Assiut University

Faculty of Computers and Information
Information Technology Department

A Proposed Technique for Distributed Denial of Service Attack Detection in Software-Defined Network

Thesis

Submitted in partial fulfillment of the Requirements for the master's degree

In

Computers and Information (Information Technology)

By

Waheed Gameel Gadallah Ghali

Demonstrator at Information Technology Department,
Faculty of Computers and Information,
Assiut University

Supervised by:

Prof. Hosny M. Ibrahim
Professor at Information Technology Department, Faculty of Computers and Information, Assiut University

Associate Prof. Nagwa M. Omar
Associate Prof. at Information Technology Department, Faculty of Computers and Information, Assiut University

Examined by:

Prof. Abd El-Majeed Amin Ali
Dean, Faculty of Computers and Information, Minia University

Prof. Moumen Taha Hanafy Ahmed
Professor and Vice Dean for Graduate Studies and Research, Faculty of Engineering, Assiut University

Prof. Hosny M. Ibrahim
Professor at Information Technology Department, Faculty of Computers and Information, Assiut University

Associate Prof. Nagwa M. Omar
Associate Prof. at Information Technology Department, Faculty of Computers and Information, Assiut University

Assiut, 2022

Acknowledgment

Acknowledgment

First of all, I would like to express my gratitude and thanks to **ALLAH**, the Most Gracious and the Most Merciful.

I would like to express my sincere gratitude and deepest thanks to **Prof. Hosny M. Ibrahim**, professor of Information Technology and Former Head of Information Technology department, Faculty of Computers and Information, Assiut University for his guidance on the research topic, kind supervision, and continuous encouragement during the whole work. Many thanks for his help and support.

I would like to express my sincere gratitude and deepest thanks to **Associate Prof. Nagwa M. Omar**, Information Technology Department, Faculty of Computers and Information, Assiut University for her support, and close supervision during the whole work. Many thanks for her help and support.

Special appreciation, pride, and gratefulness go to **my Great Father's Soul, my Dear Mother, my Beloved Brother, my Kind Sisters, and all my Family** for giving me full support during my work; without their prayers and love, I would not be capable of completing this work.

To the Soul of my all-times Great Role Model (my Beloved Father)

Abstract

Abstract

Software-Defined Networking (SDN) is a new network architecture that separates the control plane from the data plane [1]–[3]. It has central network control, and programmability facilities, therefore SDN can improve network flexibility, management, performance, and scalability. Although SDN has many benefits, it has some security challenges that threaten the proper performance of the network such as Distributed Denial of Service (DDoS) attacks that target attacking the control plane as well as the data plane [4]–[8].

There are a lot of security methods that defend against DDoS attacks in SDN which include entropy-based, traffic pattern analysis-based, and Machine Learning-based (ML) [9]–[11] [12]–[14][15]–[17]. Entropy and traffic pattern techniques have low overhead but they suffer mainly from the difficulty to adapt the network changes and different network behaviors [18]. ML techniques are preferred to detect malicious activity since they are self-learning and can adapt to several network states [19].

This thesis proposes accurate and significant techniques to detect DDoS attacks against the control plane and data plane in SDN. The introduced technique proves to detect the attack with high accuracy using ML techniques exploiting proposed advanced features obtained from traffic flow information and statistics. The developed model is trained with kernel Radial Basis Function (RBF). The technique uses advanced features such as unknown IP destination address, packets inter-arrival time, Transport Layer Protocol (TLP) header, and Type of Service (ToS) header for the control plane. Also, advanced features such as switch stored capacity, the average rate of packets with unknown destination IP address, IP Options header field, and the average number of flows are used for the data plane. The proposed features had not been used before. The features are used to create a dataset that is used as an input to the linear Support Vector Machine (SVM) classifier [13], [20]–[22]. Compromised end systems that send offensive traffic are blocked and their information is stored. The experimental results prove that the proposed ML-based technique can detect the attack with high accuracy, high detection rate, high f_measure, and low false alarm, compared to other related techniques [23]–[38].

Also, this thesis provides analytical and simulation models for TCP and UDP flow in SDN. Traffic analysis and modeling in SDN are important for tracking network activity and availability to detect abnormalities, such as security and operational problems. Therefore, Seven-Dimensional (7-D) states models with transitions are developed in this thesis to model SDN TCP and UDP flows, unlike other models that are used in previous work which use three-dimensional or four-dimensional states [39][40]. The proposed work uses multiple controllers and multiple switches with limited capacities unlike other models in related work which use one controller or one switch with an infinite buffer [39]–[52]. Using multiple controllers improves network security due to the backup capability. Also, network scalability is enhanced since there are enough controllers to serve more switches. Using multiple switches can provide more accurate results to better reflect network traffic cases. The packet average delay and loss probability are tracked to measure the performance of the developed models. Also, the proposed models can be used roughly to detect DDoS attacks by considering that the attack traffic has a Poisson input distribution with a high average arrival rate. The Simulation results are formulated and compared to analytical ones. The results prove that the proposed modeling is accurate compared to the recent work.

List of Contents

List of Contents

List of Abbreviations

List of Abbreviations

Abbreviation	Full name
7-D	Seven-Dimensional
ABF	Average number of packets of flow
API	Application Programming Interface
AQM	Active Queue Management
CNs	Conventional Networks
DDoS	Distributed Denial of Service
DF	Defined or Data Flow
FN	False Negative
FP	False Positive
FPR	False Positive Rate
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IT	Information Technology
K-FKNN	Fast K-Nearest Neighbors
KNN	K-Nearest Neighbor
ML	Machine Learning
MMPP	Markov Modulated Poisson Process
MAC	Medium Access Control
NF	New Flow
NOS	Network Operating System
OF	OpenFlow
OVS	Open vSwitch
QoS	Quality of Service
RBF	Radial Basis Function
SDN	Software-Defined Networking
SOM	Self-Organizing Maps
SVC	Support Vector Classifier
SVM	Support Vector Machine
TLP	Transport Layer Protocol
TN	True Negative

ToS	Type of Service
TP	True Positive
TPR	True Positive Rate
UX	User Experience
VoIP	Voice over Internet Protocol

List of Tables

List of Tables

List of Figures

List of Figures

Chapter 1
Introduction

1 Introduction

1.1 Motivation

Software-Defined Networking (SDN) is a new promising computer network architecture that separates the control plane from the data plane [1]–[3] [53]. This is accomplished by implementing the control plane outside the forwarding device itself. SDN implements the control plane in a special main component called an SDN controller [54]. SDN can improve network manageability, scalability, and performance through its central network control and programmability facilities [2].

However, it may suffer from different security threats such as Distributed Denial of Service (DDoS) attacks [4]–[8]. DDoS attacks can represent a major severe threat to SDN since the centralized control is implemented in the controller, which may create a single point of failure. If a DDoS attack succeeds to drop the SDN controller, the entire network will also drop [55][56].

There are many security approaches are utilized to defend against SDN DDoS attacks such as entropy-based detection and traffic pattern analysis detection [9]–[11][12]–[14]. Machine Learning (ML) is another approach that is considered strong and common to detect SDN DDoS attacks [15]–[17]. It has many popular methods that are used to create a model trained using a dataset and then the generated model can be used for detecting the attack [19].

Although entropy and traffic pattern approaches have low overhead, their fundamental drawback is their inability to adapt to changing network conditions and various network behaviors [18]. Since ML approaches are self-learning and adapt to various network situations, they are recommended for spotting malicious behavior [19]. Therefore, the current thesis presents proposed ML-based techniques to detect DDoS attacks against both the control plane and the data plane.

In SDN, analysis and modeling are important for tracking network activity and availability to detect abnormalities, such as security and operational problems. One of the motivations behind adopting traffic analysis is to use its solutions to manage networks. Network provider companies can use traffic analysis for tracking their traffic trends. Traffic modeling can be used for identifying security attacks and finding solutions to them. Also, network activity and availability are tracked by network traffic analysis, which also looks for operational and security anomalies. Traffic modeling and analysis can help to predict and expect network behavior in the future [60][61].

Therefore, it is effective to develop analytical models in this thesis to develop 7-D states models with transitions that can be used to model SDN TCP and UDP flows, unlike other models that are used in previous work which use 3-D or 4-D states [39][40]. Multiple controllers and multiple switches with limited capacities are used through the developed models, unlike models in related works which may use a single controller or a single switch with infinite capacities [39]–[52]. Furthermore, it is possible to utilize the developed models roughly to differentiate the DDoS attack traffic from the normal traffic by considering that the attack traffic has a Poisson input distribution with high average arrival rates [41][42][62][63].

1.2 Objectives

The objectives of the presented thesis are

- to provide DDoS attack detection techniques for the control plane and data plane in SDN using ML-based algorithms such as Naive Bayes, K-Nearest Neighbor (KNN), Decision Tree, Random Forest, and Support Vector Machine (SVM). The ML model is created using proposed features that are not used by the most recent research works.
- to develop analytical and simulation models using 7-D states with transitions to model TCP and UDP flows in SDN which can be used roughly to differentiate normal traffic from DDoS attack traffic by considering that the attack traffic has a Poisson input distribution with high average arrival rates. Flow-level arrivals are considered to get more accurate knowledge about modeling the SDN traffic. The constructed models should handle SDN networks with multiple OpenFlow (OF) switches in the data plane and multiple SDN controllers in the control plane with limited capacities.

The proposed techniques in this thesis should provide the following:

- Detecting DDoS attacks against the SDN controller and the data plane.
- Increasing the accuracy of DDoS attack detection in SDN: by using valuable and proposed features to train the classifiers.
- Using many ML-based algorithms and comparing them to construct and select the best detection models of them.
- Reducing DDoS attack effects using mitigation techniques: blocking the compromised end systems that send the attack packets and logging their valuable information and statistics.
- Avoiding future attack attempts: enabling the controller to install new flow rules that discard any upcoming packet that has the same flow as the flow of the detected attack packet.

- Developing 7-D states models with transitions to model SDN TCP and UDP flows.
- Differentiating DDoS attacks traffic and normal traffic using developed models and measuring their performance such as average packet delay and loss probability.
- Considering multiple controllers with finite buffers in the control plane to increase network security and scalability.
- Considering multiple switches with finite buffers in the SDN data plane to provide more accurate results and better reflect the actual case of traffic in the network.

