is explained in Figure 5.

Mean-shift algorithm
Input: Coordinates of the cluster centres
Output: Calculated gravity (mean) of the vector W
Steps: 1: Start with the data points assigned to a cluster of their own
2: Compute the centroids
3: Location of the new centroids is updated
4: Process is iterated and moved to the higher density region
5: Stop once the centroids reach a position from where it cannot move further
6: Predict the clusters

Figure 5: Mean-shift algorithm

3.3.1.2. Enhanced Clustering Algorithms

This section discusses the improved clustering algorithms employed to achieve the objectives of this research. The algorithms are integrated with homogeneity and completeness to analyse the quality of each cluster and the distance between the data points [278].

Moreover, there are many clustering algorithms to choose from, such as spectral clustering, the Gaussian mixture model, Birch, library installation, and so on. However, no single best clustering algorithm has been identified for all cases. However, to determine the comparison of the best-performing model, it is a good idea to explore a range of clustering and different configurations for each algorithm. The selected clustering algorithms are the most known and used types of machine learning algorithms. They are the most essential algorithms in any data science project that can deal with unstructured and unclassified datasets. Modern clustering algorithms exist, but none can replace the traditional algorithms in terms of the adaptation of the data and the application being built. The integrated approaches are summarized in Table 5.

Table 5: Homogeneity & completeness clustering techniques

Method	Definition
Homogeneity analysis [279]	Data points that are like each other in a cluster. Analyses the cluster of data points and provides test analysis and evaluation on the similarities of data points. The technique can make the selection of the data points in the sample based on the similarities or identical traits such that the data points can share the same cluster.
Completeness analysis [280, 281]	No gaps or missing points of data points in the cluster. Analyses the cluster of data points and considers the comprehensive wholeness of the data points. The technique analyses the cluster of data points such that there should be no gaps or other missing information on the connection between the data points instances to achieve completeness. The incomplete data point can result in errors and inconsistencies that can impact the readability, and accuracy of the model.

A. DBSCAN Enhanced Clustering Algorithm

DBSCAN enhancement algorithms measure the quality of a cluster and how close the data points are to its adjacent clusters for the variation of the number of partitions. The neighbourhood of the given radius (Eps) considers the minimum number of points (MinPts). Moreover, it identifies the distinctive clusters of the data points, by considering that the cluster in the data space is the contiguous region of the high point density that is separated from the other clusters by contiguous regions of the low point density [282, 283] such that the following formula 3.3 is achieved.

$$|N| < \min Pts \quad \text{.....3.3}$$

In realism, the proposed algorithm employs homogeneity, completeness, and silhouette coefficient to further analyse the quality and distance of data points within the clusters [278]. ε within each cluster is considered as the minimum / shortest distance in a cluster. The DBSCAN enhancement algorithm is presented in Figure 6.

DBSCAN enhancement algorithm

Input: N objects to be clustered and global parameters Eps, MinPts
Output: k cluster number
Steps: 1: Compute the degree matrix D with D (i, j)
2: Let the vector-matrix containing the vector column in matrix D
3: Compute the vector corresponding to the first value of the nth term in the matrix D
4: Compute the core samples
5: Compute the cluster core sample when true positive
6: labels the core data minimum samples
7: cluster the number of the core data points absolute cluster
8: Estimate the number of the epsilon cluster in a set D
9: Calculate the Homogeneity of the data points' accuracy
10: Compute the completeness of the data matrix accuracy
11: Score the accuracy of the labelled data points
12: Compute the noise data points in the set of data sets.
13: Estimate the data points, noise, and cluster k points

Figure 6: DBSCAN enhancement algorithm

B. Affinity Propagation Enhanced Clustering Algorithm

The objective function is to select the point with the related messages. The algorithm passes the data points between the availability and responsibility of each cluster. In our case, the enhanced algorithm is employed to analyse the variation of each data point in each cluster. The homogeneity, completeness, and silhouette are integrated to further analyse the quality and distance of data points within the clusters [284–286]. The affinity propagation enhancement algorithm is explained in Figure 7.

AF enhancement algorithm
Input: data points similarities
Output: cluster assignments
Steps: 1: Classify the data sample and fit the model
2: Compute the matrix degree D of the data set
3: Compute a matrix containing the vector as a column
4: Compute a vector corresponding to the ith data point row of the vector
5: Cluster the indices of the centre of the cluster
6: Label the cluster data points in each cluster
7: Cluster each data point till the nth data point
8: Compute the homogeneity of the data points set accuracy
9: Compute the completeness of the data points set accuracy
10: Label each data point member in the class cluster
11: Plot all the data points cluster
12: Predict the data point clusters

Figure 7: AF enhancement algorithm

C. K-means enhanced clustering algorithm

K-means enhancement algorithm analyses and identifies the distinctive data points in the points of the given cluster. It selects the k centres randomly such that the value of k is fixed. Then, the Euclidean distance is generally applied to the data point to determine the distance between each data point and the cluster. When all data points are included in the cluster the first step is achieved by grouping the data points. In our case, the K-means is enhanced to analyse the variation of each data point in each cluster. Its main objective is to calculate the nearest distance between each point of the data object and the cluster centre [287-289]. However, the Euclidean distance from one vector $X = (x_1, x_2, ...x_n)$ to another vector $Y = (y_1, y_2, ...y_n)$ has the criterion $d = (x_i, y_i)$ or

$$E = \sum_{i=1}^{k} \sum_{x \in c_i}^{n} | x - x_i |^2 \quad \text{.....3.4}$$

The K-means enhancement algorithm is explained in Figure 8.

K-means enhancement algorithm
Input: data points similarity and a constant k
Output: cluster k centroids
Steps:1: Label the number of data points samples
2: Randomly specify the state of the centroid data point
3: Compute the degree of the matrix D in the data points
4: Compute the matrix containing the vector as columns
5: Compute the vector corresponding to the ith data point in the row of vector
6: Plot the value of the x and y data point random state
7: Specify the assigned number of the k of clusters
8: Randomly initialize the k centroids
9: Repeat process 8
10: Assign each data point to the closest centroid
11: Compute the new centroid (mean) of each cluster
12: Repeat 11 till the centroid positions do not change
13: Compute homogeneity of the cluster data points accuracy
14: Compute the completeness cluster data points accuracy
15: Estimate the number of cluster centroid data points

Figure 8: K means enhancement algorithm

D) Mean shift enhanced algorithm

The algorithm is a non-parametric clustering algorithm that does not require the prior definition of the cluster number and the restriction of the shape of the clusters such that the estimation of the density forms the basis of the mean shift. Each data point is referred to as a feature vector, and the algorithm can execute the gradient ascend on the local estimation density such that the convergence is reached [290-292]. The primary objective of the algorithm is to analyse the variation of each point in a cluster such as given the set of the ***n*** points $\boldsymbol{x_i}$ in a d- d-dimensional space where the value of i = 1...., n the estimation of the multivariate density with the kernel *K(x)* calculated in the point ***x.***

$$f'(x) = \frac{1}{nh^2}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \quad \text{.....3.5}$$

The mean shift enhancement algorithm is explained in Figure 9.

Mean shift enhancement algorithm
Input: Assigned data points clusters
Output: number of similarity cluster data points
Steps: 1: Assign data points to the cluster of their own
2: Compute the degree matrix D of the data points
3: Compute the matrix containing the vector as columns
4: Compute the vector corresponding to the ith row of the vector data points
5: Compute the centroids data points
6: Update the location of the centroid data points
7: Process is iterated and moved to the higher density region
8: Stop once the centroid reaches the maximum iterations
9: Computer the number of the cluster data points
10: Compute the homogeneity data points cluster accuracy
11: Compute the completeness of data point cluster accuracy
12: Specify the number of similar cluster data points

Figure 9: Mean shift enhanced clustering algorithm

3.4 Performance Evaluation

This section presents the results obtained from the selected clustering algorithms, and so on. It starts with the performance analysis of each clustering model, etc. Then the best-performing model is adopted for the framework design of SDN-WAN.

3.4.1 Simulation Objectives

This section discusses the clustering enhancement algorithms adopted to achieve the objectives of this research. However, to achieve our comparison objectives for the CPP, the clustering algorithms are simulated on the proposed data sets to identify the most accurate clustering patterns.

Moreover, the clustered data points are considered the nodes, and the controller is the centroid data point within each subdomain traffic operation. Again, the obtained results are interpreted and implemented within the given network. The choice of the SNDlib topology has been motivated by the fact that it represents the WAN size of the network library model that can be used for test instances to make a realistic network design model [293-296]. Table 6 discusses the clustering enhancement algorithms adopted to achieve the objectives of this research.

Table 6: Adopted clustering enhancement algorithms

Algorithm	Motivation
Mean shift enhancement algorithm	Input to the algorithm consists of the data points such as (multivariate, continuous feature vectors) and the scale to track non-rigid objects. The algorithm can handle image segmentation, detection of outliers and object tracking of objects.
DBSCAN enhancement algorithm	The DBSCAN algorithm is capable of finding the associations and the structures in the data points that are considered hard to find manually, on the other hand, can be relevant and very useful to find and achieve the patterns and predictions. It can specify how close the points should be to each other to be considered a part of a cluster such that if the distance between the two data points is lower or equal to the value of the epsilon, the points can be considered neighbours.
K means enhancement algorithm	Each observation belongs to the cluster with the nearest mean (cluster centres or centroid of the cluster) and serves as a prototype of the cluster. It can provide the capability to partition a set of data points into k sets or clusters such that the data points in each of the clusters tend to be near each other. However, the standardization is considered that all the distance-based needs data to be standardized such that none of the attributes. gets undue. It considers data that has a small number of dimensions, numeric and continuous. Provides document classification, delivery store optimization, identification of localities and analysis of data within clusters.
Affinity propagation enhancement algorithm	Due to the capability of the algorithm to provide the ability to take input as general nonmetric similarities, the features make it suitable to suit the exploratory analysis of data by using unusual measures of similarity. It is a data-driven analysis technique that can be able to partition the data points into different clusters and able to identify the exemplars among them by simultaneously considering all data points as possible as the exemplars and exchanging messages between data points until a good set of exemplars and clusters can emerge.

3.4.2 Topology and Tool

To maintain our realism, this section presents the adopted topological network used for the interpretation and implementation of the obtained results of the traditional and enhancement algorithms. The proposed SNDlib network represents the network library model that can be used for test instances and to make a realistic network design model [297]. David Hock et al. [202] discussed the framework for the resilient Pareto-based optimal controller (POCO) that can provide the operator of the network with all Pareto optimal placements. To achieve the objectives, the internet2 OS3E topology was adopted on more than 140 topologies in the Topology Zoo. Moreover, the authors considered four topologies together with the dynamic traffic matrices offered by the SNDlib library. Schutz et al. [298] proposed a k-cover-based model for the reliability-aware controller placement (RCP) problem in the SDN networks. To achieve controller placement, the authors optimized the number and placement of the controllers and the latencies of the primary and backup paths between switches and controllers. Moreover, the results obtained satisfied the low latencies.
Brain--D-B-E-N-C-A-N-N [297] is one of the 20 topological network cluster models in the SNDlib library. The adopted topology consists of 161 network nodes, 332 network connection links, and

14311 demands. Moreover, the decision to adopt the topological network model is based on the wide-area network scale offered by the SNDlib library as compared to other topologies. However, to represent a wide-area network model in SDN, the selected topology is a clear representation of the WAN nodes connected during traffic communication and the placement of controllers to achieve our objectives.

Moreover, the traditional and enhancement algorithms were running on the related toolset and the packages python-professional-5.0.3 python, pandas, scipy, matplotlib and Numpy were run on a Processor Intel(R) Core (TM) i7-10610U CPU @ 1.80GHz, 2304 Mhz, 4 Core(s), 8 Logical Processor(s), System Type x64-based PC, Installed Physical Memory (RAM) 16,0 GB, OS Name Microsoft Windows 10 Pro.

3.4.3 Model Parameters

This section discusses the types of variables used in the research.

A. Independent Variable

SDN WAN data packets $(x_1 , x_2 \ldots . x_n)$ are represented by the metrics data packet delivery position (DDP) and packet and controller location (P&CL) which are the inputs.

B. Dependent Variable

D In the context of this research, these values include the target variables (y_1 , y_2) of the SDN WAN data packets or independent variables. The outcome of the dependent variable is determined by the class value. Moreover, once the experimental results of the data packet independent variable change, a dependent result may also change such that $(x_1 , y_1)(x_2 , y_2) \ldots \ldots . (x_n , y_n)$.

3.4.4 Data Collection and Characteristics

To identify the best-performing models, this research employs the sci-kit-learn sample generators that have been widely used for real-world clustering of data sets, classification, and regression analysis [299, 300]. The scikit-learn consists of the different random samples of the generators that can be utilized for building the artificial datasets for classification, clustering, etc.

The make blobs can create the multiclass of the datasets that can be able to allocate the class to one or more of the normally distributed clusters of the points. On the other hand, classification specializes in the introduction of noise that includes the correlation, redundant features, the multiple Gaussian clusters of each class, and the transformation of the linear space feature [301–303]. Moreover, the data set was frequently used as the test case for various statistical and mathematical analyses and so on.

David D. Lewis et al. [304] proposed the use of the dataset in text categorization for a detailed understanding of the real-world constraints under which the data was produced. The authors described the use of the coding policy and the quality of the control procedures used in producing the RCV1 data.

The RCV1 dataset consists of records available as downloadable files for researchers. This dataset is frequently used as a test case for various statistical and mathematical analysis approaches, particularly in machine learning. It is considered one of the best datasets for performing clustering approaches, as well as classification and regression techniques, in data mining applications [299, 305, 306].

3.4.5 Performance Metrics

These metrics are used together to analyse and evaluate the accuracy of the classification and clustering of the data packets. It also explains how the variables and the algorithms achieve the objective of this research. The performance metrics are summarized in Table 7.

Table 7: Summarised performance metrics

Metrics	Definition
Outliers [307-309]	The analysis is the variability of the measurement for the experimental errors. However, the outlier can create some serious problems in the statistical analyses.
Adjusted Rand Index (ARI) [310-312]	Analyses the similarities between the two clusters of data points. The approach relates the accuracy to an unlabelled cluster of data points.
Adjusted Mutual Information (AMI) [313-316]	The metric considers independent value permutation of labelled points that cannot change. Moreover, the metric is the symmetric switching label which is true and predicts the label that can be able to return the same value score.
V measure [317-319]	Considered as decisions between homogeneity and completeness such that V is given by (1 + beta) * homogeneity * completeness / (beta * homogeneity + completeness)

3.4.6 Methodology

To achieve our objectives, we read the collection of data packets into a powerful Python toolkit (Pandas) as the training of the packets using each algorithm. Moreover, each algorithm performs the generation of the data packets and is clustered into various subdomains such that the function of the data packets: *f(data packets) = (cluster1, cluster2, ...cluster n)* is classified, clustered and predicted for various subdomains. Additionally, the central location of each subdomain is treated as the controllers' centroid data packets. The data packets that are outside of the range of each subdomain are labelled as outliers.

Then, the results obtained from the simulation are analysed and evaluated based on the data packets within each subdomain. The results obtained from the simulation are collected for comparison. The trained and tested data packets are evaluated based on the metrics: v-measure, adjusted rand index (ARI), and adjusted mutual information (AMI) etc. The outliers or misclassified data packets for each algorithm are identified and evaluated, and The optimal placement of the controller' centroid data packet of each subdomain is identified in the form of the matrix (x, y) and placed for the quality of placement, relationship and the distance of each data packets from the controllers' centroid data packets. The minimum location of the data packets within each subdomain is identified to further analyse the balance and quality of the data packets load to answer the question: how many controllers should be allocated to the data packets node etc., and the similarity between the data packets within each subdomain within each controllers' centroid data packet.

3.5 Results and Analysis

This section discusses the results obtained using the adapted algorithms. The performance results of each algorithm are analysed, interpreted, and implemented within the existing SNDlib topological network to answer the questions: how many subdomain nodes and optimal placement of the controllers are required to minimize the path between the pairs of the nodes and controllers in the given network? Moreover, the algorithms automatically determine the size of the network based on the number of generated data points and controllers' centroid data points clustered within the network and so on.

3.5.1 Network Size

To obtain the size of the network, automatic algorithms allow the dynamic change to the scale based on the trained data set within the network. However, when the datasets are generated, the algorithms automatically determine the number and the location of the data points assigned to the controllers' centroid data points. Moreover, the obtained (x, y) length and width scale results are based on the dynamic change of the network using the criterion.

$$\sum_j A_{ij} \quad \text{.....3.6}$$

where, D represents the number of the data sets generated by the algorithms such that A_{ij} is the values of the diagonal matrix in the form of (x_i, y_i). However, $\nu \in \mathbb{R}^{n*K}$ is considered as the matrix that contains the vectors $\nu_1, ...\nu_k$ such that $y_i \in \mathbb{R}^k$ the size of the network corresponds to the ith (length and width of the scalability.

3.5.2 Optimal Placement

To achieve the optimal placement of the controllers' centroid data points on the topological network, the algorithms randomly initialize and predict the k centroids based on the trained dataset and assign each data point to its closest controller's centroid data point within each subdomain. The minimization is then applied to compute the centroids of each cluster until all the required controllers' centroids data positions are achieved, such that

G (V, E, X) = maxNumofControllers, data packets, subdomains.....3.7

where, G denotes the scale of the network having the maximum number of controllers' data points, data points allocated to the closest controllers' centroids data point within each cluster, and the maximum number of the clusters with the minimum number of data points. Moreover, the consideration matrix D with D(i,j) = $\sum_{j} A_{ij}$ is used to determine the optimal placement of the controllers' centroid data point location such that the distance of the radius is considered minimal. Again, the similarity between the data points connected to each controller's centroid is identified to reduce the propagation latency such that the criterion is defined.

$$x_i^{t+1} - x_i^t < \varepsilon \quad \text{.....3.8}$$

where, x_i denotes the data point in each cluster and ε is the minimum distance from the controllers' centroid data point of each cluster. Moreover, the experiment was conducted, and the results obtained are recorded in Tables 8 – 15.

3.5.3 Results

The results obtained from the practical experiment are presented in this subsection. The algorithms considered were analysed, compared, and evaluated based on the results of the identification, classification, clustering, and prediction of the number of subdomains, controllers' centroid data points, the number of data points within each subdomain, etc. Then, the results obtained are interpreted and implemented within the Brain--D-B-E-N-C-A-N-N [297] for reassignment of nodes and controllers to achieve optimal placement, good latency, reliability, load, and scalability. Moreover, the results of the enhancement algorithms are compared and recorded.

i. *AP Enhancement*

Table 8 summarizes the results obtained from the affinity propagation enhancement algorithm to achieve the objectives of this research.

In our case, the obtained results from the practical experiment identified 7 controllers' centroid data packets as an ideal number in the x and y plane locations within the size of the network. Thus, optimal placement (x, y), number of data packets, and subdomains were identified within the network. The results obtained proved that employing seven controllers can provide the correct assignment of the data packets. 88% homogeneity, 88% completeness, 88% v measure, ARI 91%, and AMI 87% clustering accuracy were identified by the model during traffic communication in the network. Moreover, 10% of misclassified data packets were identified by the model as well.
Furthermore, based on the obtained performance results, this algorithm cannot be adopted to identify, classify, cluster, and predict the required number of data packets and controllers' centroid data packets, etc. The overall results of the affinity propagation enhancement are recorded in Table 8.

Table 8: Summarized results of affinity propagation enhancement algorithm

Criteria	**Cluster 1**	**Cluster 2**	**Cluster 3**	**Cluster 4**	**Cluster 5**	**Cluster 6**	**Cluster 7**
Number of Controllers	1	1	1	1	1	1	1
Number of data packets	23	23	23	23	23	23	23
Homogeneity	88%						
Completeness	88%						
V-measure	88%						
ARI	91%						
AMI	87%						
Size of the network (L * B)	2.5*2.5m						
Optimal placement (x, y)	0.866, 1.049	0.972, 1.055	1.033, 1.155	1.345, 1.235	1.023, 1.012	1.132, 1.002	1.421, 1.321
Epsilon (m)	1.94						
Outliers	10%						

*Ci = Cluster 1, 2,3., k

ii. DBSCAN Enhancement

Table 9 presents the results of the DBSCAN enhancement algorithm to achieve the optimal placement of the controllers' data packets and so on. The algorithm identified nine optimal placements of the controllers' data packets within each subdomain of the network. Several data packets within each subdomain were also identified. 100% homogeneity, completeness, v measure, ARI, and AMI were obtained for the accuracy of the clustering during traffic communication in the

network. Again, the controllers' centroid data packet locations were identified within the *x* and *y* plane networks. The subdomain radius was identified by the model without encountering misclassification or outliers of the data packets within the network.

Furthermore, based on the obtained performance results, this algorithm can be adopted to identify, classify, cluster, and predict the required number of data packets and controllers' optimal placement, etc. The results obtained proved that nine controllers can achieve a good balance and assignment of data packets and controllers placed in the *x* and *y* planes. The overall results of the DBSCAN enhancement algorithm are shown in Table 9.

Table 9: Summarised results of the DBSCAN enhancement algorithm

Criteria	**Cluster 1**	**Cluster 2**	**Cluster 3**	**Cluster 4**	**Cluster 5**	**Cluster 6**	**Cluster 7**	**Cluster 8**	**Cluster 9**
No of Controllers	1	1	1	1	1	1	1	1	1
No data packets	18	18	19	17	16	14	13	18	20
Homogeneity	100%								
Completeness	100%								
V-measure	100%								
ARI	100%								
AMI	100%								
Size of the network (L * B)	2.5*2.5m								
Optimal placement (x, y)	1.412, 0.751	0.711, 0.714	0.718, 1.422	0.411, 0.345	1.023, 1.011	1.234, 1.321	1.478, 0.566	1.723, 1.112	0.212, 0.333
Epsilon (m)	1.94								
Outliers	0%								

*Ci = Cluster 1, 2,3., k

Below, Figure 10 shows the advantage of the enhanced DBSCAN the accuracy based on the criteria: cluster formation, grouping of the points that are closely packed together, handling the noise, no prior knowledge of the cluster number, efficiency, and robustness of the outliers.



Figure 10. enhanced DBSCAN clustering algorithm

iii. k Means Enhancement

Table 10 summarizes the general results of the k means enhancement algorithm to achieve the objectives of this research. Eight controllers' data packets were identified as an ideal number for deployment within each subdomain of the network. Moreover, 45% homogeneity, 35% completeness, 55% v measure, 40% ARI, and 25% AMI were obtained by the algorithm for clustering accuracy during traffic communication. Again, the size of the network was also identified by the algorithm in the length and width of the network scale. Furthermore, there is a 30% misclassification obtained by the algorithm during the clustering of the data packets.

Based on the obtained performance results, this algorithm cannot be adopted to identify, classify, cluster, and predict the required number of data packets and controllers' centroid data packets, etc**.** The algorithm adopted can result in the worst placement of the data packets and controllers placed in the *x* and *y* within the size of the network. The overall results of the k mean enhancement results are summarized in Table 10.

Table 10: Summarised results of k-means enhancement algorithm

Criteria	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8
No of controllers	1	1	1	1	1	1	1	1
No data packets	18	10	12	20	24	20	30	21
Homogeneity	45%							
Completeness	35%							
V-measure	55%							
ARI	40%							
AMI	45%							
Silhouette coefficient	25%							
Size of the network (L * B)	2.5*2.5m							
Optimal placement (x, y)	1.412, 0.751	0.711, 0.714	0.718, 1.422	0.514, 1.566	0.234, 1.333	1.567, 1.892	1.977, 1.224	2.111, 0.345
Epsilon (m)	1.94							
Outliers	30%							

*Ci = Cluster 1, 2,3., k

iv. Mean Shift Enhancement

Table 11 summarizes the general results of the mean shift enhancement algorithm to achieve the objective of this research. 10 controllers' centroid data packets were identified as having optimal placement in the x and y plane networks. The results obtained proved that employing 10 controllers can provide the correct assignment of the data packets. 54% homogeneity, 55% completeness, 55% v measure, ARI 40%, and AMI 45% clustering accuracy were identified by the model during the traffic communication. Moreover, 20% of misclassified data packets were identified by the model as well.

Furthermore, based on the obtained performance results, this algorithm cannot be adopted to identify, classify, cluster, and predict the required number of data packets and controllers' centroids, data packets, etc. in the network. Again, load among the controllers can be experienced due to the lower accuracy percentages of the metrics. Thus, this algorithm can result in the worst placement of the data packet and controllers placed in the x and y within the size of the network. The mean shift enhancement algorithm results are recorded in Table 11.

Table 11: Summarized results of mean shift enhancement algorithm

Criteria	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8	Cluster 9	Cluster 10
No of controller	1	1	1	1	1	1	1	1	1	1
No data packets	22	15	18	19	10	12	16	21	15	13
Homogeneity	54%									
Completeness	55%									
V-measure	55%									
ARI	40%									
AMI	45%									
Size of the network (L * B)	2.5*2.5m									
Optimal placement (x, y)	0.856, 1.041	0.888, 1.084	1.555, 1.268	1.345, 0.456	1.0342, 1.456	2.102, 1.432	0.700, 1.321	2.211, 0.341	0.421, 1.242	0.673, 2.312
Epsilon (m)	1.94									
Outliers	20%									

*Ci = Cluster 1, 2,3., k

v. Affinity Propagation

Table 12 presents the results obtained from the affinity propagation algorithm to achieve the objectives of this research.

The results obtained from the practical experiment identified 10 controllers' centroids as data packets placed in the x and y locations within the size of the network. Moreover, the algorithm identified several data packets per subdomain for balanced communication. 25% v measure, 45% ARI, and 40% AMI were identified by the model during traffic communication in the network. Again, there was an 18% clustering misclassification of the data packets, and the optimal placement of the controllers' centroid was identified within each subdomain.

However, adoption of this algorithm can lead to the incorrect assignment of the correct clustering of data packets and controllers in the network, etc., and can result in the worst placement within the size of the network. The overall results of the AP are recorded in Table 12.

Table 12: Summarized results of affinity propagation algorithm

Criteria	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8	Cluster 9	Cluster 10
No of controller	1	1	1	1	1	1	1	1	1	1
No data packets	20	23	13	18	19	22	10	5	28	6
Size of the network (L * B)	2.5*2.5m									
Optimal placement (x, y)	1.671, 0.299	1.054, 0.995	0.538, 1.430	0.300, 0.044	0.344, 0.936	1.138, 0.362	1.299, 0.592	0.404, 1.799	1.624, 1.838	2.866, 2.273
Epsilon (m)	1.94									
V-measure	25%									
ARI	45%									
AMI	40%									
Outliers	18%									

*Ci = Cluster 1, 2,3., k

vi. DBSCAN Propagation

Table 13 presents the results obtained from the DBSCAN algorithm to achieve the optimal placement of the controller and so on. The classified, clustered, and predicted results identified nine controllers as the ideal number of controllers required in the network.

Again, nine optimal locations and subdomain clusters were identified with multiple subdomains. Despite the behaviours of the algorithm, the obtained metrics resulted in good performance in classification, clustering, and prediction on the proposed data set. 92% v measure, 80% ARI, and 90% AMI were identified by the algorithm during the clustering traffic communication of the data packets in the network. The algorithm was able to identify 0% of misclassified data packets within the network.

Thus, due to the obtained results, the algorithm can be adopted as the right algorithm to determine how many controllers should be deployed in the network. As a result, network performance and the sharing of information are improved. The overall results of the DBSCAN algorithm are recorded in Table 13.

Table 13: Summarised results of the DBSCAN algorithm

Criteria	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8	Cluster 9
No of controller	1	1	1	1	1	1	1	1	1
No data packets	12	16	21	15	13	10	19	18	22
Size of the network (L * B)	2.5*2.5m								
Optimal placement (x, y)	0.184, 0.331	1.412, 0.751	0.711, 0.714	0.718, 1.422	0.411, 0.345	1.023, 1.011	1.234, 1.321	1.478, 0.566	1.723, 1.112
Epsilon (m)	1.94								
V-measure	92%								
ARI	80%								
AMI	90%								
Outliers	0%								

*Ci = Cluster 1, 2,3., k

vii. K Means

Table 14 presents the results obtained from the k-means algorithm during the generation of data packets to achieve the objective of this research. The algorithm identified nine controllers' centroid data packets as the optimal placement in the x and y locations within each subdomain. The number of data packets within each subdomain was also identified in numbers. 25% v measure, 45% ARI, and 55% AMI were identified during the data packet communication in the network. However, 18% of misclassified data packets were identified during traffic communication in the network. Furthermore, the location of the controller was identified in the x and y-plane subdomain networks.

Furthermore, based on the obtained results, percentage accuracy in terms of classification, clustering, and prediction. However, the adoption of this algorithm can lead to the incorrect location of the controllers in the network. Thus, this algorithm cannot be adopted to determine how many controllers, etc. in the network. The overall results of the k means are recorded in Table 14.

Table 14: Summarised results of k means algorithm

Method	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8	Cluster 9
No of controller	1	1	1	1	1	1	1	1	1
No data packets	13	15	21	16	12	10	19	18	15
Size of the network (L * B)	2.5*2.5m								
Optimal placement (x,y)	1.057, 0.43	0.734, 1.57	1.079, 1.56	0.514, 1.566	0.234, 1.333	1.567, 1.892	1.977, 1.224	2.111, 0.345	2.111, 0.345
Epsilon (m)	1.94								
V-measure	25%								
ARI	45%								
AMI	55%								
Outliers	18%								

*Ci = Cluster 1, 2,3., k

viii. Mean Shift

Table 15 presents the results obtained from the mean shift algorithm to achieve the objective of this research. The algorithm resulted in 8 controllers' centroid data packet subdomains, each having their optimal placement as the required number for deployment. Moreover, the selection of the optimal placement was considered the best placement on the network to achieve the best latency, and so on. Moreover, 55% v-measured, 40% ARI, and 45% accuracy were identified during traffic communication. However, the adoption of this algorithm can result in poor quality, incorrect controller placement, latency, and so on.

Therefore, this algorithm cannot be considered the right algorithm to determine how many controllers should be deployed in the network. The overall results of the mean shift are presented in Table 15.

Table 15: Summarized results of mean shift algorithm

Criteria	**Cluster 1**	**Cluster 2**	**Cluster 3**	**Cluster 4**	**Cluster 5**	**Cluster 6**	**Cluster 7**	**Cluster 8**
No of controllers	1	1	1	1	1	1	1	1
No data packets	15	18	19	10	12	16	21	15
Size of the network (L * B)	2.5*2.5m							
Optimal placement (x, y)	1.528, 0.139	0.888, 1.083	1.345, 0.456	1.0342, 1.456	2.102, 1.432	0.700, 1.321	2.211, 0.341	0.421, 1.242
Epsilon (m)	1.94							
V-measure	55%							
ARI	40%							
AMI	45%							
Outliers	40%							

*Ci = Cluster 1, 2,3., k

3.6 Discussions

This section discusses the comparison of the results obtained from the performance analysis of the enhancement and traditional clustering algorithms. Each model discussed operates according to its characteristics, traits, and so on. However, some of the algorithms require specifications to identify the number of clusters, optimal placement, etc. within the network. Moreover, the generalization performance based on the identification, classification, and clustering effectiveness and efficiency of each algorithm were identified, recorded, and drawn for the following conclusion.

Table 16: Summarised discussion results of the selected algorithms

Enhancement algorithms	
Criteria	**Discussions**
AP enhancement algorithm	The algorithm performance metrics (v-measured, ARI, and AMI) can achieve 80% + percentage accuracy on the datasets. It can identify 7 controllers' centroid data points as an ideal number for the implementation within the topological cluster in the x and y plane locations within the size of the network. Seven subdomains were identified within the network. Can be interpreted and implemented for reassignment to identify, classify, cluster, predict, and deploy the required number of nodes and controllers within the network.
DBSCAN enhancement algorithm	The algorithm can result in some increased percentage accuracy metrics performed on the datasets and produce a quality of 100% among each metric due to the highest relationship and similarities of the datasets. Nine optimal locations and subdomains were identified within the network as an ideal number for the implementation. Due to its high accuracy, it can be interpreted and implemented for reassignment to identify, classify, cluster, predict, and deploy the required number of nodes and controllers.
K-means enhancement algorithm	The performance metrics v-measured, ARI, and AMI resulted in some decreased percentage accuracy (- 50%) on the datasets. Thus, 8 controllers were not considered an ideal number for deployment. Due to its poor performance, implementation can result in the worst placement of the nodes and controllers placed in the x and y planes within the size of the network.
Mean shift enhancement algorithm	The algorithm can achieve poor performance on the metrics ARI, v-measure, and AMI with some decreased percentages (- 60%) of accuracy on the datasets. Due to its poor performance, load among the controllers can be experienced based on the lower accuracy percentages of the metrics. Can also result in the worst placement of the nodes and controllers placed in the x and y within the size of the network during implementation.
Traditional algorithms	
Criteria	**Discussions**
AP algorithm	It can identify 10 controllers' centroid data points placed in the x and y locations within the size of the network. V-measured, ARI, and AMI can result in some decreased percentages—50%—of poor performance in classification, clustering, and prediction. Implementation can lead to an incorrect reassignment of the nodes and controllers and result in the worst placement within the size of the network.
DBSCAN algorithm	Metrics can result in good performance in classification, clustering, and prediction. The network controller can encounter overloading and worst-case latency due to one domain of traffic flow between nodes. Cannot effectively and efficiently determine how many controllers to deploy in the given network, etc. Degrade the network performance and improve the sharing of incorrect information.
K-Means algorithm	V-measure, ARI, and AMI decreased below 50% due to the incorrect classification, clustering, and prediction of datasets. Lower percentage accuracy in classification, clustering, and prediction can be identified. It can lead to incorrect results for the required number of nodes and controllers when employed in the given network.
Mean Shift algorithm	V-measured, ARI, and AMI percentage (-60%) accuracy are lower due to classification, clustering, and prediction. It can result in poor quality or incorrect reassignment of the placement of controllers and nodes during the implementation.

The results obtained proved that some algorithms had difficulties identifying, classifying, and clustering the optimal placement of the data points and controllers' centroid data points, while others could capture the relationship of the data points with the highest predictive accuracy and consistency of classification, clustering, etc.

Due to the performance accuracy results, the algorithms that performed well include the enhancement of DBSCAN on the proposed data packets. The algorithms did not experience some challenges in clustering the data packets on the network subdomains. The enhanced DBSCAN algorithm identified the highest clustering accuracy percentages on the data packets as compared to other proposed algorithms. The algorithms were able to identify and cluster the data packets with a zero-rate percentage misclassification. Moreover, the algorithms that performed moderately on the data packets include the AP and mean shift algorithms. These algorithms were able to classify and identify the trend in the data packets; however, they also experienced some challenges in identifying the relationship of the data packets within each network cluster and resulted negatively in some identified metrics.

K means clustering algorithms, both traditional and enhanced, performed poorly on the proposed data packets and resulted in poor performance on the metrics. The algorithms were not able to identify and cluster the patterns of the data packets. Moreover, the algorithms encountered some misclassification challenges and performed poorly on the data packets to identify the controllers' centroid data point's optimal locations efficiently, etc. Thus, misclassified data packet errors were identified as the result of the inability of the algorithms to adapt to the data packets.

Both the traditional and enhancement algorithms indeed have advantages and disadvantages during the clustering of the data packets. However, the decision was informed by the fact that the algorithm has the highest predictive power compared to other algorithms, as demonstrated by the results. We observed the consistency of producing good results in terms of homogeneity, completeness, v measure, ARI, AMI, and others. Moreover, the concepts were achieved based on the traditional and enhancement models for categorizing data packets according to their classification as normal clustered data packets and abnormal clustered data packets. Moreover, outperformed algorithms (both traditional and enhancement) were observed in terms of classification, clustering, and prediction accuracy to achieve the optimal placement of the controllers' centroid data packets and the required number of subdomain data packets.

3.7 Chapter Summary

This chapter presents a literature review based on classification and clustering approaches in the SDN WAN, the traditional, and enhancement algorithms adopted to identify their effectiveness and efficiency for the placement of controllers in the network. The overview of various clustering strategies selected by traditional and enhancement algorithms was discussed in detail. Moreover, traditional and enhancement algorithms were presented and discussed for comparison. The algorithms were evaluated for classification accuracy, efficiency, effective analysis, and so on. Then, based on the obtained results from the simulation, the best-performing algorithm was selected, interpreted, and implemented to identify, classify, cluster, reassign, and predict the required number of nodes, controllers, etc. within the given network. The objective was to choose the best-performing algorithm for the development of SDN CPP WAN framework settings.

Chapter 4

Literature Review: *Intrusion Detection*

4.1. Chapter Outline

This chapter presents a review of the related works on SDN-based IDS. It starts with a brief introduction and is followed by an overview of IDS, etc. in the SDN. Again, it provides information on ML algorithms to detect intrusions in the network. Lastly, the methodology and performance analysis of the selected algorithms were explained in detail, as were discussions and a summary of the chapter.

4.2. Introduction

Security of the network is considered a large and existential challenge faced by every network. Increased security and malicious threats in network environments are in the form of Trojan attacks, other viruses' intruders, and attackers, which degrade network performance and communication channels. People may try to intrude on or attack the system to violate the integrity and confidentiality of the information stored [107, 320, 321]. Though various techniques such as firewalls, anti-viruses, etc. can be used as checkpoints of security in the network environment, these are not enough to protect the entire system and others. However, organizations have introduced and implemented IDS, ID, and prevention systems (IDPS) to achieve a balance and strengthen the security of the network from malicious and illegal access. IDPS has an impact and plays an immense role in attacks by protecting the system's information against various types of attacks [322, 323].

Moreover, many historical events have proved that the IPS approaches alone cannot satisfy the requirements of the system, such as authentication and encryption. Once such security measures are taken, the firewall, authentication, access control, and so on can block certain actions, and some users can be allowed access to other resources. IDS do not imply prevention; only intrusion is detected and can be reported to the operators. Certain IDPS approaches can be taken to react to the detection of unauthorized action occurring, such as to contain or stop the damage, e.g., to terminate the network connection [328–330]. The performance of ID can be enhanced accordingly with feature selection [331-334]. However, the primary challenge of the algorithm is its performance, which can be modified and raise false positive alarms.

4.3. Intrusion Detection Systems

An IDS is considered the node or the application of the software that can be able to monitor the network and other systems for malicious damage or activity and the violation of policies. Any detected malicious damage to the resources can be reported to the administrator or collected centrally with the security information and event management (SIEM) [335, 336]. SIEM's objective is to combine the output from multiple sources and then use alarm filtering to distinguish malicious activities from false alarms. However, a good explanation is given by the route [337-340].

Moreover, IDS is taken as a passive monitoring system since its objectives and main function are to alarm the user about the malicious activity taking place and not prevent it. IDS has a variety of advantages and functions that include the following: the system and the users are monitored; the identification of malicious activities based on statistical analysis [341, 342], the recognition of the known attack in the system activity; the correction of the configuration systems of errors; and the installation of the operating traps to record the related information about the intruders. Also, the auditing of the configuration of the system includes the vulnerabilities and misconfigurations, the management of the audit trails and highlights of the violations of the user and the policies, and the assessment of the integrity of the critical system and the data files [343-345].

Many factors can be considered to analyse the IDS, etc. However, the alarms and so on are considered critical aspects of an IDS framework. Moreover, IDS' prediction can be evaluated by its results and the ability to classify the correct events to be attacks or malicious attacks [346-353]. SY Wu *et al.* [354] presented a comparison of the accuracy, rate of detection, and alarming rate of possible attacks such as U2R, R2L, DoS, and Probe using the C4.5 and the SVM models. The results have proven that the C4.5 can outperform support vector machines in the detection of attacks compared to the false alarm rate.

4.3.1 IDS Approaches

Based on the methods of detection, are discussed below:

A. Anomaly Detection

Anomaly detection is the method that is concerned with the identification of events that are the results of abnormal attacks concerning the normal system. Its advantage is its capability to identify new and previously unseen attacks. This system establishes a baseline model that can be used to represent the normal behaviours of the system and abnormal events can be established. Recently, anomaly detection has served mainly as a tool to filter out noise, which could impair the training of

ML algorithms [355, 356]. It plays perhaps a more noble role, as the ML community has realized that anomalies are often associated with interesting or rare events. Applications of anomaly detection are vast, encompassing network security and maintenance, data storage, finance, and medicine. Moreover, ML can be used to analyse the event with consideration and classify abnormal or normal on a certain threshold of the range [357–359] and so on.

There are several proposals in this aspect. Ribeiro *et al.* [360] presented anomaly-based approaches that use ML algorithms over a continuous stream of data to detect intrusion in SDN scenarios. To achieve the intrusion, they characterized the anomalies by performing an analysis of the type of DDoS attack classified as an infrastructure attack by considering the impact of the depletion of resources and bandwidth. The experiment performed showed 99% accuracy, 99% recall, etc. for DDoS attacks on normal traffic. The results from the simulation showed the proposed technique can detect and locate the occurrence of the failure with a precision that is high and low overhead compared to the statistical approach, Naïve Bayes, and decision ML. The anomaly-based architecture is shown in Figure 10.

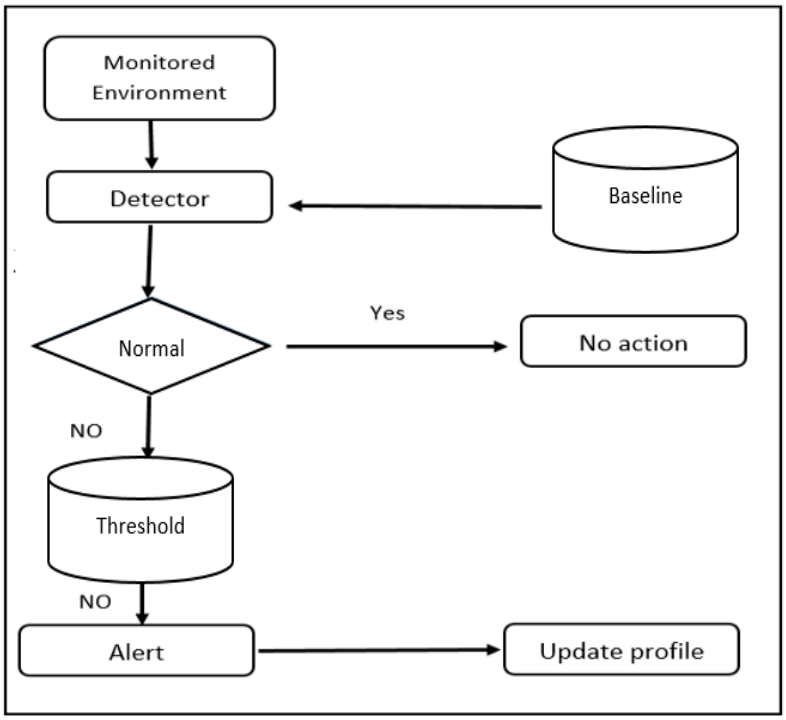


Figure 10: Anomaly-based architecture

B. The Signature Detection

Signature detection or misuse detection schemes can store the patterns of the sequence, the signature of the attack, and so on in the database system such that, in the event of an attack or intrusion, the IDS can perform signature matching with the stored signature in the database. Once

there is a successful matching of the signatures, the system generates an immediate alert. In misuse detection, the IDS analyses and gathers information for comparison purposes in large databases of the attacked signatures, and in the event of an attack, the IDS searches for the specific intruder already documented. The importance of misuse detection is centred on the use of the system. However, the architecture of the misuse is not willing to detect the attackers that are not represented in the knowledge-based system [362–364]. Moreover, the known and subtle variations may intend to evade the system if the framework is unorganized. Therefore, this can rely on the right design for baseline experts [365–367]. The signature-based architecture is shown in Figure 11.

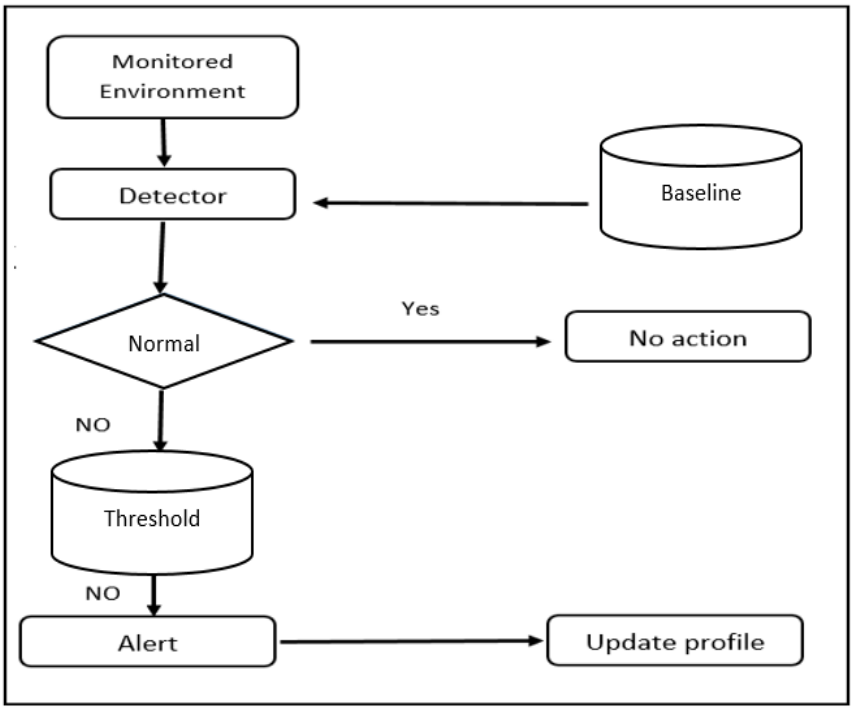


Figure 11: Signature-based architecture

Some of the works in this aspect are as follows: Muhammad *et al.* [368] proposed an approach for the identification of malicious behaviour on SDN switches using ML techniques. They proposed the system based on the signature attack, which can detect misbehaving switches that drop and swap the data packet due to their malign intent rather than the failure link. Every attack contains some signature, and the attacks are identified by the predefined signature and the behaviour. The three attacks that were identified were DDoS, port scanning, and traffic diversion attacks. The simulation results demonstrated the effectiveness of the ML and illustrated the detection mechanism that can detect attacks and achieve high detection accuracy with a low false-positive rate. In the same vein, Raja Majid *et al.* [369] proposed a fast and effective entropy-based DDoS detection approach. The authors generalize the entropy calculation by combining Shannon and Renyi to identify distributed features of the DDoS flow traffic. Traffic data were collected on the data plane with signature-based

snort detection, and the collected traffic flow was analysed. The results show that the technique achieved a relatively higher detection accuracy of 94% with 6% false-positive alerts.

C. Stateful Protocol Analysis Detection

In stateful protocol analysis detection, the deviations from the protocol state are identified to make a comparison of the observed events, such as the predetermined profiles that are generally accepted and the definitions of benign tasks [370]. Nazrul Islam *et al.* [371] implemented the model over the network. The process was controlled without the assistance of the hardware. The focus was on firewall functionalities and the available bandwidth for the traffic flow. ML techniques were also applied to build the infrastructure using the dataset, and the results show random forests achieved different communication results in the network. Similarly, Elham Ariafar *et al.* [372] presented an IDS framework based on the k-means and decision tree (DT) while using the genetic algorithm (GA) to optimize parameters such as the number of clusters, max runs, and confidence. The results with the NSL-KDD 2009 dataset revealed about 99% of the detection rate (DR) and 1.8% of the false alarm rate (FAR) achieved, demonstrating the improvement compared with the new ensemble clustering (NEC) method. The stateful protocol analysis-based methodology architecture is shown in Figure 12.

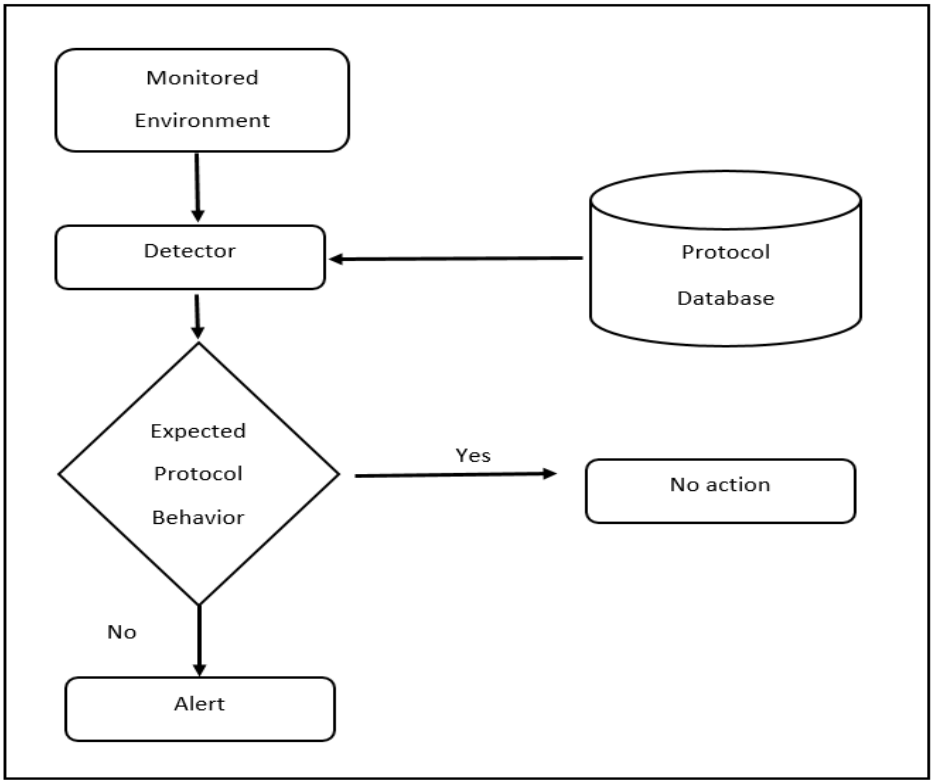


Figure 12: Stateful protocol analysis-based methodology architecture

Based on the information gained from the literature on the methodologies, Table 17 presents the pros and cons of the IDPS methodologies.

Table 17: IDPS methods pros and cons

Criteria	Pros	Cons
Anomaly [373]	Efficient detection of new and unforeseen malicious attacks. Independent on OS. Facilitation of detection of privilege abuse.	Profile accuracy is weak for observed events that are changing constantly. Not available for rebuilding profile behaviour. Fail the alerts triggering at the right time
Signature [374]	Effective to detect known attacks. Detailed contextual analysis.	Fail to detect unknown attacks and different known attacks. Difficult to keep the signature updated. Knowledge is time-consuming.
Stateful protocol [375]	Track the protocol state. Differentiate change of command sequence	Fail to inspect the attacks for the benign behaviour protocol.

4.3.2 IDS Classification: Passive and Active Response

This section discusses the classification of the IDS in the network and its approaches such as active and passive response approaches.

1) ***Passive Response*** identifies the potential of the security breach, logs the information, and so on, and its objective is not to actively stop the intrusion but to log the attack and inform the network operator with an email method, etc. It allows the user to use the product of the passive response in the decentralized active system response, such that reports are sent to the central console, which actively controls the involved system and the network devices [376-378]. This is achieved by learning and using other information gained from the system with the intent not to affect the system's resources, such as by the opponent that gains access to obtain the information that is transmitted [379, 380]. Atul Kumar et al. [381] proposed a novel ultra-lightweight authenticated framework for the passive tag that uses the XOR and the rotation operations with three phases, such as tag identification, mutual authentication, and pseudonyms. These phases identify the tag and involve updating the key between the tag and the reader. The scheme analyses the situation in terms of the communication cost between the tag and the reader and the storage for the passive tag.