1.3 Work Plan

The objectives of the presented thesis are accomplished through the following steps:

1- Reviewing the most recent detection techniques of DDoS attacks in SDN and the most recent traffic analysis models for SDN under DDoS attack traffic.

2- Reviewing the latest work that utilizes ML-based approaches to detect DDoS attacks in SDN and recognizing their advantages and disadvantages.

3- Reviewing the latest work that utilizes modeling SDN traffic and tracking its behavior using different performance measures and recognizing their advantages and disadvantages.

4- Developing efficient detection techniques for DDoS attacks in SDN based on ML algorithms that improve the accuracy and overcome the problems in similar techniques.

5- Performing experimental results to evaluate the performance of the proposed technique compared to other related techniques

6- Developing 7-D states models with transitions to model SDN TCP and UDP flows which can be used roughly to differentiate normal traffic from DDoS attack traffic. The developed analytical models should provide more efficient and accurate results for representing SDN traffic and overcome the problems in similar SDN traffic modeling works.
7- Introducing mathematical analysis for the proposed models' performance.
8- Evaluating the accuracy of the proposed queueing models compared with the simulation results as well as with other recent models.

1.4 Thesis Organization

The rest of this thesis is organized as follows:

- Chapter (2) presents a review of SDN, DDoS attacks, and related work.
- Chapter (3) proposes a DDoS detection technique based on ML in SDN using proposed features and introduces network security performance measures for its performance compared with other recent techniques.
- Chapter (4) introduces a 7-D state model with transitions which is used to model SDN TCP and UDP flows in multi-controller SDN architectures using multiple OF switches to differentiate normal traffic from DDoS attack traffic and introduces performance measures for network traffic modeling compared with other recent techniques.
- Chapter (5) presents the simulation results that evaluate the performance of the proposed ML-based DDoS attack detection technique in SDN compared with other related techniques and approaches. Also, it proposes the results that evaluate the performance of the developed queuing models for the TCP and UDP flows in SDN compared with the simulation results as well as with other related work.
- Chapter (6) presents the conclusion and suggestions for future work.

Chapter 2

Review and Related Work

2 Review and Related Work

2.1 Introduction

This chapter introduces SDN Architecture, SDN Traffic Modeling, SDN Security Threats, and SDN DDoS Attack and its Detection Techniques.

2.2 SDN Architecture

In this section SDN history and architecture are explained. A comparison between SDN and Conventional Networks (CNs) is proposed. SDN architecture advantages and disadvantages are also expressed in this section.

2.2.1 SDN History

SDN is based on separating the control plane from the data plane. In addition, it supports network programmability, which uses software applications to manage and maintain network functionality [2], [64]–[69].

In SDN, the whole functionality for processing network traffic is shifted to the controller, which represents the control plane, while data plane networking equipment such as switches become mere packet forwarding devices. As a result, the controller isolates the network's complexity. Switches become less expensive [6].

Figure (2.1) shows the evolution of SDN over the past years. The term software-defined networking was coined in 2009. The history of SDN may be divided into three phases [70]. The first phase is called active networks (from the mid-1990s to the early 2000s), which began using network programming procedures. The goal of active networking was to allow network switches to execute packet computations. The use of an active network reduces computing expenses. The second phase is the separation of control and data planes (from around 2001 to 2007), which decoupled both control and data planes. Because the control logic is not connected to hardware, the separation of the control and data planes helped in faster transformation and more flexibility. In 2003, the Internet Engineering Task Force (IETF) working group established the ForCES protocol, which was the first instance of plane separation [71]. The third phase is named OF API and Network Operating System (NOS) (from 2007 to around 2010), which introduced the first open interface between the control and data planes to make their separation effective and useful [72]. In the early days of SDN, network virtualization was a popular "use case" for it. Through network virtualization, a network's abstraction from the underlying physical hardware is made possible. Multiple virtual networks can operate over a single common infrastructure thanks to network virtualization, and each virtual network can have a topology that is far more straightforward than the underlying physical network [6][73][70].

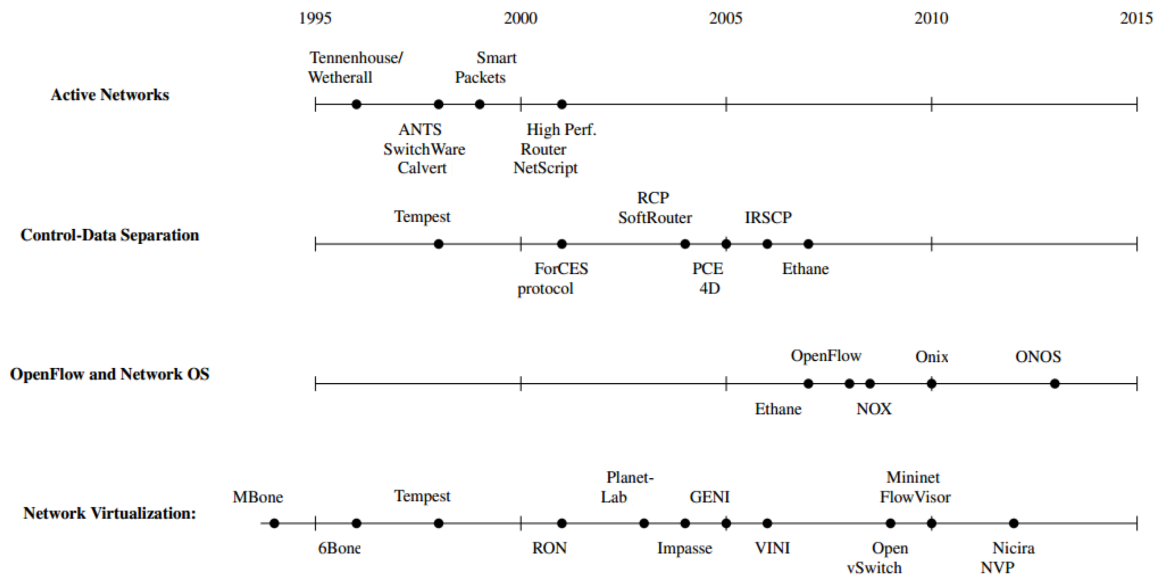


Figure 2.1: Evolution of SDN over the past years

2.2.2 SDN against CNs

As stated earlier, SDN architecture separates the forwarding plane from the controlling plane for the network layer devices (routers). It replaces the network layer device with an SDN-supported device called OF switch or Open vSwitch (OVS) which is considered the forwarding device in SDN architecture [74]. SDN increases the simplicity of the network layer device since its main function is to forward packets not to execute routing functions. However, the implementation of the network logic or control plane is done in a major component named SDN controller which does not exist in CNs infrastructure. Accordingly, within SDN architecture, the forwarding device receives its forwarding table and other configurations from the SDN controller [54].

Conversely, in CNs, the control plane is built inside the device itself besides the forwarding plane. Data routing and decision-making functions for switching or routing are completed inside the same device. Consequently, the complexity of the network device is increased which affects its performance. Also, necessary configurations for each device to function properly are done according to the device's vendor. In such a case, the configuration of each device is set individually, which causes a waste of time and effort, especially when performing network maintenance and scalability [2][64][70].

There are two key distinguishing properties for SDN architectures over CNs [75].

1. Decoupling of control plane and data plane: Implementing the control plane into a basic component which is the controller instead of each network device reduces the complexity and the cost of network devices compared with CN devices in which the control plane is distributed among

all core devices. Network operators do not need to configure and manage each network device individually. Instead, the controller can manage and control all network devices easier since it has a perfect view and global knowledge of the network and its behavior as a whole [75]–[77].

2. Network programmability: SDN provides sophisticated and suitable application programming interfaces (APIs) that facilitate the development of networking applications and software programs. Applications, for instance, traffic engineering and Quality of Service (QoS) are easy to be deployed programmatically through simple programming commands and lines of code. SDN facilitates providing and applying new configurations, troubleshooting procedures, and management operations. In contrast, CNs devices are proprietary and closed which complicates the development of such network applications [75][77].

2.2.3 SDN Planes and Interfaces

Figure (2.2) shows the general architecture of SDN [78]. SDN has mainly three layers which are called planes. The first plane is the management or application plane, the second is named the control plane, and the third is called the data or forwarding plane. The SDN planes are described as follows [78].

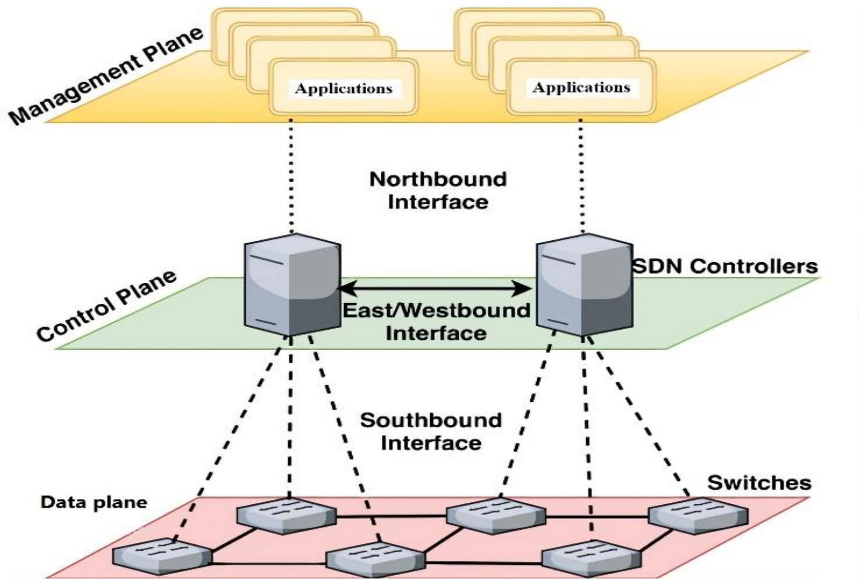


Figure 2.2: General SDN architecture

1. Management or application plane: The management plane is the set of applications, programs, as well as policies that are converted into software instructions that determine the behavior of hardware components. This wide set of network settings requires essentially a wide range of applications, including routing, policy enforcement, and firewalls [78].
2. Control plane: The control plane, sometimes known as the controller, is a decoupled entity from the data plane that has a global perspective of the whole network inside its domain. It makes use of the NOS to make network management easier. In SDN design, the control plane also serves as a strategic control point and a link between the data plane and the management plane. It uses a southbound interface to manage flow control in switches and network elements. It employs northbound APIs to interface with the management plane for network applications at the same time. It instructs data plane devices by installing flow entries and gives network state information to the management plane for application development based on its global view. Almost all controllers include a topology

manager, statistics manager, routing module, device manager, and other network functions. Examples of SDN controllers are NOX, POX, FloodLight, and OpenDaylight [79][80]. Most controllers accept OF as a southbound interface [78].

3. Data plane: The data plane contains the hardware components and forwarding devices that represent the data path. SDN only utilizes programmable switches because the routing control is implemented in the control plane instead of the forwarding devices. These switches can communicate with the controller, which is commonly done through OF protocol. Flow tables, secure channels, and the OF protocol are the three key components. Flow tables can be found on one or more devices. It communicates with the controller over a secure channel, and it communicates with external controllers via the OF protocol. Flow entries in the format of match, actions, and counters are stored in the flow table. For each packet, header matching is performed, and based on the results, a specific action is taken, and the counter is updated. Data plane devices can use control plane instructions to execute a variety of tasks, including forward to port, forward to the controller, and drop. When a switch gets a packet, it first looks it up in the flow tables to see if the packet header matches. It acts and passes the packet to a certain port if it detects a flow entry that matches this matching. If no entry is found, the packet is either dropped or passed to an external controller through the Packet-IN message. Based on the network's overall perspective, the controller responds with a Packet-OUT message and creates a flow entry for this packet [54][78].

Communications among different SDN planes accomplish using SDN interfaces, which provide suitable communication links among them.

Therefore, APIs are crucial in SDN because they allow planes to communicate with one another [78]. The SDN interfaces are expressed as follows [78].

1. Northbound API: it serves as a common interface between the controller and the application plane for application developers. It aids in the provision of underlying device information for application development, making SDN control simple and dynamic [78].
2. Southbound API: it is an SDN enabler that offers a communication mechanism between the control plane and the data plane [78][81]. This API is used to install flow entries and push configuration information to the data plane. It also provides the control plane with an abstraction of network device functions. Heterogeneity, vendor-specific network parts, and language standards are all major problems for southbound interfaces.
3. East/Westbound API: The primary characteristic of SDN is its centralized network control. Distributed controllers are becoming increasingly important as the number of network devices grows exponentially. Every controller in such a distribution has its domain with underlying forwarding devices. For a consistent global view of the entire network, controllers must share information from their respective domains. Information is imported and exported among controllers via eastbound APIs. Westbound APIs, on the other hand, can allow traditional network elements (routers, etc.) to communicate with controllers [78].

When a network has a single controller, it suffers from performance and scalability limitations. Also, it becomes vulnerable to a single point of failure problem and various attacks. Therefore, using multiple controllers in SDN adds robustness to the control plane and protects it from halting and failing the whole network by providing backup to the controller [2][82]. Kandoo [82], is a hierarchical network architecture design of an SDN control plane. As depicted in figure (2.3), Kandoo architecture represents the control plane as a hierarchy of controllers [82]. Kandoo consists mainly of double layers of controllers. The first is the root controller and the second layer includes multiple local controllers. The root controller manages all the local controllers. Each local controller is responsible for a network of multiple OF switches. The local controller tries to process requests arriving from OF switches in its network. If the local controller fails to process a request, it will forward the request to the root controller [83].

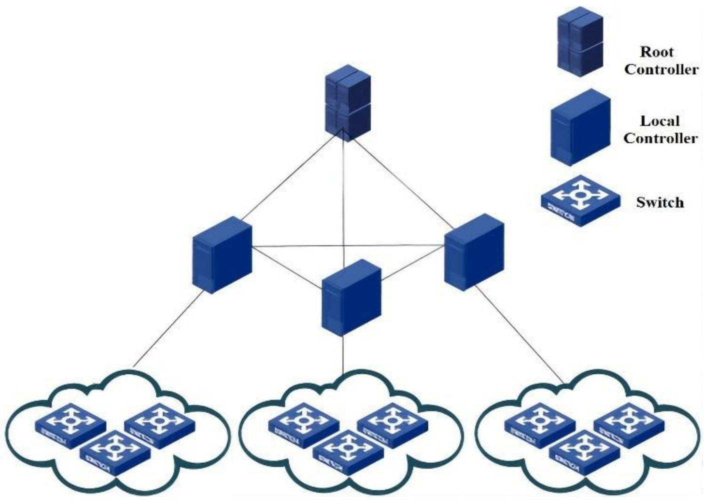


Figure 2.3: SDN network with multiple controllers (Kandoo architecture)

2.2.4 SDN Architecture Advantages and Disadvantages

Deploying SDN architecture has many benefits as follows:

1. Enhanced network security: Traditional networks require consistent firewall control, but SDN can automatically shape traffic. The network manager/administrator can introduce an application on top of the control plane to implement a firewall in SDN [84].
2. Enhanced QoS: In SDN, the control plane can run a different algorithm in the switch to manage QoS flow rules. Several network architectures use SDN features to improve QoS in their domain [84].

3. Enhanced scalability: SDN can give efficient service through programmability. Also, automated flow management allows a network administrator to set up highly scalable, flexible, and swiftly responsive to changing network needs [84].
4. Centralized control of multi-vendor environments: The SDN control software can control any OF-enabled network device from any vendor, including switches, routers, and virtual switches. Rather than having to manage groups of devices from individual vendors, Information Technology (IT) staff can use SDN-based orchestration and management tools to quickly deploy, configure, and update devices across the entire network [85].
5. Increased network management and reliability: SDN allows IT staff to make changes to network configuration without affecting the rest of the network. IT staff can specify high-level configuration and policy statements using SDN eliminating the need to individually configure network devices, and reducing the likelihood of network failures due to configuration or policy conflicts [86][85].
6. Lower operating costs: SDN can decrease operational expenses and grow administrative savings by automating and centralizing many common network administration challenges [86].
7. Consistent and timely content delivery: If you can direct and automate data flow, it is easier to provide QoS for Voice over Internet Protocol (VoIP) and multimedia transmissions. SDN also helps in the streaming of higher-quality video streams since it improves network responsiveness and, as a result, the user experience (UX) [86].

However, SDN architecture suffers from disadvantages as follows.

1. Single point of failure: SDN architecture is centralized. The control plane is represented mainly in the controller. This may cause a single point of failure problem. If the controller is attacked and drops, the entire network will drop and fail as well [87].
2. Lack of standardized security protocols: There are not any standardized security protocols for SDN. Security issues still exist despite the presence of some third-party service providers. Only those with experience managing SDN systems can stop significant threats [88].
3. Lack of built-in security in forwarding devices: SDN does not utilize traditional core devices such as routers and switches. Hence, the security built into these devices is removed [88].

2.3 SDN Traffic Modeling

Traffic modeling and analysis are valuable and effective for tracking the behavior of the networks and their traffic in the SDN architecture [89]. Furthermore, it is effective to develop models for communications in OF-based SDN networks to detect DDoS attacks [90][91].

The authors in [47] modeled the SDN switch and controller as M/M/1 and M/M/1/m Markovian queuing systems [92]. The model's accuracy was computed as the new flows probability increased [47]. However, the model considered utilizing only one controller and one switch. The authors in [48] solved this trouble by utilizing Jackson to track the packet rate from the controller to the switch. But the constructed model used a single controller and a single switch. The same authors exploited a Jackson network in [49] to model the forwarding plane. They used multiple switches with either a finite or infinite M/M/1 buffer. However, the developed model was assumed to use a single controller.

The number of processed messages over various time scales was used by the authors in [52] to calculate the controller and switch behaviors using a queuing model. However, the model considered using only one controller and one switch. In [51], a mathematical model depending on network calculus is developed to provide network scalability. However, the developed model used a single controller with infinite capacities. In [50], the same authors enhanced this work by modeling the switch output using the delay and queue length boundaries. However, they could not accurately represent the output when the network is stable [40].

The authors in [43] suggested a multistage controller targeting improving the SDN control plane's availability. The model ignored the switches and created a large delay for packets. In [44], the authors suggested a preemption-based packet-scheduling scheme to improve global fairness and reduce packet loss in SDN data planes. The data plane is assumed to consist of only one M/M/1 queue with low priority and one M/M/1 queue with high priority. The control plane was modeled using a single M/M/1/m queu. The same authors later used a Markov Modulated Poisson Process (MMPP) in SDN architectures to better model the busty nature of packet arrivals of multimedia traffic as described in [50]. The model used only one controller and assumed the queue to have infinite buffer capacity in the data plane.

The authors in [39] developed a 3-D state model using switches with two limited capacities queues and a controller with an infinite size queue. However, the developed model is assumed to use a single controller. Two traffic analysis models for TCP and UDP in SDN considering both flow-level and packet-level arrivals were constructed using 4-D states in [40]. However, the models have a single controller with a single switch and do not support network scalability.

2.4 SDN Security Threats

Centralizing the network control plane in SDN opens new security challenges and vulnerabilities. Communication channels between isolated planes can be targeted to deceive one plane into attacking the other. The control plane is more attractive to security attacks, and especially to DoS and DDoS attacks, because of its visible nature. The major and most common security challenges are described as follows:

1. Application plane security challenges: SDN applications can pose major security vulnerabilities to network resources, services, and functions because there are no standards or open specifications to support open APIs for applications to control network services and functions through the control plane [9339 Examples of the SDN applications' security challenges are authentication and authorization [94], and access control and accountability [95][96].
2. Control plane security challenges: The SDN controller is the main control unit in SDN. If an attack succeeds to drop an SDN controller, the whole network will also drop. Examples of security threats and challenges that may target the SDN controller are threats from applications [96], threats due to scalability [97], DoS attacks [97], and distributed control plane challenges [93].
3. Data plane security challenges: The first and most important security concern is distinguishing attack or malicious traffic from normal traffic since the forwarding elements do not have the right to specify and install their flow rules by themselves. The second concern is related to the buffer

space of a switch's queue. Typically, a switch's queue has a finite capacity. Therefore, there is a limitation on the maximum number of flow rules that can be saved in the switch's flow table buffers. Such shortcomings and limitations in the switch's queue buffer may expose the forwarding device to various saturation attacks [96]. Examples of security threats that may attack the data plane in SDN are man-in-middle attacks [98], and DoS attacks against OF switches as shown in figure (2.4) [99].