2) **Active Response** is considered an attack that intends to manipulate the resources and affect the operations. It modifies the streams of data and creates a stream that is false with the objective phase that is additionally related to the attackers. Moreover, the approach can block the source address of the malicious attacker, enable TCP session resetting, and so on [382–385]. Leila Mechtri et al. [386] proposed a security model denoted as MSIS-R-SA that addresses common security threats such as a black hole, a grey hole, and selfish behaviour attacks. It provides

automatic responses based on the intrusion's measured severity level in both the single and multiple intrusion scenarios while minimizing the negative consequences of adopting a typical static response. The results show intrusion response effectiveness for both single and multiple intrusions.

4.3.3 IDS Architecture

In the architecture of the network, IDS is divided into categories that include the host, network, and hybrid IDSs. In detail, the system that can monitor files of the important operating system is referred to as the HIDS example, and the analysis of the network's incoming traffic is referred to as the NIDS process. These categories are discussed as follows:

A. Host IDS

A host-based intrusion detection system (HIDS) is considered the application of software that can be installed on a system to monitor the activities on the local system. The main goal is to communicate with the installed operating system. However, it can be based on the knowledge of intrusion detection [387–389]. Moreover, there are multiple open source HIDS security systems that are free host-based IDS, such as OSSEC, which can perform log analysis, check for integrity, monitor the window registry, alert the time-based, and so on. One example of HIDS is Tripwire, which is open-source software. Again, a smartwatch is also considered the HIDS, which is the preemptive hacker tool of the Defense. The software can monitor the computer's Windows system such that changes are watched, detected, and reported immediately, and so on [390, 391].

Robin Gassais *et al.* [392] proposed a novel host-based automatic approach for intrusion detection. The authors combined user space, kernel space information, and the ML learning approach to identify the various kinds of malicious attacks on smart devices. To achieve the intrusion, the tracing solution was proposed to automatically get the behaviours of a device and process the data into a numeric array to train several ML algorithms to raise the alert whenever an intrusion is found. Several ML algorithms were implemented, including deep learning, to achieve high detection capabilities while adding little overhead to the monitored devices. Moreover, the solution was tested within a realistic automatic system with an actual threat. David Wagner *et al.* [388] proposed a framework for mimicry attacks that allows the sophisticated attacker to cloak the intrusion to avoid detection by the IDS.

The authors developed a theoretical framework for the evaluation of the security of an IDS against mimicry attacks. The authors demonstrated approaches to breaking the security of IDS with their

methods. The results obtained from the simulation confirmed the power of the mimicry attacks by giving an example of an attack on a concrete IDS implementation.

B. Network IDS

The network intrusion detection system, or NIDS, is considered the approach that monitors the packets in the network at the given link. The type of IDS can be dedicated to the hardware appliance or the application that is capable of running on the computer that is attached to the network [393]. For this reason, the packets that are NIDS monitored are not addressed to the NIDS host. However, the system can be impervious to malicious attackers. In general, it has a shortcoming that is oblivious to the local root attack [394, 395].

The protocol in the network can present multiple problems to the NIDS, which can result in crud [396-398]. Crud can appear in the stream of the network, such as errors in the implementation of the network, network link disconnection, and so on. Multiple open-source platforms are proposed, such as BRO [399], which is considered the Unix-based monitoring of the network framework. It is considered to be used for NIDS and measurement collection. etc Johari Abdullah *et al.* [400] proposed the concept of the IDS and provided the taxonomy of the notable ML and DL techniques adopted for the design of network-based IDS (NIDS). The authors discussed the challenges, including the limitations of proper solutions. The authors proposed the framework for the ML and DL-based NIDS to provide terms for the methodological approaches and so on. Md. Arafatur *et al.* [401] proposed the semi-distributed case that consists of the parallel model that runs on the edge side and area applied for the side-by-side feature selection, which was followed by the single multi-layered perceptron and the classification. Based on the study, the results obtained from the simulation and the numerical statistics demonstrated the promise of the method proposed with comparable detection accuracy to the superior centralized IDS, as well as exemplifying the inheritance trade-off between accuracy and building time performance.

C. Hybrid Based IDS

In the IDS, there is a current trend that combines the methods of network intrusion detection systems (NIDS) and host-based intrusion detection systems for the design of hybrid systems. The approach of HIDS is considered flexible and has the capability of increasing security levels.

The process of HIDS integrates the sensor IDS of the location and makes reports about the attack that are aimed at the segment of the whole network [402-404]. Yang *et al.* [405] proposed a method to increase the SVM algorithm to achieve improved intrusion detection. It was applied to the hybrid with a modification of the kernel function to deal with network intrusion detection. Krzysztof

Zaraska *et al.* [406] considered the model that provides sensors monitored by many kinds of events in the network. The advantage of the Prelude is that it can alert the collection to the scale of the wide-area network, such as the coverage of the city, country, and so on. Sun *et al.* [407] also proposed the deep learning-based intrusion detection system (DL-IDS), which uses the approaches called hybrid networks of convolutional neural networks (CNN) and long short-term memory networks (LSTM) to extract the spatial and temporal features of the network traffic data and to provide a good intrusion detection system. The authors mentioned the challenges to reducing the influence of the unbalanced number of samples of different attack types in the model training data samples on the performance of the model to improve its robustness. The results obtained from the simulation demonstrated a reliable intrusion detection dataset that covers all common intrusion and cyberattack scenarios and is capable of being updated as new threats emerge. The hybrid-based architecture is shown in Figure 13.

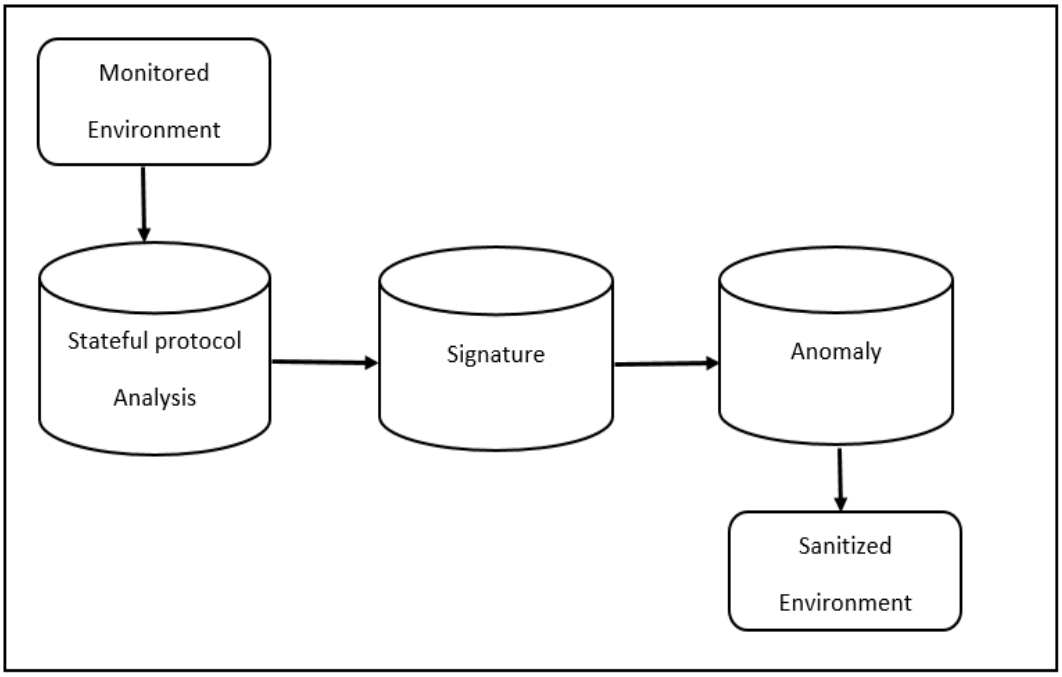


Figure 13: Hybrid Based Architecture

In this hybridized approach shown in Figure 13, the anomaly and signature-based techniques are integrated such that can result in quick detection of malicious attacks [408, 409]. Moreover, the hybridized approach is far better than an individual approach or technique. It takes advantage of multiple techniques such as a few soft computing approaches that can be integrated such that a hybrid approach can be used for detecting any type of attack.

Furthermore, several authors utilized the integration of the fuzzy genetic algorithm with the signature-based intrusion detection approaches to identify both external and internal attacks.

4.3.4 IDS Decision Making

This subsection presents the different decision-making techniques in IDS.

i. Co-operative IDS

Is considered the decision method that is widely used for networks. Due to these reasons, the circumstances, such as the existent protocol, the services and applications, and so on, conclude that MANET is considered a cooperative network that is friendly and does not accommodate the security system. It is proven that if the node can detect an anomaly, the architecture can trigger and produce the global ID with the neighbouring nodes [410–412]. Santosh *et al.* [413] proposed the idea of building a security IDS based on a cooperative scheme that can detect intrusion in MANETs using the basic game theory concepts. Similarly, Huang *et al.* [414] proposed an active intrusion monitor system (AIMS) to provide a cooperative detection model that is flexible and a platform that is effective for ID. Juneja *et al.* [415] proposed a novel two-fold mechanism to detect attacks by integrating cooperative trust management and the mitigation strategies of the attack, selecting the secure route, and policing the gradient through deep reinforcement learning. Results obtained show an improvement in the detection ratio by 20% as compared to the forecasting methods that were applied for detecting the vampire node's behaviours. K. Pazhanisamy *et al.* [416] also proposed a technique using a single motion without the implementation of multi-user cooperative motion and an efficient multi-user realization method (EME) scheme that is based on the topographical location of the data of static users. The results show the efficiency of the EME with less communication time when compared to the other methods. In the same vein, Deepaa Selva *et al.* [417] proposed intelligent selection feature methods and intrusion detection (ISTID) on the web-based on neuron genetic algorithms as software intelligent agents.

ii. Autonomous IDS

Autonomous IDS can be defined and considered a software tool that performs security function analysis and monitoring at the host. Agents are independent entities that can be removed and added to perform the reconfiguration without having to notify and restart other components and IDS. The advantage of the agent is that the individual tests on their own before being introduced to the complex environment. Due to this reason, the nodes of the network can be determined by the gathering of evidence and the criteria that involve the anomalies and activities of the intrusions [418–42]. Andalib *et al.* [423] also proposed an IDS that can reduce the amount of manual interaction and the needed knowledge to accept the performance under zero-day attacks. It uses learning techniques such as gated recurrent units (GRU), convolutional neural networks, and

random forests as an ensemble technique. Results obtained showed the potential of the system to operate with very low technician interaction under zero-day attacks.

Similarly, Alaa Alhowaide *et al.* [424] proposed the framework for the traffic heterogeneity levels and time window size of several classification methods to justify the detection selection model. It was evaluated using the BotNet-IoT dataset, and the results show the effectiveness of the classifier as its performance can significantly affect heterogeneity traffic, while the boosted tree showed the best performance. Ding-Chau Wang *et al.* [425] also presented the intrusion detection attack defence game of the Internet of Things (IoT) system, which considers autonomous IoT devices to solve the problem. The authors developed the analytical model to determine the conditions under which the malicious nodes do not have incentives to perform the attack in the ID attack defence game. Lastly, Maxine M. *et al.* [426] proposed a method for improving the realism and the ecology of AI for autonomous cyber defence by exploring the potential to use inverse reinforcement learning (IRL) to gain insight into the attacker's actions, the utilities of the actions, and the ultimate decision points that cyber deception could thwart.

This research is positioned to adapt and utilize network anomaly detection [427–432]. The IDS is placed along the network subdomains or boundaries to monitor all the traffic on that subdomain or cluster. The network anomaly detection method will be deployed at a strategic point in the network infrastructure such that it can capture and analyse data packets to detect known malicious attacks by performing a comparison of the patterns of detection of illegal activities and scanning the traffic for anomalous activity. It is intended to capture the data packets passing through the communication medium.

4.4 Network Attacks

This section discusses the attacks on the network, and these attacks are categorized as denial of service (DoS), penetration attacks, and scanning attacks.

4.4.1 Denial of Service Attacks

In the network, this type of attack intends to initiate the action and degrade network performance for the legitimate and authorized user with access. This type of attack is familiar on the internet because of the collection of hosts continually attacking the web servers with fake requests. Due to the results, economic damage can be experienced to the business of eCommerce by denying access to customers. However, there are various kinds of DoS attacks, such as flooding DoS, DDoS, and flaw exploitation DoS attacks [43–437].

i. Flooding DoS Attack: A flooding attack is considered an attacker that can send multiple requests to the target that it can handle. The flooding attack can potentially degrade performance and direct the DoS to other users in the network [438–442].

ii. DDoS: DDoS is considered the version of the attack DoS that can use the pool of large hosts to target the provided victim. A hacker, referred to as the botmaster, can initiate the attack on the DDoS to exploit some of the vulnerabilities in the computer system to take control and make the DDoS the master. However, with the attempt of a single command, the malicious attacker can be able to make instructions for attacks for specific attacks [443–447].

iii. Flow Exploitation DoS: Considered an attacker that can be able to exploit the flaw of the software by slowing and exhausting certain resources. Ping of Death (POD) can be formed by sending a malformed malicious ping to another computer. Ping is considered to be 64 bytes or 84 bytes in consideration of the IP header [448–453].

4.4.2 Penetration Attacks

A penetration attack occurs when the attacker gains unauthorized access to the system and performs modifications to the state of the system and so on [454–456]. For this reason, the penetration test can be used to identify and improve the performance of the system, whether it is under attack or when the defence mechanism is sufficient. The penetration target test can be identified as the white box, which can provide only the information that is considered basic. Moreover, there are a variety of related attacks [457–460]:

i. *User to Root (U2R):* This is explained as an attack that can gain access to the system. This can happen when the attacker starts gaining access by exploiting the vulnerabilities in the application or operating system. The process of the attacker can start by providing access to the account of the normal user on the system, such as gaining access to the password and so on, to exploit multiple vulnerabilities to gain root access to the system. Moreover, there are common attacks of the same class, such as buffer overflow, Perl [461-463], and so on.

ii. *Remote to User:* Can gain unwanted routes as a remote system and user account. R2U can launch the attacker by sending the packets to the system of the network to exploit the vulnerabilities of the local access as the operator of the machine. There are commonly known attacks on R2U, namely: Dictionary, Ftp_write, Guest [464–466, and so on].

iii. *Remote Disk reading is* considered the damage that can reach the target of one of the computer networks. For this reason, the computer system is not affected remotely. However, it gains access by finding access to the vulnerable points in the security of the network to access the system [467, 468].

iv. *Remote to Root (R2R):* In this type of attack, the user that is across the network can gain complete access to and control of the system [469–471].

4.4.3 Scanning Attack

In the network, the scanning attack is considered the attack that can send different packets in the network or system to probe for the malicious vulnerability that can be exploited. The packets (probes) are sent, and the target should respond to the request such that they are analysed to respond to the characteristics based on the target of the system to check the vulnerabilities. Hence, the scanning attack can identify the potential victim [472-474].

However, the network scanner, malicious scanner, and so on can be used to yield the topology of the network and so on. Moreover, once the victim of the system is identified, the attacker can have a chance to penetrate in a specific way. A scanning attack is again considered a legal activity, given the number of applications that are employed for scanning. There are multiple well-known scanning applications, such as web engine search and so on. Again, there is also malicious scanning that can include the vertical and horizontal scanning ports, and so on [475–477].

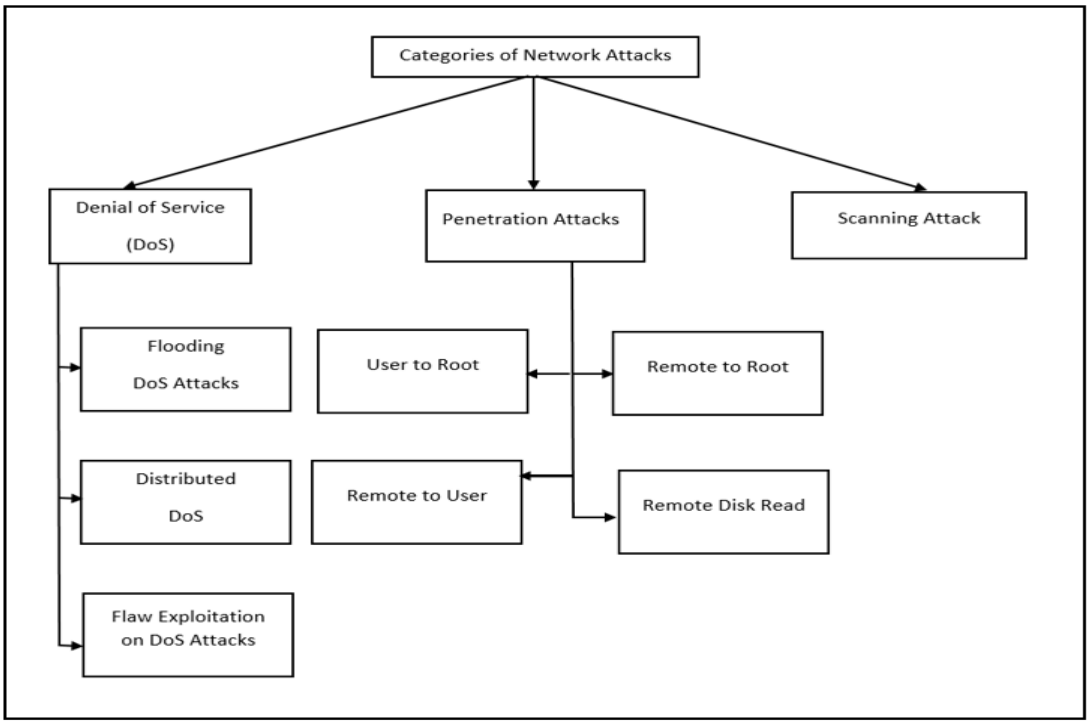


Figure 14: Network attack categories [478]

Thus, this research is positioned to address the challenges of a distributed denial of services (DDoS) which is currently the most prevalent and sophisticated threat in the SDN WAN and is increasingly difficult to prevent [479-481]. Thus, this research is positioned to address the intrusion-based DDoS in the network.

4.5 IDS and the SDN

This section presents the importance of IDS, its realization in the SDN security challenge, and the different IDS strategies that are evaluated and summarized.

In modern networks today, one of the techniques used to resolve the challenges of cyberattacks quickly is by detecting the malicious attack process early. During traffic flow or transmission, IDS is the mechanism that analyses all the traffic exchanged across the SDN network [482-485]. Once an ongoing malicious attack is identified, the IDS automatically notifies the brain of the computer or controller in the control plane, and the controller then makes a transfer to the networking switches of the data plane about some of the new flow rules to restore the normal operation of the network as quickly as possible [486, 487]. Moreover, each new packet that arrives at the system and is received by the switch port is segregated as belonging to a flow and requests the system. Then, the router processes the packet received according to the specified rule associated with the flow that the packet belongs to. Then the SDN controller installs the flow rule that is correct in the switch, which can permit packet transmission only if the packet is a valid flow [488–491]. Otherwise, the packets are dropped by the switch. That is, malicious packets can be discarded by the flow classification as “good” or "bad," which is obtained by the IDS using a set of pre-configured rules.

The IDS modules supply the flow base and the packet-based intrusion detection components based on the monitoring of the packet parser and the flow tables of the SDN switches. In a large-scale network where several IDSs are involved, once one of the IDSs detects an attack and initiates reporting to the SDN controller, the other IDS starts by announcing the identified attack to the other IDSs in the runtime. The IDSs can predict which of the time zones are involved in the attacks by considering the destination IP addresses and routing information after attack detection [492-495]. For instance, Abbas Yazdinejadna *et al.* [496] proposed a new approach based on kangaroo IDSs based on the SDN architecture for attack detection and malicious behaviour in the data plane. They designed a zone-based architecture in the KIDS that is distributed in nature and scalable. The IDS module provides the flow-based and packet detection components based on the monitoring packet parser and flow tables of the SDN switches.

The primary goal of an IDS in SDN is the ability to monitor the information sources, including the computers or networks, for unauthorized activities. This is made possible by the SDN architecture and its functionalities. It provides important opportunities for the creation of an effective intrusion detection approach. The IDS collects packets from different network systems and sources, which are then analysed for possible threats. The integration of an IDS into an SDN is potentially one of the most important approaches to providing a secure network. This is achieved by inspecting

streams of all incoming packets in the packet parser and flow tables and efficiently creating the security alarm if it senses the malicious packet [497, 498].

In the literature, various approaches to IDS have been proposed, developed, and even deployed to secure the SDN. This subsection highlights some of these research works based on their strategies, techniques, and evaluation:

Oludele *et al.* [499] proposed a multiplayer approach using anomaly detection to monitor a single host. The authors mentioned the challenges that include the detection of malicious and unwanted activities, including tampering with important files, an unauthorized connection, and so on. To achieve intrusion, the authors introduced a multi-layer that was able to harness the existing capabilities of the signature and anomaly methodologies. Moreover, the authors implemented the approach on a single host; in the future, the implementation can be further extended to multiple hosts, such as an agent installed on each computer and the manager. Aso, Guimaraes *et al.* [500] discussed the intrusion detection approach for the snort and source fire for a multinational company. The authors discussed challenges that come with the snort and source fire, which provide high flexibility.

To achieve intrusion into the network, the authors introduced the snort detection strategy approach. However, there are multiple drawbacks to the snort. Similarly, Shibli *et al.* [501] proposed a framework that can provide security in the network. The authors proposed challenges that include hacking the system and so on. To achieve the intrusion, the authors proposed a mobile agent that monitors the system, etc., using automated responses. However, there were multiple challenges with the proposed system, such that when the target of the malicious attacker is the mobile agent, the model can experience challenges, and it becomes difficult to protect the system from being hacked. Hence, some of the security structures can be adopted to protect the mobile agent.

David Wagner *et al.* [502] discussed the approach that shows the interaction of the application and the operating system. The proposed model addressed the challenges of how the IDS can be broken without detection using the matching technique, malicious insertion sequence, and so on. To achieve intrusion, the theoretical framework and evaluation of the security of IDS against the mimicry attack were developed to achieve the objectives. The approach has shown effectiveness and capability to break security threats. Lin Tan *et al.* [503] proposed a string-matching technique that can be optimized for other matching algorithms. The proposed framework addresses the challenges of any malicious activities that can occur, and the system can broadcast the intruder's information to the other state machine using predefined signature detection. To achieve the intrusion, the integration of the algorithm and framework was used for effectiveness. However, the author mentioned that their practical implementation is a challenge for the approach, which can

require a large amount of memory. For this reason, the algorithm fails to detect the behaviour, which is anomaly intrusion. S. Mrdovic *et al.* [504] proposed a technique called distributed IDS for the analysis of the system such that multiple sensors can be placed in the network segments that are selected to monitor the challenges of network traffic behaviour. To achieve the intrusion, SNORT was used to analyse the engine. Moreover, the distributed IDS was able to be managed by the console management, which can monitor and configure the IDS. The advantage of the IDS is its capability to provide greater protection against attackers because of the multiple computers that can be monitored continuously and prevent malicious attacks on the network [505].

Yeubin Bai *et al.* [506] proposed the framework for the security of the IDS. The proposed model includes two techniques for the IDS, such as misuse and anomaly detection methodologies. The proposed methods addressed the challenges of detection with different approaches, such as data mining, data fusion, and so on, used in the IDS. To achieve intrusion into the network, both methods provide information about the existing detection technologies. The authors discussed that the methods can detect intrusion, are sufficient and effective for the IDPS, and can respond to anomalies in real-time [503, 507]. Sang-Jun Han *et al.* [508] also proposed a technique for intrusion detection that can combine multiple hosts for the detection of multiple intrusions to reduce the false-positive rate. To achieve ID, the hidden Markov model (HMM) was proposed as the speech recognition model for the modelling of the call events of the system. Again, the decision tree algorithm is also used as the classification model for intrusion into future challenges. Ramaprabhu Janakiraman *et al.* [505] proposed the intrusion detection and rapid action (INDRA) that can be used to provide the IDPS, which has the capability of using peer-to-peer techniques for network security in the distributed environment to make the distribution of information about the intruder on the network peer-to-peer. The author mentioned the challenges that can affect the affected packets in the network. To address IDPS, when the INDRA can find the interruption, the security agent can cut off the packets that are affected. The advantage of IDPS is the capability to be reliable and efficiently trusted. To perform effectively, a large amount of data and memory is needed to store the information that is collected about the malicious [509, 510].

Laureano *et al.* [507] proposed a framework to protect the HIDS of the virtual machine by observing the behaviour of the system inside the virtual machine. The proposed model was able to address the intrusion into the network. To achieve intrusion, techniques were developed for the efficiency and duplication of the operating system for the intruders. Again, the proposed approach can be run simultaneously on the same hardware and has a major benefit that is cost-effective, and so on [502].

Similarly, Zhang *et al.* [511] presented the adaptive flow counting scheme for the anomaly methodology. The proposed approaches provided efficient techniques. To achieve the detection, the scheme of the prediction proposes to change the granularity and the measurement between the temporal and spatial dimensions. The mechanism was for updating the flow counting rules and aggregating the traffic that can detect anomalies while at the same time reducing the monitoring overhead. However, to be able to control the load of the individual network device (switch), the rules must be delegated to the wide network. Moreover, the greedy-based rules were proposed and can reduce the average number that is assigned to each device switch.

Garg *et al.* [512] proposed the adaptive flow counting approach for the detection of anomalies in SDN. The proposed challenges encountered were network overhead, controller overload policies, and so on. To achieve anomaly detection, the proposed algorithm was able to improve the other schemes and provide the ability to reduce complexity by eliminating the overhead and setting flags on the data that is aggregated. Moreover, the proposed technique can cover both small and heavy entries. Garg *et al.* [513] presented an adaptive aggregation of the flow entries and the detection of an anomaly scheme. The author mentioned the challenges that include the overhead and the monitoring of the accuracy of the anomalies. To achieve the detection of intrusion, the author merged the algorithm of two steps into a single procedure that is simplified. The proposed algorithm eliminated the setting step for the flags. However, instead of setting the flag value on the aggregate, the algorithm can directly update the policy rules. As a result, complexity and time can decrease, and algorithm efficiency can improve significantly.

Ha, T *et al.* [514] presented the sampling technique of the traffic in the SDN-wide area networks. The challenge addressed was to utilize the inspection capability of the intrusion. To achieve the objectives, the author formulated the problem of optimization to achieve the appropriate rate sample of each device of the network, such as routers, and the flow of traffic in the network according to the sample that is optimal to the rate of the functionalities of the SDN. Granby B.R. *et al.* [515] presented the development of the SDN platform for anomaly detection applications (PANDA).
Hommes, S. *et al.* [516] investigated the flow table saturation of the OpenFlow switches in the attack (DoS). They performed the analysis of the topologies in the network with various attacks, the criterion distance of a table, the status of the active entries of the switch flow tables, the lack of success of the balance of the load algorithms, and so on. To achieve the objectives, a protocol called NetFlow was utilized for the intended features of the traffic on the network to reduce the load.

Carvalho, L.F. *et al.* [518]: In this study, the authors presented the anomaly detection and mitigation strategy of real-time in SDN by using the analysis of the traffic and flow packet. The authors mentioned the challenges that include the anomaly detection of traffic in the SDN. To achieve the

intrusion, the authors proposed a scheme that consists of 4 phases, such as (1) that make the collection of the flows through the protocol of the OpenFlow (2) the normal traffic strategies for the creation of a digital signature of the segments that are analysed; (3) the anomaly mitigation effect-based network; and (4) the reports of the attacks identified to validate the results of the system. Lee, S. *et al.* [519] presented the approach of anomaly detection in SDN by using the two phases of the clustering approach. To achieve intrusion in SDN, the first phase, which is feature selection, was proposed and used to make the selection of the features necessary for the dataset. The advantage of the scheme was to estimate the maximal information coefficient (MIC), which measures the correlation between the two features. After making the selection, the algorithm of clustering that is based on the peak of the density can be classified by reducing the dataset into normal and malicious patterns.

Carvalho *et al.* [520] presented the ecosystem that is based on the SDN to detect and mitigate malicious behaviour. The authors discussed the challenges that involve the detection of traffic and the flow level anomaly detection strategies. To achieve intrusion, the authors presented a technique that includes four modules. (i) traffic collection modules that can be able to collect the flow of the network; (ii) anomaly detection modules that can be able to detect the anomalies using the traffic behaviour signature and the profile; (iii) the module that can mitigate an anomaly; and (iv) the module report that produces various reports of the validation of the attack. Peng *et al.* [521] discussed the DDoS detection of anomalies framework for the SDN based on centralized control. The author presented the challenges that come with the DDoS attack in the SDN. To achieve the anomalies, the author proposed two separate modules called pre-processing (normalized vector flow feature) and anomaly detection. Again, a large number of the feature flow vectors are obtained from the controller of the SDN.

From the above-discussed literature surveys, the SDN is still faced with various issues and challenges, including malicious attacks, etc., while trying to meet the demands of the end-users. However, there are various proposed papers, and some of them deal with the issues and challenges of intrusion detection and controller placement separately. Moreover, this has multiple challenges due to the placement of the controller and identification of the intrusions while still meeting the objectives of the SDN to monitor the network and systems for malicious damage or activity, the violation of policies, and other critical issues that affect the performance of the network. Thus, to achieve a practical solution to CPP, other related problems in the network must be tackled collectively under a large integrated framework. Therefore, this research proposes a robust and efficient solution to the SDN scalability issues through CPP that is both efficient and attack aware.

The objective is to achieve the best trade-off between network performance and security in the SDN-based WAN.

Table 18: Summarized IDS challenges and proposed solutions

Ref.	Challenge	Proposed Solution	Approach	Tool	Algorithm used	Methodology	Attack Types
J. Ashraf *et al.* [21]	DDoS attack, bandwidth, and network resource	A Framework for DDoS Attack	ML	N/A	Support Vector Machines, Artificial Neural Networks (ANN), Decision Trees, Genetic Algorithm (GA), Fuzzy Logic, Bayesian Network	Anomaly flow detection	DDoS attack
J. D. Gadze *et al.* [522]	Man in the middle, DDoS	Framework detection and mitigation	ML	Mininet, Scikit-learn with Python, Wireshark, Hping and Virtual Box	N/A	Anomaly flow detection	DDoS attack
T. A. Tang *et al.* [523]	Logical centralisation of a controller, the security of the network, potential threats of the network and flow-based anomaly detection.	Framework for deep learning approach for a flow-based anomaly detection system	DL for flow-based anomaly detection	N/A	Deep Neural Network (DNN)	Anomaly flow detection	DDoS attack
Q. Niyaz *et al.* [524]	Security of the managed network, intrusion detection systems, incoming normal traffic or attack	The framework consisted of a deep learning-based DDoS detection system	DL for feature reduction	MATLAB 2016a	Deep learning for feature reduction	Anomaly flow detection	DDoS attack
L. Zhu *et al.* [525]	Detection in a single domain by SDN, multiple network domains, Abnormal traffic	The framework used Predis, a privacy-preserving cross-domain attack detection scheme	ML	N/A	Efficient algorithm k-Nearest Neighbor (KNN)	Anomaly flow detection	DDoS attack
Y. Yu *et al.* [526]	Safety on the road, the driving experience, reduced traffic congestion, multiple attacks, and compromised nodes	The platform that efficiently detects and can rapidly respond to DDoS malicious attacks in a virtual	Open flow protocol for response and flow feature extraction	Open-source Mininet network simulator	SVM	Anomaly flow detection	DoS/DDoS attack

		network	strategy				
L. Barki *et al.* [527]	Forwarding of the incoming packets, intrusion detection	Traffic Classification Framework	ML algorithms	Mininet	Naive Bayes	Anomaly flow detection	DDoS attack
A. Le *et al.* [528]	DoS and probe attack, cost and latency	Framework for IDPS consisting of the 1999 DARPA dataset	IDPS for classifying traffic in the management server	Java library of WEKA	C4.5 Decision tree	Anomaly flow detection	Denial of Service and Probe attack
P. Amaral *et al.* [22]	Type of data, OpenFlow protocol, traffic classification	Traffic framework for enterprise network using open-flow protocol	ML techniques for classification datasets using supervised learning	N/A	Random Forests, Stochastic Gradient Boosting, and Extreme Gradient Boosting	Anomaly flow detection	DDoS attack
A. Dawoud *et al.* [529]	Buffer and stack overflow, anomaly detection	Intrusion detection model	Unsupervised DL	TensorFlow (TF)	K-means clustering algorithm	Anomaly flow detection	Denial of Service attack
T. V. Phan *et al.* [530]	DDoS attacks, SDN-based cloud environment.	The framework consists of a history-based IP filtering scheme (eHIPF)	The approach consists of a new hybrid ML	BoNeSi (DDoS attack tool)	Support vector machine and self-organizing map algorithms (SOM)	Anomaly flow detection	DDoS attack

4.6 Performance Evaluation

This section presents the performance analysis of the selected ML algorithms.

4.6.1 ML Objective

The objective is to evaluate the performance of each algorithm for intrusion detection performed on the identified dataset with the intent of identifying the best-performing model in terms of accuracy and other evaluation metrics. The identified model will then be adopted for the design of the proposed IDS framework in this research. In this case, supervised ML algorithms are employed, and any ML model that is found to be efficient and effective in the form of classification accuracy will be employed in the development of automatic pattern detection in the architecture of the SDN setting. The choice of the proposed ML algorithms is based on their general acceptance as the best approach for the modelling of data and provides good generalization performance due to the

effective analysis and efficient classification of the datasets [531–536]. Thus, this section performs the classification and prediction on the chosen datasets to analyse and evaluate the performance of the chosen models for intrusion.

4.6.2 System properties and tools

To achieve the objectives, hardware components and software were used such as Processor Intel(R) Core(TM) i7-10610U CPU @ 1.80GHz, 2304 MHz, 4 Core(s), 8 Logical Processor(s), System Type x64-based PC, Installed Physical Memory (RAM) 16,0 GB, OS Name Microsoft Windows 10 Pro. To determine which of the models is more effective in terms of the accuracy of the classification, prediction and so on, the ML algorithms were employed and run on the software tool called Rapid Miner Studio.

4.6.3 Data Collection and Description

To identify which of the proposed models is effective and efficient in the classification and prediction of intrusion, we adopted the NSL-KDD data set [537] to achieve the training and testing of the proposed models. Several kinds of literature have shown that this dataset is the most effective and efficient for the development of the IDS in terms of training and testing models. That is, many researchers have used data sets to achieve various objectives. For instance, S. Revathi *et al.* [538] proposed a framework using ML techniques to achieve the objectives of the intrusion using the WEKA tool.

Vipin Kumar *et al.* [539] employed clustering methods by using a data set for training and testing attacks. Similarly, K. M. Faraoun *et al.* [540] tested the IDS based on the algorithms called the neural network and SVM, while H. Oh *et al.* [541] performed the experiments in real-time IDS development based on the self-organizing map (SOM). N. Ye *et al.* [542] also proposed the use of a data set to evaluate the generic programming approach for multi-category pattern classification for networks. However, benchmarking supplies the data sets that are labelled to compare the intrusion detection system.

Table 19: Attribute value type and attacks

Type	Features
Nominal	Protocol_type(2), Service(3), Flag(4)
Binary	Land(7), logged_in(12), root_shell(14), su_attempted(15), is_host_login(21),, is_guest_login(22)
Numeric	Duration(1), src_bytes(5), dst_bytes(6), wrong_fragment(8), urgent(9), hot(10), num_failed_logins(11), num_compromised(13), num_root(16), num_file_creations(17), num_shells(18), num_access_files(19), num_outbound_cmds(20), count(23) srv_count(24), serror_rate(25), srv_serror_rate(26), rerror_rate(27), srv_rerror_rate(28), same_srv_rate(29) diff_srv_rate(30), srv_diff_host_rate(31), dst_host_count(32), dst_host_srv_count(33), dst_host_same_srv_rate(34), dst_host_diff_srv_rate(35), dst_host_same_src_port_rate(36), dst_host_srv_diff_host_rate(37), dst_host_serror_rate(38), dst_host_srv_serror_rate(39), dst_host_rerror_rate(40), dst_host_srv_rerror_rate(41)
Attack Class	**Attack Type**
DoS	Back, Land, Neptune, Pod, Smurf, Teardrop, Apache2, Udpstorm, Process table, Worm (10)
Probe	Satan, Upsweep, Nmap, Portsweep, Mscan, Saint (6)
R2L	Guess_Password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Warezclient, Spy, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named (16)
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps (7)

Table 20 shows the distribution of normal and attack records that are available in the various NSL-KDD datasets.

Table 20: Normal and attack data set [543, 544]

Data Set Type	Total No. of					
	Records	**Normal Class**	**DoS Class**	**Probe Class**	**U2R Class**	**R2L Class**
KDD Train+ 20%	25192	13449	9234	2289	11	209
		53.39 %	36.65 %	9.09%	0.04 %	0.83 %
KDD Train+	125973	67343	45927	11656	52	995
		53.46 %	36.46	9.25%	0.04 %	0.79 %
KDD Test+	22544	9711	7458	2421	200	0.89 %
		43.08 %	33.08 %	10.74 %	0.89 %	12.22 %

4.6.4 Machine Learning Models

The selected ML algorithms considered to identify the best-performing model for intrusion detection are discussed in this subsection. The chosen models are analysed and evaluated based on their different performance characteristics. Each model selected is tested for classification and prediction analysis using the NSL-KDD data set [537] and the detection of the intrusion of each model on the classification accuracy, precision, and so on. The dataset is visualized using various methods and approaches from different perspectives.

Table 21: A summarized description of the chosen models

Model	Description and motivation
Logistic regression [545, 546]	A model is used to model the probability of a particular class or the event that exists, such as pass or fail. The statistical model can use the logistic function to model the binary dependent variable. The estimation of the parameters of the logistic model is in the form of binary regression. The dependent variable has two possible values that can be represented by the variable indicator, where the two values are "0" and "1". It has the capability and is used for various fields such as machine learning, social science, and so on.
K-nearest neighbours [547-549]	Non-parametric classification approach that uses classification and regression analysis. Output is considered the membership of the class and property value instance. The function only approximates the local, and all the computation is deferred until the evaluation. Rely on the distance of the classification, and features are represented by various physical units of different scales that normalize the data training to improve accuracy. The training of the algorithm consists of the storage of the feature vectors and the labelled class of the training data instances.
Naïve bayes Model [550-552]	Considered the family of probabilistic classifiers of Bayes' theorem with strong independent assumptions about the features. Highly scalable, which requires a linear parameter number for the variable number (predictions or features) for problem learning. Is used for the construction of the classifier models that assign the labelled class to the problem instances by the representation of the feature of vector values such that the labelled class can be drawn for some finite set. It can make assumptions that the value of the feature is independent of other feature values.
Decision stump [553-555]	The algorithm consists of a decision tree, which is one-level. It is considered a decision tree with one internal node, which can be connected immediately to the node's terminal. Makes the prediction based on the value of the single input feature. It uses only a single attribute for splitting and is a weak classification model that is made of a simple tree structure and consists of one split, which can be considered a level decision tree. Due to its performance and simplicity, the model often demonstrates low predictive performance.
Random forest [556-558]	Learning methods for classification, regression, and others that operate by the construction of a multitude of decision trees at training time. The output of the classification tasks is the class selected by most trees. Consists of the same hyperparameters as a decision tree. adds additional randomness to the framework while growing the trees. The model searches for the best feature among a random subset of the features. It utilizes ensemble learning, which is an approach that combines classifiers to provide the solution to complex problems.
Gradient boosted [559-561]	Combining the weak learners with a single strong learner can be done iteratively. It combines the learning algorithm in the series to achieve a strong learner from the sequentially connected weak learner. Moreover, the weak learners are the decision trees, and every time a new one is added, it can fit on the modified version of the initial dataset. The algorithm performs the regression and the classification by combining the outputs from the individual trees. The model relies on the intuition that the best next model, when combined with the previous, can minimize the overall prediction error. The intuition of the model is to set the outcome results for the next model to minimize the error.
Deep learning [562-564]	It is considered the field of artificial intelligence such that the machine can be able to do tasks that typically require human intelligence. encompasses ML, where machines can learn by experience and acquire skills without human involvement. DL is a subset of ML where artificial neural networks, algorithms inspired by the human brain, learn from large amounts of data, such as how to learn from experience. Perform a task repeatedly, each time tweaking it a little to improve the outcome.
Decision tree [565-567]	Consider the model that uses tree representation to solve the problem that each of the node leaves can correspond to the class label and attributes that are represented on the internal node of the tree. It is used for handling non-linear data sets effectively. It is divided into two types, such as categorical and continuous variable decision trees. It is used for both classification and regression problems and for achieving the classification problems. A tree-structured classifier is such that the internal nodes are represented by the features of a dataset, the branches are represented by the decision rules, and each of the leaf nodes represents the outcomes of the instances of the problem.

4.6.5 Evaluation Metrics

This section discusses the techniques for the evaluation of the proposed models used for the simulation of the dataset. These techniques are used for comparison, analysis, evaluation, and so on to find the best-performing model from the proposed algorithms. The ML algorithmic models are evaluated for classification accuracy, classification error, absolute error, relative error, squared error, correlation, kappa, and squared correlation and are explained as follows:

Table 22: Summarized evaluation metrics

Performance metrics	Description
Classification accuracy [568-570]	Data analysis is considered a metric that can be used to summarize the performance of a classification model such that the number of correct predictions can be divided by the total number of predictions. The process is easy to calculate and intuitive to understand, and it can be the most common metric used for evaluating classifier models. In general, the CA is simply the rate of the correct classifications, which can be an independent test set or some of the variations of the cross-validation set.
Classification error [571-573]	Is considered a type of measurement error such that the respondent cannot provide a true response to a survey item. However, for a nominal data category, this can occur in either one of two ways, such as a false negative response or a false positive response. This approach aims to measure the misclassification rates and enable the correct estimation of the proportions for the misclassification.
Absolute error [574, 575]	The absolute error is considered as the magnitude or the size of the difference between a measured value and a true or the exact value such that Absolute Error = |True value – Measured value|
Relative error [576-578]	The relative error is considered to be the absolute error from a perspective such that the size of the absolute error and the size of the true value can be compared. Relative Error = |True Value - Measure Value|/ True Value
Squared error [579, 580]	The squared error accepts the input of the distance from the points to the regression line, such that the distance is considered the error and squaring off a distance. The process of squaring is important to reduce and remove negative signs. It also gives more weight to the larger differences, such as the mean squared error, by finding the average of a set of errors.
Correlation [581, 582]	Correlation or dependence is considered the relationship of the statistical analysis that can be casual or not between the random variables or the data bivariate. In simple terms, it is considered any statistical association, or commonly, the degree to which a pair of variables are linearly related such that they can change at a constant rate. For these reasons, when one variable increases, the other variable also increases, or one variable decreases as the other decreases.
Kappa [583, 584]	Is considered the statistical measure of the inter-rated reliability for the categorical variables. It is used when two rates can both apply a criterion based on a measurement tool to assess whether some condition does occur. In simple terms, the numerator can represent the discrepancy between the observed probability of success and the probability of success under the assumption of an extremely bad case.
Squared correlation [585-587]	The square of the correlation is considered the useful value in the linear regression model. This analysis can represent the fraction of the variant in one variable that may be explained by the other variable. The correlation analysis can be able to relate directly to the process of the regression, such that Y = a + bx for any of the two variables.

4.6.6. Methodology

As shown in Figure 15, we read the NSL-KDD data sets from the storage location in the form of the metrics performance. This data set was used in the model as training and testing data. The generated data set was used in the model to identify the data packet nodes that are likely to be affected or attacked. Then these data sets are trained using the selected ML algorithms for classification analysis. Moreover, each model is used to classify the data set according to the dependent variables.

To achieve the goal of this research, we designed the binary classification and prediction model for the selected algorithms for the data packets, where the dependent variables were treated as either normal or abnormal behaviour of the data packets. Furthermore, the data packets were organized using performance metrics as records, and each was considered the vector attribute followed by the target output of the data packets in the network.

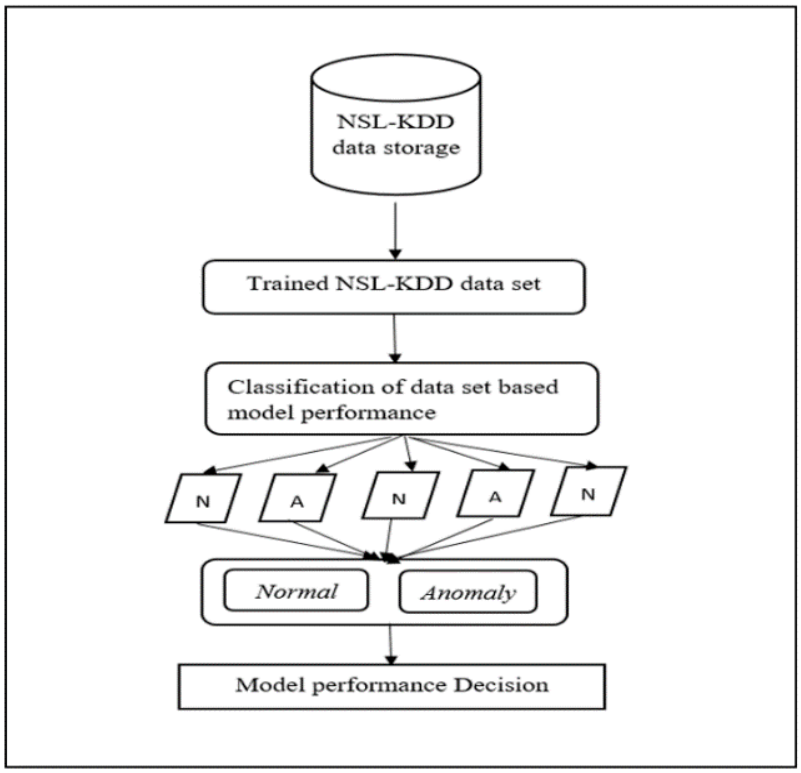


Figure 15: NSL-KDD data set classification model

4.7 Results and Analysis

In this section, we present the results obtained from the performance analysis of the selected models to achieve the objectives of this research. It commences with the analysis of the results-based classification and prediction and further gives the comparative results analysis based on accuracy, efficiency, execution time, prediction, and so on. Table 23 records the results obtained from the selected algorithms.

Table 23: Selected algorithms performance analysis

Model	Accuracy	kappa	Absolute error	Squared error	Execution Time (s)
Decision tree	96.98%	0.939	0.058	0.029	39s
Logistic regression	97.44%	0.949	0.949	0.018	4s
Gradient boosted	98.41%	0.968	0.313	0.100	5s
Naïve bayes	90.33%	0.806	0.097	0.094	1s
Deep learning	99.46%	0.989	0.007	0.005	14s
K nearest neighbour	99.57%	0.991	0.005	0.003	26s
Random forest	97.20%	0.984	0.039	0.008	1s
Decision stump	92.22%	0.843	0.144	0.072	1s

Moreover, to further analyse the quality of the classification accuracy we have adopted the ROC to better analyse the classification and prediction accuracy of each model, as shown in Figure 16. Table 24 presents the summarised ROC results of the selected models.

Table 24: Summarised ROC results

Criteria	KNN	Logistic Regression	Gradient Boosted	Decision tree	Naïve Bayes	Deep Learning	Random Forest	Decision stump
Classification accuracy	99.57%	97.44%	98.41%	96.98%	90.33%	99.46%	97.20%	92.22%
Sensitivity (True Positives Rate)	0.94	0.90	0.92	0.86	0.85	0.92	0.92	0.92
Specificity (False Positives Rate)	0.8	0.6	0.6	0.4	0.6	0.4	0.4	0.2
Cut point	9-10	9-10	9-10	9-10	9-10	9-10	9-10	9-10
Threshold	0.84	0.84	0.84	0.84	0.84	0.84	0.84	0.84

4.8 Discussion

Figure 16 presents the ROC curves of the selected models presented in Table 23. The figure shows the results of the best-performing models based on the detection accuracy of the intrusion in the network. The model that performed best was adopted in the implementation of the framework.

In general, a decision tree can classify and identify 96.98% of the normal and anomaly datasets with a 3.02% classified error, which shows how wrongly some of the data sets are misclassified as normal and anomalous. LR, on the other hand, identified 97% of the normal and anomaly-classified NSL-KDD data sets, and 2.56% of the classified error was achieved, which is less compared to the DT misclassified data set. Moreover, the GB algorithm was able to produce 98.41%, which proved that the algorithm was able to adapt to normal and anomaly detection processes compared to the DT and LR algorithms. Again, there is a balance due to the classified errors of 1.59%, which is two times less compared to the decision tree and LR. Furthermore, 90.33% of the normal and anomaly

data sets were classified and identified, and 9.67% of the classified errors of the normal and anomaly datasets were identified and predicted by the NB, which performed poorly in adapting to the proposed NSL-KDD data set.

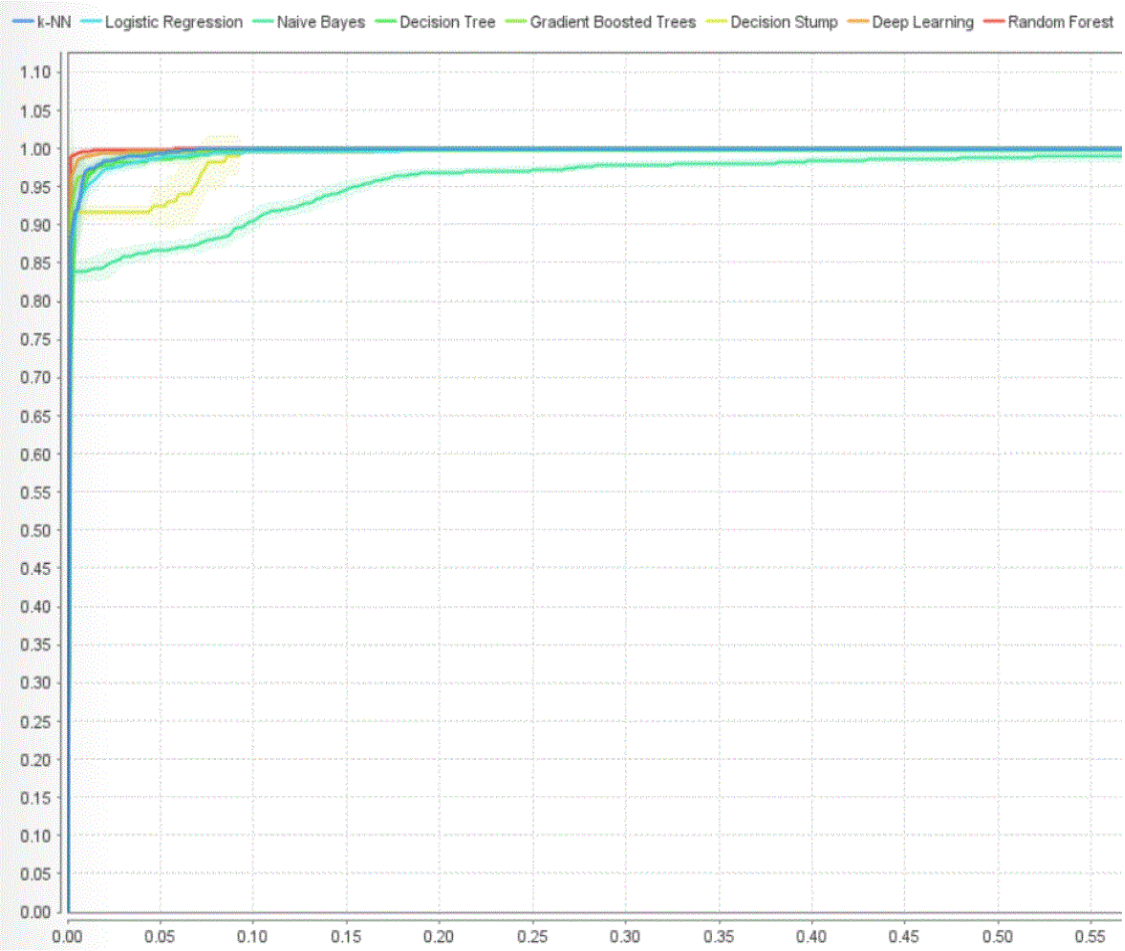


Figure 16: ROC curves of the selected models

Furthermore, DL produced 99.46% detection accuracy of the classified normal and anomaly dataset with 0.54% fewer classified errors, which is far less compared to the decision tree, LR, GB, and NB algorithms. This means the model can excellently identify, classify, and predict the normal and anomaly data sets more effectively and efficiently. On the other hand, KNN was able to produce 99.57% classified accuracy for the normal and anomaly data sets. This implies that the model resulted in good performance accuracy on the normal and anomalies within the NSL-KDD data set. Again, the 0.43% classified error was identified to be far less compared to the LR, NB, DL, DT, and GB algorithms. An RF can produce 97.20% of the classified normal and anomaly detections. However, they can excellently achieve classified errors for normal and anomaly data sets of 0.8%, which shows good performance for the identification and prediction of the algorithm. Decision stump, which is the last algorithm, can produce 92.21 of the identified accuracy classifications for

normal and anomaly detection, which is 7.79%, which is more compared to the DT, LR, GB, DL, KNN, and RF.