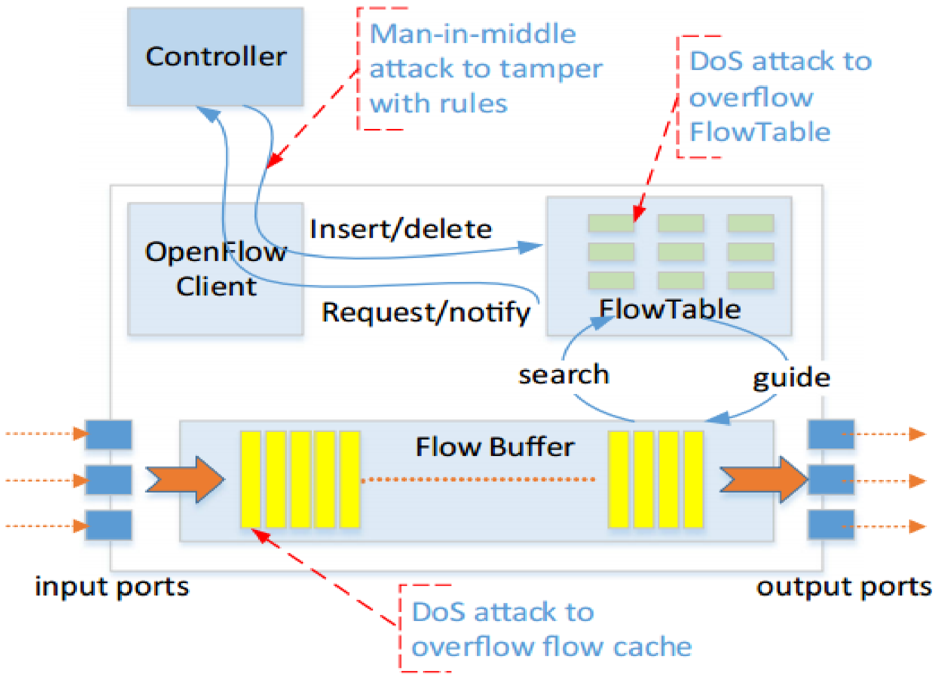

Controller
Man-in-middle attack to tamper with rules
DoS attack to overflow FlowTable
OpenFlow Client
Insert/delete
Request/notify
FlowTable
search
guide
Flow Buffer
input ports
output ports
DoS attack to overflow flow cache

Figure 2.4: Man-in-middle attack and DoS attack against OF switches

2.5 SDN DDoS Attack and its Detection Techniques

In this section, DDoS attack in SDN as well as DDoS attack detection are explained.

2.5.1 Introduction to DoS and DDoS attacks

A DoS attack [5] is an availability-based attack since it targets preventing the victim from being available to provide services to legitimate users. In a DoS attack, an attack system seeks to exhaust the resources of a victim so that the entire targeted system drops and becomes unable to function properly. Over time, the DoS attack evolutes to a DDoS attack where an attacker hijacks some other vulnerable devices on the Internet to attack the same network resource at the same time multiplying the effect of the DoS attack [100].

DDoS attacks consist mainly of four major components [101][102].

1. Main attacker: It is the genuine source of the attack procedure.
2. Handlers: They are hijacked systems that host and execute the attacking software to control multiple agents or zombies.
3. Agents or zombies: They execute the attacking software and produce packet floods targeted at the victims.
4. Targeted victims: It is the network component that is targeted by the main attack launcher.

As shown in (figure 2.5), through the operation of a DDoS attack, the main attacker tries to hijack and control many devices called handlers that host and execute the attacking software [101]. Handlers can compromise multiple agents called zombies that execute the attacking software and produce packet floods targeted at the victims. In the SDN architecture, many components can be targetted by the main attacker such as an OF switch and the SDN controller. Since the controller represents the control plane of a network in SDN, it is usually the main target. Consequently, through a DDoS attack in the SDN architecture, a large number of packets with spoofed source IP addresses are sent through the network to attack the controller. In such a situation, the entire SDN architecture may lose its controlling plane since the controller is considered the main control plane unit in SDN architecture. Consequently, the whole network will fail and will not work properly [103].

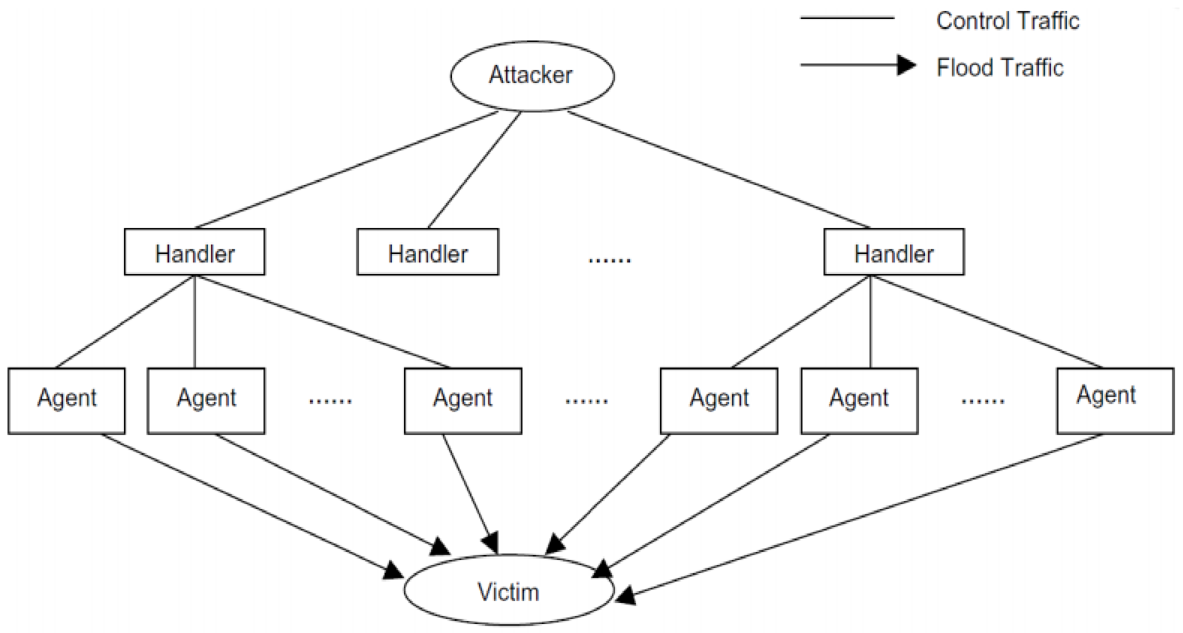


Figure 2.5: DDoS attack structure

2.5.2 DDoS Attack Detection

DDoS attack detection has many security approaches [19][104]–[106] which are explained in the following subsections.

2.5.2.1 Entropy-based Detection

Entropy may be utilized to determine the field’s randomness within a duration of time. Entropy techniques are used in [55][107] to detect DDoS attacks in SDN [19].

High entropy values mean a more distributed distribution of probability while low entropy values denote distribution concentration. Entropy can be computed on a variety of attributes, such as network flows, IP addresses, and packet variety. This method has low overhead calculations but it cannot adapt to network state changes [18]. Also, entropy-based techniques may be unsuitable for all cases of network traffic [108]. The authors in [55] propose an SDN DDoS detection system using entropy. A threshold was selected experimentally. The approach is not reliable for all network conditions. The provided technique suffers mainly from a lack of stability and adaptability to network state changes.

Another entropy-based DDoS detection system was introduced in [107]. The technique is fast and effective. The authors utilized generalized entropy computation which can combine Shannon and Renyi entropy to identify DDoS traffic. Also, the technique allowed SDN controllers to deal with high traffic effectively.

2.5.2.2 Traffic Pattern Analysis-based Detection

The algorithms in this category imply that the attacked devices behave differently from relevant patterns that vary from normal devices. Attacked computers are normally managed by a single botmaster in a botnet attack condition. Relevant traffic patterns are seen due to sending instructions to several members of the same botnet triggering relevant actions [109].

The authors in [110] presented a time and space-efficient technique to detect SDN DDoS attacks. The method tracks characteristics of identifying compromised hosts. Different traffic statistics are used to identify abnormal behavior. After that, the method uses hash functions in switches and stores integer variables for each host. Hence, a threshold is set and used to classify compromised and normal hosts. The technique is efficient since it saves time and space. However, it violates the SDN standard since it implements logic into switches. Also, the detection performance and accuracy should be improved.

The authors in [41] present an examination of the impact of spoofed and non-spoofed TCP-SYN flooding DDoS attacks on the controller resources in SDN. They also propose an Intrusion Detection System (IDS) based on ML. The authors recommend using queueing systems with M/M/1 and detecting

DDoS assaults on high arrival rates for the developed modeling. The traffic is classified using five distinct classification models from a range of families, and each model is assessed using a separate set of performance characteristics. The classification models are validated using the cross-validation method. Better features may be retrieved thanks to this work, which also allows for the classification of DDoS assault traffic.

In [42], the authors suggested a dynamic resource allocation method to defend against DDoS assaults in SDN against specific cloud customers. When a DDoS attack happens, they leverage the idle cloud resources to swiftly create enough intrusion prevention servers for the victim to guarantee the QoS for legitimate users while simultaneously filtering out attack packets. The authors develop a mathematical model based on queueing theory, M/M/1 queueing, to approximate the requirements of their resource investment and detect the attack. The authors conclude that they can mitigate DDoS attacks in a cloud setting after carefully analyzing the system and doing experiments with real-world data sets.

2.5.2.3 ML-based Detection

ML-based techniques such as Bayesian networks, SVM, random forest, and decision trees are widely used to distinguish attack traffic from normal network traffic [111][112][113]. These algorithms are efficient and implemented extensively in both wired and wireless networks [113]. These techniques consider the different characteristics of the network and the traffic to detect the existence of attacks.

ML-based techniques are preferred to detect malicious activity based upon the network's abnormal behavior since they have the potential to self-learn, enabling them to adapt to the traffic flow and recognize malicious flow.

They depend on the training dataset to detect malicious behaviors using an already available attack dataset. The dataset contains records for normal traffic in addition to attack traffic. The training process is used to learn a classifier to distinguish attack traffic from normal one based on specific features extracted from the dataset. Hence, an ML-based technique typically differentiates network flows based on such traffic-related features and classifies them as malicious and normal [55][114].

ML may involve many popular methods that are described as follows [13], [20]–[22], [115], [116].

1. Naive Bayes: It's a classification method based on the Bayes Theorem and the assumption of predictor independence. A Naive Bayes classifier, in simple terms, assumes that the existence of one feature in a class is unrelated to the presence of any other feature. The text categorization industry is the primary focus of Naive Bayes. It's mostly used for clustering and classification purposes, and it's based on the conditional probability of occurrence. Naive Bayes is fast, but their used features must be independent [20][115][116].
2. SVM: SVM is an efficient ML classification algorithm. It uses a dataset to perform training, with each record labeled with one of two classes. SVM extracts fields from training records. It creates a classifier that can categorize a new data record. SVM selects the extreme vectors (support vectors) that support creating the hyperplane. It is fast and preferable for classification into two labels, complex problems, and data with large dimensions. Also, it requires less memory space and is less prone to the overfitting problem. However, choosing a convenient kernel is complex, besides, SVM results may be unclear [13], [20]–[22].

3. KNN: KNN is another ML-based method. It classifies the instance by considering its neighbors. The specified class is chosen due to being the most frequent among its K (an integer number) nearest neighbor. On the advantages side, KNN implementation is easy, effective, and robust. On the disadvantages side, it suffers from the high cost of its calculations, and K's value must be determined [20][22].
4. Decision Tree: The decision Tree technique is an algorithm that takes data of attributes along with its classes and generates a set of rules that may be used to identify the data. It creates a model that looks like a tree form. This method is simple and can deal with data that is either numerical or categorical. Unfortunately, the generated tree can be complex, and unstable [13], [20], [21].
5. Random Forest: It is an ensemble method that may be used with decision trees. These decision trees may be created while the training process and the result may be regression or classification. This algorithm reduces the classifier's overfitting problem. However, it is slow and its implementation is difficult [13], [20]–[22].

The authors in [26] propose a solution based on Self-Organizing Maps (SOM). They gather flow statistics from forwarding devices. The parameters used for training SOM are considered from the switch perspective, not the controller view. They include the average number of packets in the flow, average bytes in the flow, average duration in the flow, percentage of pair flows, the growth of a single flow, and growth of single ports. This technique violates SDN principles since it built intelligence in the data plane and the efficiency has to be enhanced.

A technique that detects DoS attacks that runs on top of the NOX controller is presented in [27]. The technique is called Sguard and has two modules, the access control, and the classification operation. The first module uses authorization information to take protective steps to locate the genuine source of a packet. Information is collected, such as source Medium Access Control (MAC) address, source IP address, source port number, and switch ID. The second module executes an artificial neural network algorithm, which is SOM to decide the class of network traffic, using a feature vector. The technique is lightweight, and effective in detecting the attack but it suffers from a large training time and low accuracy.

A DoS detection method that has two stages is implemented in [23]. In the first stage, the packet rate is computed using flow statistics. The ML-based model is the second stage. This approach is effective but suffers from a long detection time and low accuracy.

The authors in [24] provided a method for detecting anomaly SDN streams in SDN architectures. They used the DPTCM-KNN method [24] as the core algorithm of flow classification and attack detection. The technique is effective but has low accuracy and classification performance.

The authors in [32] use function virtualization in SDN to reduce the effects caused by DDoS attack traffic. The detector has three main components, which are a trigger of attack detection, detection of an attack, and attack backtracking. The algorithm is efficient, but has large time complexity, low performance, and lacks stability.

A technique that detects SDN DDoS attacks is introduced in [33]. The algorithm depends on K-means++ [33] and Fast K-Nearest Neighbors (K-FKNN). The technique is efficient but it has low performance, it takes long durations of time to classify and detect the attack, and it puts high loads on

SDN resources.

An SDN DDoS detection method with two parts is implemented in [34]. In the first part, the traffic is analyzed and the second part includes filtering, wrapping, and embedded-based feature selection methods. The technique has low accuracy and performance.

In [28], an LSTM-FUZZY SDN DDoS attack detection technique with mitigation is implemented. It has three phases: characterization, anomaly detection, and mitigation. The modular design of the system allows the maintenance and adaptation of other techniques to characterize traffic, detection, and mitigation of anomalies in SDN. However, the system takes a large training time and memory and suffers from overfitting problems.

The authors in [36] present a method that uses Snort IDS and a deep learning-based model. They exploit information that is based on sFlow and adaptive polling. This technique gives promising results, although its detection performance and accuracy still have to be improved.

A framework for DDoS attack detection and defense against SDN data planes is implemented in [29]. The authors build a DDoS trigger system and employ an ML method based on K-Means and KNN. The controller then defends against the attacks. They present a brand-new DDoS detection framework, but the accuracy needs enhancement.

The authors in [30] use ML algorithms to detect automated DDoS assaults against SDN data planes. The authors concentrate on TCP SYN food assaults and derive data plane DDoS detection features. They implemented a real-time DDoS detection system, but its accuracy has to be improved.

The authors of [37] exploit stateful data planes and allow switches to keep track of handled packets in persistent memory to detect SDN data planes DDoS attacks. The authors execute various ML-based classification algorithms.

StateSec [38] is a unique technique developed for protecting stateful SDN data planes against DDoS attacks. StateSec relies on in-switch processing capabilities. When compared to sFlow, StateSec is more efficient. However, to integrate this entropy-based detection mechanism into the switch, the authors need to adapt StateSec to the broader in-switch processing abstraction. The technique needs to include more sophisticated mitigation strategies.

The authors in [25] implemented FloodDefender which is a scalable and protocol-agnostic protection mechanism for OF networks against SDN-based DoS attacks. FloodDefender constructs a robust data plane flow table by dividing the flow table into "flow table region" and "cache region". FloodDefender enhances the utilization of flow tables, time delay, and packet loss rate. However, it needs a high cost to be deployed.

The authors in [31] offered OverWatch, which is an SDN-based high-efficient cross-plane DDoS attack defense framework with collaborative intelligence. It divides defense functions across the data and control planes. Experiments have shown that OverWatch is capable of high-precision detection and real-time defense response. However, it places a significant overhead on the data plane as well as the control plane.

This chapter provides a review of SDN and related work of DDoS detection using ML and traffic modeling. The next chapter explains the proposed ML-based detection technique.

Chapter 3

Machine Learning Technique for Detecting Distributed Denial of Service Attacks

3 Machine Learning Technique for Detecting Distributed Denial of Service Attacks

3.1 Introduction

In this chapter, a proposed technique is proposed to detect DDoS attacks with high accuracy using ML-based techniques in the SDN architecture. The proposed technique depends on developing proposed features that are obtained from traffic flow information and statistics. The developed model is trained with the kernel Radial Basis Function (RBF). The technique uses advanced features such as unknown destination addresses, packets inter-arrival time, TLP header, and ToS header. The utilized features are not used by the most recent research works. Also, we use features from related works such as average packet payload length (L_{avg}) and average number of packets of flow (ABF). The features are used to create a dataset that is used as an input to the linear SVM classifier. The classifier is used to train the model with the RBF kernel. As a mitigation procedure, the introduced technique blocks compromised end systems that send the attacking packets and stores various information and statistics about them. Additionally, the controller installs proposed rules that drop any packet that has the same flow as the packet flow, to avoid future attack attempts.

The current chapter is outlined as follows. Section (3.2) describes the proposed DDoS detection technique. Section (3.3) provides the performance measures of the proposed technique compared with other techniques.

3.2 The Proposed DDoS Detection Technique

The proposed DDoS Detection technique is based on the ML approach. We use some efficient ML-based techniques such as Naive Bayes, KNN, Decision Tree, and Random Forest, in addition to SVM to create classification models [13], [20]–[22]. The proposed technique uses advanced proposed traffic-based flow features to create the model. The generated model can classify SDN traffic flow packets as normal or DDoS attacks. Features are gathered from traffic flow headers, information, and statistics.

The proposed methodology steps are illustrated in the following subsections.

3.2.1 Generating Normal and DDoS Attack Traffic

The proposed technique begins with creating packets that have the necessary header fields. Some of the generated packets are normal traffic packets and others are a type of DDoS attack traffic. The headers, besides the resulting flow statistics, can be used as features to detect a DDoS attack.

For the detection technique in the control plane, some of the features are new and others are from previous works. The advanced proposed features are the following:

1. The destination IP address: In the case of a normal traffic packet, the destination IP address is usually well known to the controller. But, in the case of DDoS traffic, the destination IP address can be unknown. The attacker can plan to transmit many packets in a short time duration that are directed to unidentified hosts. Hence, the controller receives more packets asking for service. In this case, the SDN controller’s resources may be hung down, which makes the controller not available to valid Packet-In messages. We can trace the time duration between the arrivals of sequential packets, which all have unknown destination addresses. When the values of these inter-arrival times are small enough, this will indicate a DDoS attack.

2. The inter-arrival time between successive packets: In the normal case, the value of this parameter would be relatively large (e.g., 0.1 seconds or more) since, within the normal state, the traffic is usually not large and does not overwhelm the network. In the opposite case, for a DDoS attack, there is a flooding of packets that are transmitted over the network targeting its components, especially its controller. For this reason, the inter-arrival time between every two successive packets is usually small (such as 0.01 seconds) in the case of DDoS traffic.
3. Type of Service (ToS) header field: This attack tends to specify the ToS header field of the attack packets to high values such as the value ranging from 4 to 7. This procedure may enable these packets to be served in higher precedence than other normal traffic packets.
4. Transport Layer Protocol (TLP) header field: A DDoS attack may use the TLP header field which is one of the network layer headers to send DDoS attack traffic. The attacker specifies this value with null instead of using a protocol such as "TCP", "UDP", or "ICMP". The attacker acts this behavior to enable the attack packet to avoid security procedures implemented within "TCP" and "UDP". When the target host exposes this attack, its resources would be frequently exhausted. This DDoS attack is called IP null attack [117].