However, in our case, the best-performing algorithm is based on the capability to classify, identify, and achieve excellent prediction. However, KNN has shown 99.57% of classified accuracy and predicted results analysis for normal and anomaly detection of intrusion. This implies that the algorithm resulted in good performance accuracy on the NSL-KDD data set and the prediction of normal and anomaly datasets. Again, the 0.43% classified error was identified to be far less compared to the LR, NB, DL, Decision Tree, and GB algorithms. The obtained results prove that the KNN outperforms the other models in terms of classification accuracy and prediction analysis on the dataset.

The choice of the KNN in this chapter is informed by the highest percentage for classification and prediction accuracy of the intrusion as compared to other models, as demonstrated by the results. The KNN has the consistency and capability of producing good results in terms of classification, prediction, and others. Then, this algorithm will be adopted and utilized in the development of the IDS framework settings that will be achieved in Chapter 5.

4.9 Chapter Summary

This chapter presents a background review of intrusion detection systems (IDS) in software-defined networking (SDN). It began with IDS classification methodologies, discussing various classifications such as passive and active responses. The IDS architecture was discussed in detail. Moreover, the classification of the IDS for decision-making, the IPS classification, and network attacks were further discussed. Information about the performance evaluations, which include the selected algorithms, metrics, tools, and the description of the data set, was explained. Again, classification and prediction results, analysis and evaluation, and comparative results were presented. Lastly, the KNN algorithm was selected based on the classified and predicted data set for normal and anomaly detection of intrusion compared to other models, as demonstrated by the results.

Chapter 5

Research Methodology and Design

5.1 Chapter Outline

This chapter presents the adopted research methodology and design of this research. Moreover, different research paradigms were discussed in detail. The design science research (DSR) process was presented in detail. Furthermore, the qualitative and quantitative approaches were discussed in detail. In addition, the implemented approach of this research was further explained in detail. The research tools and technologies for this research were also discussed. Lastly, the research validity, reliability, and evaluation were presented in detail.

5.2 Chapter Introduction

Research in computer science (CS) is the study of systematic investigation and findings to achieve related facts and draw conclusions [588–590], etc. It involves the method of discovering new or collating facts in the form of a scientific approach to the study and other factors of critical investigation [591–593]. However, the research can range from the theoretical underpinning, practicality, and implementation of the projects [594-598], etc. Moreover, there are still other formulated and related topics that can draw particular attention, such as the design of the algorithms, the performance of the software, the analysis of the program, verification, testing [599–605], and so on. In addition, the aspects of research in CS are problem-solving, design, development, analysis [606–608], and so on. These approaches involve the investigation of systematic methods, namely, information gathering, discovery, uncovering, and exploration of the criteria [609–613].

Moreover, most aspects of the research in CS involve methodology. A research methodology provides the researcher with legitimacy and provides scientific findings. It is considered a detailed plan that can assist the researcher in staying on track with the research, making the process smooth, and making the research effective and manageable. It involves research methods that include the test of theory based on explanation for understanding [614, 615]. The objective of this criterion is to provide and introduce rules and concepts about the techniques for research communication, identification of opportunities, risk management, identification and uncovering potential problems [616–619], and so on. On the other hand, the research design and research methodology are considered the framework plan for conducting the research. It provides a set of rules, plans, and procedures that can be used to produce reliable data. This framework should be aligned with the type of research questions being proposed and the type of collected data.

In addition, it assists the researcher in coming up with the correct decisions, identifying major and minor parts of the research, easily framing the right objectives of the research work, and others.

In CS, we refer to the research methods used to test our hypothesis and compare two or more things associated with each other and the actual phenomenon we observe. The methods produce answers formed from the research questions posed in the form of a working hypothesis that allows us to derive theories about the observations, etc. It allows the researchers to predict and derive future events and phenomena. In CS, several research methods can be used, including experimental, simulation, and theoretical [620]. To gather and measure the information on the appropriate variable of interest, we consider the data collection to answer the stated research questions, test the hypothesis, and evaluate the outcomes. Moreover, the data can be partitioned into four main types based on the methods: observational, simulation, derived, experimental, and others. Furthermore, research tools refer to the devices or types of instruments used for the collection of data; they are used to conduct the measurement of variables and collect the required information needed to answer the RQs and others.

Furthermore, upon completion of the data collection process, the next step involves the research evaluation. It is defined as the disciplined and systematic process that can be carried out to achieve the assessment of an object, practice, and other systematic activities intended to provide the required information that will be used in making the decision. It also provides a systematic method that studies programs, practices, and interventions to understand how well the goal was achieved. Again, it assists in determining what can work well and what needs to be improved in the program and others.

In the context of this research, as shown in Figure 17, a preliminary literature review was performed to identify the research gap about the SDN environment. Moreover, our study contributed some new perspectives to the existing knowledge of SDN networks. Furthermore, we have reviewed other studies closely related to our research to fill the gap and extend prior studies. We have designed the framework that established the importance of our study and compared our results with other findings. We have integrated what other researchers have done, criticized the previous scholarly works, built bridges between the related topics, and provided the central issues in the study.

In addition, we have employed software tools, technologies, and methods for our data analysis and evaluation. The empirical analysis based on the SDN CPP and IDS performance-based algorithms was evaluated for the best ML algorithmic model. This was performed based on the clustering and classification algorithms. From the performed model, the model we utilized for the designed integrated CPP and IDS in the SDN-enabled WAN settings.

Furthermore, the integrated SDN CPP and IDS evaluations were compared with the related papers. Again, the connection between our designed framework and our contribution was presented in detail. Then, the conclusion of this research was drawn in terms of the future direction.

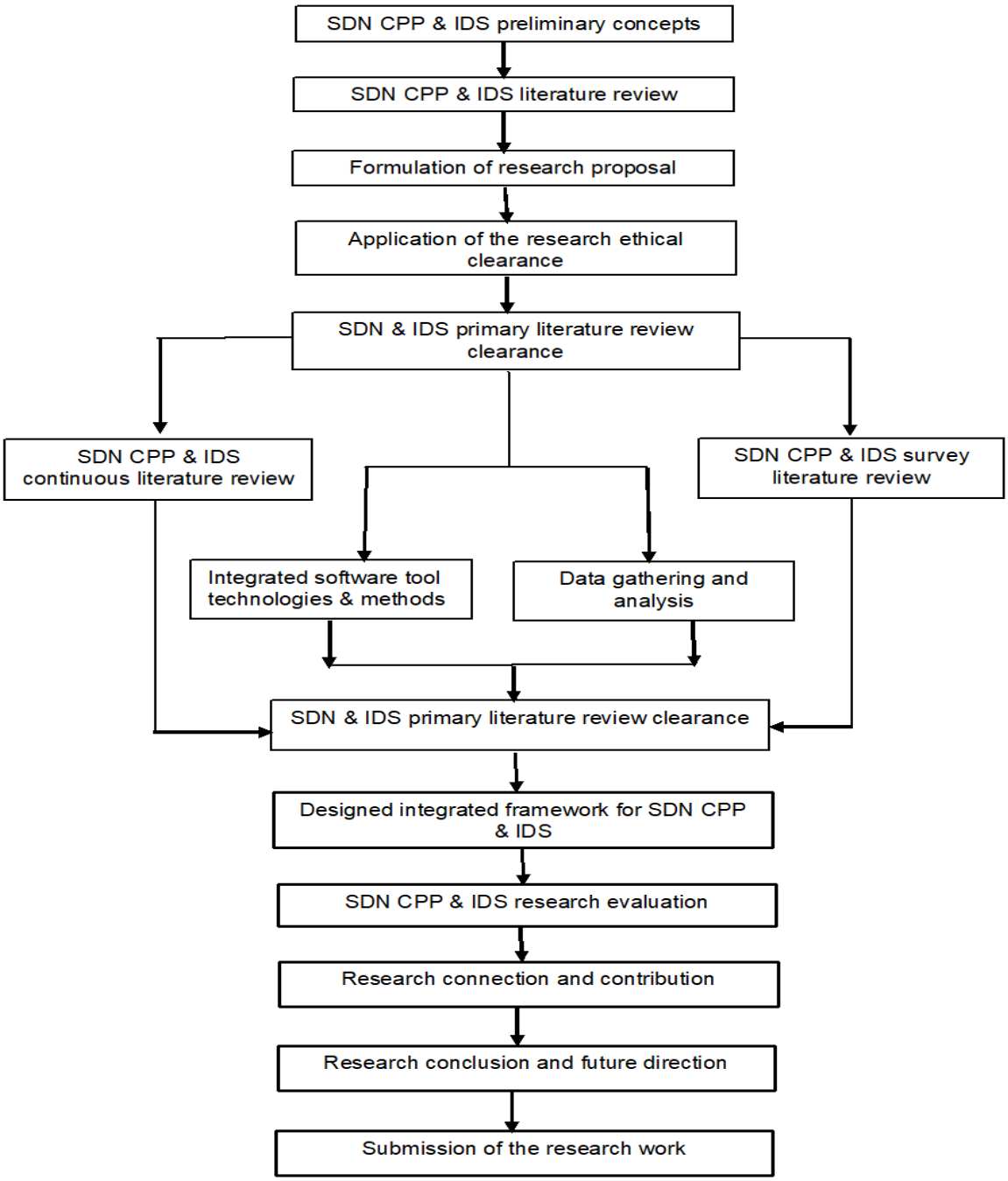


Figure 17: The flow of the proposed research work

5.3 Research Paradigms

The research paradigm is the set of philosophical beliefs that can be used for the study and interpretation of knowledge [621]. Moreover, it is a set of concepts, practices, values, and

assumptions that establishes a way to consider reality [622, 623]. In this section, we discuss the different applications of paradigms that include positivism, interpretivism, and others.

5.3.1 Positivism

Positivism is considered an independent, self-governing existence of the truth. It is considered the research paradigm that is established on the objective that reality and truth are independent and free of the observer and viewer [624]. Moreover, the positivism paradigm involves the process of gathering the data, observing the irregularities, and extracting the law. A. Aliyu et al. [624] also discussed that the methodologies that can be employed by positivist researchers include confirmatory analysis, nomothetic, qualitative, and others. Furthermore, it can also emphasize that the genuine, factual, and happening can be studied and observed scientifically and empirically and can be elucidated in the form of lucid and rational analysis and investigation. In some cases, researchers employ positivism in the process of determining the forecasts of human occurrences to gain some knowledge of what constitutes the truth [625].

5.3.2 Pragmatism

The pragmatism paradigm involves shared meanings represented in the form of joint actions. It considers that theory can be generalized and contextualized by the analysis of its transferability to other situations. Tran et al. [626] discussed that pragmatism involves the ability to convert observations into theories and assess the theories through actions.

Scott et al. [627] mentioned that it involves meaning that is based on the belief that the meaning of a set of ideas should lie in the consequences rather than the ideas themselves. Morgan et al. [628] mentioned that the method involves abductive reasoning that can be able to move back and forth between induction and deduction, i.e., linking data to theory or working back and forth between the methods of qualitative and quantitative data. Furthermore, Guthrie et al. [629] mentioned that the method enables the researchers to combine the methodologies in the same project based on the structure of the problem statement at hand.

5.3.3 Interpretivism

Aliyu et al. [624] mentioned that interpretivism is quite new everywhere among non-positivist scholars and searchers. Bryman et al. [630] discussed the interpretivism research paradigm as an epistemological position that requires the social scientist to grasp the subjective meaning of social actions. Moreover, Cohen et al. [631] discussed that appreciating the world of human occurrence is the main role of understanding the method in the research. For this reason, interpretivism is considered to replace positivism since it belongs to the social truth or reality. In addition, Creswell

et al. [632] discussed that the method relies on the views of the participants in the subject being investigated.

5.3.4 Socio-critic

This contrasts with the traditional theory that can explore and confirm the status quo, challenges the status quo, and strives for a balanced and democratic society. This paradigm is concerned with the issue of power and other relations with society and the interaction of race, class, education, gender, and others by contributing to the social system. Horkheimer et. al. [633] discussed the criteria for the correct critical theory, such as (i) explaining what is wrong with the current social reality, (2) identifying the action to change it, and (3) providing both the clear norms of criticism and transformation.

Based on the above discussion, the interpretivism paradigm refers to the understanding of the individual and their interpretation; hence, it is referred to as the constructivist paradigm. The positivism paradigm assumes that a single tangible reality can be understood, measured, and identified. On the other hand, the socio-paradigm is based on the dialectical critical nature of knowledge construction as well as the role of the ideology of the scientific process. Lastly, the pragmatic paradigm bridges the gap between the scientific method and the structural orientation of the older approaches and the naturalistic methods.

It involves a research design that involves operation decisions based on what will work best by finding the answers to the questions under investigation. Thus, in this research, we have employed

the pragmatic research paradigm [634, 635] to solve issues and challenges based on SDN, CPP, and IDS. The paradigm employed in this study was adopted based on the following criteria: problem-solving, research-driven and application, results and evaluation, theory and practice, and is discussed as follows:

i. *Problem Solving*: We employed and applied the technical tools, systems, techniques, etc. to address the SDN CPP objectives as well as to provide security measurement for the number of placed controllers in the network.

ii. *Research Driven and Application*: We have designed an integrated framework that addresses the CPP-IDS practical solutions and aims to produce remarkable and usable results in the network. Moreover, it aimed to provide tangible results that can be implemented and applied by the SDN user applications.

iii. *Results and Evaluation*: In this study, we evaluated the SDN CPP-based IDS practical results and outcomes that involve the system's performance, usability, and impact of the

technology on the existing research studies based on SDN CPP-enabled WAN and attack-aware strategies. We evaluated the effectiveness, efficiency, applicability, etc. of our system in the real-world SDN environment.

iv. *Theory and Practice*: In this study, we bridged the research gap between theory and practical application. We have reviewed previous related papers and employed and gathered concepts, techniques, methods, tools, etc. to bridge the gap between the theory and practical applications based on SDN CPP. We provided the development of SDN theories and the designed models relevant to solving the challenges based on SDN CPP-IDS in the network. The main aim was to ensure both theoretical and practical contributions to the SDN research objectives.

5.3.5 Design Science Research (DSR) Methodology

DSR is considered an established field for the development of innovative solutions to real-world challenges [636, 637]. It is a research paradigm in which the researcher can provide relevant answers to address human problems, create useful artefacts, and make some contribution to existing knowledge [638, 639]. DSR methodology is a highly context-dependent and iterative process. Moreover, every single DSR project follows its process in the instance stage, considering both the specific opportunity and the designed constraints [640, 641].

Iteratively, it seeks to understand and conceptualize the space of the problem, the analysis of the space solution, and the designed knowledge. Furthermore, the contributions to the evaluation of the DSR have developed in the immediate evaluation [642, 643]. Thus, in this section, we have employed the

DSR for the integrated CPP and IDS framework in the SDN-enabled WAN. The involved DSR process, as shown in Table 25, included awareness of the problem, development, and evaluation [644-646].

In our study, we followed the practice of DSR methodology to contribute to the following criteria:

i. *Artefact Creation*: We designed and developed new artefacts that include the algorithm and framework employed for the practical outcomes in the network. We provided a structured approach during the creation of the system to make sure that the artefacts were designed for the right purpose and met the right requirements to address the research gap in the SDN CPP-enabled WAN.

ii. *Practical Orientation*: We have employed and adopted software tools and techniques, etc., to solve practical problems and address the challenges faced by the SDN network. We have

reviewed previous papers, systems, etc., identified specific existing gaps, and provided innovative solutions to address the CPP-IDS in the network.

iii. *Validation and Evaluation*: We employed various techniques, methods, algorithms, etc. to practically evaluate and validate our framework model. We evaluated the performance, usability, efficiency, and impact of our research study.

iv. *Contribution to Knowledge*: We employed the methodology to provide contributions to both practical and theoretical knowledge. We provided advanced knowledge by developing and validating the new design and principles, models, and theories about the nature of the SDN CPP and attack-aware strategy. We followed the methodological strategy to provide and generate practical knowledge through a developed innovation that can be applied in the real-world SDN environment.

5.4 Qualitative vs Quantitative

5.4.1 Qualitative

In this section, we have employed the pragmatic paradigm discussed in Section 5.3.2. on the LAN collected traffic flow rules to gather knowledge about the nature of the integrated SDN, CPP, and IDS. The generated and decoded data were statistically reported based on the results of the data features.

The adopted pragmatic paradigm method was used during the analysis, evaluation, and understanding of the meaningful outcome of individual data during the practical implementation based on CPP and IDS. The traffic data was typically collected during the same time interval and integrated into the statistical software tool for analysis and evaluation. Moreover, the CPP objectives and ID problem

Formulations discussed in Chapter 6 were used for evaluation. Then, the data was analysed for practical decisions based on CPP and IDS, and the gathered results were used to gain knowledge about the nature of the SDN environment.

Table 25:Design science research steps

Awareness of the problem	Based on our awareness of the problem, we proposed to address the challenges of the nature of the SDN architecture. We brought to our attention that the SDN architecture is advantageous to address the challenges and complexity of the traditional network, but it also brings some security issues since it was not considered in the initial design. The proposed solutions lacked a robust built-in security model that effectively identifies normal, and other activities such as CPP, traffic prediction, and others could suffer from the lack of trusted connections between the layers, which would make them unable to defend against attacks such as distributed denial of service (DDoS).
Suggestion	We designed an integrated framework solution based on SDN environment issues through CPP that is both efficient and attack-aware. The tentative design was based on the best trade-off between the network performance and the security in the SDN-based WAN. We proposed to incorporate the ML techniques into the overall framework design that addresses the challenges of securing, efficiently monitoring, and managing the reliability of the controllers during the exchange of control messages in the integrated framework, thereby improving the performance of the SDN in WAN settings.
Development	We employed the known ML technique and related packages for the implementation of our model. The model, parameters, and metrics were used to perform the analysis. Then, the results obtained were used to determine the deployed number of the controller, controller-to-switch assignment, load balancing, reliability, processing latency, and security. We performed our simulation using the 1.15 controller to analyse the flow rules. Moreover, we performed the simulation to address the CPP objectives as well as to provide countermeasures for the anomaly attack in the network.
Evaluation	In this study, we performed and evaluated our results with other well-known SDN controllers, OpenDayLight, Ryu, and Floodlight. We evaluated the controllers deployed on a designed network topology to achieve CPP and IDS in the SDN environment. Moreover, we have evaluated other ML algorithms for intrusion detection for the proof of concept. The results obtained from the simulation were analysed and recorded.
Conclusion	In the end, we considered the conclusion of our research based on the satisfying results. Moreover, we took into account our deviations, such as the accuracy of our designed models and predicted results. Moreover, the knowledge gained from the DSR process was categorized as facts and lessons learned and applied to other situations in the SDN WAN settings. We concluded on the solutions that brought together and addressed the security and potential challenges faced in the context of SDN, as well as proposed countermeasures aimed at improving the security and dependability of the SDN WAN settings.

5.4.2 Quantitative

In this section, the applied pragmatic qualitative process discussed in Section 5.4.1 was simulated for CPP objectives and IDS. The processed pragmatic qualitative process was used to quantify, analyse, interpret, and present the feedback results to achieve the formulated CPP objectives and IDS. The quantified output results from the practical implementation were used to answer the formulated CPP objectives and IDs and examine the relationship among the qualitative data features in the SDN networks. The computed qualitative statistical results were observed for the output-based CPP and anomaly attack detection. Then, pragmatic quantitative accuracy, validity, and feedback about the nature of the SDN environment were gained for analysis and evaluation. The

feedback was used to analyse the CPP and IDS, and the operational framework model was built from the drawn conclusions.

5.5 Research Design and Methods

In this section, we present the research design and methods used in this research. We explained in detail the research gap and objectives, literature review, integrated framework design, and implementation, as shown in Figure 18.

5.5.1 Research Design

In the context of the research design, as shown in Figure 18, a preliminary literature review of the nature of the SDN was conducted. The knowledge obtained from the literature was for the formulation of SDN problems related to the SDN environment. Moreover, the research aim based on the research statement and research objectives about the nature of the SDN were discussed. Moreover, they highlighted the research question that addressed the critical challenges faced by the SDN environment and proposed practical solutions to the research questions. The solutions to the challenges involved the deployed number of the controller, switch assignment, load balancing, reliability, processing latency, and reliability. Furthermore, the proposed solutions to the security IDS due to the nature of the SDN were designed and implemented in the SDN environment.

In addition, the collection of data used to experiment was conducted using software tools and technologies. The implementation of the designed model was achieved in the SDN, enabling the WAN to achieve the CPP objectives and IDS. The performance evaluation of the designed model was conducted for the analysis of the results, etc. Then, we validated, and the contribution of the nature of the SDN WAN was discussed. We further provided the future direction of our integrated SDN, CPP, and IDS framework.

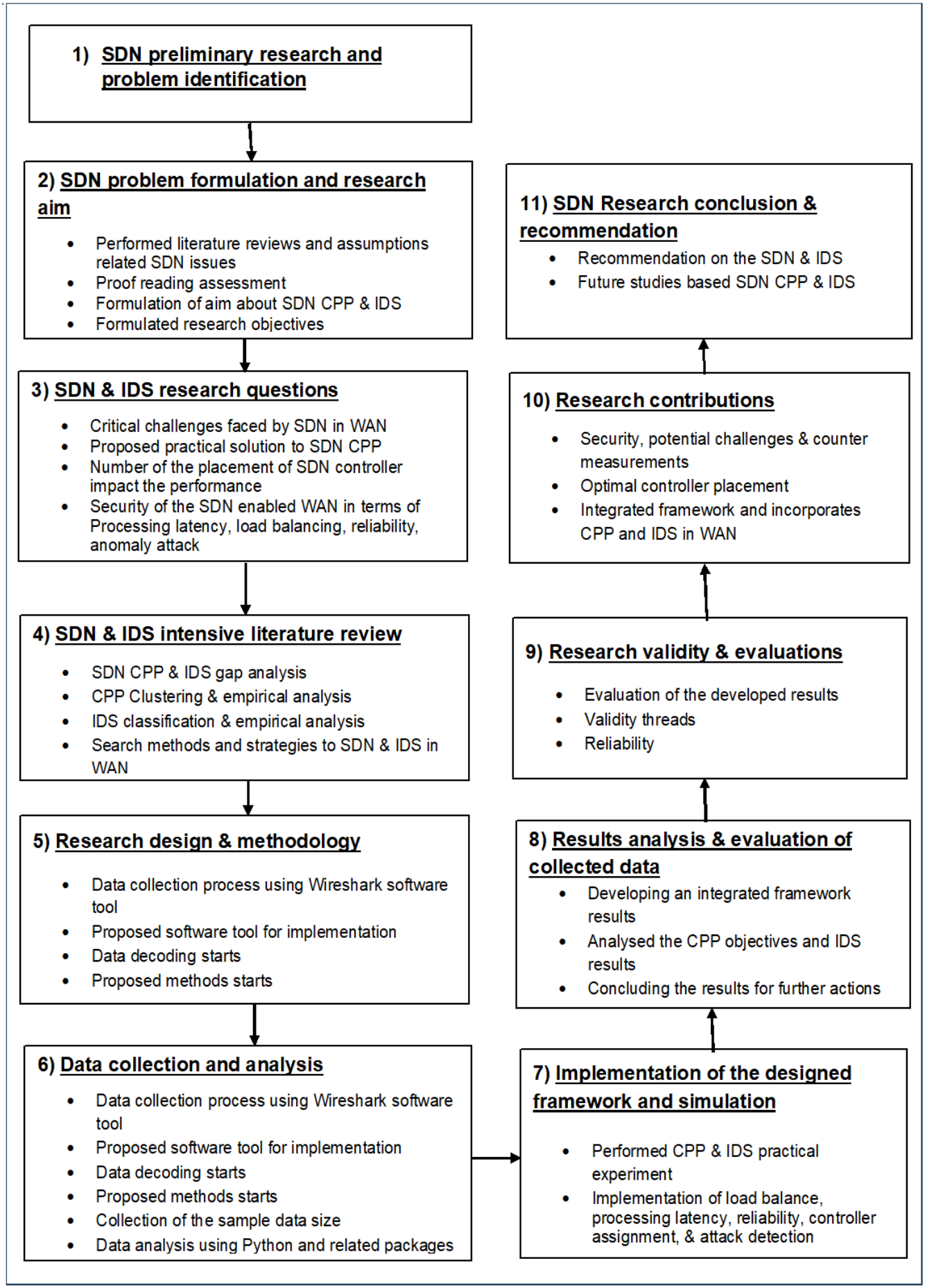


Figure 18: Research design workflow

5.5.2 Research Methods

In the context of this research, the research methods involve a systematic plan that resolves the research problem. It is considered a specific procedure for the collection and analysis of the data.

i. *Literature Review:* In this study, we identified the research gap in the SDN environment, defined our research problem and objectives, and established the theoretical foundation of our research. We reviewed SDN-related papers, etc., for guidance on the research methodology and aligned our knowledge study with the existing studies. We gathered informative concepts, provided a comprehensive understanding of the subject, made informed decisions, and advanced our knowledge.

ii. *Experimental Analysis:* In this study, we achieved performance evaluation, data analysis, security evaluation, benchmarking, etc. We gathered the empirical evidence, performed informed decision-making, provided hypotheses on the related research objectives, and contributed to the improvement and advancement of the research gap, knowledge, and practices.

iii. *Model Design*: We designed the model to provide an understanding of our systems, performed predictions and simulations, optimized the decision-making, validated the solutions, and presented the research outcomes. We employed the model to provide the structure and presentation of real-world phenomena, analysis, and exploration and provided innovative domains in various disciplines within the SDN environment.

iv. *Simulations*: We employed various tools to support our research knowledge and used related software tools with controlled and repeatable environments to study, optimize, analyse, validate, and perform predictions about the system, process, etc. We performed the simulations to access the performance of the network, algorithm. Moreover, we performed predictive analysis for the designed system, validated and tested it, and conducted scientific experimentation to assess the risks and provide decision-making processes.

5.6 Data Collections and Analysis

This section presents the discussion about the collected traffic statistics and analysis. Moreover, the description of the data was discussed in detail. Furthermore, how the data was analysed to gather the qualitative and quantitative information results for CPP objectives and IDS was discussed in detail.

5.6.1 Data Collection and Description

5.6.1.1 Qualitative and Quantitative Data

In the context of this research, we explored our simulated data to gain an understanding of the underlying reasons for the nature of the SDN network. We simulated the traffic network data from the private network and achieved the CPP-IDS conceptual findings related to our collected traffic statistics through experimental observation. Our traffic data consisted of attributes such as device ID, packets, bytes, payload, length, and others. We evaluated the responses based on the traffic flow rules in real-time monitoring. On the contrary, our quantitative data included the numerical value of the data measurements for (a) the number of the received packet, which measures the id number of the packet flow rule; (b) the received data packet, which represents the received time of the flow; (c) the source IP address, which refers to the address of the host in the network; (d) the destination IP address, which refers to the delivered data address; (e) the protocol, which refers to the type of protocol used from the host to the destination; (f) length, which refers to the length of the communicated data packet; and (g) information, which refers to the details about sent and delivered data packets. In our case, we evaluated the analysis of our flow rules based on the response from the implemented model to achieve the CPP-IDS in the SDN network.

5.6.2 Data Analysis

In this study, attribute-based flow rules were used to measure the CPP-IDS in the network. The qualitative and quantitative data were gathered, analysed, and presented in the form of graphs, diagrams, etc.

Once the results-based CPP-IDS was recorded, we compared and analysed the outcomes. The recorded results consisted of the deployed controllers, controller-to-switch assignment, load balancing, reliability, and identified anomaly attacks. Moreover, the ML technique was used to identify the DDoS attack in the SDN network. The results obtained from the simulation were represented in graphs, tables, etc. That is how many abnormal flow rules were classified and identified in the network. The obtained results were compared and represented using the bar and line graphs.

Moreover, we employed tables to record our analysed and simulated flow statistics. To achieve this, we used Microsoft Excel for the visual representation of our analysed data in the form of a table and line graph. Through the use of tables, we summarized the results obtained from the simulated data to gain knowledge about the CPP-IDS in the SDN network.

Furthermore, once the data was simulated, we graphically represented our CPP and IDS data with diagrams and a visual presentation of how the model responded to the simulated traffic flow rules.

Lastly, we empirically evaluated our proposed controllers in the SDN network, and the results obtained from the simulation were analysed by graphs, tables, and diagrams. The employed controllers were compared and practically evaluated. For IDS, we evaluated the employed ML algorithms together with our deployed algorithm. The collected data was presented in a bar graph and table.

5.7 Research Tools and Technologies

In this section, we have presented the technological tools employed during practical experiments. We have employed the GUI application to allow the virtual combination of the devices in the network to achieve the CPP in the SDN network. Moreover, we have employed the 1.15 controller to provide communication services in the SDN. Furthermore, we have employed the KNN ML technique for the analysis of threat detection in the SDN environment. These software tools were running on a personal computer (PC) with the installed Ubuntu 20.04 version operating system having a core ™ i7-10610U CPU processor and 16 GB random access memories (RAM).

i. *Ubuntu 20.04 LTS:* is an open-source software operating system that can be able to run from the desktop, cloud, and the connected internet. We downloaded and installed the operating system from the site: https://ubuntu.com/download/desktop. The operating system was used to run the different types of the controller software.

ii. *1.15:* is an open network operating system that provides the control plane for the SDN, that manages the components of the network such as switches, links and other software programs or modules. The software was downloaded from the site: https:///Downloads. The software controller was used to perform the CPP analysis such as latency, reliability, switch assignment, load balancing and IDS for security counter measurement.

iii. *OpenDayLight:* is a modular open platform for customizing and automating networks of any size and scale. The software was downloaded from the site:

https: //www.opendaylight.org/getting-started/downloads-and-documentation. The software controller was used for the CPP performance evaluation of the designed framework.

iv. *Ryu:* is a component-based software-defined networking, the controller can provide the software components with API to create the topological network, control application, and management. The controller can support different protocols for managing the devices of the

network. The software was downloaded from the site: https://ryu-sdn.org/. The controller software was employed for the performance evaluation of the adopted controller.

v. *Floodlight:* is an open-source SDN controller developed by the big switch network. It is supported by the community of developers, including the number of the engineered from the big switch networks. The software was downloaded from the site: https://sourceforge.net/projects/floodlight/. The software controller was used for the evaluation of the controller.

5.8 Research Evaluation, Validity and Reliability

In this section, we presented the strategies taken to ensure the evaluation, validity, and reliability of this research.

5.8.1 Evaluation

This subsection presents the list of measurements and metrics employed for the performance evaluation of the SDN network.

a) Processing Latency: was employed as one of the CPP performance measurements; it was used for the evaluation-based time delay that occurs during the processing of the network traffic information. In this measurement, we evaluated the time taken for the system to receive the traffic input data, perform the necessary operations, and produce the results.
b) Load balancing measures the distribution of the traffic flow rules across multiple paths to optimize the performance of the SDN network. Moreover, the measurement was used for efficiently distributing the incoming network traffic flow rules among various paths to avoid congestion, etc.
c) Reliability: This measures the network's ability to consistently deliver the services of the network and connectivity during failures or disruptions. It was used for the assurance of the operational network and its availability to meet the requirements of the connected devices in the network.
d) Distributed controller: measures the appropriate number of required controllers based on the specific architecture of the network and the desired operation and scale of the SDN environment.
e) Security: The measurement was used to secure the SDN environment as the central point of defence in the network. Implementation-based IDS security was used to safeguard against potential malicious attacks on the network.

f) Classification Accuracy: A measurement metric was used to evaluate the effectiveness and performance of a classifier. We measured the degree and correctness of the model against the classification and prediction of the network traffic statistics for any given input data.

5.8.2 Validity Threats

In this section, several threats invalidate the results of this research. As discussed in the previous section, the nature of the injected anomaly attack cannot be compared with other attacks to perform the same operation. We have launched the anomaly attack to mimic the nature of the attack in the network. Thus, in this study, adjustments were made to affect the predictive capacity of our model. However, there might be other approaches that can be used and suitable metrics that can outperform the approach we have employed in this research. Again, there might be other improved algorithms, technological tools, etc. that can be adopted to improve performance. The implementation of this research was based on the performance of the SDN-trained model; hence, there might be other suitable approaches or metrics that we did not see or utilize in this research that can be implemented on the real network to improve the predictive power of the model. Hence, we cannot generalize our recorded results in this research to the placement of the controllers, intrusion detection, and mitigation strategies in the SDN-enabled WAN settings. However, we are confident the framework approach discussed in this research can produce high accuracy in controller placement, intrusion detection, and mitigation of anomaly attacks in SDN WAN settings if adopted.

5.8.3 Reliability

For reliability in the context of this research, we have employed the comparison approaches of the various controllers to prove the performance of our model. We evaluated the efficiency and effectiveness of the proposed models based on CPP in the SDN environment. To achieve this, we evaluated our proposal and performed a comparison with other controllers, including Ryu, Floodlight, and OpenDaylight, for performance accuracy.

For the reliability of anomaly flow detection, we performed a comparison with other well-known ML classification and detection algorithms to prove that our proposal outperforms others in terms of detection rate and others. Then, the models were analysed for the detection of the anomaly attack in the SDN-enabled WAN settings**.**

5.9 Administration and Ethical Consideration

The proposal for this study was developed to address the challenges faced by the SDN network; the literature studies performed were conducted based on the identified gaps from the related review. The research questions and objectives were drawn from the identified problem statement to address the main challenges faced by the SDN network.

The ethical issues are part of everyday research. There are major dimensions when dealing with ethical issues in research that are considered procedural ethics and the practice of research ethics. Procedural ethics involves seeking the relevant approvals for conducting the research. In this study, consent for the proposal and approval were sought from the faculty of the departmental committee of the CS department, thereby ensuring that the study of this research was relevant to the field of CS and information systems. Moreover, procedural ethics is the ethics in the practice of research, which involves informed consent and approval of the research about the involved approach in this study.

To ensure the correctness of our proposal, the consent form approval from the faculty was signed, provisioned, and acknowledged to allow the ownership of the research. Moreover, the signed form provided the rules for the committee to refuse the researcher's participation in the research study. The consent form provided great responsibility to ensure that no research participants should be involved.

The literature studies performed in this study were only related to the main objectives of this research; no additional information was added to support the collected surveys. The selected models were chosen based on the empirical performance analysis of the study. The results obtained were used to gain knowledge and are only related to the approach of this research. Only the comparison model was considered for the performance of this study. For this reason, we do not contest that other models can outperform the selected models. Moreover, the data used for the simulation was downloaded and utilized for this study. The designed CPP and IDS frameworks were practically implemented on the designed SDN-enabled WAN.

Hence, the distribution of the data is not guaranteed to obtain the same results as other practical approaches. The designed framework and results should not be interpreted to tackle the challenges of the network but as a practical implementation and applicability in the network. The results obtained from this research should not be evaluated with other metrics to analyse the performance of the model. Moreover, the same comparison evaluation should be followed to obtain the same results; no other techniques should be followed to evaluate the framework.

5.10 Chapter Summary

This chapter presented an overview of the adopted methodology, methods, and tools used to achieve the objectives of this research and to answer the research questions. It first discussed how the research methodology, design methods, etc. were employed in this study. The chapter discussed how the dataset was extracted and simulated for the analysis and evaluation results. Moreover, the research paradigm and the adopted paradigm in this research were presented in detail. The qualitative and quantitative approaches to this research were presented in detail. The adopted research tools and technology, as well as the collection and description of the data from this research, were presented in detail. Furthermore, administration and ethical considerations were considered for this research. Chapter 6 presents the integrated framework, which is considered the main goal of this research study.

Chapter 6

Integrated Controllers Placement-IDS Framework

6.1. Chapter Outline

This chapter presents an integrated CPP framework and IDS. It first discusses the introduction, followed by a detailed overview of the CPP, problem formulation, equations, and others. Moreover, the chapter presents how the parameters, algorithms, and others were utilized together to achieve both CPP and anomaly detection. Furthermore, the IDS mitigation policies were presented in detail.

6.2 Chapter Introduction

In this chapter, we designed an integrated CPP and IDS framework to devise the security means for the number of placed controllers in the SDN environment. The designed framework is positioned to address the CPP objectives: deployed number of the controller, controller to switch assignment, processing latency, reliability, etc. in the SDN environment. Moreover, we have presented the CPP problem formulation to optimize the SDN environment for the number of controllers placed in the network, etc.

On the other hand, the designed CPP framework integrated the ID system that detects the anomaly attack in the SDN network. The IDS modules: traffic monitoring and classification, detection, and mitigation of attacks, as well as providing a report about the mitigated attack in the network, were discussed in detail. The essence of the designed IDS was to provide a defence mechanism for the number-generated flow statistics against attackers. Moreover, how the algorithm, metrics, problem formulation, and others were employed together for the designed framework to gain knowledge about the nature of the SDN environment was presented in detail.

6.3 Proposed Controllers Placement

6.3.1 Overview

In this section, we presented an overview of the proposed CPP in the SDN environment. Moreover, we presented the problem formulation of the deployed number of controllers, controller-to-switch assignment, processing latency, load balance, reliability, and others. We have presented the formulation of the CPP objectives by using the equation given by,

$z' \in [x_i, y_j] \; \forall \; v \in V$.....*(6.1)*

where z' denoted the active controller in the network, *v* denoted an independent traffic statistic such that V is the overall collected traffic statistics in the network,

$z' \rightarrow V$.....*(6.2)*

Moreover, we considered eq.(6.2) i.e., controller to switch assignment, where v denotes the independent collected traffic statistics associated with the controller z',

$d(v, z') \in MinPts = \beta \leq \varepsilon$ .....*(6.3)*

The distance between the z' and v was maximized such that the *MinPts*= β denoted the minimum independent collected traffic, *v* belongs to the maximized distance *D*, and ε denoted the maximized distance within the subdomain cluster, and

$z'' \ni \sum_1^n V$ s.t $C_{LBD} \leq \varepsilon$.....*(6.4)*

Moreover, z'' denoted the overall placed controllers within the set of collected independent traffic statistics such that C_{LBD} denotes the controller load detector.

In general, the designed CPP achieved the processing latency, load balance, deployed number of the controller, and others during the shared number of connected devices in the SDN environment.

6.3.2 Problem Formulation

In this section, we presented the problem formulations of the CPP for the number of deployed controllers in the SDN environment.

We deployed the number of the controllers *(N_{ctr})*, that is how many required controllers are for the designed network topology. Secondly, we evaluated the number of the generated flow stats on the assigned controller to switch subdomain networks. Thirdly, we evaluated the minimized or required processing latency, that is the intra and inter-processing latency for the number of the connected controllers. Fourthly, we calculated the load balance detector (C_{LBD}) on the using eq.(6.8- a&b). Lastly, we evaluated the maximized processing latency to achieve reliability using eq.(6.9) such that the overall equation for the CPP objectives is calculated using eq.(6.10).

6.3.3 Objective Parameters

In this section, we discussed the objective parameters used to achieve the CPP objectives. To evaluate the deployed number of the controllers, we defined the parameters.

$N_{ctr} = min \sum_{k=1}^{n} L(C_k)$.....*(6.5)*

Eq.(6.5) presented the parameters utilized to evaluate the optimum controllers. C_k denoted the k_{th} number of the placed controller $L(C_k)$.

$$N_{Ctr \rightarrow Sw} = min \sum_{v \in V}^{k} d(v, z') \quad \text{.....(6.6)}$$

Eq. (6.6) presented the controller to switch assignment connection. The $d(v, z')$ denoted the shortest distance evaluated between the controller z' and switches ($v \in V$). We also evaluated the processing latency i.e., inter-processing latency $a(i)$ = S2C and intra processing latency *b(i)* = C2C. The $a(i)$ denoted the shortest inter-distance (controller to switch), and $b(i)$ denoted the shortest distance (controller to controller) that minimized the processing latency.

$$C_{Capacity} = \frac{Collected\ traffic\ statistics\ \varphi(x)}{Oveall\ placed\ controller\ (\varphi(z''))} = \varphi(y) \quad \text{.....(6.8a)}$$

and

$$C_{LBD} = \frac{\beta(x) \times d(v, z')}{Collected\ traffic\ statistics\ \varphi(x)} \quad \text{.....(6.8b)}$$

Eq. (6.8b) was used to compute the load balance detector (C_{LBD}) on the placed controller z', $\varphi(x)$ denoted the collected traffic statistics, $\varphi(z'')$ denoted the number of the placed controllers in the network, and $\varphi(y)$ denoted the capacity of the single controller. We evaluated that the load on the controller should not exceed its capacity in the network i.e., $C_{LBD} \leq \varphi(y)$.

$$MCP_{Ctr[1] \rightarrow Ctr[2]} = MCP_{Ctr[2] \rightarrow Ctr[3]} = \frac{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} - a(i)}{\max\{a(i), b(i)\}} = \theta \quad \text{.....(6.9)}$$

Furthermore, we computed the multiple control path (*MCP*) between the controllers z''. We computed the C2C and S2C time processing using the multi-path alignment strategic method. That is, the time required to forward the load of the failed controller to other active controllers should not exceed the required processing latency in the network.

$$CPP = N_{ctr} + N_{Ctr-Sw}(C_{LBD} + Re(P_i)) + \max \sum_{k=1}^{n} S$$

$$= \min \sum_{k=1}^{n} L(C_k) + \min \sum_{v \in V}^{k} d(v, z') + [C_{LB} + Rel(P_i)] + \max \sum_{k=1}^{n} S \quad \text{.....(6.10)}$$

Lastly, we computed the overall objective parameters defined by *eq.6.10*. N_{ctr} denoted the evaluated controller placement. Moreover, N_{Ctr-Sw} denoted the controller to switch assignments, C_{LBD} denoted the computed load balance detector, and $MCP(P_i)$ denoted the multi-control paths between the employed controllers, and *S* ensures the maximum security in the SDN network.

6.3.4 Optimum Placement Algorithm

All the procedures required for optimal placement in the SDN environment are summarized in Figure 19. Once the traffic network is active, line 1 clustered the number of the generated traffic statistics to the deployed controllers in the network, line 2 clustered the k-nth flow stats belonging to the subdomain cluster network, line 3 checked the generated flow stats belonging to the normal and abnormal data, line 4 evaluated step 3 for the anomaly flow data, line 5 validated the data and the output belonging to the anomaly data, line 6 raised the action alert of the data belonging to the anomaly data, line 7 implemented the enforced policy on the detection data alert, line 9 forwarded the evaluated data belonging to the normal stats, line 10 counted the number of the flow stats on the subdomain cluster networks, line 11 counted the number of the controller assigned to the switches, line 12 computed the required load on each controller, i.e, that is the number of the flow stats belonging to the subdomain network.

Controller Placement Algorithm:

Input: Receive the generated traffic data $v \in V$

Output: Number of controllers, clusters, and switches

1. Cluster the number of traffic and overall controllers.
2. Cluster the *k* - nth flow stats feature data ***x's*** **and** ***y*** **subdomain cluster**
3. heck the data belonging to normal and abnormal data
4. Switch (**d (**x's**, y) == k)**
5. if **d (**x's**, y) == k** → belongs to anomalous flow event (A)
6. alert and count **d (**x's**, y) == k** → anomaly storage event list (A's)
7. implement function policy 1 – 4 on the data
8. else
9. Forward the flow data belonging to the normal
10. Count the number of the cluster subdomain flow stats
11. Count the number of switch assignments and controllers
12. Calculates the load required on the cluster subdomain network
13. End

Figure 19: Optimum placement algorithm

6.4. kNN Anomaly Flow Detection Technique

In this section, we present the applied knn anomaly detection technique in the SDN network. We discuss how the generated traffic flow statistics were used and evaluated for classification and characterization, intrusion detection, mitigation, etc. in the network. The IDS network traffic stats evaluated in this section are presented in Appendix A.

6.4.1 Intrusion Detection Architecture Design

This section presents the designed IDS architecture over the SDN environment. Moreover, the designed architecture showed how its components interacted with the SDN network, etc. The

essence of the designed model was to identify attacks in the SDN environment. We merged the designed IDS with the SDN architecture to potentially provide a secure SDN environment during traffic communication. Moreover, our model provided a positive effect and increased security efficiency in the SDN environment. Again, our IDS modules found in the control plane controller were used for attack alerts, detection, etc. The kNN anomaly detection-based SDN architecture is shown in Figure 20, and the process is shown in Figure 21.

IDS is the security mechanism designed to monitor and analyze the traffic on the network and to respond to potential security or malicious activities. The main aim of IDS is to identify and alert the security administrator about malicious activity, etc. This approach enables the administrator to quickly respond to security incidents, minimizing the potential damage, etc., to the network. In our study, we employed anomaly-based detection, which involved creating the baseline of normal behaviours for the network to flag any deviations from the baseline as intrusions. Since our designed IDS model is meant to identify anomaly flow based on the network, We employed the supervised traffic flow statistics performed on the ML technique together with various level modules: characterization and classification, detection and mitigation, etc.

For the operation, our IDS model received the traffic flow statistics in the following format: deviceId, bytes, packets, duration, priority, tableid, and payload. The traffic flow rules extracted were evaluated for the correct inputs, predictions, and outcomes of the model. The predicted outcome of the flow rules was the payload associated with the input traffic flow statistics in the network. The traffic flow rules were fitted and classified as normal and anomalous. We iterated our model to read the generated traffic flow statistics from $1 \rightarrow n$ row attributes and performed the detection on the data, where $k = (A's$ – anomalous attack). Figure 20 presents the designed intrusion detection architecture.

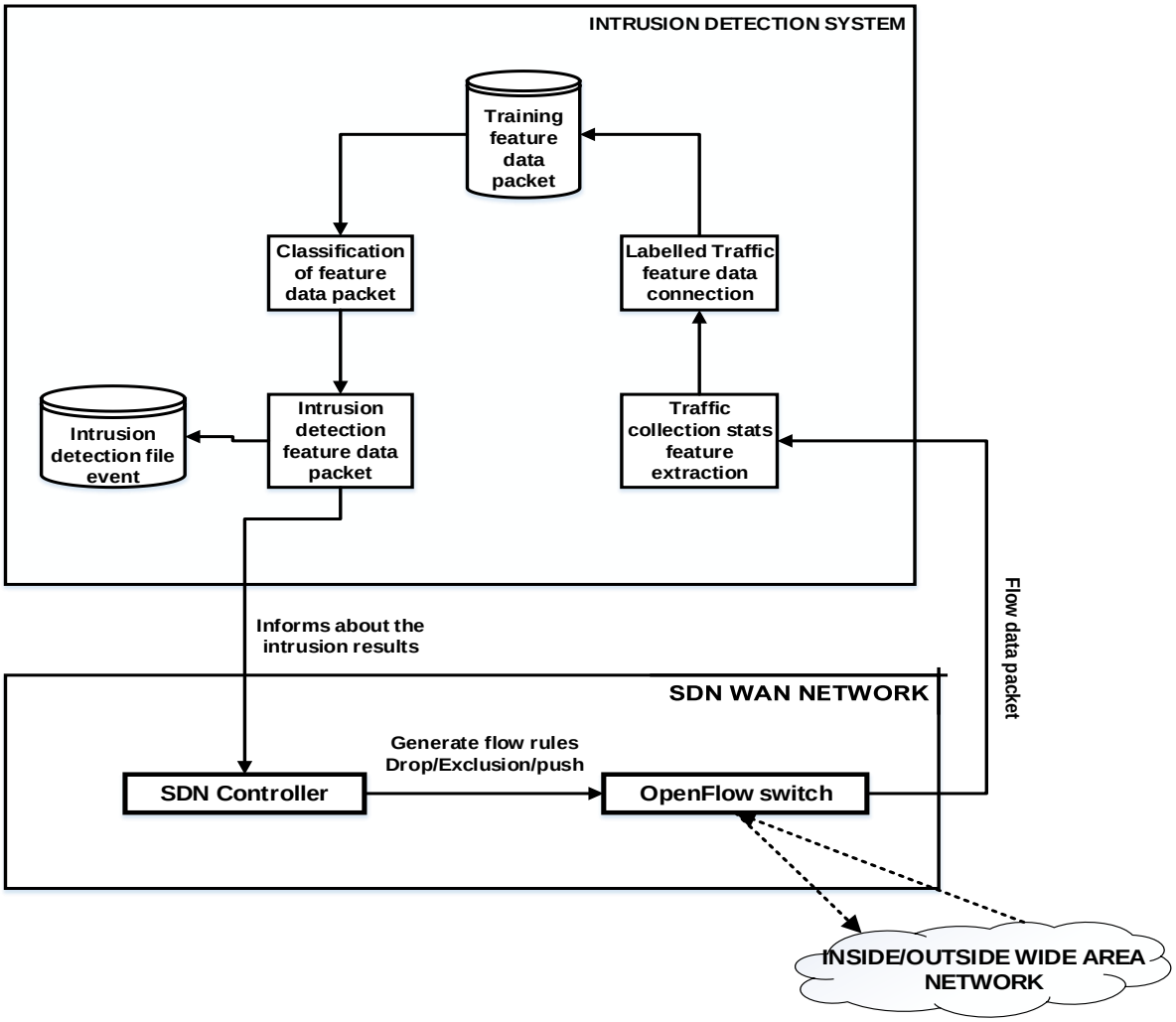


Figure 20: Designed intrusion detection architecture

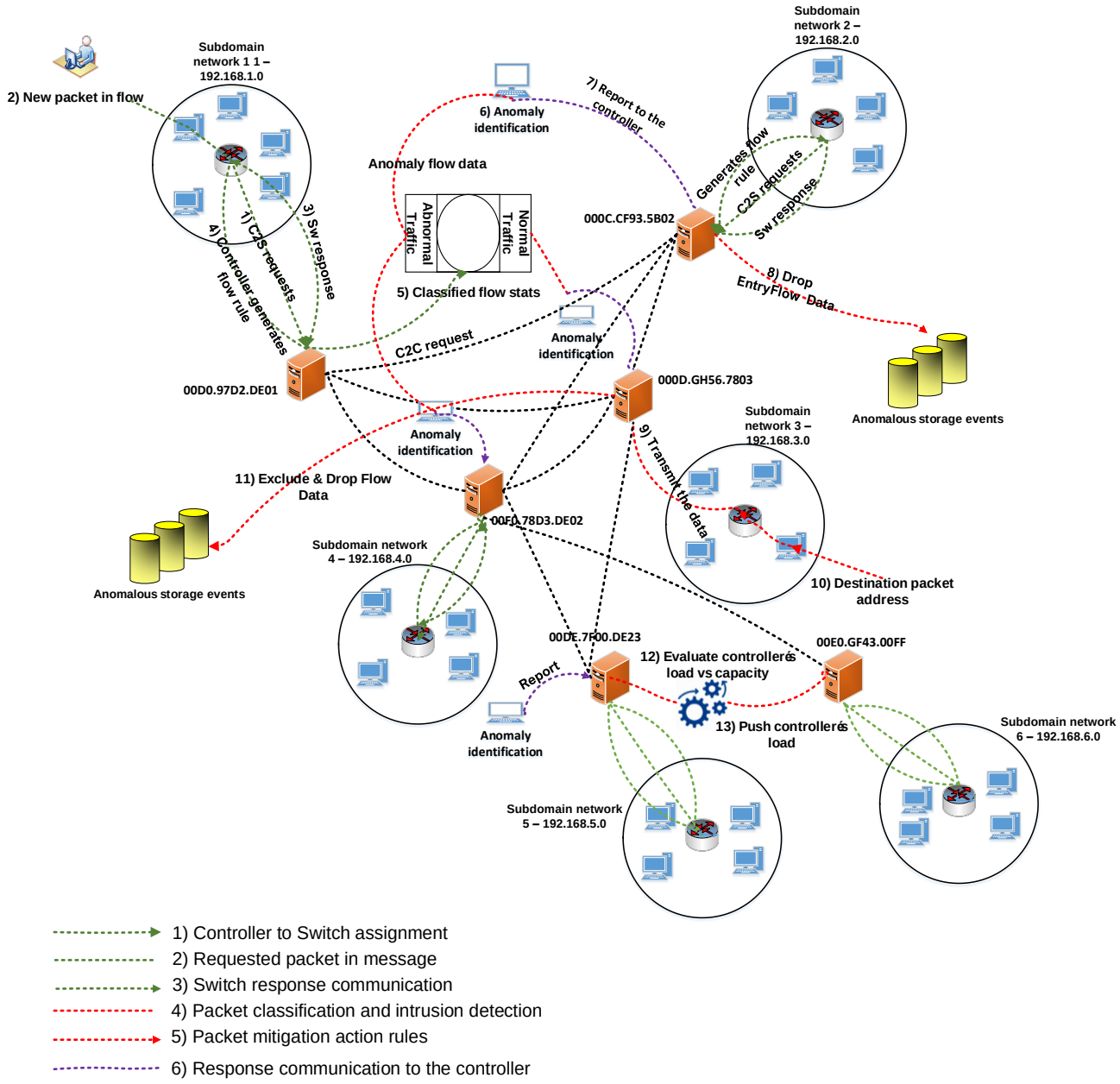


Figure 21. Proposed intrusion detection flow process

6.4.1.1 Traffic Collection Module

In this module, we evaluated our traffic flow rules from the SDN network. We performed the traffic characterization on the generated flow information and periodically exported it to the anomaly detection module. In our case, we considered the native OpenFlow (OF), which relied on the OF protocol for the collected flow statistics. The requested OF statistics were performed by the controller for the ordinary collection of all the flow entries residing in the OF switch and their corresponding counters. The required flow entries in the flow table were a combination of variables, as shown in Table 26.