The features used in related works include the following:

1. Average payload length of packets (L_{avg}): The size of the payload part of the packet can be used as a feature for the existence of a DDoS attack when it is small such as 120 bytes [118]. To increase its load efficiency and harmfulness, a DDoS attack usually creates small-sized packets, so that it can create the largest possible number of packets in a small interval of time

[118]. We compute the average payload lengths of the packets that have the same source IP address and are sent within a specific interval of time using the following equation.

$$L_{avg} = \sum_{i=1}^{N} \frac{L_i}{N} \tag{3.1}$$

where, L_{avg} is the average payload length of packets with the same source IP address, L_i is the payload length of packet i, and N is the number of flows.

2. Average number of packets of flow (ABF): The DDoS attack usually tries to hide its identity on the internet by spoofing the source IP address. In this case, the source IP addresses of the attack packets vary continuously, which complicates the mission of tracking the attack's original address. According to the definition of a flow, variation in the source IP address field would result in generating a new flow. Hence, in the case of a DDoS attack, the average number of packets that belong to a specific flow in the switch would be small (e.g. 3 packets) [118] and is calculated using equation (3.2).

$$ABF = \sum_{i=1}^{N} \frac{M_i}{N} \tag{3.2}$$

where ABF is the average number of packets in a specific flow, M_i is the number of packets in flow i, and N is the number of flows.

The previous features are used to create a dataset for the control plane DDoS detection that includes 50,000 records for normal traffic in addition to 50,000 records for attack traffic.

For the detection technique in the data plane, the advanced proposed features are the following:

1. Switch stored capacity: This property can be used as a feature for detecting DDoS attacks against OF switches in the SDN data plane. In a DDoS attack, an attacker can send many packets toward an OF switch. In this scenario, an OF switch must store the packets in its flow buffers, until the reply of the controller reaches. This can quickly fill up the switch's limited storage capacity, which makes the switch’s buffers unavailable to legitimate users’ packets. The stored capacity of the switch can be calculated as.

   $$C = \sum_{i=0}^{N} S_i \qquad (3.3)$$

   where C is the switch stored capacity, N is the number of arriving packets in the switch, and S_i is the packet size for packet i.

2. Average rate of packets with unknown destination IP addresses: The attacker can target sending packets with unknown destination IP addresses so that the switch cannot serve these packets directly. Instead, the switch must store these packets temporarily in its storage buffers waiting for a response from the SDN controller. This may typically exhaust the storage capacities of the data plane switch and hang it down denying its service to normal users’ traffic. In addition, the controller replies to the switch with suitable new flow rules which need to be added to the flow table of the switch. This may fulfill the storage capacity of the switch’s flow table rapidly which makes it unavailable for legitimate new flows.

3. IP Options header field: A DDoS attacker may target increasing the header length of a packet so that the packet occupies more space in the switch’s buffer. This typically helps in fulfilling the free space of a switch rapidly making it unavailable to legitimate packets. The attacker can achieve this

using the IP Options field of a packet header by utilizing many routing options such as requesting a specific routing path for the packet.

4. Average number of flows: When the attacker floods the data plane switch with high attack traffic, the average number of flows in the switch flow buffers increases sharply. Therefore, this may be used as a feature for DDoS attacks against the data plane in the SDN architecture.

The features used in related works include ToS header, TLP header, ABF, and Average arrival rate.

The previous features are used to create a dataset for the data plane DDoS detection that includes 50,000 records for normal traffic in addition to 50,000 records for attack traffic.

3.2.2 Proposed Model

The proposed DDoS detection model is created utilizing the dataset which has both normal and DDoS attack traffic packets. The dataset is divided as 70 % of it is utilized to train the classifier and the other 30 % is used to test it. It is mainly structured as a table, which has the number of created packets as the number of rows, and the number of features plus the classification label as the number of columns.

Many efficient ML-based techniques are customized and used to create classification models. For instance, we use SVM, which seeks to obtain the best dataset generalization. Therefore, it looks for the optimal hyperplane that can accomplish that [119][120]. It creates a model that may determine whether a new data entity belongs to some class or not. Assume having a dataset S. $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ where, $x_i \epsilon R^n$, $y \epsilon \{-1, +1\}$, and x_i is the entered transformed vector and y_i is the output. SVM has only two class labels which are represented as -1 and +1. SVM designs the hyperplane, which is optimal, so that it can decouple the input vectors into two different entities. The hyperplane is described by the next equation according to [119][120].

$$x_i \in R^n: (\vec{w}:\vec{x}) + b = 0, \vec{w} \in R^n, b \in R \tag{3.4}$$

SVM should look for the hyperplane, which results in the largest possible distance between training samples as:

$$f(\vec{x}) = sign(\vec{w}:\vec{x}) + b \tag{3.5}$$

SVM considers that the data can be separated into two categories. Therefore, the optimal hyperplane may be mixed by the following inequality.

$$y_i\{(\vec{w},\vec{w}) + b\} \geq 1, s.t. i = 1, \dots, n \tag{3.6}$$

Hence, the problem of optimization may be represented in:

$$minimization \frac{1}{2}(w^T, w) s.t. y_i(w.x + b) \geq 1 \tag{3.7}$$

Inversely, whether data cannot be separated, the optimization is shown by:

$$minimization \frac{1}{2}(w^T, w) + c \sum_{i=1}^{n} \xi_i s.t y_i(w.x + b) + \xi_i \geq 1; \xi_i \geq 0 \tag{3.8}$$

Where ξ_i is the slack operand which enhances choosing the best hyperplane, and c is the cost magnitude which represents the regularization factor. The user may select the value of c on trial.

We utilize a linear Support Vector Classifier (SVC) with the RBF [121] as a kernel to determine the optimal decision border that can separate 6-dimensional space into two classes. The functionality of the RBF kernel for two entities x_1 and x_2 is to calculate the closeness between them. It is described in:

$$K(x_1 + x_2) = \exp\left(-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right) \tag{3.9}$$

where, σ^2 is the variance, $(x_1 - x_2)$ is the Euclidean distance between x_1 and x_2.

RBF Kernel is well-known since it is similar to KNN Algorithm [20]. It has the merits of KNN and solves the problem of needing a high space cost. While the training operates, SVM's RBF Kernel must maintain only the support vectors instead of the dataset. RBF Kernels have two hyper parameters, C for SVM and γ for the RBF Kernel. Here, γ is inversely proportional to σ as:

$$\gamma \propto \frac{1}{\sigma^2} \tag{3.10}$$

In addition to SVM, techniques such as Naive Bayes, KNN, Decision Tree, and Random Forest are also utilized and compared. SVM is the preferred one to be used later for testing operations since it has the highest accuracy. The proposed model for DDoS detection using SVM is expressed in figure (3.1).

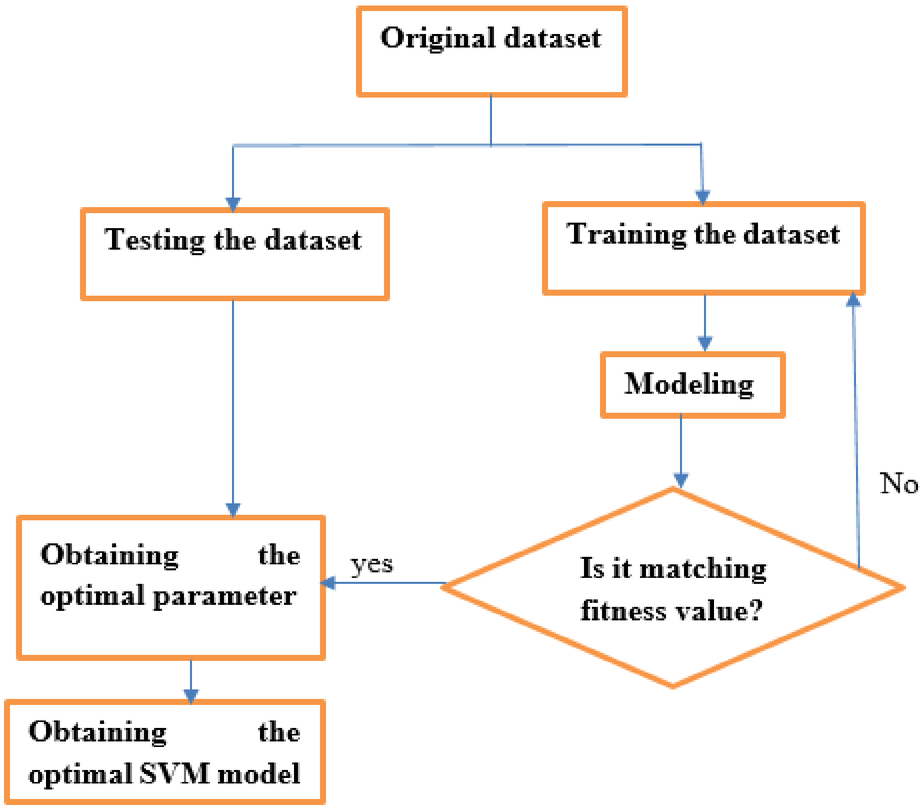

Original dataset
Testing the dataset
Training the dataset
Modeling
No
Is it matching fitness value?
yes
Obtaining the optimal parameter
Obtaining the optimal SVM model

Figure 3.1: Flow diagram of the proposed SVM model

3.2.3 Detection Process

In this subsection, the detection process of the proposed models in the control plane and data plane is described.

For the control plane detection, the controller receives an OF packet from the switch. First of all, the source IP address of the ongoing flow packet is extracted and compared against IP addresses in a blacklist. If this address exists, the packet will be dropped and valuable statistics such as the number of times this source address sent a packet, the time stamp, and other valuable information are logged and taken into account. Conversely, if there is no match between the source address and any element of the blacklist, the packet headers along with the necessary statistics are used as features that are entered as input to the classifier for testing purposes. If the classifier decides that the incoming packet is normal, this packet will be accepted and served in a normal manner. On the other side, if the newly arrived packet is classified as a DDoS attack, the compromised sending end system can be added to a blacklist accordingly. Also, packet information is recorded. Besides, various traffic flow statistics are logged, such as packet arrival time stamps, as well as the number of packets sent from the same compromised system. Figure (3.2) shows a flow chart for the control plane detection technique. Algorithm #1 describes the proposed technique. The description of each component of the proposed detection technique is as follows:

Packet-in message receiver: This component is responsible for receiving Packet-In messages from OF switches.