The evaluated traffic statistic was accomplished by switches replying to the flow statistics requests from the associated OF controller. The switch replied with large chunks of the flow table contents, etc. Each chunk contained a portion of the flow entries corresponding to the packet counters of each flow.

Table 26: Extracted features

Extracted feature	**Description**
Deviceid	The device used for the transmission of the traffic flow
Bytes	The bytes associated with the transmitted flow data
Packets	The size associated with the transmitted data
Duration	The time associated with the delivery date
Priority	The priority of the data in the network
TableId	The identity of the table associated with the delivered date
Payload	The payload associated with the data flow

6.4.1.2 Anomaly Detection Module

In this section, the evaluated traffic flow rules were subsequently fed to the anomaly detection module. For our mechanism, we use the *KNN*-based ML technique for traffic classification and monitoring in the network. Our method inspected and correlated the specific network stats for classification and monitoring.

6.4.1.3 Traffic Monitoring and Classification

In this section, we evaluated the kNN model based on our prepared traffic flow rules for monitoring and classification. The higher the classification accuracy signified a dispersed probability distribution, while the lower denoted the concentration of the distribution. The flow statistics related to traffic feature distribution employed for this work are presented in Table 27. We performed the traffic characterization on the flow statistics and extracted the output in the form of normal and abnormal traffic profiles. We tested the performance of our model using the following metrics: classification report, F1 score, and precision.

Table 27: Normal and anomaly effects on the traffic features

k	Type	Affected feature traffic pattern				
		DevideID	Byte	Packet	Info	Length
1	Normal	+/1	+/1	+/1	1	+/1
2	Anomaly	-/0	-/0	-/0	0	-/0
3	Anomaly	-/0	-/0	-/0	0	-/1
4	Anomaly	-/0	-/0	-/0	0	-/1
5	Normal	+/1	+/-	+/-	1	+/1

6.4.1.4 Anomaly Identification

In the previous subsection, we trained and characterized the traffic flow statistics during traffic monitoring and classification. In this study, we perform intrusion detection in the SDN environment, which is how we can detect the oncoming DDoS anomaly attack in the network.

In this study, we configured the controller to perform the attack detection on the flow statistics associated with the DDoS packet information in the SDN network. The main aim was to evaluate how the configured controller generates an alert on the packet associated with the DDoS attack and provide mitigation actions for the number of oncoming traffic flow rules in real-time monitoring. We configured the controller and defined the rules that trigger an alert when the traffic exceeds a certain threshold. That means creating and defining the rules that check the traffic statistics against the defined anomaly threshold.

To perform the attack detection, we evaluated the input attributes that is deviceId, bytes, packets, duration, priority, & tableId. The payload **denoted the corresponding traffic flow output *d(x, y),* and a certain threshold denoted anomaly class attack such that *d (x's, y)* == *k,*** where *k* = anomalous event *(A).* The traffic flow was alarmed as true if anomalous *(A),* otherwise normal traffic communication. The model was continued while ***d (x, y) != k***. The traffic flow stats associated with malicious attacks were further processed in the mitigation module**. In Figure 22 we present how the intrusion detection module was handled during the suspicious anomaly attack in the SDN environment.**

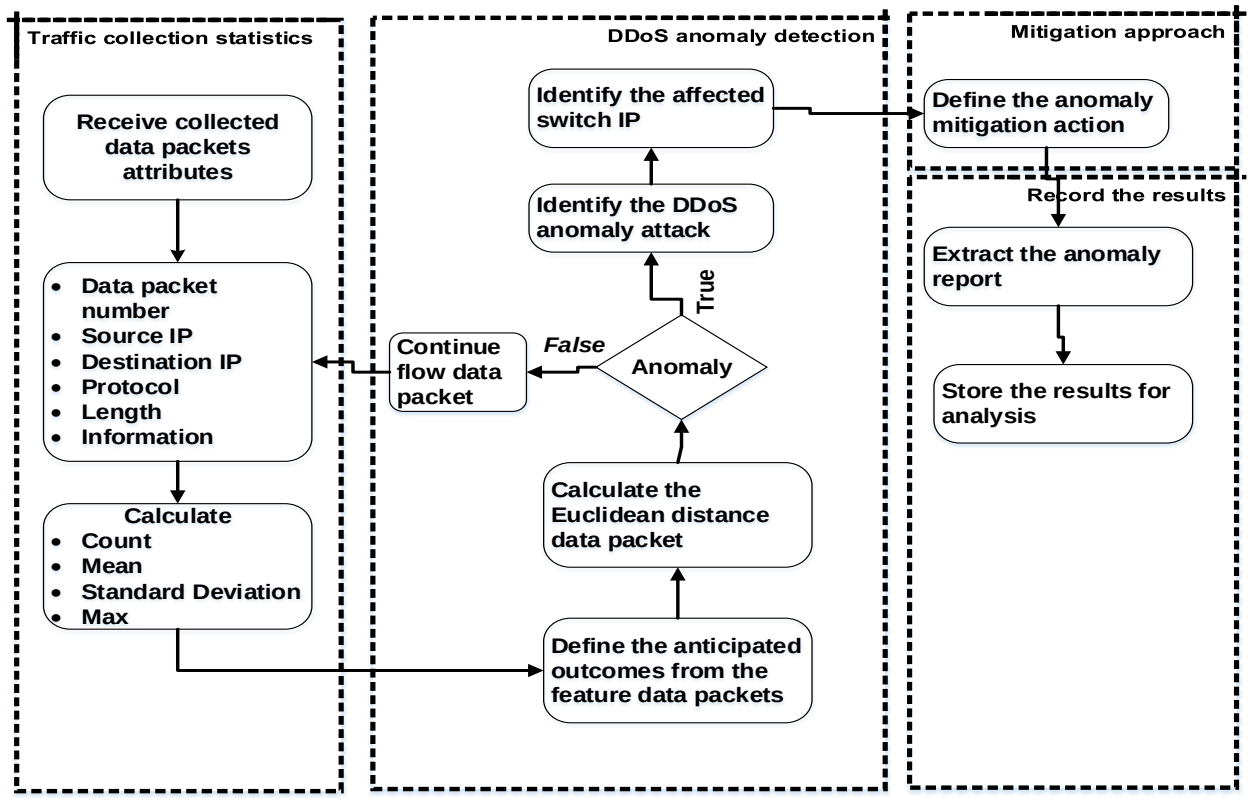


Figure 22: Anomaly detection and mitigation flow process

6.4.1.5 Mitigation Module

In this study, the flow rules **associated with the identified anomaly attacks are directed to the mitigation module for further processing.** While the anomaly detection module provides traffic monitoring and classification, in this subsection, we apply the mitigation policy to the identified anomaly attack in the network.

A. Implemented Policy

The policy in this study defines the rule that determines the presence and future decisions of malicious events in the network. In this subsection, we briefly discuss the adopted policies to achieve the specific actions on the malicious attack in the network. In this study, we defined the event as the identified anomaly attack in the network, rules as the set of conditions to achieve the specific event, and action as the measurement to manage the occurrence of the identified attack. The applied policy includes dropping, exclusion, and pushing. This countermeasurement was used to manage the identified anomaly attack in the network. Table 28 provides a summary of the implemented policies required to manage the occurrence of the detected attack in the network.

Table 28: List of implemented policies

Policy	Description
Drop flow rule	Drop the traffic flow rule associated with the DDoS anomaly attack
Exclude flow rule	Exclude the flow rule belonging to the anomaly DDoS flow attack
Drop entry flow	Drop the entry flow rule belonging to the anomaly DDoS attack
Push controller's load	Push the load's flow to other neighbouring controllers

6.4.1.6 Report Module

Once the policy is applied, the detailed report forming part of the anomalous events is recorded and saved in the generational report. The recorded details were used to predict any form of identical anomalous event intended to invade the resources in the network. Then, any new identical anomaly flow rules and services associated with the recorded details of the anomalous event were prioritized, and the countermeasures action policy associated with the attack was applied before any damage to the resources took place in the network.

6.4.2 The Intrusion Detection Algorithm

In this section, we briefly discussed how the kNN ML technique was applied during the DDoS anomaly flow detection. The applied techniques were used to detect the attack to provide security measures in the network. We provided the detailed step-by-step process and the format algorithm.

6.4.2.1 The kNN - IDS Technique

This subsection presents a detailed discussion of the performed kNN - ML technique to identify the DDoS attack in the network.

In line 1 of the pseudocode, the traffic flow input *x's* is characterized: deviceid, bytes, packet, length etc, **line 2 validated the *y* - the corresponding traffic flow output: payload associated with the input traffic flow *x's*, line 3 of the source code evaluated *d (x, y)* and identified the anomaly attack such that *d (x, y)* == *k*, where *k* =** anomalous flow event *(A)*, line 4 evaluated the conditions: if ***d (x, y) == k* or *m* where (*m)* is the normal traffic flow, line 5 evaluated the condition if *d (x's, y) == k*** and **resulted to** anomalous traffic flow *(A),* line 6 alert and forward ***d (x's, y) == k*** → anomalous traffic flow to the storage event lists *(A's),* line 7 evaluated the while condition ***d (x, y) !=k* that is (*m* - normal traffic flow)**, line 8 forwarded the evaluated data to the destination address, line 9 repeated the step 2 on the newly updated flow data, line 10 of the pseudocode terminates the end of file **and line 11 stopped the program.**

6.4.2.2 The kNN-IDS pseudocode

Distributed Intrusion Detection Algorithm:

Input: Receive and define the collected traffic flow stats ***x's***

Output: Identified anomaly flow rule ***d (x's, y)*** == *k* → ***(A's)***

Steps: 1. Characterised the inputs *x's* traffic flow rules attributes

2. Predict the outcome of *x's* to the corresponding *y* – dependent variable

3. Calculate the square distance ***d (x's, y)*** **from the** received ***x's*** **and** ***y***

4. Check the two conditions: ***d (x's, y) == k or m***

5. if ***d (x's, y) == k*** → belongs to anomalous flow event *(A)*

6. Alert ***d (x's, y) == k*** → anomaly storage event list *(A's)*

7. while (***d (x's, y) !=k) == m*** → **belongs to** normal events

8. forward the normal traffic flow rules x's

9. Repeat statement → line 2

10. Terminate EOF

11. Stop

Figure 23: kNN anomaly detector algorithm

6.4.3 Enforced Policy

In this section, we briefly present the implemented policy on the identified DDoS anomaly attack. The formulated policy was used to manage the DDoS attacks on the network. That is how we can apply the policy to secure the SDN network against the identified DDoS.

6.4.3.1 The Policy Algorithm

This subsection presents the overall function policy-based algorithm to provide countermeasures and management of the identified DDoS attacks in the network.

Line 1 of the pseudocode received the input traffic stats *x's*, line 2 characterized and evaluated the input data *x's* **traffic attributes belonging to the output flow attribute, lines 3 - 5 evaluated the** traffic flow belonging to the anomaly traffic **attributes,** line 6 evaluated ***d (x's, y) == k*** **is true, line 7** forwarded the evaluated normal data to the destination address, line 8 - 13 **evaluated the enforced policy actions (drop, exclude, dropping, push) on the identified traffic flow data, and line 14 stopped the program.**

Enforced Policy Algorithm:

Input: Receive the ***x's*** – traffic attribute flow stats
Output: drop / exclude/push flow data
Steps: 1. Receive the input traffic flow stats *x's*
2. Characterised *x's* → y - payload
3. Knn fits *y* → *x's* – flow rules
4. if ***(x's, y)*** → ***d (x's, y)*** == *k*
5. forward ***k*** → anomalous storage event *(A)*
6. *while* ***(d (x's, y) !=k***
7. *forward* ***d (x's, y)*** → *N* - x's → normal
8. if ***(d (external'k, y) ==k)*** ∈ *e'k* → *(external input attack)*
9. **drop entry *d (e'k, y) == k***
10. **forward** *e'k* → anomalous storage event *(e'k)*
11. else if (*d(internal'k, y) ==k)* ∈ *inter'k* → *(internal input attack)*
12. drop flow *d(internal'k, y) == k*
13. forward internal'*k* → anomalous storage events *(A)*
14. End

Figure 24: Policy-based IDS algorithm

6.5 Integrated Framework

This section presents the designed integrated framework for CPP and IDS in the SDN architecture. Modules including the traffic collector, intrusion detection, mitigation, and report were presented in detail. Furthermore, the CPP objectives including the processing latency, reliability, load balancing, and the deployed number of the controller were presented in below Figure 25.

In the below figure 25, the designed CPP-IDS framework generates the traffic flow stats (v_1, $v_2, v_3 \ldots . v_k$) ∈ V using the rules of the controller. Then, the devices located in the data plane forward the packets according to the rules assigned by the controller. Our designed CPP-IDS consisted of OpenFlow switches, controllers, and links. The enhanced DBSCAN was employed for clustering of the flow stats on a WAN environment. Once the flow stats are requested, the KNN algorithm is used to detect the DDoS attacks in the network. Furthermore, we evaluated the criteria that include the processing latency, switch assignment, number of controllers, load balancing, reliability, and security in the WAN.

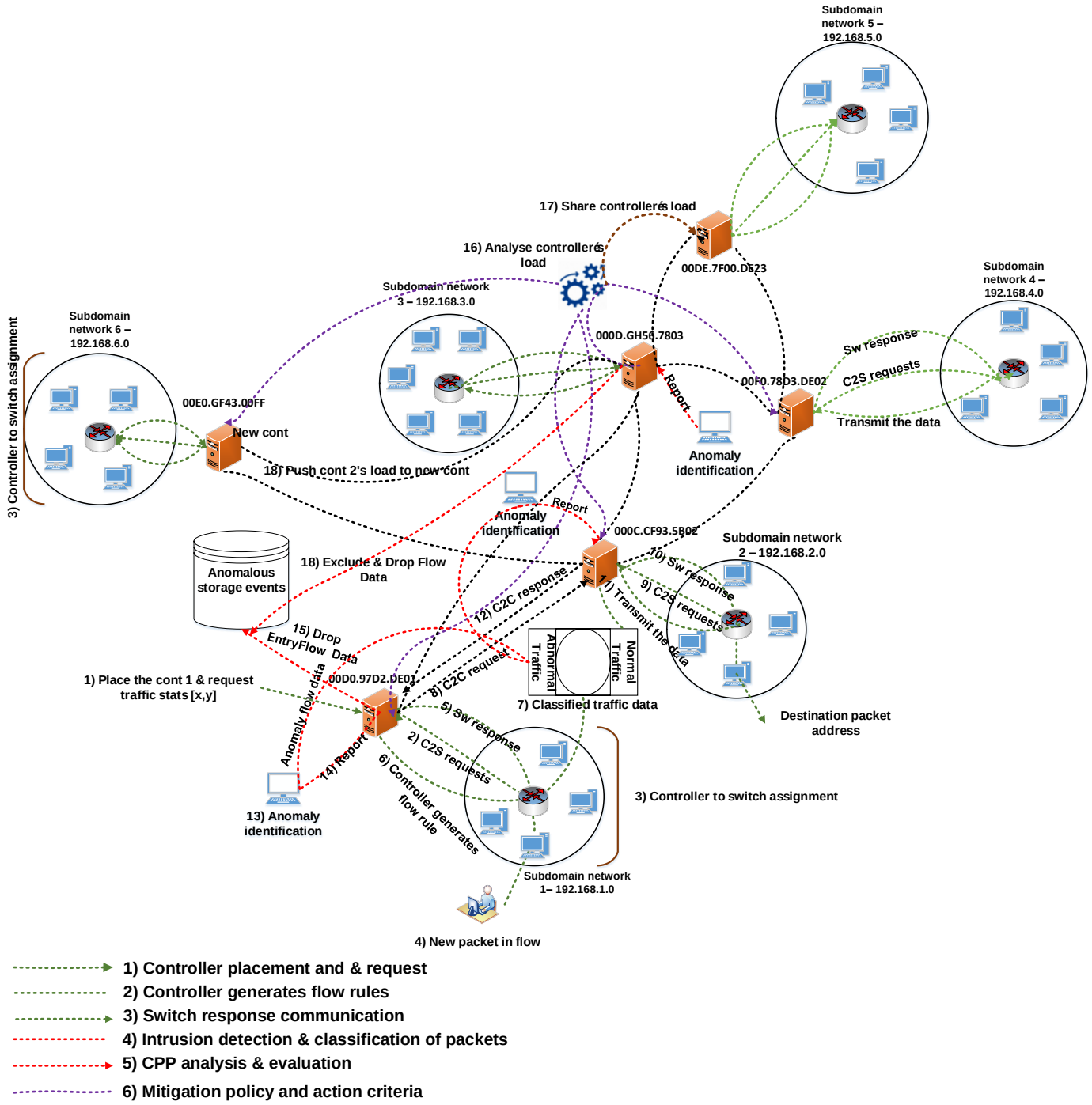


Figure 25: Integrated controllers' placement - IDS framework

In Fig. 25, we present an integrated CPP-IDS framework that achieves the best trade-off between network performance and security in the SDN-based WAN. The designed framework collectively tackles the solutions through SDN CPP-IDS that are both efficient and attack-aware. The ML technique was employed during the detection of the attack to address the security issues in the network. Moreover, the load balancing, processing latency, reliability of the controllers, etc. during the exchange of control messages were addressed in detail.

A. SDN Network

The designed framework presented in Figure 25 consisted of several SDN switches i.e., (v_1, $v_2, v_3 \ldots . . v_k$) $\in$ V. Our deployed controllers located in the control plane requested the traffic flow statistics *x's* from the switch v_1. The switch device located in the data plane forwarded the traffic flows using the instructions from the controller. Then the collected traffic flow stats were further evaluated for both CPP and IDS.

B. Controllers' Placement and Switch Assignments

To achieve the switch assignments for the number of deployed controllers, we evaluated the generated traffic flow statistics *x's* to determine the number of required controllers in the network. We simulated the SDN environment and evaluated the increased number of deployed controllers related to the generated flow statistics. That is how many OVS, host devices, links, etc. are required for the increased number of deployed controllers and assigned switches in the network.

C. Network Intrusion Detection

The IDS module found in the control plane controller was evaluated for attack detection in the network. We received the generated number of the traffic flow statistics *x's* and performed the attack detection. We configured the controller to classify and evaluate the traffic flow statistics associated with the DDoS packet information in the SDN network. The main aim was to evaluate how the configured controller generates the alert based on the traffic packet associated with the DDoS attack and provides mitigation actions based on the number of oncoming traffic flow rules in real-time monitoring. To achieve this, we configured the controller and defined the rules that trigger an alert when the traffic data exceeds a certain threshold. That means creating and defining the rules that check the traffic statistics against the defined threshold.

D. Report

Once the anomaly traffic flow statistic is identified, the details of the traffic results are recorded as a report. That is, for any identical anomaly, traffic flow associated with the report results was prioritized, and the relevant action policy was applied to minimize any damage to the resources.

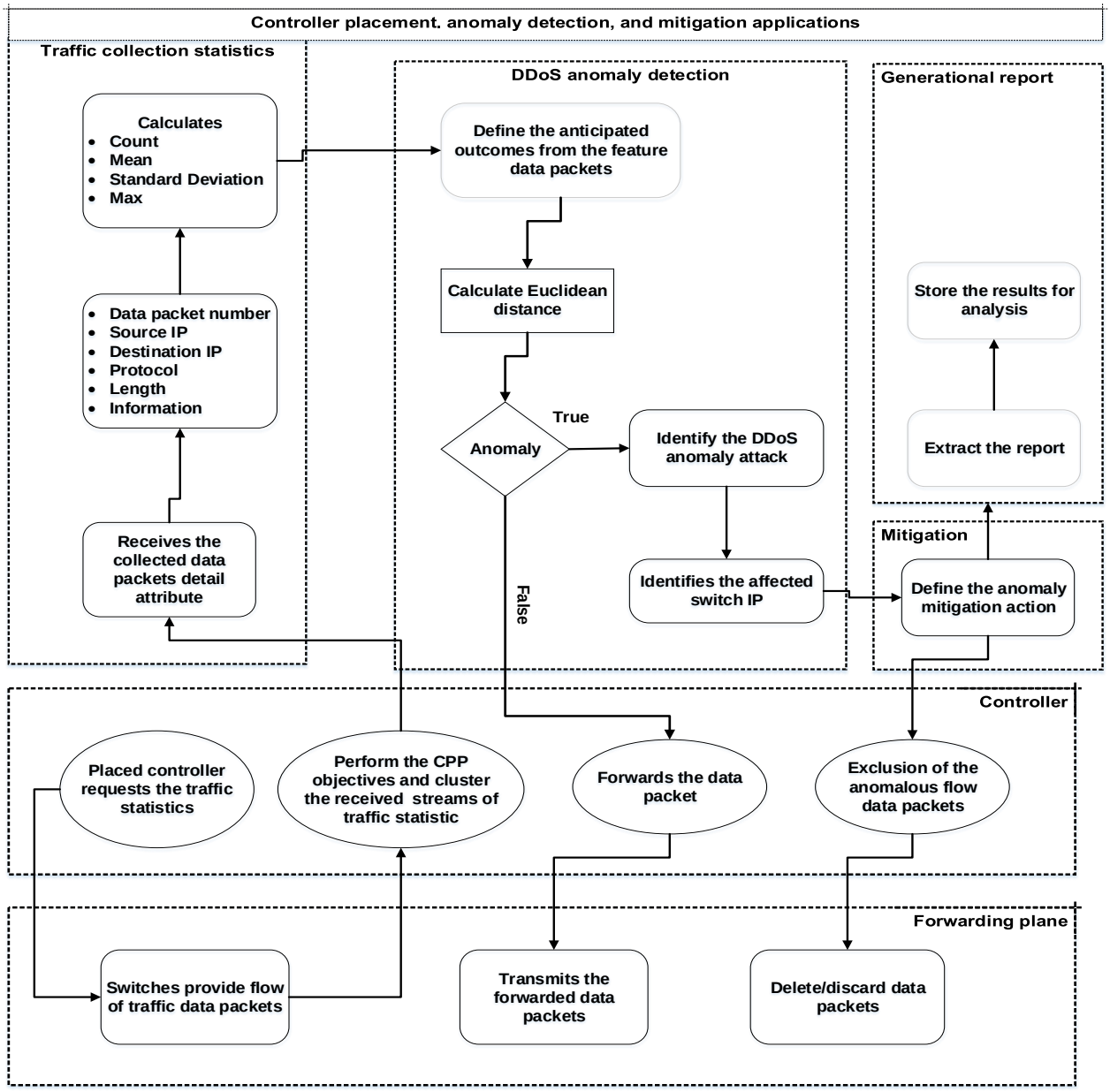


Figure 26: CPP – IDS process

Figure 26 presents the CPP-IIDS process in the SDN environment. The process explains how the SDN environment was utilized for the traffic and collected statistics. Moreover, the process presents how the DDoS anomaly attack was detected in the SDN environment during the anomaly detection module. Once the DDoS was identified, the details of the packets were pushed to the mitigation module, where the enforced policy was implemented in the SDN environment. Lastly, the extracted information about the attacks was stored in the generational report.

6.6 Chapter Summary

This chapter presents the integrated framework for IDS-CPP in the SDN environment. The essence was to design an automatic framework that is more effective and efficient in detecting the attack while achieving the CPP objectives to devise a security means for the number of placed controllers in the SDN environment. Moreover, the problem formulation, including the number of required and deployed controllers, switch assignment, processing latency, etc., was discussed in detail. Furthermore, the designed IDS and the related functions were discussed in detail. In the next section, we present the implementation of the designed framework as well as the results.

Chapter 7

Implementation and Results

7.1 Chapter Outline

This chapter presents the implementation of the integrated CPP-IDS framework presented in Figure 25. It first presents the chapter introduction, system properties, etc. Moreover, the simulation setup and parameters were presented in detail.

7.2 Chapter Introduction

In this chapter, we presented the implementation of an integrated CPP–IDS. The implementation performed the number of the required controllers *(N_{ctr})*, switch assignments (N_{Ctr-Sw}), processing latency in terms of controller-to-controller (C2C) and switch-to-controller (S2C) delays, controller processing load or load balance detector (C_{LBD}), and reliability to avoid a single point of failure while ensuring the security of the network via IDS in the SDN-enabled environment.

Based on the above discussion, we have performed our experiment on the Ubuntu 22.04 operating system. Our simulation was performed on the virtual machine configured, tested, and troubleshooted in the virtual SDN-enabled lab environment. Moreover, an open network operating system () X-Wing (LTS) controller was employed to manage the network components, such as switches, links, hosts, etc., and to assist with communication services and other neighbouring networks. On the other hand, we evaluated the detection of the kNN ML technique-based DDoS attack in the network. Moreover, we evaluated the performed IDS models to prove the efficiency of our model. The detailed step-by-step process and the format algorithm were presented in detail. Furthermore, the policy was enforced to provide countermeasures and management of the identified attack.

In the context of the CPP, we have evaluated our controller with the widely known controllers: OpenDayLight, Ryu, and Floodlight. These controllers were adopted based on individual robust and flexible frameworks that manage and efficiently control the network devices. They have a built-in modular architecture that allows the user to remove and add device components based on specific needs. Moreover, they can allow the extensive integration of new features, applications, and protocols and adapt to the various use cases and requirements of the network. The controllers are built with various protocols and APIs, such as OpenFlow, NETCONF, etc., that can provide efficient communication on a diverse range of network devices and applications. Lastly, they provide a dynamic network configuration that manages network devices, handles network policies,

traffic flows, and behaviour of the network, and allows suitable automation as compared to other controllers.

In this study, we evaluated the performance of the adopted controllers based on the deployed number of controllers, processing latency, load balancing, and reliability in the SDN environment. The results obtained from the experiment were compared and analysed, and a conclusion was drawn based on the controller CPP's effective performance.

7.3 System Properties

To analyse the effectiveness and efficiency of our implementation, several parameters etc. were employed during the practical experiment and presented in detail in the next subsection.

7.3.1 System Properties

In this subsection, we presented the system parameters employed during the practical experiment. The involved parameters include software, hardware etc. as shown in Table 29.

Table 29: System properties

Parameter	Value
Software	Ubuntu 20.04 LTS, 64-bit operating system, x64-based processor
	Mininet
	Pycharm 5.02
	X-Wing (LTS) version 1.15.0 API-1.15.0
	Docker container (DC)
	Ryu
	Opendaylight
	Floodlight
	Java JDK 17
	Virtual machine (VM)
Hardware	512GB hard disk SSD
	RAM 16,0 GB
	Processor: Intel(R) Core(TM) i7-10610U CPU @ 1.80GHz 2.30 GHz

7.3.2 Simulations Set-up

In this subsection, we presented the simulation set-up performed in this research. The system properties presented in Table 29 were employed in the practical experiment to achieve the CPP - IDS in the SDN environment. The performed experiment was run on the Ubuntu 20.04 LTS, 64-bit operating system i7-10610U CPU processor, installed on the 512GB hard disk SSD, with RAM 16.0 GB.

In the context of this research, we designed the topological SDN-enabled environment consisting of OpenFlow switches, host devices, controllers, links etc. as shown in Figure 27. Thus, to achieve the CPP - IDS, we employed the parameters recorded in Table 30.

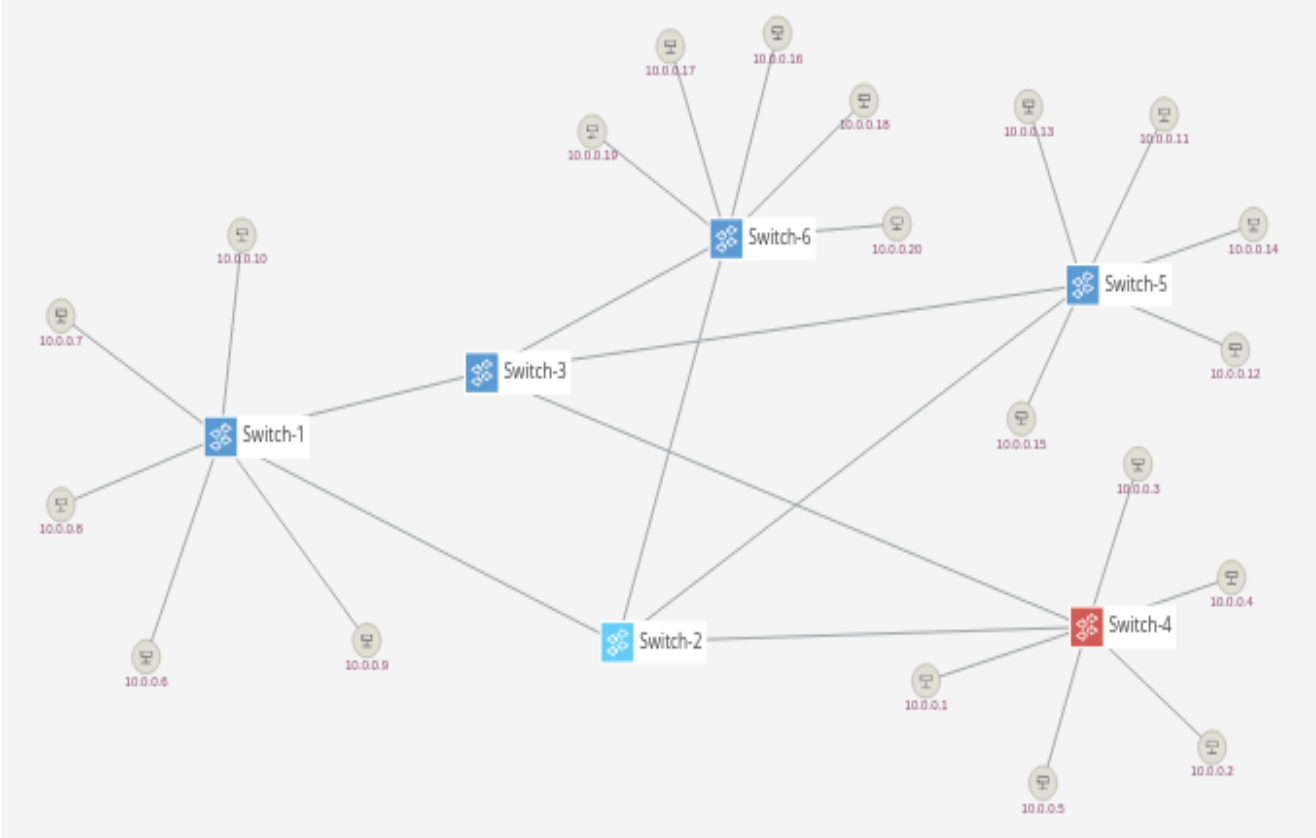


Figure 27. SDN environment

Table 30: Simulation parameters

Simulation Parameter(s)	Values
Simulation area - S_A	SDN enabled environment
Simulators - S_T	Mininet, web UI, VM, Python
Number of nodes N_A	20
Number of the switches and controllers - S_C	6, 3
Controllers - C_N	, ODL, Ryu, Floodlight
SDN domain - S_D	5
Simulation time - S_T	300s
Traffic type - T_T	UDP and TCP
Size of flow table size - S_T	120 entries
Flow size - S_S	100 byte
Number of the packets - N_P	25, 50, 125, 175, 100
Packet size - P_S	510 bytes
Number of runs - N_r	10
Link bandwidth - L_B	10 Mbps

In Table 30, we have presented the parameters adopted for the simulation of the CPP-IDS WAN. The adopted simulation parameters were selected based on the nature of the SDN environment, a simulator that includes the mininet, web UI, and VM, the number of nodes, OpenFlow switches, and the size of the packages used to perform the simulation.

7.4 Implementation

In this section, we present the implemented approach based on CPP – IDS in the SDN network. Firstly, we built the topological network consisting of connected hosts, switches etc. Secondly, we evaluated the designed SDN-enabled network-based CPP performance. Thirdly, we applied the kNN ML technique to perform the detection of the attack on the generated data. The results obtained from the experiment were presented in detail. The performed SDN experiment was run on the installed Ubuntu 20.04 LTS and VM.

7.4.1 Proposed Controllers' Placement

In this section, we simulated the SDN traffic network designed in Figure 27 defined by $\mathrm{v} \in \mathrm{V}$ where *v* denoted the collection of host devices $(\mathrm{v}_1, \mathrm{v}_2, \mathrm{v}_3 \ldots . \mathrm{v}_\mathrm{k})$ connected to the switches - S_n and the controllers C_k in the network.

7.4.1.1 Deployed Number of Controllers

To achieve the number of deployed controllers in the network, we defined our parameters $L(C_k)$ as the optimal number ***(k)*** of the controller, where ***k*** = 1,..., N is the number of the deployed controller. Each subdomain network was validated with the number of the minimum host devices $\beta(x)$. We simulated the SDN environment and evaluated the increased number of deployed controllers and the generated flow statistics. That is how many OVS, host devices, links etc. required for the increased number of deployed controllers in the network. In Table 31, we present the detailed information required to achieve the deployed number of controllers in the network.

Table 31: Deployed number of the controller

Criteria	Explanation
Total host devices	How many host devices are connected to the subdomain network
OvS	How many open-flow switches are connected to the controller
Number of flow rules	Number of the deployed controller flow rules in the network
CPU memory	The CPU memory for the number of deployed controller
CPU utilisation	CPU utilisation for the number of deployed controller
Controller response time	The controller response time in the network
Links	The number of links required for the deployed controller

$C_k \in [x_i, y_j]$

7.4.1.2 Switch Assignment $- N_{Ctr-Sw}$

To achieve the switch assignment in the context of this study, we evaluated the required number of switches related to the deployed number of controllers and flow rules in the network.

That is how many switches were required to be assigned to the deployed number of controllers during the generation of the network traffic flow. In this study, we evaluated how the increased number of assigned switches, etc., impacted the number of deployed controllers for the generated number of traffic flows in the network. In Table 32, we present the parameters required to achieve the controller switch assignment in the network.

Table 32: Summarised controller-to-switch results

No	Optimal controller	No. of connected OVS	Host devices (β)	Flow rules	Links
1	$\mathbf{C_1}$	θ_1	$\boldsymbol{\beta}$	$\boldsymbol{\Omega_1}$	***No(1)***
2	$\mathbf{C_2}$	θ_2	$\boldsymbol{\beta}$	$\boldsymbol{\Omega_2}$	***No(2)***
3	$\mathbf{C_3}$	θ_3	$\boldsymbol{\beta}$	$\boldsymbol{\Omega_3}$	***No(3)***
n	$\mathbf{C_n}$	θ_n	$\boldsymbol{\beta_n}$	$\boldsymbol{\Omega_n}$	***No(n)***

$C_k \in [x_i, y_j]$

7.4.1.3 Processing Latency $\approx Gap_{Ctr \rightarrow Ctr}(k)$

In this study, we evaluated the required processing latency in the network. We evaluated the time required for the sent-out message to achieve S2C and C2C during traffic communication. That is the required processing latency during traffic communication. To achieve this, we pinged the traffic communication from the source to the destination IP address and evaluated the, which denoted the controller-to-switch (S2C) processing latency, and b(i) denoted the controller-to-controller (C2C) processing latency (s). In Table 33, we present the parameters required to achieve processing latency.

Table 33: Processing latency

Criteria		Time(s)
Inter-latency $a(i)$	S2C	*s*
Intra-latency *b(i)*	C2C	*s*

$\mathbf{C_k} \in [x_i, y_j]$

7.4.1.1 Load Balance Detector

To achieve load balancing in the context of this study, we evaluated how many traffic flow statistics were generated in the SDN environment and calculated the load and capacity of the controller. We

computed the capacity and load on the controller using equations 6.8a and 6.8b. That is the required load for the deployed controller in the network. We computed the required capacity and load related to the increased number of controllers in the network. In Table 34, we present the summarized parameters required to achieve the load balance and capacity of the controller.

Table 34: Load balancing in the network

Criteria	Value 1	Value 2	Value 3
Controller	$\mathbf{C_1}$	$\mathbf{C_2}$	$\mathbf{C_3}$
Host devices	β	β	β
Traffic flows	$\boldsymbol{\Omega_1}$	$\boldsymbol{\Omega_2}$	$\boldsymbol{\Omega_3}$
Load balancing	lb($\mathbf{C_1}$)	Lb($\mathbf{C_2}$)	Lb($\mathbf{C_3}$)
Open flow switch	θ_1	θ_2	θ_3

$\mathbf{C_k} \in [x_i, y_j]$

7.4.1.1 Reliability

The reliability in the context of this study was evaluated based on the failure rate of the controller. In the SDN network, the controller can experience abnormal failure and be disconnected from the network. In this study, we evaluated the transferred flow rules from the failed controller to the neighbouring controllers. We applied the multi-path alignment strategy to export the flow rules from the failed controller to another controller. In Table 35, we present the summarized parameters required to achieve reliability in the network.

Table 35: Summarised reliability results

Criteria	Value 1	Value 2	Value 3
Controller	$\mathbf{C_1}$	$\mathbf{C_2}$	$\mathbf{C_3}$
Host devices	$\boldsymbol{\beta}$	$\boldsymbol{\beta}$	$\boldsymbol{\beta}$
Traffic flows	$\boldsymbol{\Omega_1}$	$\boldsymbol{\Omega_2}$	$\boldsymbol{\Omega_3}$
OVS	θ_1	θ_2	θ_3
Status reliability	True/False	True/False	True/False

$\mathbf{C_k} \in [x_i, y_j]$

7.4.2 Proposed Intrusion Detection

In this section, we present the implementation of the intrusion detection technique to ensure maximum security in the SDN environment. Firstly, we present how the generated traffic flow statistics were collected in the network. Secondly, we present how the flow statistics were used for traffic monitoring and classification. Thirdly, we present how the potential DDoS anomaly attack was detected in the SDN environment.

A) Traffic collection

In this study, we evaluated the traffic flow statistics from the designed network presented in Figure 27. The collected traffic flow consisted of the following attributes: deviceid, bytes, packet, length, etc. The data attributes were fitted for the training and testing models corresponding to the payload output results. In Table 36, we present the extracted information details used during the training and testing of the data.

Table 36: Traffic collection attribute details

Extracted feature	Description
Identity device	The identity devices associated with the traffic flow
Byte	The byte associated with the details of the packet flow
Packet size	The size of the delivered flow rule associated with the flow
Packets	The details of the delivered data packets in the network.
Payload	Transmitted payload associated with the packet.

B) Anomaly Detection Module

In this section, the traffic flow was subsequently trained and tested for traffic monitoring and classification. We inspected the traffic flow for the classification and identification of related information. We first present how the collected traffic flows were monitored for traffic classification, then we present how the controller evaluates the injected or oncoming traffic flows in the SDN network.

i. Traffic Classification

In this section, we performed training and testing on the generated traffic statistics for monitoring and classification. That is how much data was classified as normal and abnormal in the network. The employed data were split into training and testing; that is, 80% was used for training and 20% for testing. The higher the classification accuracy, the more dispersed the probability distribution, while the lower the concentration of the data distribution. Moreover, normal data involved a good distribution of the data, and abnormal data involved an anomalous distribution of the data. The generated, trained, and tested data from this research is presented in *Appendix A.*

ii. Anomaly Identification

In the context of the attack detection, we simulated the SDN environment presented in Figure 27 and evaluated how the configured controller analyses the oncoming / injected flows in the SDN environment. That is, how can we detect the oncoming DDoS anomaly attack in the network during

real-time monitoring? In the next subsection, we present how the alert generation was achieved, and the details of the injected DDoS attack in the SDN environment.

i. Alert Generation

In this study, we simulated the SDN environment presented in Figure 27 and evaluated the configured controller based on the oncoming traffic flows in the SDN environment. That is, how can we detect the DDoS attack in the SDN environment to provide security means for the number of oncoming traffic flows?

We defined the threshold value that determines the anomaly flow attack, configured the controller, and defined the rules that trigger an alert when the traffic exceeds a certain threshold. That is, the flow rules above the threshold are alerted as a potential attack on otherwise normal data flow**.** Once identified as a potential attack, the traffic flow was pushed to the mitigation module for further processing.

ii. ***Injected DDoS Attacks***

In this study, we launched the DDoS attack in the SDN network and evaluated how the configured controller analyses the packet-travelled information. The main aim of this study was to evaluate how the controller alarms the packet related to the DDoS attacks and provides mitigation action for the number of oncoming traffic flow rules in real-time monitoring. Table 37 presents the extracted packet information of the injected DDoS attack.

Table 37: Injected DDoS Anomaly attack details

No.	Extracted information	Description
1	ping	Used to send the ICMP echo request packets to the target host or the IP address.
2	-c	Number of the sent data packets in the network
3	-s	The sent packet size in the network
4	Target IP address	The information about the target communication address
5	Time	Round trip time about the packet details
6	Packet loss	Information about the transmitted packet

C) Attack Mitigation

Once the injected DDoS attack packet information presented in Table 37 was performed, the enforced policy matching the DDoS attack criteria was configured on the CLI to suspend the data associated with the DDoS anomaly attack in the network. In Table 38, we present the enforced policy matching the flow criteria required to suspend the identified DDoS anomaly attack in the SDN environment.

Table 38: DDoS anomaly attack mitigation policy

Criteria	Match criteria	Action
DropFlowData	We used the SDN controller and defined our flow table in SDN switches, our rule dictates how the network traffic is processed.	Drop the flow rule/s associated with the identified DDoS attack. > drop the source IP flow rule associated with the id anomaly attack.
ExcludeFlowData	We set the matching packet rule and identify any related packets associated with the attack attributes.	Exclude the identified flow rule associated with the DDoS flow rule required to duplicate the traffic flow rule. > Exclude the id flow rule required to mimic the source and destination IP address.
DropEntryFlow	We dictated any flow rule matching the IP address associated with the anomaly attack in the SDN network.	Drop the source IP address flow matching the entry flow data packet related to the anomaly attack. > drop the source IP address related to the ID anomaly attack
PushControllerLoad	We evaluated the failure rate of the controller in the SDN network and how flow rules are forwarded from the source to the destination.	Push the load of the affected controller to the target controller.

7.5 Results and Analysis

In this section, we present the results obtained from the practical experiment-based CPP-IDS in the SDN environment. Firstly, we presented the results obtained from the CPP objectives. Secondly, we present the attack detection in the network. Thirdly, we applied the mitigation policy to suspend the identified attack on the network. The experiment was performed on the VM installed on Ubuntu 20.04.

7.5.1 Proposed Controller Placement

In this subsection, we presented the results obtained from the number of deployed controllers, switch assignments, load balancing, processing latency, and reliability in the SDN environment. The SDN experiments were run on the VM installed on Ubuntu 20.04 LTS.

7.5.1.1 Deployed Number of Controllers

In this subsection, we simulated the SDN environment presented in Figure 27 based on the various numbers of the deployed controllers. In our study, the results were analysed based on the $L(C_k)$ - the number of controllers required during the traffic communication in the network. That is, 20 host devices were required to achieve 18 flow rules and 6 connected flow switches. 2GB CPU memory was utilized on the generated traffic flow. Based on the deployed controllers, 30 active links were recorded. Table 39. summarises the results obtained from the number of deployed controllers in the SDN environment.

Table 39: Deployed no. of controller

No.	Criteria	Value 3	Value 2	Value 1
1	Total host devices	20	10	5
2	OvS	6	3	1
3	Number of flow rules	18	10	3
4	CPU memory	64GB	64GB	64GB
5	CPU utilisation	30%	10%	10%
6	Controller response time	50ms	30ms	20ms
7	Links	30	20	6
8	Device status	Reachability	Reachability	Reachability

In Figure 28, we present the results obtained from the deployed number of controllers in the SDN environment. In this study, the increased number of controllers resulted in a higher 2GB CPU memory percentage utilization as compared to the lower deployed number of controllers. That is, the traffic pattern increases when the 20 host devices are connected to the 6 switch subdomain networks.

Moreover, the response time of the controller constantly increased based on the number of connected switches, generated flow rules, and connected host devices. To this end, the number of switches was adjusted based on the increased and deployed number of controllers in the network.

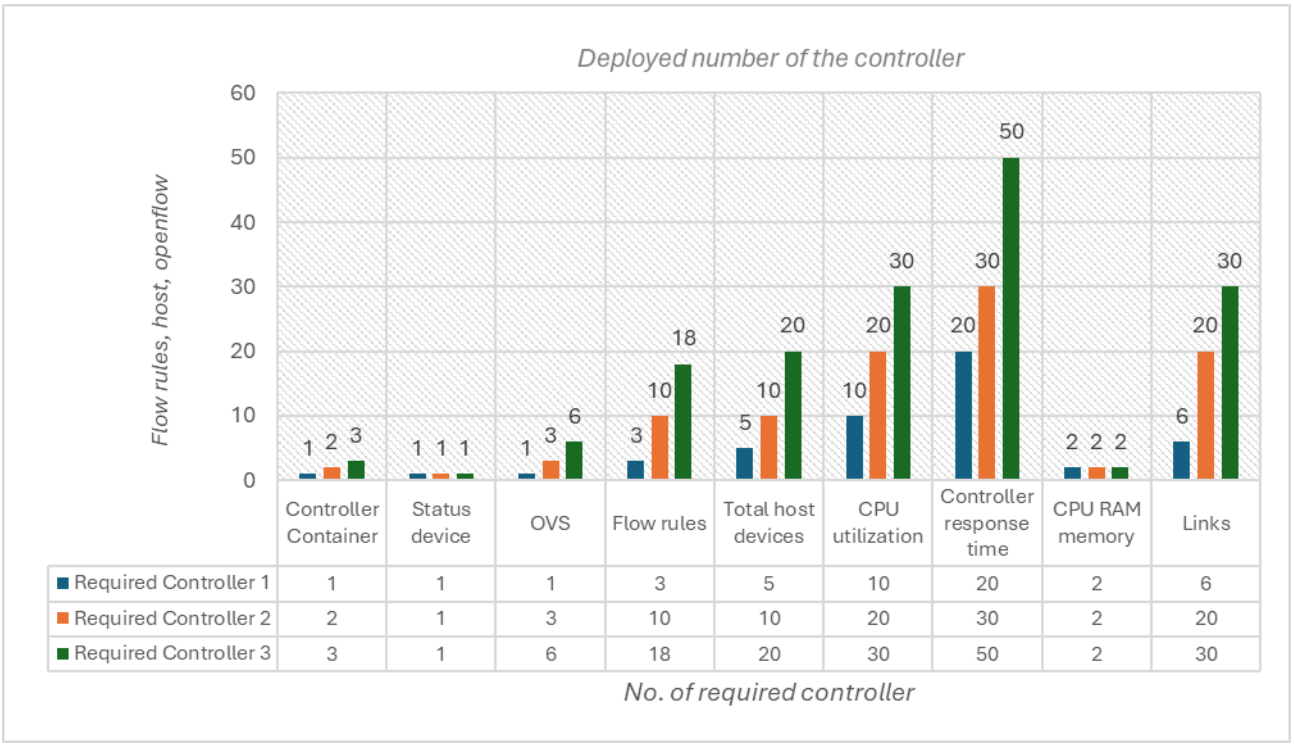


Figure 28. No. of required controllers

7.5.1.2 Switch Assignment

In this study, we evaluated the status of the assigned switch to the controller in the network. That is how many switches are required to be assigned to one controller for the number of traffic flow rules. Based on our results, one assigned switch generated fewer flow rules as compared to the second and third. That is, the higher the number of assigned switches, the higher the generation of flow rules in the network. Table 40 presents the summarized results of the assigned and required OVS on the generated flow rules in the network.

Table 40: Summarised switch assignment results

No.	Criteria	Value 1	Value 2	Value 3
1	Controller container	1	1	1
2	No. of connected switch	6	2	1
3	Open flow rules	18	10	5
4	Connection	True	True	True
5	Network interface	172.192.10.101	172.192.10.101	172.192.10.101
6	Container	172.168.10.1	172.168.10.2	172.168.10.3
7	Links	30	20	6
8	OpenFlow Port	8181	8181	8181

In Figure 29, we presented the results obtained from the assigned controller switch for the number of generated flow rules. In Figure 29, the number of flow rules increased based on the number of assigned switches. That is, 1 switch recommended the minimum of 3 flow rules, 2 switches recommended 10 minimum flow rules in the network, and lastly, 3 switches accommodated 18 flow rules in the network. That is, the higher the number of switches assigned to the controller, the higher the generated traffic statistics in the network.

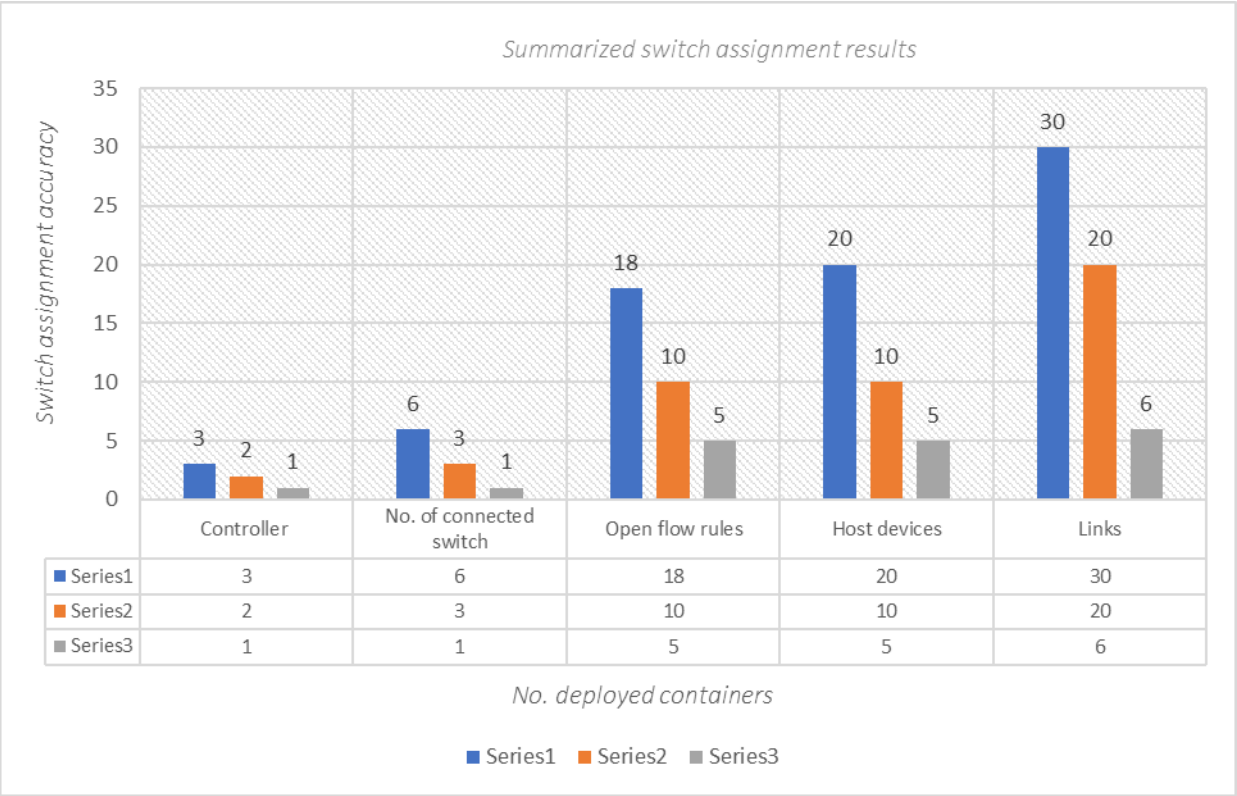


Figure 29: Switch assignment

7.5.1.3 Processing Latency

In this study, we evaluated the results obtained from the processed packet latency in the network. We evaluated the time taken for the packet to send out a message from the controller to the specific switch port. Our analysis recommended $a(i) = 0.958$s (C2S) as the minimized processing latency from the controller to the switch in the network. Moreover, $b(i) = 1.891$s (C2C) was recommended as the minimized processing latency required to process the traffic flow rule from one controller to the other. That is the traffic flow communication exceeded the recommended $a(i) > 0.958$s and $b(i) > 1.891s$ was excluded from the communication. Table 41 presents the required processing latency during SDN traffic communication.

Table 41: Network processing latency

Criteria	Processing latency (s)	Values
Inter-latency $a(i)$ = S2C	to switch	0.958s
Intra-latency $b(i)$ = C2C	to	1.891s

In Figure 30, we presented the time taken for the packet to send out a message from the controller to the specific switch port. In this study, we evaluated how long the packet was forwarded from the controller to the switch port. 0.958s was the required time during the forwarding of the packet from the controller to switch port 7. On the other hand, the forwarded packet from the controller to the controller was much higher as compared to the C2S. To this end, the forwarded packet-based C2S

and C2C recommended not to exceed the minimum required time to achieve a stable traffic network pattern.

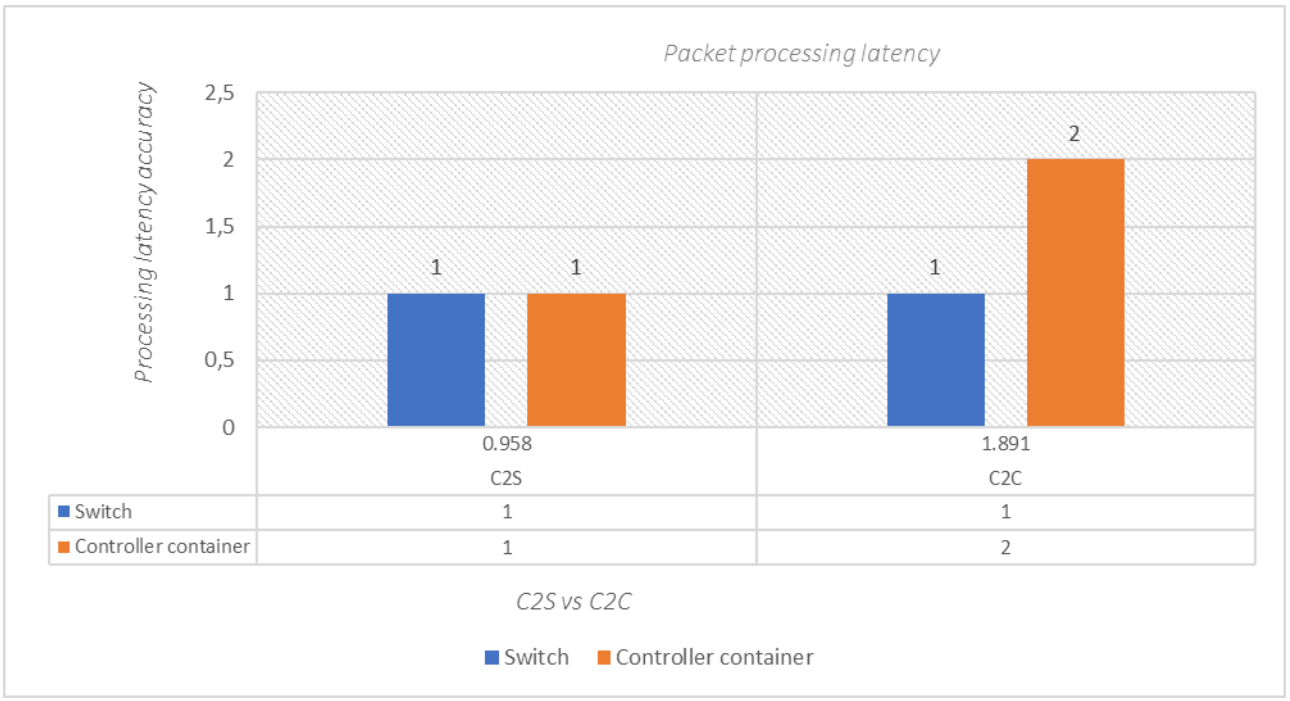


Figure 30: Processing latency

7.5.1.4 Controller Load Balance

In this subsection, we report the evaluation of the controller's performance to ensure efficient network operation. We evaluated how many traffic statistics, etc., were captured during the run time and calculated the capacity and load on the controller. We computed the capacity and load on the controller using equations 6.8a and 6.8b. In this study, we recorded the results of the generated traffic statistics, host devices, and deployed controllers. Based on Eq. (6.8a), we determined that the capacity of the controller decreased based on the increased number of controllers. On the other hand, based on Eq. (6.8b), we evaluated the load increase based on the increased number of controllers. That is, the load required to accommodate 3 flows is 0.069 lb, 10 flows are 0.027 lb, and 18 flows are 0.833 lb. Table 42 presents the required load balance and capacity on the controller in the network.

Table 42:Summarised load balance results

No.	Criteria	Value 1	Value 2	Value 3
1	Controller container	1	2	3
2	Host devices	5	10	20
3	Traffic flows	3	10	18
4	Traffic statistics	72	72	72
5	Controller capacity	72cc	36cc	24cc
6	Load balancing	0.069lb	0.027lb	0.833lb
7	Open flow switch	1	3	6

In Figure 31, we present the results obtained from the calculated load balancing and capacity in the network. The capacity of the controller decreased based on the increased number of controllers in the network. That is, the load was shared across the controllers to achieve balance in the network. That is, 0,042 lb is required by controller 2 and the connected 3 switches to achieve 0.027 lb. Moreover, 0,803 lb is required by controllers 3 and 6 connected switches in the network to achieve 0.833 lb.

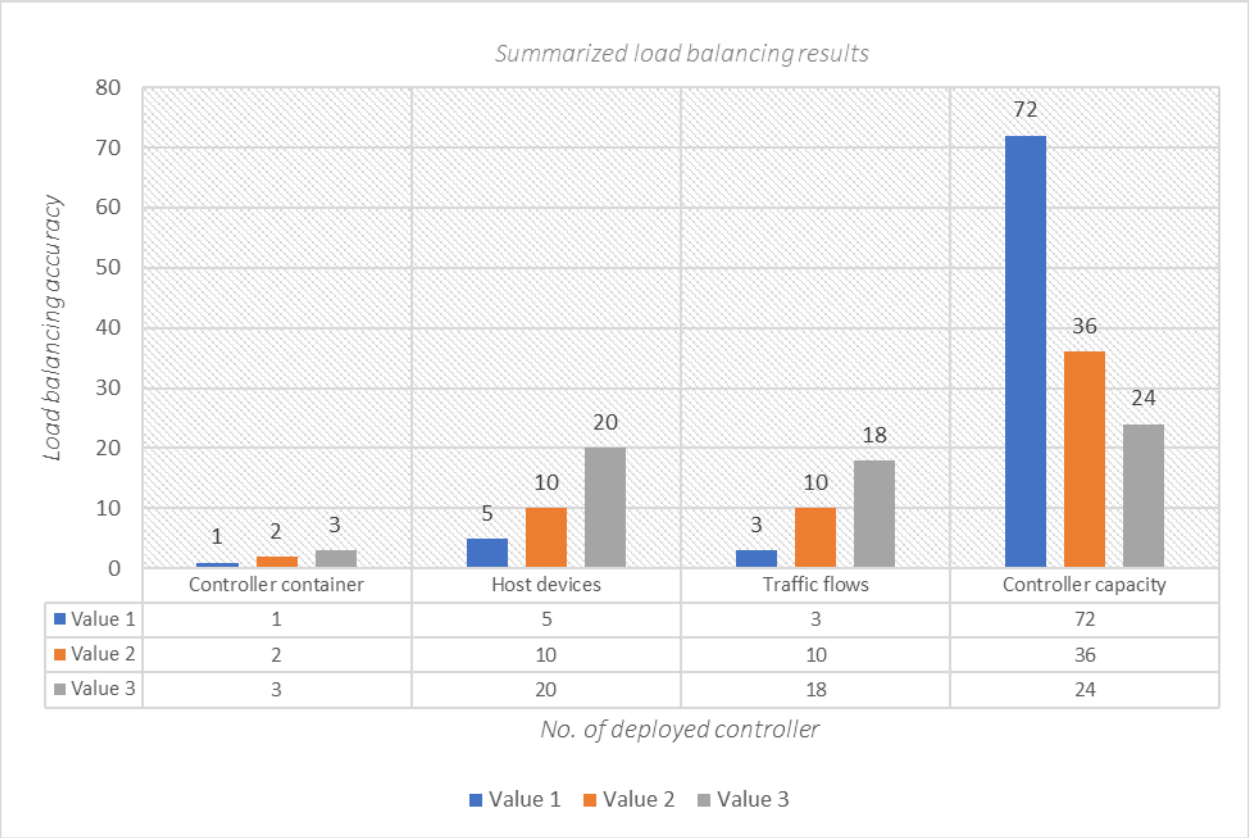


Figure 31: Controller load balance

7.5.1.5 Reliability

In this subsection, we evaluated the transferred flow rules based on the failure rate of the controller. In our study, we evaluated the nature of the flow rules in the network based on the failure rate of the controller in the network. In this study, we evaluated the status of controller 1 and the number of flow rules on the other active controllers. The failed controller was disconnected from the traffic communication, and the flow rules associated with the affected controller were shared among the active controllers. In Table 43, we summarized the results obtained during the failure of the controller in the SDN environment.

Table 43: Summarised reliability results

No.	Criteria	Value 1	Value 2	Value 3
1	Controller container	1	2	3
2	Host devices	0	10	20
3	Traffic flows	0	10	18
4	OVS	0	3	6
5	Status reliability	0	1	1

In Figure 32, we present the results obtained from the failure of controller 1. From the simulation, 1 switch, 5 hosts, and 3 flow rules associated with controller 1 were disconnected and connected to other active controllers, hosts, and switches in the network. The controller status, flows, devices, etc., were recorded in detail. The results obtained from the simulation affected our SDN WAN in terms of reliability for the number of required controllers, traffic flows, and controller status based on reliability 1, 2, and 3 of the controllers.

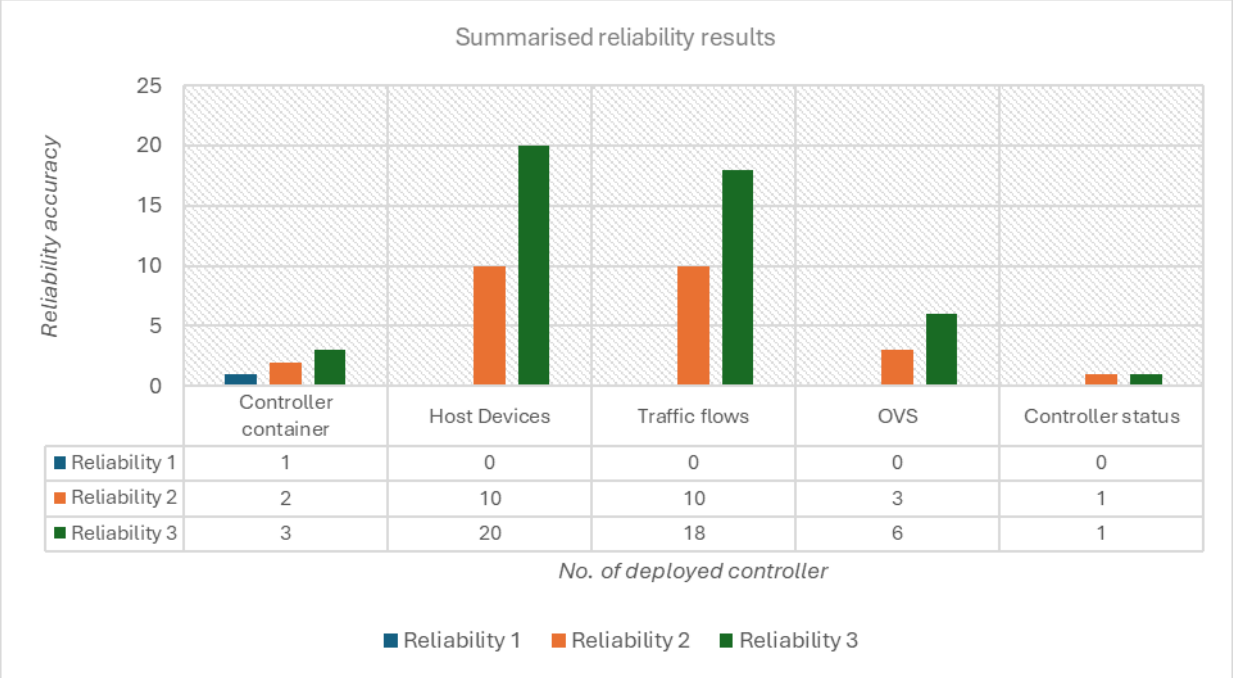


Figure 32: Reliability results

7.5.2 Detecting Network Attacks

This section presents the actual results obtained during the detection of the DDoS attack on the network. The obtained results were performed on the virtual machine installed on the Ubuntu 20.04 LTS. We utilize the designed SDN environment presented in Figure 27. Firstly, we present how traffic monitoring and classification were achieved. Secondly, we injected the DDoS attack into the SDN environment and evaluated how the configured controller evaluates the oncoming traffic communication.

7.5.2.1 *Traffic Classification Results*

Table 44 captures the collected information from the traffic monitoring and classification technique while utilizing the data packets. That is how many generated flow statistics were classified as normal and abnormal, ensuring the accuracy of the model to ensure maximum security in the SDN

environment. Our model recommended 100% detection accuracy, precision, and F1-score. To this end, eight flow statistics were classified as anomaly attacks in the network.

Table 44: Traffic monitoring results

No.	Criteria	Results
1	Anomaly attack	8
2	Detection accuracy	100%
3	Precision	100%
4	F1 - score	100%
5	Threshold percentile	95%
6	Detected anomaly-based threshold	1.096
7	N_neighbor	5

In Figure 33, we presented the results obtained from the traffic monitoring and classification in the network, i.e., eight anomaly attacks were classified as attacks in the network. The 1.096 threshold was considered for traffic monitoring and classification based on the 95% percentile.

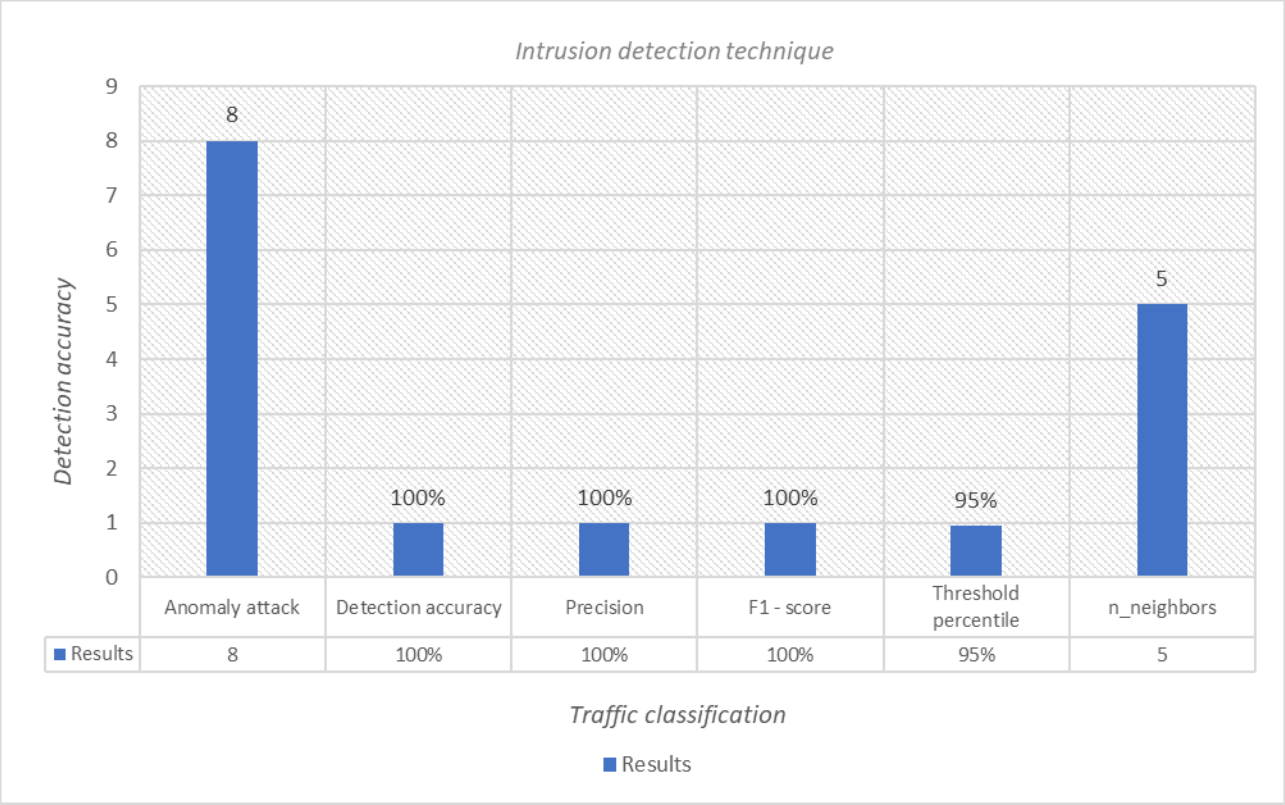


Figure 33: Traffic monitoring and classification results

7.5.2.2 Traffic Monitoring Scatter Plot

Figure 34 illustrates the results obtained during traffic monitoring and classification in the network. We observed how the number of anomaly attacks was classified based on the n_neighbors and threshold percentiles. The results showed a 100% classification report on the data packets, etc. That

is when the data packets travel in the network and are monitored and classified as normal and abnormal traffic flow. To this end, our model showed the capability of monitoring and classifying the data in the network.

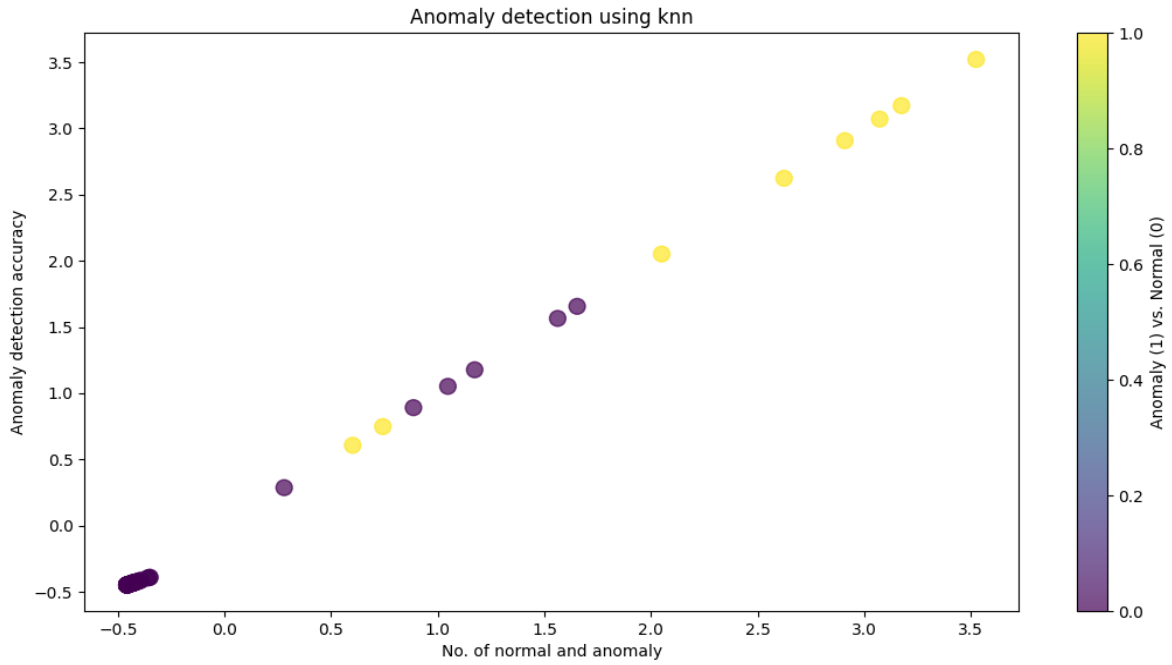


Figure 34. Traffic monitoring scatter plot

7.5.2.3 SDN Environment Attack

In this section, we present the results obtained from the simulated SDN environment. We firstly present detailed information about the injected DDoS attack in the network; secondly, we present how the configured controller was used to suspend the packet associated with the DDoS attack to provide security means for the number of oncoming traffic communications in the network.

i. ***Injected DDoS Packet Details***

Table 45 presents the details of the injected traffic associated with the DDoS anomaly attack on the network. The main aim was to evaluate how the configured controller analyzes the injected DDoS associated with the malicious flows in the SDN environment. Table 45 presents the summarized injected DDoS packet details in the network.

Table 45: Summarized DDoS Anomaly Attack details

No.	DDoS attack	Description
1	ping -c 5 -s 64 10.0.0.2	Sent the 5 packets of 64 bytes to the host IP address 10.0.0.2
2	ping -c 6 -s 64 10.0.0.4	Sent the 6 packets of 64 bytes to the DST IP address 10.0.0.4
3	ping -c 4 -s 64 10.0.0.3	Sent the 4 packets of 64 bytes to the host IP address 10.0.0.3
4	ping -c 2 -s 64 10.0.0.5	Sent the 2 packets of 64 bytes to the host IP address 10.0.0.5
5	ping -c 2 -s 64 10.0.0.14	Sent the 2 packets of 64 bytes to the DST IP address 10.0.0.14

ii. ***DDoS Alert Results***

In this study, we configured the controller and implemented the alert logic to check oncoming traffic exceeding the 1.096 threshold. That is how many DDoS flow attacks exceeded the threshold in the network.

We simulated the SDN-controlled environment and ensured the detection of oncoming traffic based on the 1.096 threshold alerts in the network. Based on the injected packets presented in Table 45, our model resulted in five alerts that exceeded the threshold of 1.096 in the network.

7.5.3 Attack Mitigation

This section presents the actual results obtained from the mitigation strategy for the DDoS attacks. The main aim is to provide security measurements for the number of identified attacks in the network. That is how the action policy can be applied to the alerted anomaly DDoS attack in the network. In this study, we present how the action policy was used on the matching traffic criteria to suspend the flow rules associated with the alerted anomaly DDoS attack.

7.5.3.1 Implemented Matching Rules

In this subsection, we evaluated how the enforced action policy responded to the identified traffic flow attack in the network. That is how many DDoS anomaly attacks were dropped, excluded from the network, etc. to improve the performance of the network. In Table 46, we present the enforced action policy intended to minimize the occurrence of injected DDoS traffic in the network.

Table 46: Enforced policy matching criteria

No.	Matching criteria
1	> flow-objective add-ipv4_src=10.0.0.2 action=drop
2	> flow-objective add-ipv4_src=10.0.0.4 action=drop
3	> flow-objective add-ipv4_dst=10.0.0.3 action=drop
4	> flow-objective add-ipv4_dst=10.0.0.5 action=drop

7.5.3.2 *Implemented Policy Results*

In this section, the enforced policy was implemented to suspend the alerted DDoS attack on the SDN network. We evaluated the results obtained from the simulated action policy, that is, the response time of the controller and packet loss of the injected packets in the network. To this end, we evaluated how many flow alerts were analysed by the enforced policy in the network. We also evaluated how the traffic flow rules associated with the failed controller were shared among other active controllers. Table 47 summarizes the enforced policy results in the simulated SDN environment.

Table 47: Enforced policy results

No.	Criteria	Results
1	DropEntryFlow attack	5
2	ExcludeFlow attack	2
3	DropFlow attack	2
4	PushControllerFlow	3
5	No.of controllers	3

In Figure 35, we present the results of the enforced policy in the network. We evaluated how the enforced policy was used to suspend the number of alerted DDoS attacks on the network. That is the number of suspended, excluded, and dropped flow data points. We also evaluated how many flow rules were required to be migrated into the network as a result of the failed controller.

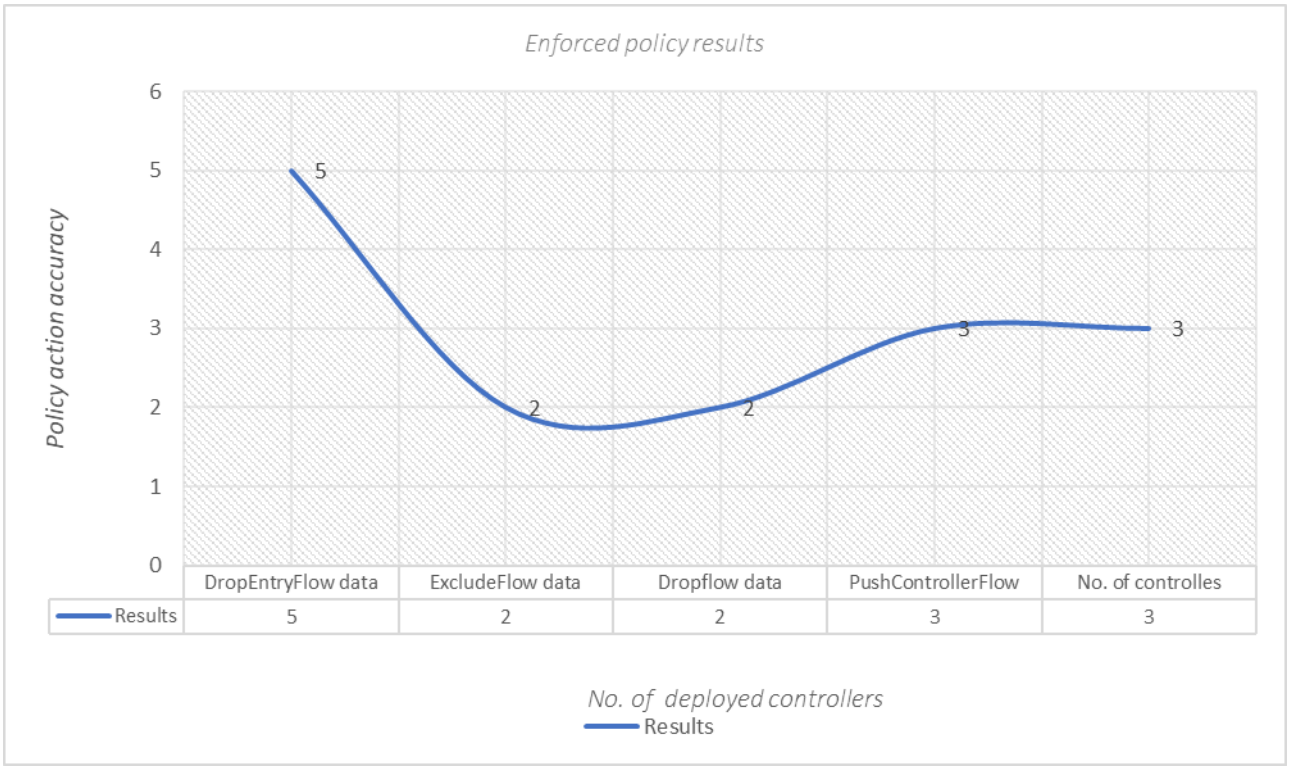


Figure 35: Enforced action policy results

7.6 CPP Comparison and Evaluation

This section presents the comparison of the selected controllers in the SDN environment. We employed the OpenDayLight (ODL) 16.0, Ryu 1.5 version, and Floodlight 1.2.0 on the designed SDN network topology presented in Figure 27. The results obtained from the experiment were presented and compared with the performance results.

7.6.1 Proposed ODL Placement

This section presents the results obtained from the number of employed ODLs, switch assignments, load balancing, processing latency, and reliability in the SDN environment. The SDN experiments were run on the VM installed on Ubuntu 20.04 LTS.

7.6.1.1 Deployed Number of Controllers

This subsection presents the results obtained from the number of deployed ODL controllers. In this study, we evaluated the results based on the number of controllers required during traffic communication in the network. In Table 48, we captured information from the number of deployed controllers, i.e., 20 host devices were required to achieve 25 flow rules connected to 6 switches. 2 GB of CPU memory was utilized during the simulation. In Table 48, we present the information captured during the deployment of the controller in the SDN network.

Table 48: Deployed No. of Controllers

No.	Criteria	Value 1	Value 2	Value 3
1	Total host devices	5	10	20
2	OvS	1	3	6
3	Number of flow rules	3	8	25
4	CPU memory	64GB	64GB	64GB
5	CPU utilisation	10%	10%	30%
6	Controller response time	20ms	30ms	50ms
7	Links	6	20	30
8	Device status	Reachability	Reachability	Reachability

$C_k = [x_i, y_j]$

7.6.1.2 Switch Assignment

In this subsection, we evaluate the status of the switch assignment to the controller in the network. That is how many switches are required to be assigned to one controller for the number of traffic flow rules. Based on our obtained results, one assigned switch generated fewer flow rules as compared to the second and third flow switches. That is, the higher the number of assigned switches, the higher the generation of flow rules in the network. In Table 49, we present the summarized results of the required OVS concerning the generated flow rules in the network.

Table 49: Summarised switch assignment results

No.	Criteria	Value 1	Value 2	Value 3
1	Controller container	1	2	3
2	No. of connected switch	1	3	6
3	Open flow rules	3	10	30
4	Links	6	20	30
5	Device status	Reachability	Reachability	Reachability
6	Host devices	5	10	20

7.6.1.3 Processing Latency

In this subsection, we evaluated the packet sent out during the traffic communication. That is the time it takes for the packet sending out the message to reach the specific port. We evaluated the intra-processing latency b(i) and inter-processing latency in the SDN network. Our captured information recommended 0.977s as the minimized processing latency required to be applied from switch to controller. Moreover, b(i) = 1.431s was recommended as the required processing latency to send flow rules from one controller to another in the network. Table 50 summarizes the required processing latency during the communication.

Table 50: Network processing latency

Criteria	Processing latency (s)	Values
Inter-latency $a(i)$	Controller to switch	0.977s
Intra-latency *b(i)*	Controller to controller	1.431s

$C_k \in [x_i, y_j]$

7.6.1.4 Controller Load Balance

This subsection presents the load balance and capacity of the controller to ensure efficient network operation. We captured the number of traffic statistics and calculated the load and capacity on the controller using equations 6.8a and 6.8b. That is the required capacity and load on the controller in the network. In this study, the increased load was associated with the decreased capacity of the controller. In Table 51, we present the required load for the number of employed controllers in the network.

Table 51: Summarised load balance results

No.	Criteria	Value 1	Value 2	Value 3
1	Controller container	1	2	3
2	Host devices	5	10	20
3	Traffic flows	6	10	30
4	Traffic statistics	100	100	100
5	Controller capacity	100cc	50cc	33cc
6	Load balancing	0.05lb	0.2lb	0.6lb
7	Open flow switch	1	3	6

$C_k \in [x_i, y_j]$

7.6.1.5 Reliability

In this subsection, we evaluated the forwarded flow rules based on the failure rate of the controller. We evaluated the status of the failed controller and the number of captured flow rules in the network. The failed controller was disconnected from the traffic communication, and the flow rules associated with the affected controller were shared among active controllers. In Table 52, we present the results obtained during the failure of the controller.

Table 52: Summarised reliability results

No.	Criteria	Value 1	Value 2	Value 3
1	Controller container	1	2	3
2	Host devices	0	15	20
3	Traffic flows	0	10	18
4	OVS	0	4	6
5	Status reliability	0	1	1

$C_k \in [x_i, y_j]$

7.6.2 Proposed Ryu's Placement

In this section, we present the results obtained from the practical experiments performed using the Ryu controller. The evaluated results included the deployed number of the controller, switch assignment, load balancing, processing latency, and reliability. The performed experiments were run on the VM installed on Ubuntu 20.04 LTS.

7.6.2.1 Deployed Number of Controllers

In this subsection, we present the result obtained from the number of controllers in the SDN environment. The evaluated results were based on the required number of controllers during network traffic communication. In this study, 6 flow rules were generated on controller 1, 10 flow rules were generated from controller 2, and 18 flow rules were generated from controller 3. Moreover, various CPU memories were generated based on the deployed number of the controller. In Table 53, we summarized the results obtained from the deployed number of the controller in the SDN network.

Table 53: Deployed No. of Controllers

No.	Criteria	Value 1	Value 2	Value 3
1	Total host devices	5	10	20
2	OvS	1	3	6
3	Number of flow rules	6	10	18
4	CPU memory	1.11GB	1.35GB	2GB
5	CPU utilisation	10%	10%	30%
6	Controller response time	20ms	30ms	50ms
7	Links	6	20	30
8	Device status	Reachability	Reachability	Reachability

$C_k = [x_1, y_j]$

7.6.2.2 Switch Assignment

To perform the switch assignment in the context of this research, we evaluated how many switches were required to be assigned to the controller for the number of traffic flow rules in the network. Based on our simulation, 1 switch assignment generated 7 flow rules, 2 controllers generated 10 flow rules, and 3 controllers generated 25 flow rules. That is, the number of flow rules increases based on the increased number of switches and controllers in the network. In Table 54, we summarized the results obtained from the number of assigned switches in the SDN environment.

Table 54: Summarised switch assignment results

No.	Criteria	Value 1	Value 2	Value 3
1	Controller container	1	2	3
2	No. of connected switch	1	3	6
3	Open flow rules	7	10	25
4	Links	6	20	30
5	Device status	Reachability	Reachability	Reachability
6	Host devices	5	10	20

$C_k = [x_1, y_j]$

7.6.2.3 Processing Latency

In this subsection, we evaluated the processing latency based on C2S and C2C during the packet sent out message in the network. Our model recommended $a(i) =$ 0.849s as the minimized processing latency required for effective communication in the network. Moreover, *b(i) = 1.531s* was considered the best processing latency between the controllers. In Table 55, we present the summarised results obtained from the processing latency in the SDN environment.

Table 55: Network processing latency

Criteria	Processing latency (s)	Values
Inter-latency $a(i)$	Switch to controller	*0.849s*
Intra-latency *b(i)*	Controller to controller	*1.531s*

$C_k \in [x_i, y_j]$

7.6.2.4 Controller Load Balance

In this subsection, we evaluate the load required to ensure efficient network operation. We evaluated how many traffic statistics were captured during the run time and calculated the capacity and load on the controller. We computed the capacity and load on the controller using equations 6.8a and 6.8b. Based on the calculated results, controller 1 required 0.025 lb, controller 2 0.1 lb, and controller 3 0.3 lb. Our model predicted that the higher the number of deployed controllers, the more balanced loads are required. That is; to achieve load balance on controller 2, 0.075 lb was required to achieve 0.1 lb. In Table 56, we summarize the results obtained from the load and capacity of the controller.

Table 56: Summarised load balance results

No.	Criteria	Value 1	Value 2	Value 3
1	Controller container	1	2	3
2	Host devices	5	10	20
3	Traffic flows	7	10	25
4	Traffic statistics	200	200	200
5	Controller capacity	200cc	100cc	67cc
6	Load balancing	0.025lb	0.1lb	0.3lb
7	Open flow switch	1	3	6

$\mathbf{C_k} \in [x_i, y_j]$

7.6.2.5 Reliability

In this subsection, we evaluated the state of the network and the transferred flow rules from the source to the target based on the multi-path alignment. We evaluated the status of the failed controller and the number of captured flow rules in the network. The failed controller was disconnected from the traffic communication, and the flow rules associated with the affected controller were shared among active controllers. In Table 57, we summarized the results obtained based on the failure rate of controller 1.

Table 57: Summarised reliability results

No.	Criteria	Value 1	Value 2	Value 3
1	Controller container	1	2	3
2	Host devices	0	15	20
3	Traffic flows	0	17	25
4	OVS	0	3	6
5	Status reliability	0	1	1

$\mathbf{C_k} \in [x_i, y_j]$

7.6.3 Proposed Floodlight Placement

In this section, we present the results obtained from the floodlight controller in the SDN network. The performance is based on the number of deployed controllers, switch assignments, load balancing, processing latency, and reliability in the SDN environment. The SDN experiments were run on the VM installed on Ubuntu 20.04 LTS.

7.6.3.1 Deployed Number of Controllers

In this subsection, we evaluated the number of flow rules based on the number of deployed controllers in the network. The analysed results were based on $L(C_k)$ - the number of controllers required for the number of flow rules during the traffic communication in the network. In this study, controller 1 was evaluated with 6 generated flow rules, while controller 2 generated 10 flow rules, and controller 3 generated 16 flow rules in the network. That is the higher the number of the controller, the increased traffic flow rules in the network. In Table 58, we summarised the required number of controllers related to the flow rules.

Table 58: Deployed No. of controllers

No.	Criteria	Value 1	Value 2	Value 3
1	Total host devices	5	10	20
2	OvS	1	3	6
3	Number of flow rules	6	10	16
4	CPU memory	2GB	1.35gb	2gb
5	CPU utilisation	10%	10%	30%
6	Controller response time	20ms	30ms	50ms
7	Links	6	20	30
8	Device status	Reachability	Reachability	Reachability

$\mathbf{C_k} = [\mathbf{x_1}, \mathbf{y_j}]$

7.6.3.2 Switch Assignment

In this subsection, we evaluate the results obtained from the assigned switch in the network. The analyzed results answer the question of how many switches are required for the number of generated traffic flow rules. We increased the number of switches and evaluated the number of flow rules in the network. Based on our obtained results, one assigned switch generated fewer flow rules as compared to the second and third flow switches. That is, the higher the number of assigned switches, the higher the generated number of flow rules in the network. In Table 59, we present the summarized results of the required OVS and generated flow rules in the network.

Table 59: Summarised switch assignment results

No.	Criteria	Value 1	Value 2	Value 3
1	Controller container	1	2	3
2	No. of connected switch	1	3	6
3	Open flow rules	6	10	30
4	Links	6	20	30
5	Device status	Reachability	Reachability	Reachability
6	Host devices	5	10	20

$C_k = [x_1, y_j]$

7.6.3.3 Processing Latency

In this study, we evaluated the minimized processing latency and b(i) in the network. In our case, the model recommended minimizing 0.749s as the required time to achieve floodlight switching communication. On the other hand, b(i) = 1.231s was recommended as the required C2C processing latency in the network. In Table 60, we summarized the network processing latency required during the network communication.

Table 60: Network processing latency

Criteria	*Processing latency (s)*	*Values*
Inter-latency $a(i)$	Switch to controller	*0.749s*
Intra-latency *b(i)*	Controller to controller	*1.231s*

$C_k \in [x_i, y_j]$

7.6.3.4 Controller load balance

In this subsection, we compute the load and capacity of the network controller to ensure efficient network operation. We captured the number of traffic statistics and calculated the load on the controller using eqs. 6.8a and 6.8b. That is the required capacity and load on the controller in the network. Based on our captured results, controller 2 required 0.12 lb to achieve 0.16 lb in the network. Moreover, controller 3 required 0.34 lb to achieve 0.5 lb in the network. In Table 61, we summarized the results obtained to achieve load balancing in the network.

Table 61: Summarised load balance results

No.	Criteria	Value 1	Value 2	Value 3
1	Controller container	1	2	3
2	Host devices	5	10	20
3	Traffic flows	6	10	30
4	Traffic statistics	120	120	120
5	Controller capacity	120cc	60cc	40cc
6	Load balancing	0.04lb	0.16lb	0.5lb
7	Open flow switch	1	3	6

$C_k \in [x_i, y_j]$

7.6.3.5 Reliability

In this study, we evaluated the forwarded flow rules based on the failure rate of the controller in the network. We evaluated the status of the failed controller and the forwarded number of the captured flow rules in the network. Based on the failure rate of controller 1, the disconnected controller's flow rules were shared among the active controllers. In Table 62, we present the results obtained due to the failure rate of controller 1.

Table 62: Summarised reliability results

No.	Criteria	Value 1	Value 2	Value 3
1	Controller container	1	2	3
2	Host devices	0	15	20
3	Traffic flows	0	16	30
4	OVS	0	3	6
5	Status reliability	0	1	1

$C_k \in [x_i, y_j]$

7.7. SDN-based CPP Comparison

In Table 63, we present the comparison and evaluation of the existing CPP schemes in the SDN networks. The performed comparison was analysed based on the utilized methods, metrics, limitations etc.

The SDN CPP can indeed be tackled differently using various methods, techniques, approaches etc. However, there are multiple solutions proposed to address the issues and challenges in the SDN. Moreover, some of the proposed solutions were achieved individually etc. In Table 63, we summarised some of the SDN CPP-reviewed papers and performed a comparison in detail.

Table 63: Comparison with the existing CPP scheme

Author	Methods & algorithm	Placement Metric (s)	Evaluation	Limitation
Jiang Liu et al. [205]	Clustering algorithms	Reliability	An optimal and sub-optimal solution	Multiple objectives optimization
Heller et al. [113]	K centre	Switch to controller & worst-case latency	k-centres can consider up to n − 1 controller failures	Fully distributed control is more resilient than decoupled control planes
Maryam Tanha et al. [195]	Capacitated k centre	Switch to controller latency Reliability	Minimize the total incurred cost incremental design	Design of heuristic algorithm, testing random topologies
He et al. [204]	Mixed-integer programming	Average flow setup time	For low densities, an adaptive flow controller	The cost of controller migration, depending on its load, and the cost of switch reassignment
Guodong Wang et al. [180]	Optimized k means	Latency minimization, load balancing, resistance/reliability	Maximum latency between the centroid and its associated nodes	Decrease the complexity of the controller placement
Liu et al.[186]	Simulated Annealing	Reliability	Allows for finding close optimal results	Consequent control traffic
Hemamalini et al. [212]	Mathematical formulation, heuristic	Scalability, latency, load balance	Controller deployment	Traffic Engineering
Guang et al. [183]	k-centre	Latency, load balance	Network topology	Load of the controller
Rong et al.[216]	DL algorithms	Transmission and processing delay	Kuhn Munkres model	Controller capacity matching
Yuanxiang et al. [213]	Deep neural networks	Latency, load balance	CIFAR- 10 benchmarks	Response time, policy
This Research	CPP, Anomaly detection technique, ML algorithm, and mathematical formulation	Processing latency, reliability, load balancing, optimal controller, controller to switch assignment	Opendaylight, Ryu, Floodlight's optimal controller, processing latency, controller to controller, controller to switch assignment, load balancing, and reliability.	Precise statistical measurements, instruments, techniques, and tools were adopted during the practical implementation. Real network implementation.

7.8 IDS Comparison and Evaluation

This section shows the IDS comparison and evaluation of the proposed ML techniques. We experimented with and analysed the traffic pattern during traffic monitoring and classification in the SDN network. We employed Naive Bayes (NB) and Random Forests (RF), and the trained and tested data is presented in Appendix A.

In this study, the chosen ML models are also very effective in the classification and analysis of the datasets. We employed various metrics that include detection accuracy, precision, and F1-score to evaluate the performance of the proposed models. The experiment was run on Ubuntu 20.04 with the installed PyCharm 5.03 IDE.

7.8.1 NB Algorithm

In this subsection, we evaluate the NB-based IDS in the SDN network. We evaluated the performance of the NB algorithm on the traffic flow in the SDN network. That is how many attacks were detected by the NB ML technique and the accuracy required to ensure maximum security in the SDN environment. Based on the simulated SDN environment data packets, we achieved 12 DDoS anomaly attacks with the corresponding detection accuracy, precision, and F1 score. Table 64 summarizes the NB anomaly detection results.

Table 64: NB anomaly detection results

No.	Criteria	Results
1	DDoS anomaly attack	12
2	Alerts	12
3	Detection accuracy	84%
4	Precision	78%
5	F1 - score	80%

7.8.2 RF Algorithm

In this subsection, we evaluated the RF attack detection in the SDN network to answer the question: How many anomaly attacks can be identified by the RF in the SDN network? We ran the RF and assessed how the algorithm responded to the generated data packet statistics. During our simulation, 10 alerts related to the anomaly attack were identified in the network. Moreover, our model analyzed the data packets with 79% detection accuracy, 80% precision score, and 82% accuracy based on the F1 score.

Table 65: RF anomaly detection results

No.	Criteria	Results
1	DDoS anomaly attack	12
2	Alerts	12
3	Detection accuracy	79%
4	Precision	80%
5	F1 - score	82%

7.8.3 IDS Discussion

This subsection presents the discussion based on the performed IDS models in the SDN environment. The performed models were compared with the kNN-based IDS in the SDN network. The main aim of the study was to evaluate how effectively and efficiently our model outperformed other proposed models based on detection accuracy in the SDN environment.

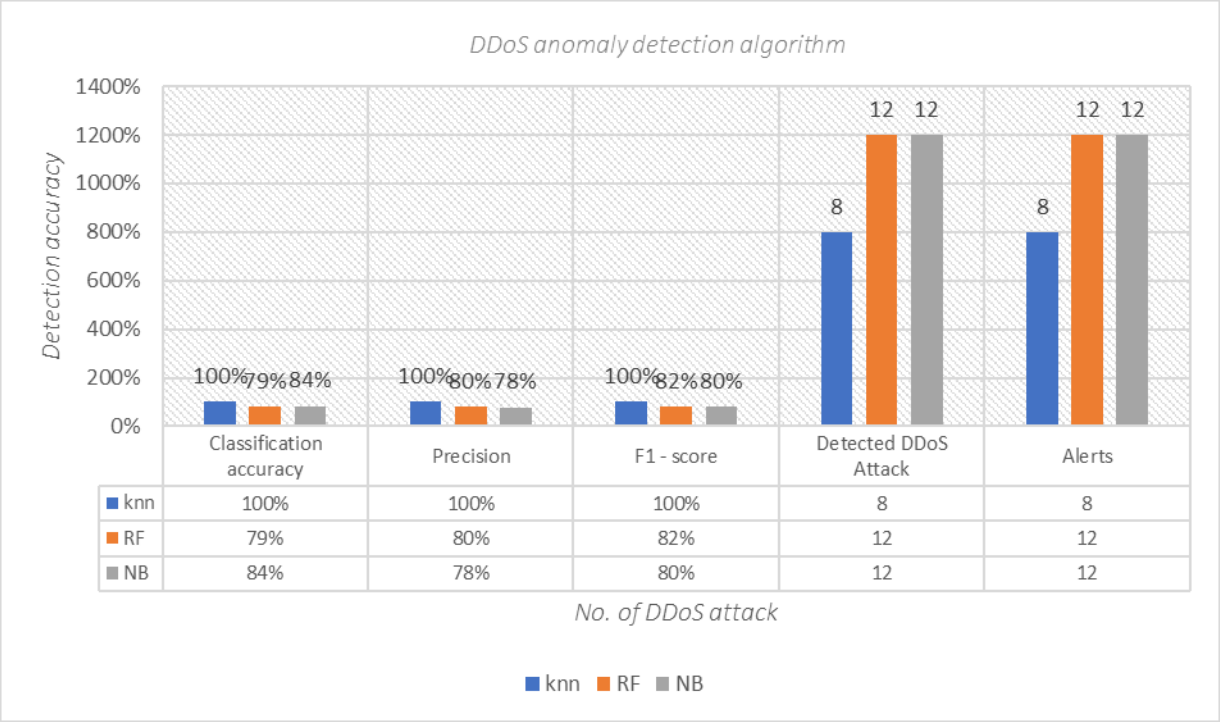


Figure 36: IDs performance models

In Figure 36, the NB-based IDS raised 12 alerts in real-time with 84% detection accuracy as compared with the 100% accuracy of the kNN algorithm. The number of alerts was higher as compared to the alerts detected by the KNN algorithm. On the other hand, the RF-based IDS raised 12 alerts in the network as compared to the 84% accuracy of the NB; however, the 79% accuracy was less as compared to the kNN algorithm in the network. That is, the model's performance accuracy decreased as a result of the increased number of identified alerts.

To this end, the kNN-based IDS was able to detect eight alerts in the network as compared to other proposed ML models. The kNN outperformed other ML algorithms with 100% detection accuracy in the network. That is, the higher the number of alerts, the higher the detection accuracy of the model.

7.9 SDN-based IDS Comparison

In this subsection, we perform the SDN-based IDS comparison. The comparison was analysed based on the employed methods, methodology, challenges, and type of attack.

It is true from the above discussions that the performance of an IDS is based on how efficient the model response is during the detection of the attack in the network. Moreover, there are multiple IDS-proposed solutions; however, an efficient algorithm still needs to be proposed to achieve the highest detection accuracy in the SDN network. In Table 66, we summarized some of the proposed SDN-based IDS challenges and solutions. The main aim was to gather information about the

previously related papers on SDN IDS techniques and provide a comparison with the existing SDN-based IDS.