Headers reader: It reads information from various packet headers.

Features extractor: This component applies calculations on the packet and also various traffic statistics to extract features.

SVM model: It runs the classifier on the obtained features to determine whether the attack exists or does not.

Normal service module: It runs when the traffic is normal to serve the packet.

Mitigation module: It means blocking the compromised end system, logging its information, and installing flow rules to drop all incoming packets in the future with the same flow.

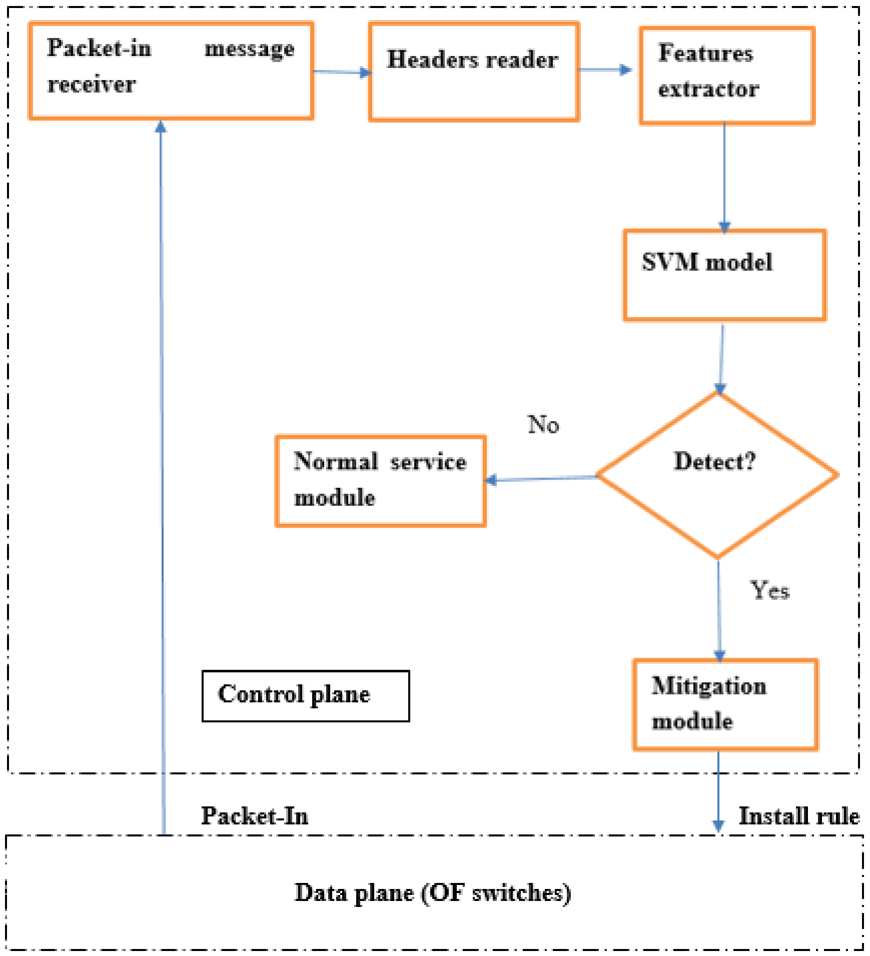


Figure 3.2: Flow diagram of the control plane detection technique

Algorithm 1 Proposed DDoS Detection Technique

Input: Packet-In message

Output: Attack Detection

```
1. while true do
2.     Packet-In message is sent by an OF switch to the controller
3.     Extracts source IP address from the packet
4.     if (source IP address matches an entry in DDoS attack's blacklist)
       then
5.         Log necessary statistics about the packet and its source IP address
6.         Set flow rules to drop the packet, and send them to suitable switches
7.     else
8.         Store packet arrival time for upcoming calculations
9.         Extract packet's destination IP address, port numbers, transport
           layer protocol, type of service, and payload length headers
10.        Obtain features by performing necessary calculations using
           information extracted from headers and traffic information
11.        Input features to the SVM model
12.        if (The model detects a DDoS attack) then
13.            Add the source IP address to the blacklist
14.            Log necessary statistics about the packet and its source IP
               address
15.        else
16.            Serve the packet normally and store necessary statistics about it
17.        end if
18.    end if
19.end while
```

For the data plane testing, a packet arrives at the switch which looks for the source IP address in a blacklist. If a match occurs, the switch drops the packet, and its information is stored. Conversely, if no match is found, the packet headers are extracted. Then, extracted headers are used to perform effective calculations to obtain data plane detection features. The obtained features and relevant statistics are entered as input to the classifier for testing. Figure (3.3) shows a flow chart for the data plane detection technique. Algorithm #2 describes the proposed technique. The description of each component of the proposed detection technique is as follows:

Packet arrival: This component is responsible for receiving a packet by an OF switch.

Drop packet: It drops the receiving packet if it is sent by a compromised end system.

The explanations of the remaining components are similar to the corresponding components in the control plane detection.

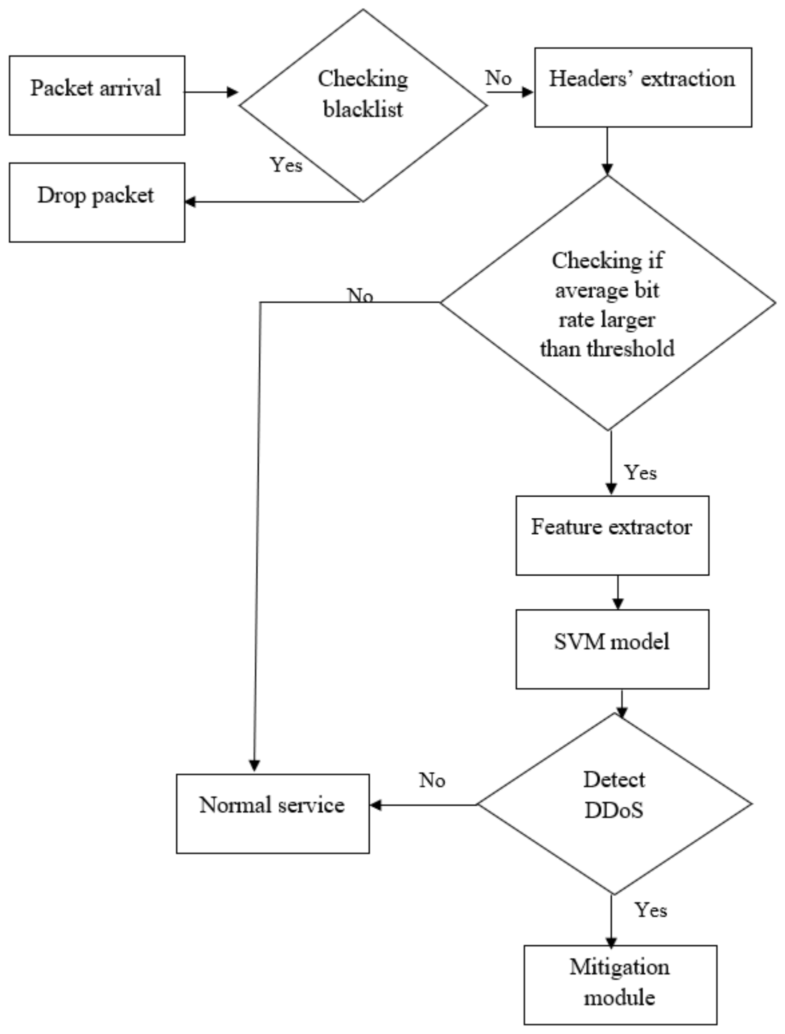


Figure 3.3: Flow diagram of the data plane detection technique

Algorithm 2 Proposed data plane DDoS detection technique

Input: Packet arrival to switch

Output: DDoS attack detection

```
1. while true do
2.   packet arrival to an OF switch
3.   Extract source IP address of the packet
4.   if (packet's source IP address matches an entry in a blacklist)
5.     log necessary packet's statistics and information
6.     drop the packet
7.   else
8.     store packet arrival time for upcoming calculations
9.     extract packet's destination IP address, IP Option, Type of service,
       Transport layer protocol headers
10.    calculate average arrival bit rate for switch traffic with unknown destination
IP address
11.    if (average arrival bit rate is less than a threshold)
12.      serve packet normally
13.    else
14.      obtain features by performing necessary calculations using information
extracted from headers and traffic flow statistics
15.      input features to the detection classifier
16.      if (the detection model detects DDoS attack)
17.        execute the third detection phase by decreasing the trust value of a packet
sender
18.        add the sender to the blacklist and log its information when the trust
becomes zero
19.      else
20.        serve packet normally and log valuable statistics about it
21.      endif
22.    endif
23.  endif
24. endwhile
25. For each time duration $t$
26. For each sender $s$ in trust list $tv$
27.     if (sender $s$ does not send in time slot $t$)
28.      update its trust by increasing its value
29.     endif
30.  endfor
31. endfor
```

3.3 Performance Measures

To track the performance of the technique, we should consider the following.

1. True Positive (TP): It is the number of attack packets that are truly considered an attack.
2. True Negative (TN): It is the number of normal packets that are truly classified as normal traffic.
3. False Positive (FP): It is the number of normal packets that are considered by fault as attack traffic.
4. False Negative (FN): It is the number of attack traffic packets that are classified by fault as normal.

There are many performance measures in network security, which are used to measure the performance of the introduced technique as follows.