Table 66: Comparison with existing SDN-based IDS

Author	Method & Algorithm	Methodology	Challenge(s)	Type of attack
Nadia Niknami *et al.* [650]	Entropy-KLML	Anomaly detection method	Vulnerabilities between the control plane and the data plane	DoS
Jorge Eliecer *et al.* [651]	Decision Trees (DT), Random Forest (RF), and a Neural Network architecture	Network anomaly detection	SDN security	DDoS
Jin Wang *et al.* [652]	Convolutional Neural Network and Extreme Learning Machine (CNN-ELM)	Anomaly detection model	SDN security vulnerabilities	DoS
Mahmoud Said *et al*. [653]	Information Gain (IG) and Random Forest (RF)	Anomaly detection system	Threat detection systems software applications	DDoS
L. Barki *et al.* [527]	ML algorithms traffic Naïve Bayes Classification	Anomaly flow detection	Forwarding of the data packets	DDoS
A. Le *et al.* [528]	C4.5 Decision tree	Anomaly flow detection	Traffic classification DoS and probe attack, cost, and latency	DoS and probe
P. Amaral *et al.* [22]	Random Forests, Stochastic Gradient Boosting, and Extreme Gradient Boosting	Anomaly flow detection	Type of data, OpenFlow protocol, traffic classification	DDoS
J. Ashraf *et al.* [21]	Support Vector Machines, Artificial Neural Networks (ANN)	Anomaly flow detection	DDoS attack & bandwidth	DDoS
Gustavo Frigo *et al*. [654]	DenStream algorithm	Unsupervised online anomaly detection	Network forwarding decisions	DDoS and portscan
This Research	SDN-based KNN IDS technique	Anomaly flow-based DDoS detection	Insufficient techniques, resources, and real network attack implementation.	DDoS

7.10 Chapter Summary

This chapter presents the implemented approach for the designed CPP and IDS. Moreover, the experimental parameters and system properties were presented in detail. The evaluation and comparison for both CPP and IDS were presented in detail. It also presents how the CPP objectives were achieved in the SDN environment. The design of the IDS used to detect the DDoS anomaly attack in the SDN network was presented in detail. Furthermore, the mitigation policies acting on the identified DDoS anomaly attack were presented in detail. Lastly, we presented the evaluation and comparison of our designed CPP and IDS framework using the related paper.

Chapter 8

Summary, Conclusion and Recommendations

8.1 Chapter Outline

This chapter presents a summary based on the conducted research study. Firstly, we provide a brief discussion about the written chapters. We briefly discuss how the objectives of this research were met. We further discuss the scope and limitations of this research. Lastly, we provide the conclusion of this research. Furthermore, the recommendations and directions based on this research were discussed in detail.

8.2 Summary

This study is summarised as follows: Chapter 1 provided a brief introduction to SDN relevant to this research. We presented the problem statement, aim, research questions, and objectives to achieve the CPP and IDS frameworks. Moreover, we also presented the following research methods for this research: Then, after carefully performing the reviews, we concluded that the issues and challenges of the SDN environment should be resolved, and the most significant challenge was the security of the SDN controllers. In Chapter 2, we have performed in-depth literature reviews on the SDN CPP. Moreover, an overview of the SDN, its architecture, and single and distributed controller placement strategies was discussed in detail. Furthermore, we have discussed different types of controllers as well as the factors that influenced the CPP. Lastly, we presented a CPP classification approach.

In Chapter 3, we presented the clustering empirical analysis of the SDN-enabled WAN and achieved the CPP objectives. We first presented an ML overview, clustering strategies, traditional clustering, and enhanced clustering algorithms. We performed the empirical analysis on the selected algorithms; the essence was to identify more effective and efficient algorithms based on CPP objectives. Among the algorithms performed, the enhanced DBSCAN algorithm was selected based on its 100% clustering performance accuracy. For this reason, the selection was based on the performance accuracy of the model and was adopted in the CPP framework designed in Chapter 6. In Chapter 4, we performed an empirical analysis of the selected ML techniques. We reviewed the related works on the SDN-based IDS. We then evaluated and selected algorithms based on the accuracy IDs in the SDN. To this end, KNN outperformed the other selected models in terms of ID accuracy. Thus, the KNN was developed in the designed ID framework settings. Chapter 5 presents the adopted research methodology. Moreover, the different paradigms were presented in detail, and

the pragmatic paradigm for this study was selected. Additionally, the DSR was discussed in depth. Furthermore, the qualitative and quantitative methods used for this research were presented.
in detail. The research design methods were also presented. In addition, the collected data and its utilization were presented in detail. The technological tools and parameters for this research were explained in detail. The research evaluation and validity of this research were also presented in detail. Lastly, the administration and ethical considerations of this research were also presented.

Furthermore, in Chapter 6, we designed an integrated framework for CPP and IDS. We first presented the CPP problem formulation, followed by the IDS. The objective parameters of the study were presented in detail. The optimum placement and intrusion detection algorithms were presented. Moreover, we have designed the IDS architecture and shown its operation in the SDN environment. Furthermore, we have presented how the mitigation policy actions were applied to the identified anomaly attack in the network. In addition, the overall IDS algorithm was presented in detail. Again, the integrated framework for CPP and IDS-related functions was further presented. In Chapter 7, we presented the implementation of our models. The CPP objectives and IDS were implemented on the installed Ubuntu 22.04 operating system and VM. The simulation was performed on an open-source virtual machine. Moreover, the open network operating system () X-Wing (LTS) version 1.15 was employed to manage the network components. The parameters and system properties, such as hardware, etc., were employed for the practical implementation. Then, the results obtained from the simulation were used to analyse the nature of the SDN environment-based CPP-IIDS.

How the ROs were met in this thesis are as follows:

RO1: Conducting a comprehensive literature review on SDN and related studies in terms of challenges such as scalability and security issues as well as existing solutions: To achieve RO1, we conducted a thorough and comprehensive literature review on the SDN environment and identified significant issues and challenges such as reliability, security against DDoS attacks, latency, switch assignment (S2C), controller to controller (C2C), and the number of required controllers. To this end, we dedicated Chapter 2 to SDN CPP and conducted an analysis of the CPP clustering techniques in Chapter 3 and intrusion detection in Chapter 4. We empirically analysed different approaches to select the optimum model for both CPP and IDS and applied the DSR to achieve an integrated CPP-IDS framework solution.

RO2: Designing and implementing a robust integrated framework that incorporates both security and CPP for SDN-enabled WAN: RO2 was achieved by developing a robust CPP-IDS framework

to address SDN performance, reliability and security. The proposed framework leveraged the supervised ML clustering technique models for both CPP and IDS, which achieved the worst-case scenario, average latency, maximized the control path, resilience, and SDN control plane attack-aware strategy. While the enhanced DBSCAN model leveraged the CPP objectives—number of controllers, switch assignment, latency, reliability, and load balancing. The kNN was adopted to achieve detection accuracy in a given WAN. Based on the evaluation results, the kNN achieved a high detection accuracy compared to other models such as Random Forest and Naïve Bayes-based detection of network anomalies.

8.3 Conclusion

Although the nature of SDN architecture is very advantageous to address the complex issues and challenges in traditional networks, it also brings some security issues and challenges since security was not considered in the initial design. Without a robust built-in security model that can effectively and efficiently identify anomalous flows, the whole network could suffer from a lack of trusted connections between the layers. In addition, different security mechanisms were proposed and developed in the SDN environment; however, robust and efficient approaches are still required to be implemented in the network. Furthermore, recent research has shown that SDN-based CPP practical solutions can't be achieved in isolation since they depend heavily on solutions to other problems such as security problems, controller load balancing problems, and others. This means that to achieve a practical solution to SDN CPP, other related network problems should be tackled collectively under a large integrated framework. In our study, we presented a designed integrated framework for CPP-IDS that is both efficient and attack-aware, thereby improving the performance of the SDN in WAN settings. Based on the effective and efficient operation of our model, we, therefore, recommended that the integrated framework-based SDN CPP-IIDS will go a long way in making the SDN WAN resilient to failure as well as providing secure means for the number of placed controllers in the SDN environment.

8.4 Research Limitation

The proposed integrated CPP-IDS framework discussed in this thesis has limitations, and some of these limitations are as follows: The proposed framework is capable of identifying only the specific attack in the network, which is the DoS/DDoS attack. The designed framework is not intended to identify any form of attack apart from the DDoS in the network. Once the traffic was evaluated using the attack-aware strategy on the SDN control plane, the minimized time was reported; that is, how fast the alarm was sent to the administrator once the DDoS attack was detected. Moreover, less than 4 seconds was reported for any identical traffic flow associated with the DDoS attack in the

SDN environment, and the alarming size of the DDoS traffic flow was more than 64 MB. Furthermore, less of the time-enforced policy was evaluated by the administrator to suspend, manage, and respond to the identified DDoS attack on the network. The designed framework was not meant to be deployed in any kind of environment, apart from the SDN environment.

8.5 Recommendations

In the SDN environment, several studies have proven that signature-based techniques are not efficient for the SDN due to its architecture; hence, attention was shifted to flow-based anomaly detection that was implemented to overcome the limitations of the signature-based ID system. Moreover, we recommended that the application of ML techniques in the SDN bring considerable attention because it can detect patterns within the application that are based on the selected feature set and classify the encrypted traffic at a lower computational cost, QoS and QoE, and security. Furthermore, on the large scale of SDN environments, multi-controllers' are essential in ensuring QoS, efficiency, and effective performance; thus, the number of controllers to be deployed and their location should be carefully considered in the network.

Without proper location and allocation of the controller, the performance and QoS of the network can be negatively impacted, and it may fail to defend against anomaly attacks. Furthermore, to achieve optimal controller placement in SDN, several techniques must be proposed, including algorithms and other metrics to improve clustering and classification performance. In addition, the practice of data features during clustering and classification analysis should be compared to achieve effective and efficient clustering and classification accuracy. To minimize the likelihood of the connected switches being affected by anomaly attacks, the SDN environment should be provided with tighter security measurements.

8.6 Future Works

The solution-based CPP is to find the optimal controller placement in the SDN network. Due to the dynamic nature of the SDN, adaptive-based CPP is the way forward in the research field [62, 63]. Based on the controller placement in the network, we found out that relocation within its domain cannot assist in improving the traffic load imbalance across the SDN domains [614, 615]. Thus, controller relocation alone cannot be an effective approach. Based on the CPP traffic prediction, we found that the decision on the switch migration was based on the current and past traffic conditions [620]. Developing a method that predicts traffic behaviour in the next epoch and deciding on the switch reassignment can prove to be much more beneficial. Furthermore, based on the addition and deletion of the controllers, we have found that the issue that was overlooked was the cost of turning

on and switching off a controller in terms of energy and time [35]. It may be less productive to switch off the controllers for a short period if they are required to be powered up again.

For anomaly detection, the use of ML techniques has gained importance due to their efficiency in evaluating network security. Since the nature of the attacks is dynamic, the adaptability of other detection methods is required [62, 63]. Moreover, developing a feature selection method with a classifier that can reduce the dimensions of the dataset is still an ongoing challenge [54]. Furthermore, designing a centralized SDN controller that can monitor and implement real-time intrusion detection in the high-speed network should be taken as a possible future direction [75, 76].

8.7 Appendix A: SDN environment simulated flow statistics

deviceId, Bytes,Packets,Duration,Priority,TableId,PayLoad
100007a585b6f,0,0,0,40000,0,normal
100009465555a,0,0,0,40000,0,normal
10000ea6f4b8e,0,0,0,40000,0,normal
1000002bbd8d4,702,9,0,40000,0,normal
10000c70edd85,0,0,0,40000,0,normal
10000dc56d70b,702,9,0,40000,0,normal
10000646f55aa,2262,29,2,40000,0,normal
10000a6288ee9,93240,2220,1,40000,0,abnormal
1000011326ce6,0,0,0,40000,0,normal
10000687421fe,468,6,4,40000,0,normal
10000c0cb82c3,702,9,0,40000,0,normal
10000e1a5f53e,78834,1877,0,40000,0,normal
10000ee9234f2,702,9,0,40000,0,normal
1000019e87bbb,49434,1177,2,40000,0,normal
100002d691d89,624,8,5,40000,0,normal
10000710b8f9c,624,8,5,40000,0,normal
1000001717bca,0,0,0,40000,0,normal
1000051debc3d,0,0,0,40000,0,normal
10000ce551323,0,0,0,40000,0,normal
100004086df16,85050,2025,5,40000,0,normal
100004308a118,2496,32,8,40000,0,abnormal
100004d20d7bf,2496,32,8,40000,0,normal
100004ea11ac0,936,12,0,40000,0,normal
10000e91c3202,936,12,0,40000,0,abnormal
10000f0feacfe,0,0,0,40000,0,normal
100003233c5c0,702,9,13,40000,0,normal
100007aa3d8d8,702,9,13,40000,0,normal
10000d78dcf79,28098,669,12,40000,0,normal
10000494083df,702,9,0,40000,0,normal
100007a585b6f,0,0,0,40000,0,normal
100009465555a,0,0,0,40000,0,normal
10000ea6f4b8e,0,0,0,40000,0,normal
1000002bbd8d4,702,9,0,40000,0,normal
10000c70edd85,0,0,0,40000,0,normal
10000dc56d70b,702,9,0,40000,0,normal
10000646f55aa,2262,29,2,40000,0,normal
10000a6288ee9,93240,2220,1,40000,0,abnormal
1000011326ce6,0,0,0,40000,0,normal
10000687421fe,468,6,4,40000,0,normal

10000c0cb82c3,702,9,0,40000,0,normal
10000e1a5f53e,78834,1877,0,40000,0,normal
10000ee9234f2,702,9,0,40000,0,normal
1000019e87bbb,49434,1177,2,40000,0,normal
100002d691d89,624,8,5,40000,0,normal
10000710b8f9c,624,8,5,40000,0,normal
1000001717bca,0,0,0,40000,0,normal
1000051debc3d,0,0,0,40000,0,normal
10000ce551323,0,0,0,40000,0,normal
100004086df16,85050,2025,5,40000,0,normal
100004308a118,2496,32,8,40000,0,abnormal
100004d20d7bf,2496,32,8,40000,0,normal
100004ea11ac0,936,12,0,40000,0,normal
10000e91c3202,936,12,0,40000,0,abnormal
10000f0feacfe,0,0,0,40000,0,normal
100003233c5c0,702,9,13,40000,0,normal
100007aa3d8d8,702,9,13,40000,0,normal
10000d78dcf79,28098,669,12,40000,0,normal
10000494083df,702,9,0,40000,0,normal
100005f2b175b,702,9,0,40000,0,normal
10000648d5a4f,0,0,0,40000,0,normal
1000026a0ffae,1172,15,0,40000,0,normal
1000096b12177,58716,1398,0,40000,0,normal
10000f5cb6cad,1172,15,0,40000,0,normal
1000061e06af9,0,0,0,40000,0,normal
10000b4f580c9,0,0,0,40000,0,normal
10000c023e2a3,0,0,0,40000,0,normal
100001eb71f99,1014,13,3,40000,0,normal
100002145004b,35238,839,3,40000,0,normal
10000cc389fc9,1014,13,3,40000,0,normal
1000046ad8304,0,0,0,40000,0,normal
10000d91ddbef,702,9,4,40000,0,normal
10000fe4ce66d,702,9,4,40000,0,normal
100003c692fe0,234,3,0,40000,0,normal
100006e22ed0b,234,3,0,40000,0,normal
10000b3951493,0,0,0,40000,0,normal
100000a6016cb,0,0,0,40000,0,normal
10000602f0307,312,4,0,40000,0,normal
10000c09f8512,312,4,0,40000,0,normal
100004b9ea92b,780,10,3,40000,0,normal
i10000e78874b9,780,10,3,40000,0,normal
10000f272a734,82656,1968,1,40000,0,normal
100001188d205,72156,1718,8,40000,0,normal
100005e7ca7f8,702,9,8,40000,0,normal
10000edec9085,624,8,8,40000,0,normal
100000eb2cb7a,390,5,7,40000,0,normal
100005c6d1f55,390,5,7,40000,0,normal
10000f97f6548,31458,749,3,40000,0,normal
100003378e52e,0,0,0,40000,0,normal
10000f3560271,0,0,0,40000,0,normal
1000038aaca10,156,2,7,40000,0,normal
10000b9843412,156,2,7,40000,0,normal
10000e5b7fcac,17262,411,1,40000,0,normal
100006c1db14a,0,0,0,40000,0,normal
100007ec64994,0,0,0,40000,0,normal
10000e1fdb1e3,0,0,0,40000,0,normal
10000be2d48f1,234,3,9,40000,0,normal

10000f4d2082b,312,4,10,40000,0,normal
10000fd90cb94,42,1,8,40000,0,normal
10000104a4232,0,0,0,40000,0,normal
100006b5a3170,0,0,0,40000,0,normal
10000c023bc27,0,0,0,40000,0,normal
10000634f6d84,702,9,8,40000,0,normal
10000806deb31,38178,909,5,40000,0,normal
10000f16433ca,702,9,7,40000,0,normal
10000300c561a,0,0,0,40000,0,normal
1000036c9c2b9,0,0,0,40000,0,normal
10000c2c72253,0,0,0,40000,0,normal
i100001cc97dc4,1482,19,12,40000,0,normal
100002634c13b,47292,1126,2,40000,0,normal
100009c6f89da,1482,19,12,40000,0,normal
10000b2edd22d,0,0,0,40000,0,normal
100007a585b6f,0,0,0,40000,0,normal
100009465555a,0,0,0,40000,0,normal
10000ea6f4b8e,0,0,0,40000,0,normal
1000002bbd8d4,702,9,0,40000,0,normal
10000c70edd85,0,0,0,40000,0,normal
10000dc56d70b,702,9,0,40000,0,normal
10000646f55aa,2262,29,2,40000,0,normal
10000a6288ee9,93240,2220,1,40000,0,abnormal
1000011326ce6,0,0,0,40000,0,normal
10000687421fe,468,6,4,40000,0,normal
10000c0cb82c3,702,9,0,40000,0,normal
10000e1a5f53e,78834,1877,0,40000,0,normal
10000ee9234f2,702,9,0,40000,0,normal
1000019e87bbb,49434,1177,2,40000,0,normal
100002d691d89,624,8,5,40000,0,normal
10000710b8f9c,624,8,5,40000,0,normal
1000001717bca,0,0,0,40000,0,normal
1000051debc3d,0,0,0,40000,0,normal
10000ce551323,0,0,0,40000,0,normal
100004086df16,85050,2025,5,40000,0,normal
100004308a118,2496,32,8,40000,0,abnormal
100004d20d7bf,2496,32,8,40000,0,normal
100004ea11ac0,936,12,0,40000,0,normal
10000e91c3202,936,12,0,40000,0,abnormal
10000f0feacfe,0,0,0,40000,0,normal
100003233c5c0,702,9,13,40000,0,normal
100007aa3d8d8,702,9,13,40000,0,normal
10000d78dcf79,28098,669,12,40000,0,normal
10000494083df,702,9,0,40000,0,normal
100005f2b175b,702,9,0,40000,0,normal
10000648d5a4f,0,0,0,40000,0,normal
1000026a0ffae,1172,15,0,40000,0,normal
1000096b12177,58716,1398,0,40000,0,normal
10000f5cb6cad,1172,15,0,40000,0,normal
1000061e06af9,0,0,0,40000,0,normal
10000b4f580c9,0,0,0,40000,0,normal
10000c023e2a3,0,0,0,40000,0,normal
100001eb71f99,1014,13,3,40000,0,normal
100002145004b,35238,839,3,40000,0,normal
10000cc389fc9,1014,13,3,40000,0,normal
1000046ad8304,0,0,0,40000,0,normal
10000d91ddbef,702,9,4,40000,0,normal

10000fe4ce66d,702,9,4,40000,0,normal
100003c692fe0,234,3,0,40000,0,normal
100006e22ed0b,234,3,0,40000,0,normal
10000b3951493,0,0,0,40000,0,normal
100000a6016cb,0,0,0,40000,0,normal
10000602f0307,312,4,0,40000,0,normal
10000c09f8512,312,4,0,40000,0,normal
100004b9ea92b,780,10,3,40000,0,normal
i10000e78874b9,780,10,3,40000,0,normal
10000f272a734,82656,1968,1,40000,0,normal
100001188d205,72156,1718,8,40000,0,normal
100005e7ca7f8,702,9,8,40000,0,normal
10000edec9085,624,8,8,40000,0,normal
100000eb2cb7a,390,5,7,40000,0,normal
100005c6d1f55,390,5,7,40000,0,normal
10000f97f6548,31458,749,3,40000,0,normal
100003378e52e,0,0,0,40000,0,normal
10000f3560271,0,0,0,40000,0,normal
1000038aaca10,156,2,7,40000,0,normal
10000b9843412,156,2,7,40000,0,normal
10000e5b7fcac,17262,411,1,40000,0,normal
100006c1db14a,0,0,0,40000,0,normal
100007ec64994,0,0,0,40000,0,normal
10000e1fdb1e3,0,0,0,40000,0,normal
10000be2d48f1,234,3,9,40000,0,normal
10000f4d2082b,312,4,10,40000,0,normal
10000fd90cb94,42,1,8,40000,0,normal
10000104a4232,0,0,0,40000,0,normal
100006b5a3170,0,0,0,40000,0,normal
10000c023bc27,0,0,0,40000,0,normal
10000634f6d84,702,9,8,40000,0,normal
10000806deb31,38178,909,5,40000,0,normal
10000f16433ca,702,9,7,40000,0,normal
10000300c561a,0,0,0,40000,0,normal
1000036c9c2b9,0,0,0,40000,0,normal
10000c2c72253,0,0,0,40000,0,normal
i100001cc97dc4,1482,19,12,40000,0,normal
100002634c13b,47292,1126,2,40000,0,normal
100009c6f89da,1482,19,12,40000,0,normal
10000b2edd22d,0,0,0,40000,0,normal
10000e5b1bf14,624,8,1,40000,0,normal
10000f59c1910,624,8,0,40000,0,normal
10000a18b080b,234,3,0,40000,0,normal
10000a635a102,234,3,0,40000,0,normal
10000c5bdce55,0,0,0,40000,0,normal
100000c0c7e8a,24780,590,8,40000,0,normal
100002cb6ed06,546,7,9,40000,0,normal
100002eae5884,546,7,9,40000,0,normal
10000e5b1bf14,624,8,1,40000,0,normal
10000f59c1910,624,8,0,40000,0,normal
10000a18b080b,234,3,0,40000,0,normal
10000a635a102,234,3,0,40000,0,normal
10000c5bdce55,0,0,0,40000,0,normal
100000c0c7e8a,24780,590,8,40000,0,normal
100002cb6ed06,546,7,9,40000,0,normal
100002eae5884,546,7,9,40000,0,normal
100005f2b175b,702,9,0,40000,0,normal

10000648d5a4f,0,0,0,40000,0,normal
1000026a0ffae,1172,15,0,40000,0,normal
1000096b12177,58716,1398,0,40000,0,normal
10000f5cb6cad,1172,15,0,40000,0,normal
1000061e06af9,0,0,0,40000,0,normal
10000b4f580c9,0,0,0,40000,0,normal
10000c023e2a3,0,0,0,40000,0,normal
100001eb71f99,1014,13,3,40000,0,normal
100002145004b,35238,839,3,40000,0,normal
10000cc389fc9,1014,13,3,40000,0,normal
1000046ad8304,0,0,0,40000,0,normal
10000d91ddbef,702,9,4,40000,0,normal
10000fe4ce66d,702,9,4,40000,0,normal
100003c692fe0,234,3,0,40000,0,normal
100006e22ed0b,234,3,0,40000,0,normal
10000b3951493,0,0,0,40000,0,normal
100000a6016cb,0,0,0,40000,0,normal
10000602f0307,312,4,0,40000,0,normal
10000c09f8512,312,4,0,40000,0,normal
100004b9ea92b,780,10,3,40000,0,normal
i10000e78874b9,780,10,3,40000,0,normal
10000f272a734,82656,1968,1,40000,0,normal
100001188d205,72156,1718,8,40000,0,normal
100005e7ca7f8,702,9,8,40000,0,normal
10000edec9085,624,8,8,40000,0,normal
100000eb2cb7a,390,5,7,40000,0,normal
100005c6d1f55,390,5,7,40000,0,normal
10000f97f6548,31458,749,3,40000,0,normal
100003378e52e,0,0,0,40000,0,normal
10000f3560271,0,0,0,40000,0,normal
1000038aaca10,156,2,7,40000,0,normal
10000b9843412,156,2,7,40000,0,normal
10000e5b7fcac,17262,411,1,40000,0,normal
100006c1db14a,0,0,0,40000,0,normal
100007ec64994,0,0,0,40000,0,normal
10000e1fdb1e3,0,0,0,40000,0,normal
10000be2d48f1,234,3,9,40000,0,normal
10000f4d2082b,312,4,10,40000,0,normal
10000fd90cb94,42,1,8,40000,0,normal
10000104a4232,0,0,0,40000,0,normal
100006b5a3170,0,0,0,40000,0,normal
10000c023bc27,0,0,0,40000,0,normal
10000634f6d84,702,9,8,40000,0,normal
10000806deb31,38178,909,5,40000,0,normal
10000f16433ca,702,9,7,40000,0,normal
10000300c561a,0,0,0,40000,0,normal
1000036c9c2b9,0,0,0,40000,0,normal
10000c2c72253,0,0,0,40000,0,normal
i100001cc97dc4,1482,19,12,40000,0,normal
100002634c13b,47292,1126,2,40000,0,normal
100009c6f89da,1482,19,12,40000,0,normal
10000b2edd22d,0,0,0,40000,0,normal
10000e5b1bf14,624,8,1,40000,0,normal
10000f59c1910,624,8,0,40000,0,normal
10000a18b080b,234,3,0,40000,0,normal
10000a635a102,234,3,0,40000,0,normal
10000c5bdce55,0,0,0,40000,0,normal

100000c0c7e8a,24780,590,8,40000,0,normal
100002cb6ed06,546,7,9,40000,0,normal
100002eae5884,546,7,9,40000,0,normal

8.8 References

[1] M. R. Parsaei, R. Mohammadi, and R. Javidan, "A new adaptive traffic engineering method for telesurgery using ACO algorithm over software defined networks," *European Research in Telemedicine/La Recherche Europeenne en Telemedecine,* vol. 6, no. 3-4, pp. 173-180, 2017.

[2] R. Chai, Q. Yuan, L. Zhu, and Q. Chen, "Control plane delay minimization-based capacitated controller placement algorithm for SDN," *EURASIP Journal on Wireless Communications and Networking,* vol. 2019, no. 1, pp. 1-17, 2019.

[3] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification," *ACM SIGCOMM Computer Communication Review,* vol. 36, no. 5, pp. 5-16, 2006.

[4] M. E. Ahmed, H. Kim, and M. Park, "Mitigating DNS query-based DDoS attacks with machine learning on software-defined networking," in *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM)*, 2017, pp. 11-16: IEEE.

[5] T. J. Parvat and P. Chandra, "A Novel approach to deep packet inspection for intrusion detection," *Procedia Computer Science,* vol. 45, pp. 506-513, 2015.

[6] B. Mirkhanzadeh *et al.*, "Demonstration of joint operation across openROADM metro network, OpenFlow packet domain, and OpenStack compute domain," in *Optical Fiber Communication Conference*, 2020, p. W3C. 3: Optical Society of America.

[7] J. Xie *et al.*, "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Communications Surveys & Tutorials,* vol. 21, no. 1, pp. 393-430, 2018.

[8] J. Yan and J. Yuan, "A survey of traffic classification in software defined networks," in *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*, 2018, pp. 200-206: IEEE.

[9] P. Foremski, "On different ways to classify Internet traffic: a short review of selected publications," *Theoretical and Applied Informatics,* vol. 25, 2013.

[10] X. Ji *et al.*, "Overview of 5G security technology," *Science China Information Sciences,* vol. 61, no. 8, p. 081301, 2018.

[11] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On Controller Performance in Software-Defined Networks," in *2nd {USENIX} Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE 12)*, 2012.

[12] S. Vissicchio, L. Vanbever, and O. Bonaventure, "Opportunities and research challenges of hybrid software defined networks," *ACM SIGCOMM Computer Communication Review,* vol. 44, no. 2, pp. 70-75, 2014.

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

[14] W.-X. Liu, J. Zhang, Z.-W. Liang, L.-X. Peng, and J. Cai, "Content popularity prediction and caching for ICN: A deep learning approach with SDN," *IEEE Access,* vol. 6, pp. 5075-5089, 2017.

[15] H. Mostafaei, M. Menth, and M. S. Obaidat, "A learning automaton-based controller placement algorithm for software-defined networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1-6: IEEE.

[16] L. Zhu, R. Chai, and Q. Chen, "Control plane delay minimization based SDN controller placement scheme," in *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, 2017, pp. 1-6: IEEE.

[17] A. Abusnaina, A. Khormali, D. Nyang, M. Yuksel, and A. Mohaisen, "Examining the robustness of learning-based DDoS detection in software defined networks," in *2019 IEEE Conference on Dependable and Secure Computing (DSC)*, 2019, pp. 1-8: IEEE.

[18] M. Abderrahim, A. B. Letaifa, A. Haji, and S. Tabbane, "A game theory-based effective network management in SDN networks," in *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, 2018, pp. 390-395: IEEE.

[19] E. Z. Abdevand, S. Ghanbari, Z. Umarova, and Z. Iztayev, "Introducing a New Intrusion Detection Method in The SDN Network to Increase Security Using Decision Tree and Neural Network."

[20] M. E. Ahmed, H. Kim, and M. Park, "Mitigating DNS query-based DDoS attacks with machine learning on software-defined networking," in *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM)*, 2017, pp. 11-16: IEEE.

[21] J. Ashraf and S. Latif, "Handling intrusion and DDoS attacks in Software Defined Networks using machine learning techniques," in *2014 National Software Engineering Conference*, 2014, pp. 55-60: IEEE.

[22] P. Amaral, J. Dinis, P. Pinto, L. Bernardo, J. Tavares, and H. S. Mamede, "Machine learning in software defined networks: Data collection and traffic classification," in *2016 IEEE 24th International conference on network protocols (ICNP)*, 2016, pp. 1-5: IEEE.

[23] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based DDoS detection system in software-defined networking (SDN)," *arXiv preprint arXiv:1611.07400*, 2016.

[24] V. Huang *et al.*, "A Scalable Approach to SDN Control Plane Management: High Utilization Comes with Low Latency," *IEEE Transactions on Network and Service Management*, 2020.

[25] S. H. Haji *et al.*, "Comparison of software defined networking with traditional networking," pp. 1-18, 2021.

[26] N. Ahuja, G. Singal, D. Mukhopadhyay, N. J. J. o. N. Kumar, and C. Applications, "Automated DDOS attack detection in software defined networking," vol. 187, p. 103108, 2021.

[27] O. Aouedi, K. Piamrat, and B. J. F. i. Parrein, "Intelligent Traffic Management in Next-Generation Networks," vol. 14, no. 2, p. 44, 2022.

[28] P. A. Agg and Z. C. J. G. Johanyák, "Energy savings in SDN networks," vol. 8, no. 1, pp. 205-210, 2021.

[29] R. Sebopelo, B. Isong, N. Gasela, and A. M. Abu-Mahfouz, "A review of intrusion detection techniques in the SDN environment," in *2021 3rd International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, 2021, pp. 1-9: IEEE.

[30] R. Sebopelo, B. J. J. o. I. S. Isong, and Informatics, "An Integrated Framework for Controllers Placement and Security in Software-Defined Networks Ecosystem," vol. 6, no. 1, pp. 464-494, 2024.

[31] T. Guesmi, A. Kalghoum, B. M. Alshammari, H. Alsaif, and A. J. S. Alzamil, "Leveraging software-defined networking approach for future information-centric networking enhancement," vol. 13, no. 3, p. 441, 2021.

[32] R. Kunze, D. Telekom, and D. Beller, "Internet Engineering Task Force D. Hiremagalur, Ed. Internet-Draft G. Grammel, Ed. Intended status: Standards Track Juniper Expires: May 3, 2018, G. Galimberti, Ed. Cisco."

[33] N. Oliveira, I. Praça, E. Maia, and O. J. A. S. Sousa, "Intelligent cyber attack detection and classification for network-based intrusion detection systems," vol. 11, no. 4, p. 1674, 2021.

[34] A. A. M. Alraawi and S. A. N. Adam, "Performance Evaluation of Controller Based SDN Network Over Non-controller Based Network in Data Center Network," in *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, pp. 1-4: IEEE.

[35] A. A. Ahmad, "Solution Model for Intrusion Detection in Software Defined Networking (SDN) using Machine Learning."

[36] D. Archana and S. Prakasam, "Load Balancing Using SDN in Wireless Sensor Network Environment."
[37] M. R. Haque *et al.*, "Unprecedented smart algorithm for uninterrupted SDN services during DDoS attack," vol. 70, no. 1, pp. 875-894, 2022.
[38] M. R. Macedonia and D. P. J. C. Brutzman, "MBone provides audio and video across the Internet," vol. 27, no. 4, pp. 30-36, 1994.
[39] B. Mirkhanzadeh *et al.*, "Demonstration of joint operation across openROADM metro network, OpenFlow packet domain, and OpenStack compute domain," in *Optical Fiber Communication Conference*, 2020, p. W3C. 3: Optical Society of America.
[40] L. C. Costa *et al.*, "OpenFlow data planes performance evaluation," vol. 147, p. 102194, 2021.
[41] R. Wazirali, R. Ahmad, and S. J. A. S. Alhiyari, "SDN-OpenFlow Topology Discovery: An Overview of Performance Issues," vol. 11, no. 15, p. 6999, 2021.
[42] S. Miano, F. J. I. J. o. E. Risso, and C. Engineering, "Transforming a traditional home gateway into a hardware-accelerated SDN switch," vol. 10, no. 3, p. 2668, 2020.
[43] D. Erickson *et al.*, "A demonstration of virtual machine mobility in an OpenFlow network," ed: Aug 2008.
[44] I. N. Khoerotunisa, S. N. Hertiana, and R. M. J. J. T. I. Negara, "ANALYSIS OF USER MOBILITY PERFORMANCE ON SOFTWARE DEFINED WIRELESS NETWORK USING DIJKSTRA ALGORITHM," vol. 2, no. 2, pp. 127-133, 2021.
[45] B. Wang, Z. Bin, M. Yao, and W. Wang, "Virtual Tenant Network Research based on Price Mechanism in SDN Architecture," in *First International Conference on Information Sciences, Machinery, Materials and Energy*, 2015, pp. 663-666: Atlantis Press.
[46] S. Badotra and S. N. J. C. C. Panda, "Evaluation and comparison of OpenDayLight and open networking operating system in software-defined networking," vol. 23, no. 2, pp. 1281-1291, 2020.
[47] J. Wang *et al.*, "Dynamic SDN Control Plane Request Assignment in NFV Datacenters," vol. 8, no. 1, pp. 680-694, 2020.
[48] H. Kim and N. J. I. C. M. Feamster, "Improving network management with software defined networking," vol. 51, no. 2, pp. 114-119, 2013.
[49] G. J. S. M. J. Hu, "A Separation Property of the Observer-Based Stabilizing Controller for Linear Delay Systems," vol. 62, no. 4, pp. 763-772, 2021.
[50] T. Spalink, S. Karlin, L. Peterson, and Y. J. A. S. O. S. R. Gottlieb, "Building a robust software-based router using network processors," vol. 35, no. 5, pp. 216-229, 2001.
[51] R. Sultana, J. Grover, and M. J. V. C. Tripathi, "Security of SDN-based vehicular ad hoc networks: State-of-the-art and challenges," vol. 27, p. 100284, 2021.
[52] S. El Jaouhari, A. Bouabdallah, A. A. J. J. o. A. I. Corici, and H. Computing, "SDN-based security management of multiple WoT Smart Spaces," vol. 12, no. 10, pp. 9081-9096, 2021.
[53] H. A. Hassan, E. E. Hemdan, W. El-Shafai, M. Shokair, and F. E. Abd El-Samie, "An Efficient Intrusion Detection System for SDN using Convolutional Neural Network," in *2021 International Conference on Electronic Engineering (ICEEM)*, 2021, pp. 1-5: IEEE.
[54] G. Wang, Y. Zhao, J. Huang, and W. J. I. N. Wang, "The controller placement problem in software defined networking: A survey," vol. 31, no. 5, pp. 21-27, 2017.
[55] R. Ferrús *et al.*, "SDN/NFV-enabled satellite communications networks: Opportunities, scenarios and challenges," vol. 18, pp. 95-112, 2016.
[56] K. Sood, Y. J. J. o. c. Xiang, and I. Networks, "The controller placement problem or the controller selection problem? " vol. 2, no. 3, pp. 1-9, 2017.
[57] T. Das and M. Gurusamy, "INCEPT: INcremental ControllEr PlacemenT in software defined networks," in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, 2018, pp. 1-6: IEEE.

[58] J. Yu *et al.*, "Dual use SDN controller for management and experimentation in a field deployed testbed," in *2020 Optical Fiber Communications Conference and Exhibition (OFC)*, 2020, pp. 1-3: IEEE.
[59] G. Schütz and J. A. J. C. C. Martins, "A comprehensive approach for optimizing controller placement in Software-Defined Networks," vol. 159, pp. 198-205, 2020.
[60] N. Nambiar, "Attack Resilience of Cache Replacement Policies: Implementation and Experimentation In SDN," 2021.
[61] B. Singh, V. Mukherjee, P. J. R. Tiwari, and S. E. Reviews, "A survey on impact assessment of DG and FACTS controllers in power systems," vol. 42, pp. 846-882, 2015.
[62] D. Scano, L. Valcarenghi, K. Kondepu, P. Castoldi, and A. Giorgetti, "Network Slicing in SDN Networks," in *2020 22nd International Conference on Transparent Optical Networks (ICTON)*, 2020, pp. 1-4: IEEE.
[63] F. Bannour, S. Souihi, and A. J. F. G. C. S. Mellouk, "Adaptive distributed SDN controllers: Application to Content-Centric Delivery Networks," vol. 113, pp. 78-93, 2020.
[64] C. Qian, K. Yu, H. J. J. o. B. Gu, and I. Marketing, "Flexibility mechanisms in a dynamic distribution network," 2020.
[65] D. Staessens, S. Sharma, D. Colle, M. Pickavet, and P. Demeester, "Software defined networking: Meeting carrier grade requirements," in *2011 18th IEEE workshop on local & metropolitan area networks (LANMAN)*, 2011, pp. 1-6: IEEE.
[66] S. H. Haji *et al.*, "Comparison of software defined networking with traditional networking," pp. 1-18, 2021.
[67] H. Mahantesh, M. N. Guptha, and M. J. W. P. C. Hema, "Optimized Path and Reduced Rule Caching Cost for Software Defined Network (SDN) Based Internet of Things (IoT)," pp. 1-17, 2021.
[68] F. Meneses, D. Corujo, C. Guimaraes, and R. L. Aguiar, "An abstraction framework for flow mobility in multi-technology 5G environments using virtualization and SDN," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, 2017, pp. 1-5: IEEE.
[69] S. Yang, L. Cui, Z. Chen, W. J. I. T. o. N. Xiao, and S. Management, "An efficient approach to robust SDN controller placement for security," vol. 17, no. 3, pp. 1669-1682, 2020.
[70] M. Uddin, S. Mukherjee, H. Chang, and T. J. I. J. o. S. A. i. C. Lakshman, "SDN-based multi-protocol edge switching for IoT service automation," vol. 36, no. 12, pp. 2775-2786, 2018.
[71] R. Vilalta *et al.*, "Experimental evaluation of control and monitoring protocols for optical SDN networks and equipment [Invited Tutorial]," vol. 13, no. 8, pp. D1-D12, 2021.
[72] C. Dong and R. J. J. o. B. L. Franklin, "From the digital Internet to the physical Internet: A conceptual framework with a stylized network model," vol. 42, no. 1, pp. 108-119, 2021.
[73] A. Bukharina, "Advantages of Method of Collective Dynamic Routing in IP Networks," in *2021 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO*, 2021, pp. 1-4: IEEE.
[74] G. Altangerel, T. Chuluuntsetseg, and D. J. a. p. a. Yamkhin, "Performance analysis of SDN controllers: POX, Floodlight and Opendaylight," 2021.
[75] L. Zhu, M. M. Karim, K. Sharif, F. Li, X. Du, and M. J. a. p. A. Guizani, "SDN controllers: Benchmarking & performance evaluation," 2019.
[76] M. T. Islam, N. Islam, and M. J. W. P. C. Al Refat, "Node to node performance evaluation through RYU SDN controller," pp. 1-16, 2020.
[77] Q. Li, Y. Chen, P. P. Lee, M. Xu, and K. J. I. A. T. o. N. Ren, "Security policy violations in SDN data plane," vol. 26, no. 4, pp. 1715-1727, 2018.
[78] A. A. Seyedkolaei and S. A. H. J. T. J. o. S. Seno, "Multi-criteria decision-making for controller placement in software-defined wide-area networks," pp. 1-27, 2021.
[79] M. el Yaakoubi, P. Ravesteijn, A. Prinsen, H. Hooimeijer, and M. van der Ven, "Data Driven Decision Support: The Role of the Controller in Decision-Making Processes," in

16th European Conference on Management, Leadership and Governance, 2020, p. 73: Academic Conferences International Limited.

[80] A. Shaghaghi, M. A. Kaafar, R. Buyya, S. J. H. o. C. N. Jha, and C. Security, "Software-defined network (SDN) data plane security: issues, solutions, and future directions," pp. 341-387, 2020.

[81] D. Lunagariya and B. Goswami, "A Comparative Performance Analysis of Stellar SDN Controllers using Emulators," in *2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, 2021, pp. 1-9: IEEE.

[82] R. S. Tessinari *et al.*, "Demonstration of a Dynamic QKD Network Control Using a QKD-Aware SDN Application Over a Programmable Hardware Encryptor," in *Optical Fiber Communication Conference*, 2021, p. M2B. 3: Optical Society of America.

[83] X. Zhang, L. Cui, F. P. Tso, W. J. I. T. o. N. Jia, and S. Management, "pHeavy: Predicting Heavy Flows in the Programmable Data Plane," 2021.

[84] A. Panda, W. Zheng, X. Hu, A. Krishnamurthy, and S. Shenker, "{SCL}: Simplifying Distributed {SDN} Control Planes," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 329-345.

[85] Y.-Z. Cai, Y.-T. Wang, and M.-H. J. C. N. Tsai, "Dynamic adjustment for proactive flow installation mechanism in SDN-based IoT," vol. 194, p. 108167, 2021.

[86] R. Fitria, D. Yulisda, and S. J. J. S. I. Fachrurrazi, "Software-Defined Networking (Sdn) In Modern Network Management," vol. 5, no. 1, 2021.

[87] S. B. H. Said, Q. H. Truong, and M. J. A. S. R. Boc, "Sdn-based configuration solution for IEEE 802.1 time sensitive networking (TSN)," vol. 16, no. 1, pp. 27-32, 2019.

[88] P. Saint-Andre, K. Smith, R. Tronçon, and R. Troncon, *XMPP: the definitive guide*. " O'Reilly Media, Inc.", 2009.

[89] E. Al-Masri *et al.*, "Investigating messaging protocols for the Internet of Things (IoT)," vol. 8, pp. 94880-94911, 2020.

[90] A. Lara and L. Quesada, "Performance analysis of SDN northbound interfaces," in *2018 IEEE 10th Latin-American Conference on Communications (LATINCOM)*, 2018, pp. 1-6: IEEE.

[91] M. Paliwal, D. Shrimankar, and O. J. I. A. Tembhurne, "Controllers in SDN: A review report," vol. 6, pp. 36256-36270, 2018.

[92] Y. Tseng, M. Pattaranantakul, R. He, Z. Zhang, and F. Naït-Abdesselam, "Controller DAC: Securing SDN controller with dynamic access control," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1-6: IEEE.

[93] S. Chattopadhyay, "SDN for Large Scale IoT Networks," 2021.

[94] F. Benamrane, M. B. Mamoun, R. J. I. J. o. C. N. Benaini, and D. Systems, "New method for controller-to-controller communication in distributed SDN architecture," vol. 19, no. 3, pp. 357-367, 2017.

[95] Q. Qin, K. Poularakis, G. Iosifidis, and L. Tassiulas, "SDN controller placement at the edge: Optimizing delay and overheads," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, 2018, pp. 684-692: IEEE.

[96] E. Ahvar, S. Ahvar, S. M. Raza, J. Manuel Sanchez Vilchez, and G. M. J. N. Lee, "Next Generation of SDN in Cloud-Fog for 5G and Beyond-Enabled Applications: Opportunities and Challenges," vol. 1, no. 1, pp. 28-49, 2021.

[97] S. Rao, J. Sarlashkar, S. Rengarajan, C. Sharp, and G. Neely, "A controls overview on achieving ultra-low NOx," SAE Technical Paper0148-7191, 2020.

[98] M. Joselli *et al.*, "An architecture with automatic load balancing for real-time simulation and visualization systems," vol. 1, no. 3, pp. 207-224, 2010.

[99] N. McKeown *et al.*, "OpenFlow: enabling innovation in campus networks," vol. 38, no. 2, pp. 69-74, 2008.

[100] D. Erickson, "The beacon OpenFlow controller," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, 2013, pp. 13-18.

[101] Q. Ilyas and R. Khondoker, "Security analysis of FloodLight, ZeroSDN, Beacon and POX SDN controllers," in *SDN and NFV security*: Springer, 2018, pp. 85-98.
[102] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, "Advanced study of SDN/OpenFlow controllers," in *Proceedings of the 9th Central & Eastern European software engineering conference in Russia*, 2013, pp. 1-6.
[103] S. A. Shah, J. Faiz, M. Farooq, A. Shafi, and S. A. Mehdi, "An architectural evaluation of SDN controllers," in *2013 IEEE International Conference on Communications (ICC)*, 2013, pp. 3504-3508: IEEE.
[104] A. Voellmy and J. Wang, "Scalable software defined network controllers," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, 2012, pp. 289-290.
[105] A. Kumari and A. S. J. a. p. a. Sairam, "A survey of controller placement problem in software defined networks," 2019.
[106] A. J. A. o. M. Krylatov and A. Intelligence, "Column generation for the equilibrium route-flow traffic assignment problem," pp. 1-17, 2021.
[107] A. Mishra, N. Gupta, and B. J. T. s. Gupta, "Defense mechanisms against DDoS attack based on entropy in SDN-cloud using POX controller," vol. 77, no. 1, pp. 47-62, 2021.
[108] H. L. Aung, S. Chaudhary, M. Saadi, and L. Wuttisittikulkij, "Implementing TrafficEngineering with Segment Routing using OpenDaylight Controller and Pathman-SR," in *2021 36th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, 2021, pp. 1-4: IEEE.
[109] J. Chen, A. Gopal, and B. Dezfouli, "Modeling control traffic in software-defined networks," in *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, 2021, pp. 258-262: IEEE.
[110] Y. Wang, D. Jiang, L. Huo, Y. J. M. N. Zhao, and Applications, "A new traffic prediction algorithm to software defined networking," vol. 26, no. 2, pp. 716-725, 2021.
[111] L. X. Liao, J. Wang, H.-C. Chao, and B. J. J. o. I. T. Qin, "Distributed and Efficient Network Hypervisor for SDN Virtualization," vol. 22, no. 3, pp. 625-636, 2021.
[112] M. Kulkarni, M. Baddeley, and I. J. a. p. a. Haque, "Embedded vs. External Controllers in Software-Defined IoT Networks," 2021.
[113] B. Heller, R. Sherwood, and N. J. A. S. C. C. R. McKeown, "The controller placement problem," vol. 42, no. 4, pp. 473-478, 2012.
[114] M. B. Mabrouk, D. Rehoune, and A. Fotouhi, "Intelligent Routing Approach based on Machine Learning and SDN for Heterogeneous IoTs," in *2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR)*, 2021, pp. 1-6: IEEE.
[115] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop distributed file system," in *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, 2010, pp. 1-10: Ieee.
[116] J. Moeyersons, P. J. Maenhaut, F. Turck, and B. J. I. J. o. N. M. Volckaert, "Pluggable SDN framework for managing heterogeneous SDN networks," vol. 30, no. 2, p. e2087, 2020.
[117] L. Guillen, H. Takahira, S. Izumi, T. Abe, and T. J. I. A. Suganuma, "On Designing a Resilient SDN C/M-Plane for Multi-Controller Failure in Disaster Situations," vol. 8, pp. 141719-141732, 2020.
[118] J. Cui, Q. Lu, H. Zhong, M. Tian, L. J. I. t. o. n. Liu, and s. management, "A load-balancing mechanism for distributed SDN control plane using response time," vol. 15, no. 4, pp. 1197-1206, 2018.
[119] F. J. Ros and P. M. J. C. C. Ruiz, "On reliable controller placements in software-defined networks," vol. 77, pp. 41-51, 2016.
[120] E. Amiri, E. Alizadeh, and K. Raeisi, "An efficient hierarchical distributed SDN controller model," in *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*, 2019, pp. 553-557: IEEE.

[121] Q. Xu, L. Li, J. Liu, and J. Zhang, "A scalable and hierarchical load balancing model for control plane of SDN," in *2018 Sixth International Conference on Advanced Cloud and Big Data (CBD)*, 2018, pp. 6-11: IEEE.
[122] B. Renukadevi and S. D. M. Raja, "Deep packet inspection management application in SDN," in *2017 2nd International Conference on Computing and Communications Technologies (ICCCT)*, 2017, pp. 256-259: IEEE.
[123] R. Vilalta *et al.*, "TeraFlow: Secured Autonomic Traffic Management for a Tera of SDN Flows," in *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, 2021, pp. 377-382: IEEE.
[124] T. Das and M. J. I. C. L. Gurusamy, "Controller placement for resilient network state synchronization in multi-controller sdn," vol. 24, no. 6, pp. 1299-1303, 2020.
[125] S. Tahmasebi, M. Safi, S. Zolfi, M. R. Maghsoudi, H. R. Faragardi, and H. J. S. Fotouhi, "Cuckoo-PC: an evolutionary synchronization-aware placement of SDN controllers for optimizing the network performance in WSNs," vol. 20, no. 11, p. 3231, 2020.
[126] H. Li, G. Shou, Y. Hu, and Y. J. I. S. J. Liu, "Sdn/nfv enhanced time synchronization in packet networks," 2020.
[127] M. Allman, K. Christensen, B. Nordman, and V. Paxson, "Enabling an Energy-Efficient Future Internet Through Selectively Connected End Systems," in *HotNets*, 2007.
[128] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe, "Design and implementation of a routing control platform," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, 2005, pp. 15-28.
[129] K. L. Calvert, W. K. Edwards, and R. E. Grinter, "Moving Toward the Middle: The Case Against the End-to-End Argument in Home Networking," in *HotNets*, 2007.
[130] N. Gude *et al.*, "NOX: towards an operating system for networks," vol. 38, no. 3, pp. 105-110, 2008.
[131] T. Manuel, B. H. J. I. J. o. R. T. Goswami, and Engineering, "Experimenting with scalability of beacon controller in software defined network," vol. 7, no. 5S2, pp. 550-555, 2019.
[132] A. Eftimie and E. Borcoci, "SDN controller implementation using OpenDaylight: experiments," in *2020 13th International Conference on Communications (COMM)*, 2020, pp. 477-481: IEEE.
[133] M. Dagli, S. Keskin, Y. Yigit, and A. Kose, "Resiliency Analysis of and Opendaylight SDN Controllers Against Switch and Link Failures," in *2020 Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, 2020, pp. 149-153: IEEE.
[134] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, 2014, pp. 1-6: IEEE.
[135] T. Koponen *et al.*, "Onix: A distributed control platform for large-scale production networks," in *OSDI*, 2010, vol. 10, pp. 1-6.
[136] J. Singh and S. J. C. S. R. Behal, "Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions," vol. 37, p. 100279, 2020.
[137] R. Tulloh, M. Iqbal, M. Rifki Baihaqi, A. J. I. J. O. C. Bayu Aji Pamungkas, and D. System, "Load Balancing Implementation Strategy for Various Services in Software Defined Network using Controller," 2021.
[138] S. Rowshanrad, V. Abdi, and M. J. I. E. J. Keshtgari, "Performance evaluation of SDN controllers: Floodlight and OpenDaylight," vol. 17, no. 2, pp. 47-57, 2016.
[139] D. Chasaki and C. Mansour, "Detecting Malicious Hosts in SDN through System Call Learning," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2021, pp. 1-2: IEEE.
[140] H. Zhong, J. Fan, J. Cui, Y. Xu, and L. J. C. N. Liu, "Assessing Profit of Prediction for SDN controllers load balancing," vol. 191, p. 107991, 2021.

[141] H. Mokhtar, X. Di, Y. Zhou, A. Hassan, Z. Ma, and S. J. C. N. Musa, "Multiple-level threshold load balancing in distributed SDN controllers," vol. 198, p. 108369, 2021.
[142] S. Farshidi, S. Jansen, M. J. I. Deldar, and S. Technology, "A decision model for programming language ecosystem selection: Seven industry case studies," p. 106640, 2021.
[143] M. P. Fernandez, "Comparing openflow controller paradigms scalability: Reactive and proactive," in *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, 2013, pp. 1009-1016: IEEE.
[144] A. K. Singh and S. J. I. J. o. N. M. Srivastava, "A survey and classification of controller placement problem in SDN," vol. 28, no. 3, p. e2018, 2018.
[145] F. X. Wibowo, M. A. Gregory, K. Ahmed, K. M. J. J. o. N. Gomez, and C. Applications, "Multi-domain software defined networking: research status and challenges," vol. 87, pp. 32-45, 2017.
[146] L. Mamushiane, A. Lysko, and S. Dlamini, "A comparative evaluation of the performance of popular SDN controllers," in *2018 Wireless Days (WD)*, 2018, pp. 54-59: IEEE.
[147] H. Kim, X. Chen, J. Brassil, and J. J. A. S. C. C. R. Rexford, "Experience-driven research on programmable networks," vol. 51, no. 1, pp. 10-17, 2021.
[148] S. Asadollahi, B. Goswami, and M. Sameer, "Ryu controller's scalability experiment on software defined networks," in *2018 IEEE international conference on current trends in Advanced Computing (ICCTAC)*, 2018, pp. 1-5: IEEE.
[149] A. Bondkovskii, J. Keeney, S. van der Meer, and S. Weber, "Qualitative comparison of open-source sdn controllers," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, 2016, pp. 889-894: IEEE.
[150] S. Costanzo, I. Fajjari, N. Aitsaadi, and R. Langar, "SDN-based network slicing in C-RAN," in *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2018, pp. 1-2: IEEE.
[151] R. K. Arbettu, R. Khondoker, K. Bayarou, and F. Weber, "Security analysis of OpenDaylight, Rosemary and Ryu SDN controllers," in *2016 17th International Telecommunications Network Strategy and Planning Symposium (Networks)*, 2016, pp. 37-44: IEEE.
[152] M. Shtern, R. Sandel, M. Litoiu, C. Bachalo, and V. Theodorou, "Towards mitigation of low and slow application DDoS attacks," in *2014 IEEE International Conference on Cloud Engineering*, 2014, pp. 604-609: IEEE.
[153] H. Padekar, Y. Park, H. Hu, and S.-Y. Chang, "Enabling dynamic access control for controller applications in software-defined networks," in *Proceedings of the 21st ACM on Symposium on Access Control Models and Technologies*, 2016, pp. 51-61.
[154] S. Shin *et al.*, "Rosemary: A robust, secure, and high-performance network operating system," in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, 2014, pp. 78-89.
[155] A.-B. Zainab and M. B. Al-Somaidai, "Latency evaluation of an sdn controlled by flowvisor and two heterogeneous controllers," in *International Conference on New Trends in Information and Communications Technology Applications*, 2020, pp. 3-16: Springer.
[156] M. F. Monir and D. Pan, "Application and Assessment of Click Modular Firewall vs POX Firewall in SDN/NFV Framework," in *2020 IEEE REGION 10 CONFERENCE (TENCON)*, 2020, pp. 991-996: IEEE.
[157] H. Hassen, S. Meherzi, and S. Belghith, "Performance Analysis of POX and OpenDayLight Controllers Based on QoS Parameters," in *AINA (2)*, 2021, pp. 745-756: Springer.
[158] R. Jawaharan, P. M. Mohan, T. Das, and M. Gurusamy, "Empirical evaluation of sdn controllers using mininet/Wireshark and comparison with cbench," in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, 2018, pp. 1-2: IEEE.
[159] S.-Y. Wang, H.-W. Chiu, and C.-L. J. P. o. t. I. Chou, "Comparisons of SDN OpenFlow controllers over EstiNet: Ryu vs. NOX," p. 256, 2015.

[160] S. Kaur, J. Singh, and N. S. Ghumman, "Network programmability using POX controller," in *Proc. Int. Conf. Commun., Comput. Syst. (ICCCS)*, 2014, vol. 138, pp. 134-138.
[161] A. Nguyen-Ngoc, S. Raffeck, S. Lange, S. Geissler, T. Zinner, and P. Tran-Gia, "Benchmarking the controller with OFCProbe," in *2018 IEEE Seventh International Conference on Communications and Electronics (ICCE)*, 2018, pp. 367-372: IEEE.
[162] S. Badotra and J. J. I. J. o. A. R. i. C. S. Singh, "Open Daylight as a Controller for Software Defined Networking," vol. 8, no. 5, 2017.
[163] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "SDN controllers: A comparative study," in *2016 18th Mediterranean Electrotechnical Conference (MELECON)*, 2016, pp. 1-6: IEEE.
[164] R. Sharma, S. Agarwal, P. Singh, and G. Ansari, "Future Controller Design and Implementation Trends in Software Defined Networking," in *2018 2nd International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE)*, 2018, pp. 15-22: IEEE.
[165] E. Borcoci, R. Badea, S. G. Obreja, and M. Vochin, "On multi-controller placement optimization in software defined networking-based wans," in *ICN*, 2015, vol. 2015, p. 273.
[166] A. Fernández-Fernández, C. Cervelló-Pastor, and L. J. E. Ochoa-Aday, "Energy efficiency and network performance: A reality check in SDN-based 5G systems," vol. 10, no. 12, p. 2132, 2017.
[167] D. Santos, A. de Sousa, and C. M. Machuca, "Robust SDN controller placement to malicious node attacks," in *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2018, pp. 1-8: IEEE.
[168] B. Zhang, X. Wang, and M. J. C. C. Huang, "Multi-objective optimization controller placement problem in internet-oriented software defined network," vol. 123, pp. 24-35, 2018.
[169] G. Wang, Y. Zhao, J. Huang, Y. J. I. T. o. N. Wu, and S. Management, "An effective approach to controller placement in software defined wide area networks," vol. 15, no. 1, pp. 344-355, 2017.
[170] T. Zhang, P. Giaccone, A. Bianco, and S. J. C. C. De Domenico, "The role of the inter-controller consensus in the placement of distributed SDN controllers," vol. 113, pp. 1-13, 2017.
[171] J. Lu, Z. Zhang, T. Hu, P. Yi, and J. J. I. A. Lan, "A survey of controller placement problem in software-defined networking," vol. 7, pp. 24290-24307, 2019.
[172] A. Kumari and A. S. J. N. Sairam, "Controller placement problem in software-defined networking: A survey," vol. 78, no. 2, pp. 195-223, 2021.
[173] K. He *et al.*, "Latency in software defined networks: Measurements and mitigation techniques," in *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 2015, pp. 435-436.
[174] B. P. R. Killi, S. V. J. I. T. o. N. Rao, and S. Management, "Capacitated next controller placement in software defined networks," vol. 14, no. 3, pp. 514-527, 2017.
[175] P. Vizarreta, C. M. Machuca, and W. Kellerer, "Controller placement strategies for a resilient SDN control plane," in *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, 2016, pp. 253-259: IEEE.
[176] L. F. Müller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, and M. P. Barcellos, "Survivor: An enhanced controller placement strategy for improving SDN survivability," in *2014 IEEE Global Communications Conference*, 2014, pp. 1909-1915: IEEE.
[177] S. Scott-Hayward, "Design and deployment of secure, robust, and resilient SDN Controllers," in *Proceedings of the 2015 1st IEEE conference on network Softwarization (NetSoft)*, 2015, pp. 1-5: IEEE.
[178] S. Lange *et al.*, "Heuristic approaches to the controller placement problem in large scale SDN networks," vol. 12, no. 1, pp. 4-17, 2015.

[179] J. Liao, H. Sun, J. Wang, Q. Qi, K. Li, and T. J. C. N. Li, "Density cluster based approach for controller placement problem in large-scale software defined networking," vol. 112, pp. 24-35, 2017.
[180] G. Wang, Y. Zhao, J. Huang, Q. Duan, and J. Li, "A K-means-based network partition algorithm for controller placement in software defined network," in *2016 IEEE International Conference on Communications (ICC)*, 2016, pp. 1-6: IEEE.
[181] K. S. Sahoo, S. Sahoo, A. Sarkar, B. Sahoo, and R. Dash, "On the placement of controllers for designing a wide area software defined networks," in *TENCON 2017-2017 IEEE Region 10 Conference*, 2017, pp. 3123-3128: IEEE.
[182] A. Ksentini, M. Bagaa, T. Taleb, and I. Balasingham, "On using bargaining game for optimal placement of SDN controllers," in *2016 IEEE International Conference on Communications (ICC)*, 2016, pp. 1-6: IEEE.
[183] G. Yao, J. Bi, Y. Li, and L. J. I. C. L. Guo, "On the capacitated controller placement problem in software defined networks," vol. 18, no. 8, pp. 1339-1342, 2014.
[184] G. Yao, J. Bi, and L. Guo, "On the cascading failures of multi-controllers in software defined networks," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, 2013, pp. 1-2: IEEE.
[185] L. Liao and V. C. Leung, "Genetic algorithms with particle swarm optimization based mutation for distributed controller placement in SDNs," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2017, pp. 1-6: IEEE.
[186] S. Liu, R. Steinert, and D. Kostic, "Flexible distributed control plane deployment," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1-7: IEEE.
[187] G. Ishigaki, R. Gour, A. Yousefpour, N. Shinomiya, and J. P. Jue, "Cluster leader election problem for distributed controller placement in sdn," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, 2017, pp. 1-6: IEEE.
[188] M. Gupta and S. Singh, "Greening of the Internet," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003, pp. 19-26.
[189] A. Ruiz-Rivera, K.-W. Chin, and S. J. I. c. l. Soh, "Greco: An energy aware controller association algorithm for software defined networks," vol. 19, no. 4, pp. 541-544, 2015.
[190] N. T. Hai and D.-S. Kim, "Efficient load balancing for multi-controller in SDN-based mission-critical networks," in *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, 2016, pp. 420-425: IEEE.
[191] Y. Zhou, Y. Wang, J. Yu, J. Ba, and S. Zhang, "Load balancing for multiple controllers in SDN based on switches group," in *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2017, pp. 227-230: IEEE.
[192] K. S. Sahoo *et al.*, "ESMLB: Efficient switch migration-based load balancing for multi controller SDN in IoT," vol. 7, no. 7, pp. 5852-5860, 2019.
[193] M.-L. Chiang, H.-S. Cheng, H.-Y. Liu, and C.-Y. J. C. C. Chiang, "SDN-based server clusters with dynamic load balancing and performance improvement," vol. 24, pp. 537-558, 2021.
[194] A. Filali, S. Cherkaoui, and A. Kobbane, "Prediction-based switch migration scheduling for SDN load balancing," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1-6: IEEE.
[195] M. Tanha, D. Sajjadi, and J. Pan, "Enduring node failures through resilient controller placement for software defined networks," in *2016 IEEE Global Communications Conference (GLOBECOM)*, 2016, pp. 1-7: IEEE.
[196] Y. Jimenez, C. Cervello-Pastor, and A. J. Garcia, "On the controller placement for designing a distributed SDN control layer," in *2014 IFIP Networking Conference*, 2014, pp. 1-9: IEEE.

[197] M. F. Bari *et al.*, "Dynamic controller provisioning in software defined networks," in *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, 2013, pp. 18-25: IEEE.
[198] M. T. I. ul Huque, W. Si, G. Jourjon, V. J. I. T. o. N. Gramoli, and S. Management, "Large-scale dynamic controller placement," vol. 14, no. 1, pp. 63-76, 2017.
[199] J.-M. Sanner, Y. Hadjadj-Aoul, M. Ouzzif, and G. Rubino, "An evolutionary controllers' placement algorithm for reliable SDN networks," in *2017 13th International Conference on Network and Service Management (CNSM)*, 2017, pp. 1-6: IEEE.
[200] A. Sallahi and M. J. I. c. l. St-Hilaire, "Optimal model for the controller placement problem in software defined networks," vol. 19, no. 1, pp. 30-33, 2014.
[201] Y. Hu, W. Wang, X. Gong, X. Que, and S. J. C. C. Cheng, "On reliability-optimized controller placement for software-defined networks," vol. 11, no. 2, pp. 38-54, 2014.
[202] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-optimal resilient controller placement in SDN-based core networks," in *Proceedings of the 2013 25th International Teletraffic Congress (ITC)*, 2013, pp. 1-9: IEEE.
[203] A. Ksentini, M. Bagaa, and T. Taleb, "On using SDN in 5G: The controller placement problem," in *2016 IEEE Global Communications Conference (GLOBECOM)*, 2016, pp. 1-6: IEEE.
[204] M. He, A. Basta, A. Blenk, and W. Kellerer, "Modeling flow setup time for controller placement in SDN: Evaluation for dynamic flows," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1-7: IEEE.
[205] J. Liu, J. Liu, R. J. C. S. Xie, and I. Systems, "Reliability-based controller placement algorithm in software defined networking," vol. 13, no. 2, pp. 547-560, 2016.
[206] S. Moazzeni, M. R. Khayyambashi, N. Movahhedinia, and F. J. C. N. Callegati, "On reliability improvement of software-defined networks," vol. 133, pp. 195-211, 2018.
[207] H. Li, P. Li, S. Guo, and S. Yu, "Byzantine-resilient secure software-defined networks with multiple controllers," in *2014 IEEE International Conference on Communications (ICC)*, 2014, pp. 695-700: IEEE.
[208] Y. Wu, S. Zhou, Y. Wei, and S. Leng, "Deep Reinforcement Learning for Controller Placement in Software Defined Network," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 1254-1259: IEEE.
[209] A. K. Singh, S. Maurya, N. Kumar, and S. J. T. o. E. T. T. Srivastava, "Heuristic approaches for the reliable SDN controller placement problem," vol. 31, no. 2, p. e3761, 2020.
[210] P.-T. Tivig and E. Borcoci, "Critical Analysis of Multi-Controller Placement Problem in Large SDN Networks," in *2020 13th International Conference on Communications (COMM)*, 2020, pp. 489-494: IEEE.
[211] A. K. Singh, S. Maurya, and S. J. F. o. C. S. Srivastava, "Varna-based optimization: a novel method for capacitated controller placement problem in SDN," vol. 14, no. 3, pp. 1-26, 2020.
[212] H. Thiruvengadam, R. Gopalakrishnan, and M. Rajendiran, "Dynamic Controller Deployment in SDN Networks Using ML Approach," in *International Conference on Sustainable Communication Networks and Application*, 2019, pp. 311-318: Springer.
[213] Y. Gao, L. Chen, and B. Li, "Spotlight: Optimizing device placement for training deep neural networks," in *International Conference on Machine Learning*, 2018, pp. 1676-1684: PMLR.
[214] M. Samir, E. Samir, M. Azab, M. R. Rizk, and N. Sadek, "On Managing Controller Placement Problem for Grid-aware Software Defined Networking," in *2019 IEEE Global Conference on Internet of Things (GCIoT)*, 2019, pp. 1-5: IEEE.
[215] V. Huang, G. Chen, Q. Fu, and E. Wen, "Optimizing controller placement for software-defined networks," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019, pp. 224-232: IEEE.

[216] R. Chai, Q. Yuan, L. Zhu, Q. J. E. J. o. W. C. Chen, and Networking, "Control plane delay minimization-based capacitated controller placement algorithm for SDN," vol. 2019, no. 1, pp. 1-17, 2019.
[217] G. Cheng, H. Chen, H. Hu, and J. J. I. J. o. C. S. Lan, "Dynamic switch migration towards a scalable SDN control plane," vol. 29, no. 9, pp. 1482-1499, 2016.
[218] G. Wu, J. Wang, M. S. Obaidat, L. Yao, and K. F. J. I. J. o. C. S. Hsiao, "Dynamic switch migration with noncooperative game towards control plane scalability in SDN," vol. 32, no. 7, p. e3927, 2019.
[219] H. K. Rath, V. Revoori, S. M. Nadaf, and A. Simha, "Optimal controller placement in Software Defined Networks (SDN) using a non-zero-sum game," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, 2014, pp. 1-6: IEEE.
[220] B. P. R. Killi, E. A. Reddy, and S. V. Rao, "Cooperative game theory based network partitioning for controller placement in SDN," in *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*, 2018, pp. 105-112: IEEE.
[221] Y. Jimenez, C. Cervelló-Pastor, and A. J. Garcia, "On the controller placement for designing a distributed SDN control layer," in *2014 IFIP Networking Conference*, 2014, pp. 1-9: IEEE.
[222] I. El Naqa and M. J. Murphy, "What is machine learning? " in *machine learning in radiation oncology*: Springer, 2015, pp. 3-11.
[223] J. G. Carbonell, R. S. Michalski, and T. M. J. M. l. Mitchell, "An overview of machine learning," pp. 3-23, 1983.
[224] M. I. Jordan and T. M. J. S. Mitchell, "Machine learning: Trends, perspectives, and prospects," vol. 349, no. 6245, pp. 255-260, 2015.
[225] J. Pineau *et al.*, "Improving Reproducibility in Machine Learning Research," vol. 22, pp. 1-20, 2021.
[226] J. M. Castillo T, M. Arif, W. J. Niessen, I. G. Schoots, and J. F. J. C. Veenland, "Automated classification of significant prostate cancer on MRI: a systematic review on the performance of machine learning applications," vol. 12, no. 6, p. 1606, 2020.
[227] H. Mannila, "Data mining: machine learning, statistics, and databases," in *Proceedings of 8th International Conference on Scientific and Statistical Data Base Management*, 1996, pp. 2-9: IEEE.
[228] M. A. Alsheikh, S. Lin, D. Niyato, H.-P. J. I. C. S. Tan, and Tutorials, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," vol. 16, no. 4, pp. 1996-2018, 2014.
[229] S. B. Kotsiantis, I. Zaharakis, and P. J. E. a. i. a. i. c. e. Pintelas, "Supervised machine learning: A review of classification techniques," vol. 160, no. 1, pp. 3-24, 2007.
[230] T. O. J. N. a. i. m. l. Ayodele, "Types of machine learning algorithms," vol. 3, pp. 19-48, 2010.
[231] R. Gupta, D. Srivastava, M. Sahu, S. Tiwari, R. K. Ambasta, and P. J. M. D. Kumar, "Artificial intelligence to deep learning: Machine intelligence approach for drug discovery," pp. 1-46, 2021.
[232] S. Zroug, I. Remadna, L. Kahloul, S. Benharzallah, and S. L. Terrissa, "Leveraging the Power of Machine Learning for Performance Evaluation Prediction in Wireless Sensor Networks," in *2021 International Conference on Information Technology (ICIT)*, 2021, pp. 864-869: IEEE.
[233] P. Cunningham, M. Cord, and S. J. Delany, "Supervised learning," in *Machine learning techniques for multimedia*: Springer, 2008, pp. 21-49.
[234] S. Uddin, A. Khan, M. E. Hossain, M. A. J. B. m. i. Moni, and d. making, "Comparing different supervised machine learning algorithms for disease prediction," vol. 19, no. 1, pp. 1-16, 2019.

[235] F. Osisanwo *et al.*, "Supervised machine learning algorithms: classification and comparison," vol. 48, no. 3, pp. 128-138, 2017.
[236] B. J. I. J. o. S. Mahesh and R. . "Machine Learning Algorithms-A Review," vol. 9, pp. 381-386, 2020.
[237] D. K. Renuka, T. Hamsapriya, M. R. Chakkaravarthi, and P. L. Surya, "Spam classification based on supervised learning using machine learning techniques," in *2011 International Conference on Process Automation, Control and Computing*, 2011, pp. 1-7: IEEE.
[238] D. W. Armitage and H. K. J. E. I. Ober, "A comparison of supervised learning techniques in the classification of bat echolocation calls," vol. 5, no. 6, pp. 465-473, 2010.
[239] V. Sharma, S. Rai, A. J. I. J. o. A. r. i. c. s. Dev, and s. engineering, "A comprehensive study of artificial neural networks," vol. 2, no. 10, 2012.
[240] W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders, and K.-R. J. P. o. t. I. Müller, "Explaining deep neural networks and beyond: A review of methods and applications," vol. 109, no. 3, pp. 247-278, 2021.
[241] T. Das, V. Sridharan, M. J. I. c. s. Gurusamy, and tutorials, "A survey on controller placement in SDN," vol. 22, no. 1, pp. 472-503, 2019.
[242] H. Li, R. E. De Grande, and A. Boukerche, "An efficient CPP solution for resilience-oriented SDN controller deployment," in *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2017, pp. 540-549: IEEE.
[243] B.-W. Yuan *et al.*, "A novel density-based adaptive k nearest neighbour method for dealing with overlapping problem in imbalanced datasets," vol. 33, no. 9, pp. 4457-4481, 2021.
[244] M. Latella, F. Sola, and C. J. R. S. Camporeale, "A Density-Based Algorithm for the Detection of Individual Trees from LiDAR Data," vol. 13, no. 2, p. 322, 2021.
[245] K. K. Sharma, A. Seal, E. Herrera-Viedma, and O. J. S. Krejcar, "An Enhanced Spectral Clustering Algorithm with S-Distance," vol. 13, no. 4, p. 596, 2021.
[246] T. N. Subedi, K. K. Nguyen, and M. J. I. J. o. C. S. Cheriet, "SDN-based fault-tolerant on-demand and in-advance bandwidth reservation in data centre interconnects," vol. 31, no. 4, p. e3479, 2018.
[247] N. Pinaeva and V. Antonenko, "WARP: WAN Connection Bandwidth Monitoring and Prediction," in *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2020, pp. 228-234: IEEE.
[248] P. L. Gallegos-Segovia, J. F. Bravo-Torres, P. E. Vintimilla-Tapia, J. O. Ordonez-Ordonez, R. E. Mora-Huiracocha, and V. M. Larios-Rosillo, "Evaluation of an sdn-wan controller applied to services hosted in the cloud," in *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*, 2017, pp. 1-6: IEEE.
[249] W. Chen, C. Chen, X. Jiang, and L. J. I. A. Liu, "Multi-controller placement towards SDN based on Louvain heuristic algorithm," vol. 6, pp. 49486-49497, 2018.
[250] A. Alowa and T. Fevens, "Combined degree-based with independent dominating set approach for controller placement problem in software defined networks," in *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2019, pp. 269-276: IEEE.
[251] T. J. E. S. Singh, "A novel data clustering approach based on whale optimization algorithm," vol. 38, no. 3, p. e12657, 2021.
[252] C. N. Sminesh, E. G. M. Kanaga, and A. J. I. C. Roy, "Optimal multi-controller placement strategy in SD-WAN using modified density peak clustering," vol. 13, no. 20, pp. 3509-3518, 2019.
[253] A. Jalili, M. Keshtgari, R. Akbari, and R. J. C. C. Javidan, "Multi criteria analysis of controller placement problem in software defined networks," vol. 133, pp. 115-128, 2019.
[254] P. Xiao *et al.*, "AK self-adaptive SDN controller placement for wide area networks," vol. 17, no. 7, pp. 620-633, 2016.
[255] S. Chakraborty, D. Paul, S. J. S. Das, and P. Letters, "Hierarchical clustering with optimal transport," vol. 163, p. 108781, 2020.

[256] I. G. N. M. Jaya and H. J. G. A. Folmer, "Identifying spatiotemporal clusters by means of agglomerative hierarchical clustering and Bayesian regression analysis with spatiotemporally varying coefficients: methodology and application to dengue disease in Bandung, Indonesia," 2020.
[257] R. M. Shukla and S. J. I. o. T. Sengupta, "Scalable and robust outlier detector using hierarchical clustering and long short-term memory (lstm) neural network for the internet of things," vol. 9, p. 100167, 2020.
[258] K. Alhazmi, A. Moubayed, and A. Shami, "Distributed sdn controller placement using betweenness centrality & hierarchical clustering," in *Proceedings of the 8th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications*, 2018, pp. 15-20.
[259] P. Li and O. Milenkovic, "Inhomogoenous hypergraph clustering with applications," in *Advances in Neural Information Processing Systems*, 2017, pp. 2305-2315.
[260] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich, "Local higher-order graph clustering," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 555-564.
[261] G. Wang, Z. Zhao, J. Peng, R. Li, and H. Zhang, "An approximate algorithm of controller configuration in multi-domain SDN architecture," in *9th International Conference on Communications and Networking in China*, 2014, pp. 601-605: IEEE.
[262] L. Hu, H. Liu, J. Zhang, and A. J. E. S. w. A. Liu, "KR-DBSCAN: A density-based clustering algorithm based on reverse nearest neighbour and influence space," p. 115763, 2021.
[263] C. Atilgan, B. T. Tezel, and E. J. I. S. Nasiboglu, "Efficient implementation and parallelization of fuzzy density based clustering," vol. 575, pp. 454-467, 2021.
[264] S. Mai *et al.*, "Incremental Density-based Clustering on Multicore Processors," 2020.
[265] Z. Cai, J. Wang, and K. J. I. A. He, "Adaptive density-based spatial clustering for massive data analysis," vol. 8, pp. 23346-23358, 2020.
[266] J. Xu and K. Lange, "Power k-means clustering," in *International Conference on Machine Learning*, 2019, pp. 6921-6931: PMLR.
[267] H. Xie *et al.*, "Improving K-means clustering with enhanced firefly algorithms," vol. 84, p. 105763, 2019.
[268] W. Huang, Y. Peng, Y. Ge, and W. J. P. C. S. Kong, "A new Kmeans clustering model and its generalization achieved by joint spectral embedding and rotation," vol. 7, p. e450, 2021.
[269] S. Park, H.-S. Jo, C. Mun, and J.-G. J. S. Yook, "RRH Clustering Using Affinity Propagation Algorithm with Adaptive Thresholding and Greedy Merging in Cloud Radio Access Network," vol. 21, no. 2, p. 480, 2021.
[270] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Kdd*, 1996, vol. 96, no. 34, pp. 226-231.
[271] A. Starczewski, P. Goetzen, M. J. J. J. o. A. I. Er, and S. C. Research, "A new method for automatic determining of the DBSCAN parameters," vol. 10, 2020.
[272] N. Ohadi, A. Kamandi, M. Shabankhah, S. M. Fatemi, S. M. Hosseini, and A. Mahmoudi, "SW-DBSCAN: A grid-based DBSCAN algorithm for large datasets," in *2020 6th International Conference on Web Research (ICWR)*, 2020, pp. 139-145: IEEE.
[273] Y. Chen, L. Zhou, N. Bouguila, C. Wang, Y. Chen, and J. J. P. R. Du, "BLOCK-DBSCAN: Fast clustering for large scale data," vol. 109, p. 107624, 2021.
[274] Q. Zhu, X. Tang, and A. J. E. S. w. A. Elahi, "Application of the novel harmony search optimization algorithm for DBSCAN clustering," vol. 178, p. 115054, 2021.
[275] C. Bhadane and K. Shah, "Clustering algorithms for spatial data mining," in *Proceedings of the 2020 3rd International Conference on Geoinformatics and Data Analysis*, 2020, pp. 5-9.
[276] J. Yang, S. Rahardja, and P. J. P. R. Fränti, "Mean-shift outlier detection and filtering," vol. 115, p. 107874, 2021.

[277] V. Ramakrishnan *et al.*, "Bootstrap aggregative mean shift clustering for big data anti-pattern detection analytics in 5G/6G communication networks," vol. 95, p. 107380, 2021.
[278] J. J. Mwemezi and Y. J. N. Y. S. J. Huang, "Optimal facility location on spherical surfaces: algorithm and application," vol. 4, no. 7, pp. 21-28, 2011.
[279] C. Zhang, X. Wang, M. Chen, and T. J. R. S. M. Wang, "A comparison of hypothesis tests for homogeneity in meta-analysis with focus on rare binary events," 2021.
[280] K. S. Namjoshi, "Symmetry and completeness in the analysis of parameterized systems," in *International Workshop on Verification, Model Checking, and Abstract Interpretation*, 2007, pp. 299-313: Springer.
[281] I. Y. Torshin, K. J. P. R. Rudakov, and I. Analysis, "Combinatorial analysis of the solvability properties of the problems of recognition and completeness of algorithmic models. Part 1: factorization approach," vol. 27, no. 1, pp. 16-28, 2017.
[282] H. Zhou, P. Wang, and H. J. J. O. X. a. U. o. T. Li, "Research on adaptive parameters determination in DBSCAN algorithm," vol. 28, no. 3, pp. 289-292, 2012.
[283] M. M. A. Patwary, D. Palsetia, A. Agrawal, W.-k. Liao, F. Manne, and A. Choudhary, "A new scalable parallel DBSCAN algorithm using the disjoint-set data structure," in *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2012, pp. 1-11: IEEE.
[284] A. Schmidt, J.-W. Billig, M. A. Schlenz, and B. J. I. J. o. C. D. Wöstmann, "Do different methods of digital data analysis lead to different results? " vol. 24, no. 2, pp. 157-164, 2021.
[285] J. N. Lester, Y. Cho, and C. R. J. H. R. D. R. Lochmiller, "Learning to do qualitative data analysis: A starting point," vol. 19, no. 1, pp. 94-106, 2020.
[286] J. A. J. Q. P. Maxwell, "Why qualitative methods are necessary for generalization," vol. 8, no. 1, p. 111, 2021.
[287] S. Krishnakumar, K. J. J. o. A. I. Manivannan, and H. Computing, "Effective segmentation and classification of brain tumor using rough K means algorithm and multi kernel SVM in MR images," vol. 12, no. 6, pp. 6751-6760, 2021.
[288] D. Geng, Z. Qin, J. Wang, Z. Gao, N. J. B. S. P. Zhao, and Control, "Personalized recognition of wake/sleep state based on the combined shapelets and K-means algorithm," vol. 71, p. 103132, 2022.
[289] W. L. Al-Yaseen, A. Jehad, Q. A. Abed, A. K. J. I. J. o. I. E. Idrees, and Systems, "The Use of Modified K-Means Algorithm to Enhance the Performance of Support Vector Machine in Classifying Breast Cancer," vol. 14, no. 2, pp. 190-200, 2021.
[290] E. Cuevas, J. Gálvez, O. Avalos, Á. J. C. Chavarin, and E. Engineering, "A mean shift segmentation scheme using several pixel characteristics," vol. 90, p. 107022, 2021.
[291] J. Chen, J. Yang, J. Huang, and Y. J. J. o. I. T. Liu, "Robust Truth Discovery Scheme Based on Mean Shift Clustering Algorithm," vol. 22, no. 4, pp. 835-842, 2021.
[292] S. A. Koohpayegani, A. Tejankar, and H. J. a. p. a. Pirsiavash, "Mean Shift for Self-Supervised Learning," 2021.
[293] D. Kim, Y.-H. Kim, K.-H. Kim, J.-B. Kim, G.-S. You, and J.-M. J. T. J. o. S. Gil, "Logically isolated group network for virtual convergence environment over SD-WAN," vol. 74, no. 12, pp. 6742-6752, 2018.
[294] Z. Wang, S. Zhao, Z. Fan, and X. J. W. P. C. Wan, "Performance Modeling and Analysis of Control Plane for SDN Based on Queuing Theory," vol. 97, no. 1, pp. 591-601, 2017.
[295] T. Yuan, X. Huang, M. Ma, and J. Yuan, "Balance-based SDN controller placement and assignment with minimum weight matching," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1-6: IEEE.
[296] A. Shirmarz and A. J. T. J. o. S. Ghaffari, "Performance issues and solutions in SDN-based data centre: a survey," vol. 76, no. 10, pp. 7545-7593, 2020.
[297] Z.-I. Berlin. (2006). *Survivable fixed telecommunication Network Design*. Available: http://sndlib.zib.de/home.action

[298] G. Schütz, "A k-cover model for reliability-aware controller placement in software-defined networks," in *International conference on computational science*, 2019, pp. 604-613: Springer.
[299] X. Dai, J. Liu, X. J. E. Zhang, and Buildings, "A review of studies applying machine learning models to predict occupancy and window-opening behaviours in smart buildings," p. 110159, 2020.
[300] s.-l. developers. (2007-2021). *Generated datasets*. Available: https://scikit-learn.org/stable/datasets/sample_generators.html#generators-for-classification-and-clustering
[301] X. Zhang, G. Dong, and K. Ramamohanarao, "Building behaviour knowledge space to make classification decision," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2001, pp. 488-494: Springer.
[302] I. Letteri, A. Di Cecco, and G. J. a. p. a. Della Penna, "Dataset Optimization Strategies for MalwareTraffic Detection," 2020.
[303] P. Prettenhofer and G. Louppe, "Gradient boosted regression trees in scikit-learn," 2014.
[304] D. D. Lewis, Y. Yang, T. Russell-Rose, and F. J. J. o. m. l. r. Li, "Rcv1: A new benchmark collection for text categorization research," vol. 5, no. Apr, pp. 361-397, 2004.
[305] V. Berk and N. Fox, "Process query systems for network security monitoring," in *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense IV*, 2005, vol. 5778, pp. 520-530: International Society for Optics and Photonics.
[306] H. Wang, Z. Cao, B. J. J. o. I. Hong, and F. Systems, "A network intrusion detection system based on convolutional neural network," vol. 38, no. 6, pp. 7623-7637, 2020.
[307] P. Borah and D. J. A. I. Gupta, "Robust twin bounded support vector machines for outliers and imbalanced data," pp. 1-30, 2021.
[308] S. S. S. Abd Mutalib, S. Z. Satari, W. N. S. W. J. J. o. S. M. Yusoff, and Analytics, "Comparison of Robust Estimators' Performance for Detecting Outliers in Multivariate Data," vol. 3, no. 2, 2021.
[309] S. J. E. S. w. A. Askari, "Fuzzy C-Means clustering algorithm for data with unequal cluster sizes and contaminated with noise and outliers: Review and development," vol. 165, p. 113856, 2021.
[310] A. D'Ambrosio, S. Amodio, C. Iorio, G. Pandolfo, and R. J. J. o. C. Siciliano, "Adjusted Concordance Index: an Extensionl of the Adjusted Rand Index to Fuzzy Partitions," vol. 38, no. 1, pp. 112-128, 2021.
[311] V. Robert, Y. Vasseur, and V. J. J. o. C. Brault, "Comparing high-dimensional partitions with the Co-clustering Adjusted Rand Index," vol. 38, no. 1, pp. 158-186, 2021.
[312] N. N. Mohammed and C. J. Mohammed, "Enhanced Determination of Gene Groups Based on Optimal Kernel PCA with Hierarchical Clustering Algorithm," in *2021 55th Annual Conference on Information Sciences and Systems (CISS)*, 2021, pp. 1-5: IEEE.
[313] G. Newcomb and K. J. E. Sayood, "Use of Average Mutual Information and Derived Measures to Find Coding Regions," vol. 23, no. 10, p. 1324, 2021.
[314] N. Agrawal and K. Verma, "Hyperspectral Band Selection using Mutual Information," in *2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, 2021, pp. 1-4: IEEE.
[315] G. Rossetti, "CDlib Documentation," 2021.
[316] S. Krishnagopal and G. J. a. p. a. Bianconi, "Spectral Detection of Simplicial Communities via Hodge Laplacians," 2021.
[317] X. L. Xie, G. J. I. T. o. p. a. Beni, and m. intelligence, "A validity measure for fuzzy clustering," vol. 13, no. 8, pp. 841-847, 1991.
[318] R. A. Jarvis and E. A. J. I. T. o. c. Patrick, "Clustering using a similarity measure based on shared near neighbors," vol. 100, no. 11, pp. 1025-1034, 1973.

[319] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure," in *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007, pp. 410-420.
[320] D. Chasaki and C. Mansour, "SDN Security through System Call Learning," in *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2021, pp. 1-6: IEEE.
[321] M. Revathi, V. Ramalingam, and B. J. W. P. C. Amutha, "A Machine Learning Based Detection and Mitigation of the DDOS Attack by Using SDN Controller Framework," pp. 1-25, 2021.
[322] A. K. Rangisetti, R. Dwivedi, and P. J. C. C. Singh, "Denial of ARP spoofing in SDN and NFV enabled cloud-fog-edge platforms," pp. 1-26, 2021.
[323] T. Hu *et al.*, "SEAPP: A secure application management framework based on REST API access control in SDN-enabled cloud environment," vol. 147, pp. 108-123, 2021.
[324] J. Xing, C. Wu, H. Zhou, Q. Cheng, D. Yu, and M. J. S. C. I. S. Macas, "Efficient middlebox scaling for virtualized intrusion prevention systems in software-defined networks," vol. 65, no. 8, pp. 1-3, 2022.
[325] S. M. Alturfi, D. K. Muhsen, M. A. Mohammed, I. T. Aziz, and M. Aljshamee, "A Combination Techniques of Intrusion Prevention and Detection for Cloud Computing," in *Journal of Physics: Conference Series*, 2021, vol. 1804, no. 1, p. 012121: IOP Publishing.
[326] P. Narwal, D. Kumar, S. N. Singh, and P. J. J. O. I. T. R. Tewari, "Stochastic Intrusion Detection Game-Based Arrangement Using Controlled Markov Chain for Prevention of DoS and DDoS Attacks in Cloud," vol. 14, no. 4, pp. 45-57, 2021.
[327] A. Guezzaz, A. Asimi, Y. Asimi, M. Azrour, and S. Benkirane, "A Distributed Intrusion Detection Approach Based on Machine Learning Techniques for a Cloud Security," in *Intelligent Systems in Big Data, Semantic Web and Machine Learning*: Springer, 2021, pp. 85-94.
[328] Y. F. Mohammed, "Network-Based Detection and Prevention System against DNS-Based Attacks," University of Arkansas, 2021.
[329] W. Seo and W. J. I. A. Pak, "Real-Time Network Intrusion Prevention System Based on Hybrid Machine Learning," vol. 9, pp. 46386-46397, 2021.
[330] D. Zheng, L. Zhou, and Z. J. I. C. J. o. A. S. Song, "Kernel Generalization of Multi-Rate Probabilistic Principal Component Analysis for Fault Detection in Nonlinear Process," vol. 8, no. 8, pp. 1465-1476, 2021.
[331] I. Ahmad, M. Hussain, A. Alghamdi, A. J. N. c. Alelaiwi, and applications, "Enhancing SVM performance in intrusion detection using optimal feature subset selection based on genetic principal components," vol. 24, no. 7, pp. 1671-1682, 2014.
[332] L. Rouani, M. F. Harkat, A. Kouadri, and S. J. R. E. Mekhilef, "Shading fault detection in a grid-connected PV system using vertices principal component analysis," vol. 164, pp. 1527-1539, 2021.
[333] A. H. d. A. Melani, M. A. d. C. Michalski, R. F. da Silva, G. F. M. J. R. E. de Souza, and S. Safety, "A framework to automate fault detection and diagnosis based on moving window principal component analysis and Bayesian network," vol. 215, no. C, 2021.
[334] K. Sarita *et al.*, "Principal component analysis technique for early fault detection," no. Preprint, pp. 1-12, 2021.
[335] G. González-Granadillo, S. González-Zarzosa, and R. J. S. Diaz, "Security Information and Event Management (SIEM): Analysis, Trends, and Usage in Critical Infrastructures," vol. 21, no. 14, p. 4759, 2021.
[336] P. Radoglou-Grammatikis *et al.*, "SPEAR SIEM: A Security Information and Event Management system for the Smart Grid," vol. 193, p. 108008, 2021.
[337] C. Dubuc, "A Real-time Log Correlation System for Security Information and Event Management," ed, 2021.

[338] V. T. Nguyen, M. Q. Nguyen, and T. T. Truong, "Development of a Security information event and management system Using ELK Stack and Nagios," 2021.
[339] J. Surana, M. Gupta, and P. Sharma, "Event Log Management and Monitoring."
[340] S. I. Pérez, S. Moral-Rubio, R. J. C. Criado, Solitons, and Fractals, "A new approach to combine multiplex networks and time series attributes: Building intrusion detection systems (IDS) in cybersecurity," vol. 150, p. 111143, 2021.
[341] A. Khraisat and A. J. C. Alazab, "A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges," vol. 4, no. 1, pp. 1-27, 2021.
[342] M. Subbulakshmi, S. Sujitha, A. Vetrivel, J. N. Gandhi, and K. V. J. R. G.-G. I. E. T. Guru, "Privacy Preserving Machine Learning in Various Attacks on Security Threat Models," vol. 11, no. 2, pp. 418-428, 2021.
[343] R. Lohiya and A. Thakkar, "Intrusion detection using deep neural network with antirectifier layer," in *Applied Soft Computing and Communication Networks*: Springer, 2021, pp. 89-105.
[344] A. Alhowaide, I. Alsmadi, and J. J. I. o. T. Tang, "Ensemble detection model for IoT IDS," p. 100435, 2021.
[345] S.-W. Lee *et al.*, "Towards secure intrusion detection systems using deep learning techniques: Comprehensive analysis and review," p. 103111, 2021.
[346] S. X. Wu and W. J. A. s. c. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," vol. 10, no. 1, pp. 1-35, 2010.
[347] D.-S. Deng, W. Long, Y.-Y. Li, and X.-Q. J. P. Shi, "Building Robust Closed-Loop Supply Networks against Malicious Attacks," vol. 9, no. 1, p. 39, 2021.
[348] K. Shah and D. J. F. o. C. S. Jinwala, "Privacy preserving secure expansive aggregation with malicious node identification in linear wireless sensor networks," vol. 15, no. 6, pp. 1-9, 2021.
[349] K. L. Narayanan, R. S. Krishnan, E. G. Julie, Y. H. Robinson, and V. J. W. P. C. Shanmuganathan, "Machine Learning Based Detection and a Novel EC-BRTT Algorithm Based Prevention of DoS Attacks in Wireless Sensor Networks," pp. 1-25, 2021.
[350] X. Zhang, Y. Liu, Y. J. J. o. S. S. Zhang, and Complexity, "A Secure Clock Synchronization Scheme for Wireless Sensor Networks Against Malicious Attacks," pp. 1-14, 2021.
[351] M. Kumar, P. Mukherjee, K. Verma, S. Verma, D. B. J. I. T. o. N. S. Rawat, and Engineering, "Improved Deep Convolutional Neural Network based Malicious Node Detection and Energy-Efficient Data Transmission in Wireless Sensor Networks," 2021.
[352] D. Liu, D. J. I. T. o. N. N. Ye, and L. Systems, "Edge-Based Decentralized Adaptive Pinning Synchronization of Complex Networks Under Link Attacks," 2021.
[353] R. Valter, "Detecting and predicting malicious attacks on Internet of Things devices using deep neural networks," 2021.
[354] S.-Y. Wu and E. J. E. S. w. A. Yen, "Data mining-based intrusion detectors," vol. 36, no. 3, pp. 5605-5612, 2009.
[355] C. Janiesch, P. Zschech, and K. J. E. M. Heinrich, "Machine learning and deep learning," vol. 31, no. 3, pp. 685-695, 2021.
[356] I. H. J. S. C. S. Sarker, "Machine learning: Algorithms, real-world applications and research directions," vol. 2, no. 3, pp. 1-21, 2021.
[357] Y. Zhao, J. Chen, D. Wu, J. Teng, and S. Yu, "Multi-task network anomaly detection using federated learning," in *Proceedings of the Tenth International Symposium on Information and Communication Technology*, 2019, pp. 273-279.
[358] A. Khraisat, I. Gondal, P. Vamplew, and J. J. C. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," vol. 2, no. 1, pp. 1-22, 2019.
[359] B. Li, S. Zhao, R. Zhang, Q. Shi, and K. J. I. C. Yang, "Anomaly detection for cellular networks using big data analytics," vol. 13, no. 20, pp. 3351-3359, 2019.

[360] A. d. R. L. Ribeiro, R. Y. C. Santos, and A. C. A. Nascimento, "Anomaly Detection Technique for Intrusion Detection in SDN Environment using Continuous Data Stream Machine Learning Algorithms," in *2021 IEEE International Systems Conference (SysCon)*, 2021, pp. 1-7: IEEE.
[361] A. M. El-Shamy, N. A. El-Fishawy, G. Attiya, and M. A. J. E. I. J. Mohamed, "Anomaly detection and bottleneck identification of the distributed application in cloud data center using software–defined networking," 2021.
[362] B. Dayıoglu and A. Ozgit, "Use of passive network mapping to enhance signature quality of misuse network intrusion detection systems," in *16th Int. Symp. on Computer and Information Sciences*, 2001: Citeseer.
[363] H. Shapoorifard and P. J. I. J. C. A. Shamsinejad, "Intrusion detection using a novel hybrid method incorporating an improved KNN," vol. 173, no. 1, pp. 5-9, 2017.
[364] F. Erlacher and F. Dressler, "FIXIDS: A high-speed signature-based flow intrusion detection system," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1-8: IEEE.
[365] L. J. J. o. C. N. Wang, "Big Data in intrusion detection systems and intrusion prevention systems," vol. 4, no. 1, pp. 48-55, 2017.
[366] H. Mohapatra, S. Rath, S. Panda, and R. J. I. j. Kumar, "Handling of man-in-the-middle attack in wsn through intrusion detection system," vol. 8, no. 5, pp. 1503-1510, 2020.
[367] N. A. Azeez *et al.*, "Intrusion detection and prevention systems: an updated review," pp. 685-696, 2020.
[368] M. J. Nazar, A. Alhudhaif, K. N. Qureshi, S. Iqbal, G. J. I. J. o. S. A. E. Jeon, and Management, "Signature and flow statistics based anomaly detection system in software-defined networking for 6G internet of things network," pp. 1-11, 2021.
[369] R. M. A. Ujjan, Z. Pervez, K. Dahal, W. A. Khan, A. M. Khattak, and B. J. S. Hayat, "Entropy Based Features Distribution for Anti-DDoS Model in SDN," vol. 13, no. 3, p. 1522, 2021.
[370] Y. Yang, K. McLaughlin, S. Sezer, Y. Yuan, and W. Huang, "Stateful intrusion detection for IEC 60870-5-104 SCADA security," in *2014 IEEE PES General Meeting| Conference & Exposition*, 2014, pp. 1-5: IEEE.
[371] R. Mustafiz, A. S. M. D. Hossain, N. Islam, M. M. J. I. J. o. C. N. Rahman, and I. Security, "Analysis of QoS in Software Defined Wireless Network with Spanning Tree Protocol," vol. 9, no. 6, p. 61, 2017.
[372] E. Ariafar and R. Kiani, "Intrusion detection system using an optimized framework based on data mining techniques," in *2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, 2017, pp. 0785-0791: IEEE.
[373] V. Chandola, A. Banerjee, and V. J. A. c. s. Kumar, "Anomaly detection: A survey," vol. 41, no. 3, pp. 1-58, 2009.
[374] G. Zhu, Y. Zheng, D. Doermann, S. J. I. T. o. P. A. Jaeger, and M. Intelligence, "Signature detection and matching for document image retrieval," vol. 31, no. 11, pp. 2015-2031, 2008.
[375] M. Vallentin, R. Sommer, J. Lee, C. Leres, V. Paxson, and B. Tierney, "The NIDS Cluster: Scalable, stateful network intrusion detection on commodity hardware," in *International Workshop on Recent Advances in Intrusion Detection*, 2007, pp. 107-126: Springer.
[376] J. Tang, K.-W. J. S. M. Wang, and Structures, "Active-passive hybrid piezoelectric networks for vibration control: comparisons and improvement," vol. 10, no. 4, p. 794, 2001.
[377] S. Anwar *et al.*, "From intrusion detection to an intrusion response system: fundamentals, requirements, and future directions," vol. 10, no. 2, p. 39, 2017.
[378] S. A. Onashoga, A. D. Akinde, A. S. J. I. i. I. S. Sodiya, and I. Technology, "A Strategic Review of Existing Mobile Agent-Based Intrusion Detection Systems," vol. 6, 2009.
[379] S. N. J. J. o. C. S. Kumar and Applications, "DecenCrypto cloud: decentralized cryptography technique for secure communication over the clouds," vol. 3, no. 3, pp. 73-78, 2015.

[380] X. Wang, D. S. Reeves, and S. F. J. J. o. I. W. Wu, "Tracing based active intrusion response," vol. 1, no. 1, pp. 50-61, 2001.
[381] A. Kumar and A. K. J. I. J. o. I. T. Jain, "Mutual authentication protocol for low cost passive tag in RFID system," vol. 13, no. 3, pp. 1209-1215, 2021.
[382] B. Nowell, T. J. J. o. P. A. R. Steelman, and Theory, "Communication under fire: The role of embeddedness in the emergence and efficacy of disaster response communication networks," vol. 25, no. 3, pp. 929-952, 2015.
[383] J. Goodgion and B. Mullins, "Active Network Response Using Host-Based IDS and Software Defined Networking," in *ICMLG 2017 5th International Conference on Management Leadership and Governance*, 2017, p. 469: Academic Conferences and Publishing Limited.
[384] U. INFORMATIONSYSTEMSSEC, "Monitoring IDS."
[385] J. J. G. J. o. C. S. Nehinbe and Technology, "A review of technical issues on IDS and alerts," 2018.
[386] L. Mechtri, F. D. Tolba, S. J. P.-t.-P. N. Ghanemi, and Applications, "An optimized intrusion response system for MANET," vol. 11, no. 3, pp. 602-618, 2018.
[387] M. J. A. J. o. I. S. Bijone, "A survey on secure network: intrusion detection & prevention approaches," vol. 4, no. 3, pp. 69-88, 2016.
[388] M. Yasin and A. A. Awan, "A study of host-based IDS using system calls," in *2004 International Networking and Communication Conference*, 2004, pp. 36-41: IEEE.
[389] I. Idrissi, M. Mostafa Azizi, O. J. I. J. o. C. Moussaoui, and D. System, "A lightweight optimized deep learning-based host-intrusion detection system deployed on the edge for IoT," 2021.
[390] D. Kapil, N. Mehra, A. Gupta, S. Maurya, and A. Sharma, "Network Security: Threat Model, Attacks, and IDS Using Machine Learning," in *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, 2021, pp. 203-208: IEEE.
[391] M. R. Ayyagari, N. Kesswani, M. Kumar, and K. J. W. N. Kumar, "Intrusion detection techniques in network environment: a systematic review," pp. 1-17, 2021.
[392] R. Gassais, N. Ezzati-Jivan, J. M. Fernandez, D. Aloise, and M. R. J. J. o. C. C. Dagenais, "Multi-level host-based intrusion detection system for Internet of things," vol. 9, no. 1, pp. 1-16, 2020.
[393] C. M. Schneider, A. A. Moreira, J. S. Andrade, S. Havlin, and H. J. J. P. o. t. N. A. o. S. Herrmann, "Mitigation of malicious attacks on networks," vol. 108, no. 10, pp. 3838-3841, 2011.
[394] G. Thamilarasu and Z. Ma, "Autonomous mobile agent based intrusion detection framework in wireless body area networks," in *2015 IEEE 16th International Symposium on a world of wireless, mobile and multimedia networks (WoWMoM)*, 2015, pp. 1-3: IEEE.
[395] C. Kreibich, M. Handley, and V. Paxson, "Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics," in *Proc. USENIX Security Symposium*, 2001, vol. 2001.
[396] S. Otoum, B. Kantarci, and H. T. J. I. N. L. Mouftah, "On the feasibility of deep learning in sensor network intrusion detection," vol. 1, no. 2, pp. 68-71, 2019.
[397] A. Sahu, Z. Mao, K. Davis, and A. E. Goulart, "Data processing and model selection for machine learning-based network intrusion detection," in *2020 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, 2020, pp. 1-6: IEEE.
[398] R. K. S. Gautam and E. A. Doegar, "An ensemble approach for intrusion detection system using machine learning algorithms," in *2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 2018, pp. 14-15: IEEE.
[399] V. Paxson, S. Campbell, and J. Lee, "Bro intrusion detection system," Lawrence Berkeley National Laboratory2006.

[400] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. J. T. o. E. T. T. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," vol. 32, no. 1, p. e4150, 2021.
[401] M. A. Rahman *et al.*, "Scalable machine learning-based intrusion detection system for IoT-enabled smart cities," vol. 61, p. 102324, 2020.
[402] A. N. Cahyo, A. K. Sari, and M. Riasetiawan, "Comparison of Hybrid Intrusion Detection System," in *2020 12th International Conference on Information Technology and Electrical Engineering (ICITEE)*, 2020, pp. 92-97: IEEE.
[403] H. Mennour and S. Mostefai, "A hybrid Deep Learning Strategy for an Anomaly Based N-IDS," in *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*, 2020, pp. 1-6: IEEE.
[404] A. Golrang, A. M. Golrang, S. Yildirim Yayilgan, and O. J. E. Elezaj, "A novel hybrid IDS based on modified NSGAII-ANN and random forest," vol. 9, no. 4, p. 577, 2020.
[405] Y. Yi, J. Wu, and W. J. E. S. w. A. Xu, "Incremental SVM based on reserved set for network intrusion detection," vol. 38, no. 6, pp. 7698-7707, 2011.
[406] K. J. U. h. w. p.-i. o. d. m. p. p. p. Zaraska, "Prelude IDS: current state and development perspectives," 2003.
[407] P. Sun *et al.*, "DL-IDS: extracting features using CNN-LSTM hybrid network for intrusion detection system," vol. 2020, 2020.
[408] V. Prabhakaran and A. Kulandasamy, "Detailed Review on IDS Techniques in Cloud Computing," in *2021 6th International Conference on Communication and Electronics Systems (ICCES)*, 2021, pp. 769-774: IEEE.
[410] M. J. T. J. o. C. Ponnusamy and M. Education, "Detection of Selfish Nodes Through Reputation Model In Mobile Adhoc Network-MANET," vol. 12, no. 9, pp. 2404–2410-2404–2410, 2021.
[411] P. Malarvizhi Kumar, H. J. A. J. f. S. Choong Seon, and Engineering, "Internet of things-based digital video intrusion for intelligent monitoring approach," pp. 1-11, 2021.
[412] S. V. Simpson and G. J. W. N. Nagarajan, "An edge based trustworthy environment establishment for Internet of things: an approach for smart cities," pp. 1-17, 2021.
[413] N. Santosh, R. Saranyan, and V. Vetriselvi, "Cluster based co-operative game theory approach for intrusion detection in mobile ad-hoc grid," in *2008 16th International Conference on Advanced Computing and Communications*, 2008, pp. 273-278: IEEE.
[414] I.-H. Huang and C.-Z. Yang, "Design of an Active Intrusion Monitor System," in *IEEE 37th Annual 2003 International Carnahan Conference on Security Technology, 2003. Proceedings.*, 2003, pp. 485-492: IEEE.
[415] V. Juneja, S. K. Dinkar, D. V. J. C. Gupta, C. Practice, and Experience, "An anomalous co-operative trust & PG-DRL based vampire attack detection & routing," p. e6557, 2021.
[416] K. J. T. J. o. C. Pazhanisamy and M. Education, "Efficient multi-user efficiency realization mechanism by user co-operative motion," vol. 12, no. 11, pp. 4758-4766, 2021.
[417] D. Selva, B. Nagaraj, D. Pelusi, R. Arunkumar, and A. J. A. Nair, "Intelligent Network Intrusion Prevention Feature Collection and Classification Algorithms," vol. 14, no. 8, p. 224, 2021.
[418] K. Bartos and M. Rehak, "Distributed self-organized collaboration of autonomous IDS sensors," in *IFIP International Conference on Autonomous Infrastructure, Management and Security*, 2012, pp. 113-117: Springer.
[419] P. Prajapati, B. Bhatt, G. Zalavadiya, M. Ajwalia, and P. Shah, "A Review on Recent Intrusion Detection Systems and Intrusion Prevention Systems in IoT," in *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 2021, pp. 588-593: IEEE.
[420] U. Ahmed, J. C.-W. Lin, and G. Srivastava, "Network-Aware SDN Load Balancer with Deep Active Learning based Intrusion Detection Model," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1-6: IEEE.

[421] T. Rincy N, R. J. W. C. Gupta, and M. Computing, "Design and Development of an Efficient Network Intrusion Detection System Using Machine Learning Techniques," vol. 2021, 2021.
[422] S. Al-Emadi, A. Al-Mohannadi, and F. Al-Senaid, "Using deep learning techniques for network intrusion detection," in *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, 2020, pp. 171-176: IEEE.
[423] A. Andalib and V. T. Vakili, "An Autonomous Intrusion Detection System Using an Ensemble of Advanced Learners," in *2020 28th Iranian Conference on Electrical Engineering (ICEE)*, 2020, pp. 1-5: IEEE.
[424] A. Alhowaide, I. Alsmadi, and J. J. C. C. Tang, "Towards the design of real-time autonomous IoT NIDS," pp. 1-14, 2021.
[425] D.-C. Wang, R. Chen, and H. J. I. T. o. R. Al-Hamadi, "Reliability of Autonomous Internet of Things Systems With Intrusion Detection Attack-Defense Game Design," vol. 70, no. 1, pp. 188-199, 2020.
[426] M. Major, B. Souza, J. DiVita, and K. J. a. p. a. Ferguson-Walter, "Informing autonomous deception systems with cyber expert performance data," 2021.
[427] K. Rajasekaran, K. J. I. J. o. C. S. Nirmala, and I. Security, "Classification and importance of intrusion detection system," vol. 10, no. 8, p. 44, 2012.
[428] A. Javaid, Q. Niyaz, W. Sun, M. J. E. E. T. o. S. Alam, and Safety, "A deep learning approach for network intrusion detection system," vol. 3, no. 9, p. e2, 2016.
[429] R. Sekar, Y. Guang, S. Verma, and T. Shanbhag, "A high-performance network intrusion detection system," in *Proceedings of the 6th ACM Conference on Computer and Communications Security*, 1999, pp. 8-17.
[430] J. Shun and H. A. Malki, "Network intrusion detection system using neural networks," in *2008 Fourth International Conference on Natural Computation*, 2008, vol. 5, pp. 242-246: IEEE.
[431] O. J. C.-C. M. Almomani and CONTINUA, "A Hybrid Model Using Bio-Inspired Metaheuristic Algorithms for Network Intrusion Detection System," vol. 68, no. 1, pp. 409-429, 2021.
[432] Z. Zhang, Y. Zhang, D. Guo, M. J. I. J. o. M. L. Song, and Cybernetics, "A scalable network intrusion detection system towards detecting, discovering, and learning unknown attacks," vol. 12, no. 6, pp. 1649-1665, 2021.
[433] Y. Wang, J. Lu, and J. J. I. T. o. C. Liang, "Security control of multiagent systems under denial-of-service attacks," 2020.
[434] R. Ma, P. Shi, Z. Wang, L. J. I. J. o. R. Wu, and N. Control, "Resilient filtering for cyber-physical systems under denial-of-service attacks," vol. 30, no. 5, pp. 1754-1769, 2020.
[435] J. Liu *et al.*, "Steady-state analysis of microgrid distributed control under denial of service attacks," 2020.
[436] X.-C. ShangGuan *et al.*, "Switching system-based load frequency control for multi-area power system resilient to denial-of-service attacks," vol. 107, p. 104678, 2021.
[437] V. Punitha, C. Mala, N. J. I. J. o. A. H. Rajagopalan, and U. Computing, "A novel deep learning model for detection of denial of service attacks in HTTP traffic over internet," vol. 33, no. 4, pp. 240-256, 2020.
[438] R. R. Zebari, S. R. Zeebaree, and K. Jacksi, "Impact analysis of HTTP and SYN flood DDoS attacks on Apache 2 and IIS 10.0 Web servers," in *2018 International Conference on Advanced Science and Engineering (ICOASE)*, 2018, pp. 156-161: IEEE.
[439] K. B. Virupakshar, M. Asundi, K. Channal, P. Shettar, S. Patil, and D. J. P. C. S. Narayan, "Distributed denial of service (DDoS) attacks detection system for OpenStack-based private cloud," vol. 167, pp. 2297-2307, 2020.
[440] M. Idhammad, K. Afdel, M. J. S. Belouch, and C. Networks, "Detection system of HTTP DDoS attacks in a cloud environment based on information theoretic entropy and random forest," vol. 2018, 2018.

[441] O. Rahman, M. A. G. Quraishi, and C.-H. Lung, "DDoS attacks detection and mitigation in SDN using machine learning," in *2019 IEEE World Congress on Services (SERVICES)*, 2019, vol. 2642, pp. 184-189: IEEE.
[442] T. Lukaseder, S. Ghosh, and F. J. a. p. a. Kargl, "Mitigation of flooding and slow DDoS attacks in a software-defined network," 2018.
[443] A. R. A. Yusof, N. I. Udzir, and A. J. I. J. o. D. E. T. Selamat, "Systematic literature review and taxonomy for DDoS attack detection and prediction," vol. 1, no. 3, pp. 292-315, 2019.
[444] M. Roopak, G. Y. Tian, and J. J. I. N. Chambers, "Multi-objective-based feature selection for DDoS attack detection in IoT networks," vol. 9, no. 3, pp. 120-127, 2020.
[445] J. Cheng *et al.*, "Adaptive DDoS attack detection method based on multiple-kernel learning," vol. 2018, 2018.
[446] M. Aamir and S. M. A. J. I. J. o. I. S. Zaidi, "DDoS attack detection with feature engineering and machine learning: the framework and performance evaluation," vol. 18, no. 6, pp. 761-785, 2019.
[447] X. Yang, B. Han, Z. Sun, and J. Huang, "Sdn-based DDoS attack detection with cross-plane collaboration and lightweight flow monitoring," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, 2017, pp. 1-6: IEEE.
[448] F. M. Shirazi and S. Mostafavi, "A Cognitive DDoS Attack Detection and Mitigation Algorithm for SDN."
[449] R. Swami, M. Dave, and V. Ranga, "DDoS attacks and defence mechanisms using machine learning techniques for SDN," in *Research Anthology on Combating Denial-of-Service Attacks*: IGI Global, 2021, pp. 248-264.
[450] D. Devarajan and K. J. A. a. S. Arora, "Multiclass DDoS Detection using Machine Learning Ensemble," 2021.
[451] S. Sreenivasarao, "Application Layer DDOS Attack Detection and Defense Methods," in *International Conference on Emerging Trends and Technologies on Intelligent Systems*, 2021, pp. 1-12: Springer.
[452] J. Joshi, J. Dhobi, and D. Parikh, "Security against DDoS Attack–A Review," 2021.
[453] G. Kaur, P. J. I. o. T. i. B. T. D. a. E. Gupta, and B. S. f. I. 5.0, "Classifier for DDoS attack detection in software defined networks," pp. 71-90, 2021.
[454] D. Stiawan, "Penetration Test & Cyber Attacks Scenario: An Overview Operating System," 2020.
[455] Z. Hu, R. Beuran, and Y. Tan, "Automated Penetration Testing Using Deep Reinforcement Learning," in *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 2020, pp. 2-10: IEEE.
[456] G. Stone, D. Talbert, and W. Eberle, "Using AI/Machine Learning for Reconnaissance Activities During Network Penetration Testing," in *International Conference on Cyber Warfare and Security*, 2021, pp. 541-XIV: Academic Conferences International Limited.
[457] R. Johari, I. Kaur, R. Tripathi, and K. Gupta, "Penetration Testing in IoT Network," in *2020 5th International Conference on Computing, Communication and Security (ICCCS)*, 2020, pp. 1-7: IEEE.
[458] M. C. Ghanem and T. M. J. I. Chen, "Reinforcement learning for efficient network penetration testing," vol. 11, no. 1, p. 6, 2020.
[459] A. Maraj, E. Rogova, G. J. I. J. o. G. Jakupi, and U. Computing, "Testing of network security systems through DoS, SQL injection, reverse TCP and social engineering attacks," vol. 11, no. 1, pp. 115-133, 2020.
[460] A. Chowdhary, D. Huang, J. S. Mahendran, D. Romo, Y. Deng, and A. Sabur, "Autonomous security analysis and penetration testing," in *2020 16th International Conference on Mobility, Sensing and Networking (MSN)*, 2020, pp. 508-515: IEEE.
[461] H. Alqahtani, I. H. Sarker, A. Kalim, S. M. M. Hossain, S. Ikhlaq, and S. Hossain, "Cyber intrusion detection using machine learning classification techniques," in *International*

Conference on Computing Science, Communication and Security, 2020, pp. 121-131: Springer.
[462] R. M. Ibrahim, "Performance Assessment of Snort-based Network Intrusion Detection System," Sudan University of Science and Technology, 2021.
[463] H. I. H. Alsaadi, R. M. ALmuttari, O. N. Ucan, and O. J. C. I. Bayat, "An adapting soft computing model for intrusion detection system," 2021.
[464] H. J. I. J. o. A. T. i. C. S. Abusaimeh and Engineering, "Security Attacks in Cloud Computing and Corresponding Defending Mechanisims," vol. 9, no. 3, 2020.
[465] Q. Zhang, J. Zhang, L. Liu, J. Wang, and P. J. I. J. o. N. S. Liu, "On Security of Privacy-Preserving Remote User Authentication with k-Times Untraceability," vol. 23, no. 3, pp. 449-454, 2021.
[466] M. Akter, G. D. Dip, M. S. Mira, M. A. Hamid, and M. Mridha, "Construing attacks of internet of things (IoT) and a prehensile intrusion detection system for anomaly detection using deep learning approach," in *International Conference on Innovative Computing and Communications*, 2020, pp. 427-438: Springer.
[467] A. Belous and V. Saladukha, *Viruses, Hardware and Software Trojans: Attacks and Countermeasures*. Springer Nature, 2020.
[468] L. Visaggio, "Hacking the infrastructure Cyber-attack, physical damage."
[469] T. Chadza, "Hidden Markov models for detecting and predicting sequential network attacks," Loughborough University, 2020.
[470] N. Parth, M. Dalponte, and L. Bruzzone, "Prediction of forest aboveground biomass using multitemporal multispectral remote sensing data," 2021.
[471] M. G. Karthik and M. M. Krishnan, "Detecting Internet of Things Attacks Using Post Pruning Decision Tree-Synthetic Minority Over Sampling Technique."
[472] M. Aamir, S. S. H. Rizvi, M. A. Hashmani, M. Zubair, J. J. M. U. R. J. O. E. Ahmad, and Technology, "Machine learning classification of port scanning and DDoS attacks: a comparative analysis," vol. 40, no. 1, pp. 215-229, 2021.
[473] A. K. Sarica and P. Angin, "A Novel SDN Dataset for Intrusion Detection in IoT Networks," in *2020 16th International Conference on Network and Service Management (CNSM)*, 2020, pp. 1-5: IEEE.
[474] P. Matoušek, V. Havlena, and L. Holík, "Efficient modelling of ics communication for anomaly detection using probabilistic automata," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2021, pp. 81-89: IEEE.
[475] S. Anandita, Y. Rosmansyah, B. Dabarsyah, and J. U. Choi, "Implementation of dendritic cell algorithm as an anomaly detection method for port scanning attack," in *2015 International Conference on Information Technology Systems and Innovation (ICITSI)*, 2015, pp. 1-6: IEEE.
[476] H. Gajjar and Z. Malek, "A survey of intrusion detection system (IDS) using OpenStack private cloud," in *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, 2020, pp. 162-168: IEEE.
[477] S. Jain, A.-M. Cretu, and Y.-A. J. a. p. a. de Montjoye, "Adversarial Detection Avoidance Attacks: Evaluating the robustness of perceptual hashing-based client-side scanning," 2021.
[478] N. Sultana, N. Chilamkurti, W. Peng, R. J. P.-t.-P. N. Alhadad, and Applications, "Survey on SDN based network intrusion detection system using machine learning approaches," vol. 12, no. 2, pp. 493-501, 2019.
[479] K. S. Sahoo, S. K. Panda, S. Sahoo, B. Sahoo, and R. J. T. J. o. S. Dash, "Toward secure software-defined networks against distributed denial of service attack," vol. 75, no. 8, pp. 4829-4874, 2019.
[480] S. T. Zargar, J. Joshi, D. J. I. c. s. Tipper, and tutorials, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," vol. 15, no. 4, pp. 2046-2069, 2013.

[481] M. Conti and A. Gangwal, "Blocking intrusions at border using software defined-internet exchange point (sd-ixp)," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2017, pp. 1-6: IEEE.
[482] Z. Latif *et al.*, "A comprehensive survey of interface protocols for software defined networks," vol. 156, p. 102563, 2020.
[483] A. Lara, A. Kolasani, B. J. I. c. s. Ramamurthy, and tutorials, "Network innovation using OpenFlow: A survey," vol. 16, no. 1, pp. 493-512, 2013.
[484] F. Foresta *et al.*, "Improving OpenStack networking: Advantages and performance of native SDN integration," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1-6: IEEE.
[485] S. Haider *et al.*, "A deep CNN ensemble framework for efficient DDoS attack detection in software defined networks," vol. 8, pp. 53972-53983, 2020.
[486] A. Yazdinejad, G. Srivastava, R. M. Parizi, A. Dehghantanha, H. Karimipour, and S. R. Karizno, "Slpow: Secure and low latency proof of work protocol for blockchain in green IoT networks," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 2020, pp. 1-5: IEEE.
[487] T. Ha, S. Yoon, A. C. Risdianto, J. Kim, and H. J. I. N. Lim, "Suspicious flow forwarding for multiple intrusion detection systems on software-defined networks," vol. 30, no. 6, pp. 22-27, 2016.
[488] Y. Hande and A. Muddana, "A survey on intrusion detection system for software defined networks (SDN)," in *Research Anthology on Artificial Intelligence Applications in Security*: IGI Global, 2021, pp. 467-489.
[489] G. A. Ajaeiya, N. Adalian, I. H. Elhajj, A. Kayssi, and A. Chehab, "Flow-based intrusion detection system for SDN," in *2017 IEEE Symposium on Computers and Communications (ISCC)*, 2017, pp. 787-793: IEEE.
[490] P. Hadem, D. K. Saikia, and S. J. C. N. Moulik, "An SDN-based Intrusion Detection System using SVM with Selective Logging for IP Traceback," vol. 191, p. 108015, 2021.
[491] T.-H. Lee, L.-H. Chang, and C.-W. Syu, "Deep Learning Enabled Intrusion Detection and Prevention System over SDN Networks," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2020, pp. 1-6: IEEE.
[492] W. Li, J. Tan, and Y. Wang, "A Framework of Blockchain-Based Collaborative Intrusion Detection in Software Defined Networking," in *International Conference on Network and System Security*, 2020, pp. 261-276: Springer.
[493] B. Susilo and R. F. Sari, "Intrusion Detection in Software Defined Network Using Deep Learning Approach," in *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, 2021, pp. 0807-0812: IEEE.
[494] J.-L. Luo, S.-Z. Yu, and S.-J. J. I. A. Peng, "Sdn/nfv-based security service function tree for cloud," vol. 8, pp. 38538-38545, 2020.
[495] A. O. Alzahrani and M. J. J. F. I. Alenazi, "Designing a Network Intrusion Detection System Based on Machine Learning for Software Defined Networks," vol. 13, no. 5, p. 111, 2021.
[496] A. Yazdinejadna, R. M. Parizi, A. Dehghantanha, and M. S. J. C. N. Khan, "A kangaroo-based intrusion detection system on software-defined networks," vol. 184, p. 107688, 2021.
[497] J. A. Pérez-Díaz, I. A. Valdovinos, K.-K. R. Choo, and D. J. I. A. Zhu, "A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning," vol. 8, pp. 155859-155872, 2020.
[498] X.-f. XU, J.-h. DAI, G.-l. J. D. T. o. C. S. YANG, and Engineering, "An SDN Controller Security Cluster Scheme Based on Intrusion Detection Technology," no. msam, 2020.
[499] O. Awodele, S. Idowu, O. Anjorin, V. J. J. I. i. I. S. Joshua, and I. Technology, "A Multi-Layered Approach to the Design of Intelligent Intrusion Detection and Prevention System (IIDPS)," vol. 6, 2009.

[500] M. Guimaraes and M. Murray, "Overview of intrusion detection and intrusion prevention," in *Proceedings of the 5th annual Conference on Information Security Curriculum Development*, 2008, pp. 44-46.
[501] M. A. Shibli and S. Muftic, "Intrusion Detection and Prevention System using Secure Mobile Agents," in *SECRYPT*, 2008, pp. 107-113.
[502] D. Wagner and P. Soto, "Mimicry attacks on host-based intrusion detection systems," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, 2002, pp. 255-264.
[503] L. Tan and T. Sherwood, "A high throughput string matching architecture for intrusion detection and prevention," in *32nd International Symposium on Computer Architecture (ISCA'05)*, 2005, pp. 112-122: IEEE.
[504] S. Mrdović and E. Zajko, "Secured intrusion detection system infrastructure," 2005.
[505] R. Janakiraman, M. Waldvogel, and Q. Zhang, "Indra: A peer-to-peer approach to network intrusion detection and prevention," in *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.*, 2003, pp. 226-231: IEEE.
[506] Y. Bai and H. Kobayashi, "Intrusion detection systems: technology and development," in *17th International Conference on Advanced Information Networking and Applications, 2003. AINA 2003.*, 2003, pp. 710-715: IEEE.
[507] M. Laureano, C. Maziero, and E. J. C. N. Jamhour, "Protecting host-based intrusion detectors through virtual machines," vol. 51, no. 5, pp. 1275-1283, 2007.
[508] S.-J. Han and S.-B. Cho, "Combining multiple host-based detectors using decision tree," in *Australasian Joint Conference on Artificial Intelligence*, 2003, pp. 208-220: Springer.
[509] M. Mittal, A. Khan, C. J. I. J. o. S. Agrawal, and E. Research, "A Study of Different Intrusion Detection and Prevension System," vol. 4, no. 8, pp. 1526-1531, 2013.
[510] H. El-Taj, F. Najjar, H. Alsenawi, M. J. I. J. o. C. S. Najjar, and I. Security, "Intrusion detection and prevention response based on signature-based and anomaly-based: Investigation study," vol. 10, no. 6, p. 50, 2012.
[511] Y. Zhang, "An adaptive flow counting method for anomaly detection in SDN," in *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, 2013, pp. 25-30.
[512] G. Garg and R. Garg, "Security of networks using efficient adaptive flow counting for anomaly detection in SDN," in *Artificial Intelligence and Evolutionary Computations in Engineering Systems*: Springer, 2016, pp. 667-674.
[513] G. Garg and R. Garg, "Detecting anomalies efficiently in SDN using adaptive mechanism," in *2015 Fifth International Conference on Advanced Computing & Communication Technologies*, 2015, pp. 367-370: IEEE.
[514] T. Ha *et al.*, "Suspicious traffic sampling for intrusion detection in software-defined networks," vol. 109, pp. 172-182, 2016.
[515] B. R. Granby, B. Askwith, and A. K. Marnerides, "SDN-PANDA: software-defined network platform for anomaly detection applications," in *2015 IEEE 23rd International Conference on Network Protocols (ICNP)*, 2015, pp. 463-466: IEEE.
[516] S. Hommes, R. State, and T. Engel, "Implications and detection of dos attacks in openflow-based networks," in *2014 IEEE Global Communications Conference*, 2014, pp. 537-543: IEEE.
[517] S. W. Shin, P. Porras, V. Yegneswara, M. Fong, G. Gu, and M. Tyson, "Fresco: Modular composable security services for software-defined networks," in *20th Annual Network & Distributed System Security Symposium*, 2013: Ndss.
[518] L. F. Carvalho, G. Fernandes, J. J. Rodrigues, L. S. Mendes, and M. L. Proença, "A novel anomaly detection system to assist network management in SDN environment," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1-6: IEEE.

[519] S. Lee, J. Kim, S. Shin, P. Porras, and V. Yegneswaran, "Athena: A framework for scalable anomaly detection in software-defined networks," in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2017, pp. 249-260: IEEE.
[520] L. F. Carvalho, T. Abrão, L. de Souza Mendes, and M. L. J. E. S. w. A. Proença Jr, "An ecosystem for anomaly detection and mitigation in software-defined networking," vol. 104, pp. 121-133, 2018.
[521] H. Peng, Z. Sun, X. Zhao, S. Tan, and Z. J. I. A. Sun, "A detection method for anomaly flow in software defined network," vol. 6, pp. 27809-27817, 2018.
[522] J. D. Gadze, A. A. Bamfo-Asante, J. O. Agyemang, H. Nunoo-Mensah, and K. A.-B. J. T. Opare, "An investigation into the application of deep learning in the detection and mitigation of DDOS attack on SDN controllers," vol. 9, no. 1, p. 14, 2021.
[523] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *2016 international conference on wireless networks and mobile communications (WINCOM)*, 2016, pp. 258-263: IEEE.
[524] Q. Niyaz, W. Sun, and A. Y. J. a. p. a. Javaid, "A deep learning based DDoS detection system in software-defined networking (SDN)," 2016.
[525] L. Zhu, X. Tang, M. Shen, X. Du, and M. J. I. J. o. S. A. i. C. Guizani, "Privacy-preserving DDoS attack detection using cross-domain traffic in software defined networks," vol. 36, no. 3, pp. 628-643, 2018.
[526] Y. Yu, L. Guo, Y. Liu, J. Zheng, and Y. J. I. a. Zong, "An efficient SDN-based DDoS attack detection and rapid response platform in vehicular networks," vol. 6, pp. 44570-44579, 2018.
[527] L. Barki, A. Shidling, N. Meti, D. Narayan, and M. M. Mulla, "Detection of distributed denial of service attacks in software defined networks," in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2016, pp. 2576-2581: IEEE.
[528] A. Le, P. Dinh, H. Le, and N. C. Tran, "Flexible network-based intrusion detection and prevention system on software-defined networks," in *2015 International Conference on Advanced Computing and Applications (ACOMP)*, 2015, pp. 106-111: IEEE.
[529] A. Dawoud, S. Shahristani, and C. Raun, "A deep learning framework to enhance software defined networks security," in *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, 2018, pp. 709-714: IEEE.
[530] T. V. Phan and M. J. I. A. Park, "Efficient distributed denial-of-service attack defense in SDN-based cloud," vol. 7, pp. 18701-18714, 2019.
[531] M. Behzad, K. Asghari, M. Eazi, and M. J. E. S. w. a. Palhang, "Generalization performance of support vector machines and neural networks in runoff modeling," vol. 36, no. 4, pp. 7624-7629, 2009.
[532] A. J. Smola, B. J. S. Schölkopf, and computing, "A tutorial on support vector regression," vol. 14, no. 3, pp. 199-222, 2004.
[533] W.-B. Zoungrana, A. Chehri, and A. Zimmermann, "Automatic classification of rotating machinery defects using machine learning (ML) algorithms," in *Human Centred Intelligent Systems*: Springer, 2021, pp. 193-203.
[534] E. Tabanelli, G. Tagliavini, and L. J. a. p. a. Benini, "DNN is not all you need: Parallelizing Non-Neural ML Algorithms on Ultra-Low-Power IoT Processors," 2021.
[535] P. Cichosz, S. Kozdrowski, and S. Sujecki, "Application of ML Algorithms for Prediction of the QoT in Optical Networks with Imbalanced and Incomplete Data," in *2021 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2021, pp. 1-6: IEEE.

[536] M. Naser, V. Kodur, H.-T. Thai, R. Hawileh, J. Abdalla, and V. V. J. J. o. B. E. Degtyarev, "StructuresNet and FireNet: Benchmarking databases and machine learning algorithms in structural and fire engineering domains," vol. 44, p. 102977, 2021.
[537] D. S. P. S. L.Dhanabal. (June 2015). *A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms*. Available: https://www.unb.ca/cic/datasets/index.html
[538] S. Revathi, A. J. I. J. o. E. R. Malathi, and Technology, "A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection," vol. 2, no. 12, pp. 1848-1853, 2013.
[539] V. Kumar, H. Chauhan, D. J. I. J. o. S. C. Panwar, and E. ISSN, "K-means clustering approach to analyze NSL-KDD intrusion detection dataset," pp. 2231-2307, 2013.
[540] K. M. Faraoun, A. J. I. J. o. C. I. Boukelif, and Applications, "Genetic programming approach for multi-category pattern classification applied to network intrusions detection," vol. 6, no. 01, pp. 77-99, 2006.
[541] H. Oh, I. Doh, and K. J. I. J. C. S. A. Chae, "Attack Classification Based on Data Mining Technique and Its Application for Reliable Medical Sensor Communication," vol. 6, no. 3, pp. 20-32, 2009.
[542] N. Ye and X. Li, "A scalable clustering technique for intrusion signature recognition," in *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, 2001, pp. 5-6: Citeseer.
[543] J. Verma, A. Bhandari, and G. J. A. i. M. S. J. Singh, "Review of existing data sets for network intrusion detection system," vol. 9, no. 6, pp. 3849-3854, 2020.
[544] D. H. Deshmukh, T. Ghorpade, and P. Padiya, "Improving classification using preprocessing and machine learning algorithms on NSL-KDD dataset," in *2015 International Conference on Communication, Information & Computing Technology (ICCICT)*, 2015, pp. 1-6: IEEE.
[545] S. Nusinovici *et al.*, "Logistic regression was as good as machine learning for predicting major chronic diseases," vol. 122, pp. 56-69, 2020.
[546] P. C. Austin and J. J. S. i. m. Merlo, "Intermediate and advanced topics in multilevel logistic regression analysis," vol. 36, no. 20, pp. 3257-3277, 2017.
[547] A. Ali *et al.*, "A k-nearest neighbours based ensemble via optimal model selection for regression," vol. 8, pp. 132095-132105, 2020.
[548] P. Cunningham and S. J. J. A. C. S. Delany, "k-Nearest Neighbour Classifiers-A Tutorial," vol. 54, no. 6, pp. 1-25, 2021.
[549] P. Cunningham and S. J. J. a. p. a. Delany, "k-Nearest neighbour classifiers: (with Python examples)," 2020.
[550] S. Chen, G. I. Webb, L. Liu, and X. J. K.-B. S. Ma, "A novel selective naïve Bayes algorithm," vol. 192, p. 105361, 2020.
[551] H. Zhang, L. Jiang, and L. J. P. R. Yu, "Attribute and instance weighted naive Bayes," vol. 111, p. 107674, 2021.
[552] H.-F. Zhang, M.-M. Jia, B.-B. Xiang, and C. J. E. Ma, "Predicting missing links in complex networks via an extended local naïve Bayes model," vol. 130, no. 3, p. 38002, 2020.
[553] S. K. Sankaralingam, N. S. Nagarajan, and A. J. C. C. Narmadha, "Energy aware decision stump linear programming boosting node classification based data aggregation in WSN," vol. 155, pp. 133-142, 2020.
[554] L. Shi, Q. Duan, P. Dong, L. Xi, X. J. I. J. o. C. S. Ma, and Engineering, "Signal prediction based on boosting and decision stump," vol. 16, no. 2, pp. 117-122, 2018.
[555] T. N. Shah, M. Z. Khan, M. Ali, B. Khan, and N. Idress, "CART, J-48graft, J48, ID3, Decision Stump and Random Forest: A comparative study."
[556] J. L. Speiser, M. E. Miller, J. Tooze, and E. J. E. s. w. a. Ip, "A comparison of random forest variable selection methods for classification prediction modeling," vol. 134, pp. 93-101, 2019.

[557] T. Hengl, M. Nussbaum, M. N. Wright, G. B. Heuvelink, and B. J. P. Gräler, "Random forest as a generic framework for predictive modeling of spatial and spatio-temporal variables," vol. 6, p. e5518, 2018.
[558] R. Couronné, P. Probst, and A.-L. J. B. b. Boulesteix, "Random forest versus logistic regression: a large-scale benchmark experiment," vol. 19, no. 1, pp. 1-14, 2018.
[559] S. Si, H. Zhang, S. S. Keerthi, D. Mahajan, I. S. Dhillon, and C.-J. Hsieh, "Gradient boosted decision trees for high dimensional sparse output," in *International conference on machine learning*, 2017, pp. 3182-3190: PMLR.
[560] G. Einziger, M. Goldstein, Y. Sa'ar, and I. Segall, "Verifying robustness of gradient boosted models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, vol. 33, no. 01, pp. 2446-2453.
[561] B. A. Tama, K.-H. J. N. C. Rhee, and Applications, "An in-depth experimental study of anomaly detection using gradient boosted machine," vol. 31, no. 4, pp. 955-965, 2019.
[562] D. J. H.-D. F. C. Learning, "Deep learning," 2020.
[563] A. Kamilaris, F. X. J. C. Prenafeta-Boldú, and e. i. agriculture, "Deep learning in agriculture: A survey," vol. 147, pp. 70-90, 2018.
[564] A. Voulodimos, N. Doulamis, A. Doulamis, E. J. C. i. Protopapadakis, and neuroscience, "Deep learning for computer vision: A brief review," vol. 2018, 2018.
[565] B. Charbuty, A. J. J. o. A. S. Abdulazeez, and T. Trends, "Classification based on decision tree algorithm for machine learning," vol. 2, no. 01, pp. 20-28, 2021.
[566] G. Pappalardo, S. Cafiso, A. Di Graziano, and A. J. S. Severino, "Decision tree method to analyze the performance of lane support systems," vol. 13, no. 2, p. 846, 2021.
[567] A. Schidler and S. Szeider, "SAT-based Decision Tree Learning for Large Data Sets," in *Proceedings of AAAI*, 2021, vol. 21.
[568] I. Kim, A. Ramdas, A. Singh, and L. J. T. A. o. S. Wasserman, "Classification accuracy as a proxy for two-sample testing," vol. 49, no. 1, pp. 411-434, 2021.
[569] C. Zhang, Y. E. Chen, S. Zhang, and J. J. J. a. p. a. Li, "Information-theoretic classification accuracy: a criterion that guides data-driven combination of ambiguous outcome labels in multi-class classification," 2021.
[570] T. Hubregtsen, J. Pichlmeier, P. Stecher, and K. J. Q. M. I. Bertels, "Evaluation of parameterized quantum circuits: on the relation between classification accuracy, expressibility, and entangling capability," vol. 3, no. 1, pp. 1-19, 2021.
[571] M. Hedderich, J. Fischer, D. Klakow, and J. J. a. p. a. Vreeken, "Label-Descriptive Patterns and their Application to Characterizing Classification Errors," 2021.
[572] L. Morelli, N. Grossi, A. Scippa, and G. J. J. o. M. P. Campatelli, "Extended classification of surface errors shapes in peripheral end-milling operations," vol. 71, pp. 604-624, 2021.
[573] I. Grabovsky, J. Pace, and C. J. A. P. M. Runyon, "Cut-Score Operating Function Extensions: Penalty-Based Errors and Uncertainty in Standard Settings," p. 01466216211046896, 2021.
[574] A. Parchami, "Testing Absolute Error Loss-Based Capability Index," in *International Conference on Intelligent and Fuzzy Systems*, 2021, pp. 607-613: Springer.
[575] B. T. Atmaja and M. Akagi, "Evaluation of error-and correlation-based loss functions for multitask learning dimensional speech emotion recognition," in *Journal of Physics: Conference Series*, 2021, vol. 1896, no. 1, p. 012004: IOP Publishing.
[576] R. Chiong, Z. Fan, Z. Hu, F. J. C. M. Chiong, and P. i. Biomedicine, "Using an improved relative error support vector machine for body fat prediction," vol. 198, p. 105749, 2021.
[577] H. Ming, H. Liu, H. J. J. o. C. Yang, and A. Mathematics, "Least product relative error estimation for identification in multiplicative additive models," p. 113886, 2021.
[578] H. Wang, X. Peng, Y. Xiao, Z. Xu, X. J. C. Chen, and I. Systems, "Differentially private data aggregating with relative error constraint," pp. 1-16, 2021.
[579] T. Kim, J. Oh, N. Kim, S. Cho, and S.-Y. J. a. p. a. Yun, "Comparing Kullback-Leibler Divergence and Mean Squared Error Loss in Knowledge Distillation," 2021.

[580] K. Benhenni, D. A. Girard, S. J. A. i. P. Louhichi, and A. Mathematics, "On the excess of average squared error for data-driven bandwidths in nonparametric trend estimation," vol. 12, no. Spécial, pp. 15-35, 2021.
[581] W. Yang and W. J. P. R. Nam, "Data synthesis method preserving correlation of features," vol. 122, p. 108241, 2022.
[582] S. Liu, D. Liu, G. Srivastava, D. Połap, M. J. C. Woźniak, and I. Systems, "Overview and methods of correlation filter algorithms in object tracking," vol. 7, no. 4, pp. 1895-1917, 2021.
[583] J. A. Roldán-Nofuentes and S. B. J. M. Regad, "Estimation of the Average Kappa Coefficient of a Binary Diagnostic Test in the Presence of Partial Verification," vol. 9, no. 14, p. 1694, 2021.
[584] M. J. J. A. i. D. A. Warrens and Classification, "Kappa coefficients for dichotomous-nominal classifications," vol. 15, no. 1, pp. 193-208, 2021.
[585] D. Chicco, M. J. Warrens, and G. J. P. C. S. Jurman, "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation," vol. 7, p. e623, 2021.
[586] D. Lüdecke, M. S. Ben-Shachar, I. Patil, P. Waggoner, and D. J. J. o. O. S. S. Makowski, "Performance: An R package for assessment, comparison and testing of statistical models," vol. 6, no. 60, 2021.
[587] T. L. Greaves, K. S. S. McHale, R. F. Burkart-Radke, J. B. Harper, and T. C. J. P. C. C. P. Le, "Machine learning approaches to understand and predict rate constants for organic processes in mixtures containing ionic liquids," vol. 23, no. 4, pp. 2742-2752, 2021.
[588] C. Yang, C. Huang, J. Su, and S. J. L. H. T. Wang, "Topic analysis of academic disciplines based on prolific and authoritative researchers," 2021.
[589] K. Sharma, J. C. Torrado, J. Gómez, and L. J. E. C. Jaccheri, "Improving girls' perception of computer science as a viable career option through game playing and design: Lessons from a systematic literature review," vol. 36, p. 100387, 2021.
[590] J. Pirker, A. Dengel, M. Holly, and S. Safikhani, "Virtual reality in computer science education: A systematic review," in *26th ACM Symposium on Virtual Reality Software and Technology*, 2020, pp. 1-8.
[591] S. S. J. g. Skiena, "The Algorithm Design Manual (Texts in Computer Science)," vol. 23, p. 2516, 2020.
[592] I. J. I. J. o. E. T. i. L. Ryane, "A case study of using Edmodo to enhance computer science learning for engineering students," vol. 15, no. 3, pp. 62-73, 2020.
[593] L. J. F. i. P. Bertram, "Digital Learning Games for Mathematics and Computer Science Education: The Need for Preregistered RCTs, Standardized Methodology, and Advanced Technology," vol. 11, p. 2127, 2020.
[594] N. Kalpokas and I. J. S. R. O. Radivojevic, "Bridging the Gap Between Methodology and Qualitative Data Analysis Software: A Practical Guide for Educators and Qualitative Researchers," p. 13607804211003579, 2021.
[595] P. Xie, Y. Wang, B. He, G. J. S. P. Xie, and Experience, "AutoToolVCC: Variable consistency check tool for automotive software projects," vol. 51, no. 11, pp. 2306-2316, 2021.
[596] C. Hundhausen, A. Carter, P. Conrad, A. Tariq, and O. Adesope, "Evaluating Commit, Issue and Product Quality in Team Software Development Projects," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 2021, pp. 108-114.
[597] P. S. Kumar, H. S. Behera, A. Kumari, J. Nayak, and B. J. C. S. R. Naik, "Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades," vol. 38, p. 100288, 2020.
[598] K. Zhichkin *et al.*, "Formalization of risk analysis in software products for calculating the effectiveness of investment projects," in *Journal of Physics: Conference Series*, 2021, vol. 2001, no. 1, p. 012016: IOP Publishing.

[599] E. D. Canedo, A. C. D. S. Pergentino, A. T. S. Calazans, F. V. Almeida, P. H. T. Costa, and F. Lima, "Design Thinking Use in Agile Software Projects: Software Developers' Perception," in *ICEIS (2)*, 2020, pp. 217-224.
[600] A. Lalmi, G. Fernandes, and S. B. J. P. C. S. Souad, "A conceptual hybrid project management model for construction projects," vol. 181, pp. 921-930, 2021.
[601] Ł. J. P. C. S. Radliński, "Stability of user satisfaction prediction in software projects," vol. 176, pp. 2394-2403, 2020.
[602] J. King, "Combining Theory and Practice in Data Structures & Algorithms Course Projects: An Experience Report," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 2021, pp. 959-965.
[603] T. Aho, O. Sievi-Korte, T. Kilamo, S. Yaman, and T. Mikkonen, "Demystifying data science projects: A look on the people and process of data science today," in *International Conference on product-focused software process improvement*, 2020, pp. 153-167: Springer.
[604] J. C. R. Hauck and M. Vieira, "Towards a Guide for Risk Management Integration in Agile Software Projects," in *European Conference on Software Process Improvement*, 2021, pp. 73-87: Springer.
[605] D. Gunter and K. Beattie, "Useful practices for software engineering on medium-sized distributed scientific projects," in *Proc. Collegeville Workshop Sci. Softw*, 2020.
[606] M. Subramaniam and L. W. J. T. L. Q. Braun, "Crisis-Related Research in Service to Practice: Researchers Step Up," vol. 91, no. 1, pp. 5-18, 2021.
[607] K. Vaid and U. J. P. C. S. Ghose, "Predictive analysis of manpower requirements in scrum projects using regression techniques," vol. 173, pp. 335-344, 2020.
[608] F. A. Alekundrovich, G. G. Yurevich, N. A. Michailovich, and Y. N. Glebovna, "Approach to the Search for Software Projects Similar in Structure and Semantics Based on the Knowledge Extracted from Existed Projects," in *International Conference on Computational Science and Its Applications*, 2020, pp. 718-733: Springer.
[609] B. Pérez and Á. L. Rubio, "A project-based learning approach for enhancing learning skills and motivation in software engineering," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 2020, pp. 309-315.
[610] D. Berrocal-Macías, Z. Alizadeh-Sani, F. Pinto-Santos, A. González-Briones, P. Chamoso, and J. M. Corchado, "Services Extraction for Integration in Software Projects via an Agent-Based Negotiation System," in *Practical Applications of Agents and Multi-Agent Systems*, 2021, pp. 241-252: Springer.
[611] N. Ranković, D. Ranković, M. Ivanović, and L. Lazić, "Artificial Neural Network Architecture and Orthogonal Arrays in Estimation of Software Projects Efforts," in *2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, 2021, pp. 1-6: IEEE.
[612] C. Tam, E. J. da Costa Moura, T. Oliveira, and J. J. I. J. o. P. M. Varajão, "The factors influencing the success of on-going agile software development projects," vol. 38, no. 3, pp. 165-176, 2020.
[613] R. R. Schreiber, M. P. J. I. J. o. S. E. Zylka, and K. Engineering, "Social Network Analysis in Software Development Projects: A Systematic Literature Review," vol. 30, no. 03, pp. 321-362, 2020.
[614] M. K. Thota, F. H. Shajin, P. J. I. J. O. A. S. Rajesh, and Engineering, "Survey on software defect prediction techniques," vol. 17, no. 4, pp. 331-344, 2020.
[615] D. Dobrean and L. Dioşan, "Importance of software architectures in mobile projects," in *2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, 2021, pp. 000281-000286: IEEE.
[616] S. Sankaranarayanan *et al.*, "Comparing Example-Based Collaborative Reflection to Problem Solving Practice for Learning during Team-Based Software Engineering Projects," 2021.

[617] S. I. Khaleel, "Designing a Tool to Estimate Software Projects Based on The Swarm Intelligence."
[618] M. Yaseen, S. W. Kamal, M. Bacha, A. Khan, and S. U. J. I. J. o. C. S. E. Zaman, "Frequent and proper negotiations as success factor in global software development," vol. 9, no. 4, pp. 281-288, 2020.
[619] L. Tubino, A. Cain, J.-G. Schneider, D. Thiruvady, and N. Fernando, "Authentic individual assessment for team-based software engineering projects," in *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 2020, pp. 71-81: IEEE.
[620] Y. Cheng and B. P. J. a. p. a. Nunes, "The use of Semantic Technologies in Computer Science Curriculum: A Systematic Review," 2022.
[621] K. J. J. o. E. Khaldi and S. Research, "Quantitative, qualitative or mixed research: which research paradigm to use? " vol. 7, no. 2, p. 15, 2017.
[622] R. B. Johnson and A. J. J. E. R. Onwuegbuzie, "Mixed methods research: A research paradigm whose time has come," vol. 33, no. 7, pp. 14-26, 2004.
[623] R. Johnson and A. Onwuegbuzie, "Mixed Methods Research: A Research Paradigm Whose Time Has Come-R. Burke Johnson, Anthony J. Onwuegbuzie, 2004," 2004.
[624] A. A. Aliyu, M. U. Bello, R. Kasim, D. J. J. M. Martin, and Sustainability, "Positivist and non-positivist paradigm in social science research: Conflicting paradigms or perfect partners," vol. 4, p. 79, 2014.
[625] J. Grix and G. Watkins, *Information skills: finding and using the right resources*. Bloomsbury Publishing, 2010.
[626] T. T. J. C. Tran, "Pragmatism and constructivism in conducting research about university-enterprise collaboration in the Vietnamese context," vol. 5, 2016.
[627] L. M. J. C. M. Scott and Economics, "Theory and research in construction education: the case for pragmatism," vol. 34, no. 7-8, pp. 552-560, 2016.
[628] D. L. J. J. o. m. m. r. Morgan, "Paradigms lost and pragmatism regained: Methodological implications of combining qualitative and quantitative methods," vol. 1, no. 1, pp. 48-76, 2007.
[629] G. Guthrie, *Basic research methods: An entry to social science research*. SAGE Publications India, 2010.
[630] A. Bryman, *Social research methods*. Oxford University Press, 2016.
[631] L. Cohen and L. Manion, "Research methods in education 4th edition Routeledge," ed: London, 1994.
[632] J. W. Creswell and J. Creswell, *Research design*. Sage Publications Thousand Oaks, CA, 2003.
[633] J. J. L. S. J. Asghar, "Critical paradigm: A preamble for novice researchers," vol. 10, no. 4, pp. 3121-3127, 2013.
[634] Z. J. S. R. Zhu and B. Science, "Paradigm, specialty, pragmatism: Kuhn's legacy to methodological pluralism," vol. 39, no. 5, pp. 895-912, 2022.
[635] A. Rylander Eklund, U. Navarro Aguiar, and A. J. J. o. P. I. M. Amacker, "Design thinking as sensemaking: Developing a pragmatist theory of practice to (re) introduce sensibility," vol. 39, no. 1, pp. 24-43, 2022.
[636] J. v. Brocke, M. Gau, and A. Mädche, "Journaling the Design Science Research Process. Transparency About the Making of Design Knowledge," in *International Conference on Design Science Research in Information Systems and Technology*, 2021, pp. 131-136: Springer.
[637] K. Fahd, S. J. Miah, K. Ahmed, S. Venkatraman, Y. J. E. Miao, and I. Technologies, "Integrating design science research and design based research frameworks for developing education support systems," vol. 26, no. 4, pp. 4027-4048, 2021.
[638] T. A. Henriques and H. J. I. J. o. M. P. i. B. O'Neill, "Design science research with focus groups–a pragmatic meta-model," no. ahead-of-print, 2021.

[639] M. T. Adam, S. Gregor, A. Hevner, and S. J. A. T. o. H.-C. I. Morana, "Design science research modes in human-computer interaction projects," vol. 13, no. 1, pp. 1-11, 2021.
[640] R. Winter and J. vom Brocke, "Teaching design science research," 2021.
[641] H. Weigand, P. Johannesson, B. J. D. Andersson, and K. Engineering, "An artifact ontology for design science research," vol. 133, p. 101878, 2021.
[642] M. Apiola and E. J. C. A. i. E. E. Sutinen, "Design science research for learning software engineering and computational thinking: Four cases," vol. 29, no. 1, pp. 83-101, 2021.
[643] S. Reining, F. Ahlemann, B. Mueller, and R. J. A. S. D. t. D. f. A. i. I. S. Thakurta, "Knowledge accumulation in design science research: ways to foster scientific progress," vol. 53, no. 1, pp. 10-24, 2022.
[644] J. vom Brocke, A. Hevner, and A. Maedche, "Introduction to design science research," in *Design science research. Cases*: Springer, 2020, pp. 1-13.
[645] A. Maedche, S. Gregor, S. Morana, and J. Feine, "Conceptualization of the problem space in design science research," in *International conference on design science research in information systems and technology*, 2019, pp. 18-31: Springer.
[646] Q. Deng and S. J. P. A. j. o. t. a. f. i. s. Ji, "A review of design science research in information systems: concept, process, outcome, and evaluation," vol. 10, no. 1, p. 2, 2018.
[647] G. Jain, "Application of SNORT and Wireshark in network traffic analysis," in *IOP Conference Series: Materials Science and Engineering*, 2021, vol. 1119, no. 1, p. 012007: IOP Publishing.
[648] S. J. A. P. Gholizadeh, "Top Popular Python Libraries in Research," 2022.
[649] A. Aldahiri, B. Alrashed, and W. J. F. Hussain, "Trends in using IoT with machine learning in health prediction system," vol. 3, no. 1, pp. 181-206, 2021.
[650] N. Niknami, J. J. I. T. o. N. S. Wu, and Engineering, "Entropy-KL-ML: Enhancing the Entropy-KL-based Anomaly Detection on Software-Defined Networks," 2022.
[651] J. Arevalo-Herrera, J. E. Camargo Mendoza, and J. I. Martinez Torre, "Network anomaly detection with machine learning techniques for SDN networks," in *Proceedings of the 7th International Conference on Information and Education Innovations*, 2022, pp. 129-135.
[652] J. Wang and L. J. S. Wang, "SDN-Defend: A Lightweight Online Attack Detection and Mitigation System for DDoS Attacks in SDN," vol. 22, no. 21, p. 8287, 2022.
[653] M. S. El Sayed, N.-A. Le-Khac, M. A. Azer, A. D. J. I. T. O. C. C. Jurcut, and Networking, "A flow based anomaly detection approach with feature selection method against DDoS attacks in SDNs," 2022.
[654] G. F. Scaranti, L. F. Carvalho, S. B. Junior, J. Lloret, and M. L. J. E. S. w. A. Proença Jr, "Unsupervised online anomaly detection in Software Defined Network environments," vol. 191, p. 116225, 2022.