1) Classification accuracy:

Classification accuracy is the fragment of the number of correctly classified packets to the aggregate number of packets.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad (3.11)$$

2) Detection rate, hit rate, recall, or true positive rate (TPR):

The detection rate is the fraction of the number of attacking packets that are truly classified as an attack to the aggregate number of attack packets.

$$Recall = \frac{TP}{TP + FN}$$

(3.12)

3) False Positive Rate (FPR) or False Alarm Rate:

FPR is the fraction of the number of normal packets that are considered by fault as an attack on the aggregate number of normal packets.

$$FPR = \frac{FP}{FP + TN} \tag{3.13}$$

4) Precision:

Precision is the fraction between the number of attacking packets that are correctly classified as attacks and the total number of packets that are classified as attacking packets.

$$Precision = \frac{TP}{TP + FP} \tag{3.14}$$

5) F1 score, F-score, or F_measure:

F_measure tracks the accuracy based on the combination of precision and recall.

$$F\_measure = \frac{2 \times precision \times recall}{precision + recall} \tag{3.15}$$

SDN researchers always seek to implement security techniques that can get the highest possible detection accuracy, detection rate, precision, and f_measure. Conversely, the false alarm rate is preferred to be as small as possible.

This chapter explains the proposed ML-based detection technique. The next chapter expresses the proposed traffic queuing models and their analysis.

Chapter 4

A Seven-Dimensional State Flow Traffic Modelling for Multi-controller Software-Defined Networks Considering Multiple Switches

4 A Seven-Dimensional State Flow Traffic Modelling for Multi-controller Software-Defined Networks Considering Multiple Switches

4.1 Introduction

As stated earlier in chapter (2), modeling a network will support analyzing and tracking the traffic and its performance [89]. Moreover, it is valuable to detect DDoS attacks in SDN using traffic modeling and analysis [90][91].

Many studies considered developing analytical models for evaluating the performance of TCP and UDP flows over SDN [40][25][122][123]. Modelling TCP and UDP flow traffic in SDN can help in active queue management (AQM). Also, it can support network control and monitoring because it enables network administrators to differentiate traffic based on TCP and UDP flows. This can help to choose either TCP or UDP based on the type of network applications that are used in the network. In addition, developing mathematical modeling for TCP and UDP flows can help to predict the traffic load on the SDN controller and manage it accordingly [61][124].

The approaches that use queueing models to track SDN traffic behavior [39]–[52] have many advantages and disadvantages. They have the following advantages: (1) Developing SDN queueing models using 3-D [39] or 4-D states [40], and (2) Measuring the behavior of the constructed models using average packet delay and loss probability [39], [40], [45]–[48]. However, they have the following limitations: (1) They usually do not consider multiple controllers and multiple switches [39]–[52], (2) They suffer from lacking network scalability, backup capabilities, and network security [39]–[52], (3) They usually do not consider using 7-D states [39][40], and (4) They usually use queues with infinite storage capacity [45]–[49].

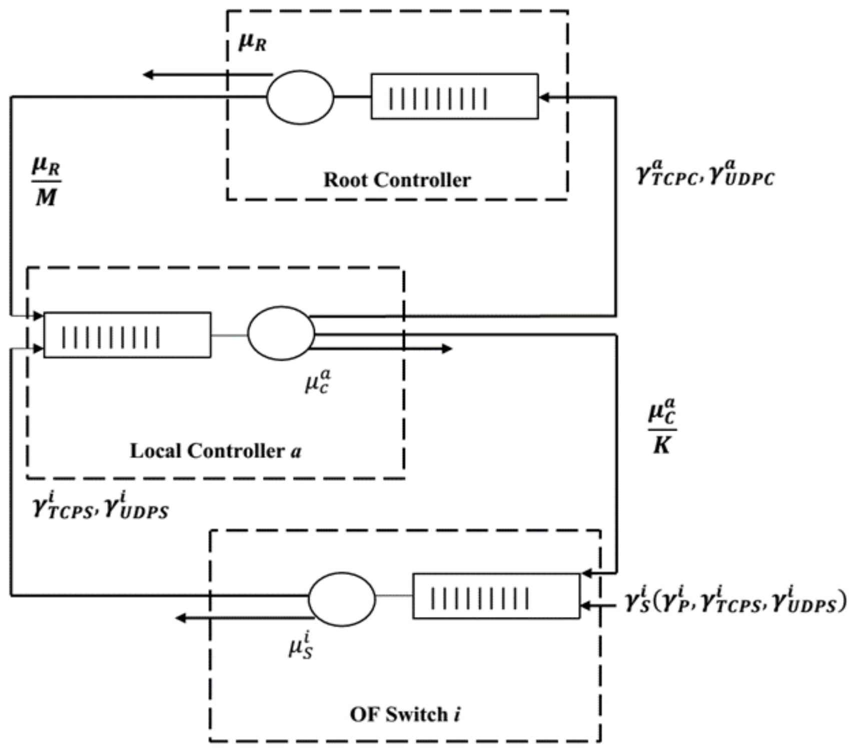
In this chapter, two traffic analysis models for TCP and UDP flows in SDN with multiple OF switches and multiple controllers are developed. Unlike models constructed in related works which assumed the network to have a single controller or a single switch [39]–[52]. Unlike queueing models in related works [45]–[49], the developed models consider all controllers and OF switches to have finite buffer capacities, which is the normal case in a real SDN environment. In both models, the mean transfer delay and the loss probability of a packet are tracked to measure the performance of a network. The proposed study uses 7-D states to complete the models. The 7-D state is effective and scalable, which makes it superior to the 3-D state [39] and 4-D state [40]. The proposed models can be used roughly to differentiate normal traffic from DDoS attack traffic by considering DDoS attack traffic has Poisson input distribution (M/M/1) with high average arrival rates.

Network scalability is improved through the suggested developed models by using multiple OF switches, which can provide network services to large numbers of end systems. Also, network security is enhanced by using multiple SDN controllers in the developed models. Using multiple controllers prevents the single point of failure problem caused by using one controller. When multiple controllers are utilized, they can provide backup capabilities to the network. This can increase network security, guarantee network stability, and allow the network to function properly even when one controller drops. All controllers and switches have limited-size queues.

The validity of the proposed analytical models is demonstrated by performing experimental simulations and comparing them against analytical models. This can help in measuring the performance of the constructed models using performance measures of traffic modeling and queueing such as the average packet delay and the packet loss probability. The proposed models are shown to roughly differentiate normal traffic from DDoS attack traffic by considering the attack traffic has a Poisson input distribution with high average arrival rates. The obtained results can show that the proposed modeling is accurate compared to the recent work.

4.2 Generic Queuing Model for SDN Traffic Considering Multiple Controllers and Multiple Switches

Figure (4.1) illustrates the general SDN queuing model for traffic considering TCP flows as well as UDP flows when the network has multiple controllers and multiple OF switches. The system model consists of a control plane as a Kandoo architecture [82] that has a root controller, and M local controllers, in addition to the data plane which has K switching nodes with each local controller. Each subnetwork consists of K switching nodes in its forwarding plane and a single local controller representing its control plane. The notations used in the model are described in table (4.1). The developed model can roughly represent DDoS attack traffic, in which a high packet arrival rate is transmitted through the network with small-time durations between successive packets. Therefore, through the model, when a zombie system sends a packet to an OF switch, the switch searches its flow table for a flow ID match with the packet. If the switch can find such a match, it executes the corresponding rules. Conversely, in the case of missing such a match, the switch forwards the packet to the local controller in its local network, which is the controller responsible for it. If the local controller succeeds to process the packet, it installs new flow rules on the suitable switches and returns the packet to the switch. But if the local controller fails to process the packet, it forwards the packet to the root controller, which replies to it by the corresponding flow rule. At this moment, the local controller adds an entry for the packet's flow in its flow tables, installs new flow rules in the relevant switches, and replies to the switch with the packet.



μ_R
Root Controller
$\frac{\mu_R}{M}$
$\gamma^a_{TCPC}, \gamma^a_{UDPC}$
μ^a_c
Local Controller *a*
$\frac{\mu^a_C}{K}$
$\gamma^i_{TCPS}, \gamma^i_{UDPS}$
$\gamma^i_S(\gamma^i_P, \gamma^i_{TCPS}, \gamma^i_{UDPS})$
μ^i_S
OF Switch *i*

Figure 4.1: Queuing model of SDN with multiple controllers and switches for TCP and UDP flows

Table 4.1: Notations used in the constructed models

λ^i_S	Average arrival rate to switch *i* from end systems
α^i_S	Total rates sent to switch *i* from other switches during routing processes
γ^i_S	Total arrival rate to switch *i*

γ_{TCPS}^{i}	Average TCP flow arrival rate to switch *i*
γ_{UDPS}^{i}	Average UDP flow arrival rate to switch *i*
R_{ij}	Probability of routing from node *i* to node *j*
γ_{P}^{i}	Packet average arrival rate per flow at switch *i*
γ_{TS}^{i}	Total TCP arrival rate to switch *i*
γ_{US}^{i}	Total UDP arrival rate to switch *i*
μ_{S}^{i}	Average service rate at switch *i*
μ_{TCPS}^{i}	Average TCP flow termination rate in switch *i*
μ_{UDPS}^{i}	Average UDP flow termination rate in switch *i*
N_{TCPS1}^{ia}	Number of TCP flows in switch *i* sending NF packet to controller *a*
N_{TCPS2}^{i}	Number of other TCP flows in switch *i*
N_{UDPS1}^{ia}	Number of UDP flows in switch *i* sending NF packet to controller *a*
N_{UDPS2}^{i}	Number of other UDP flows in switch *i*
K_{S}^{i}	Switch *i* maximum queue size
q_{S}^{i}	Switch *i* queue length
γ_{TCPC}^{a}	Average new TCP flow arrival rate to local controller *a*
γ_{TCPC2}^{a}	Average known TCP flow arrival rate to local controller *a*
γ_{TC}^{a}	Total TCP arrival rate to local controller *a*
γ_{UC}^{a}	Total UDP arrival rate to local controller *a*
γ_{UDPC}^{a}	Average new UDP flow arrival rate to local controller *a*

γ_{UDPC2}^{a}	Average known UDP flow arrival rate to local controller *a*
μ_C^a	Average service rate in local controller *a*
N_{TCPC1}^{a}	Number of TCP flows in local controller *a* sending requests to root controller
N_{TCPC2}^{a}	Number of other TCP flows in local controller *a*
N_{UDPC1}^{a}	Number of UDP flows in local controller *a* sending requests to root controller
N_{UDPC2}^{a}	Number of other UDP flows in local controller *a*
K_C^a	Local controller *a* maximum queue size
q_C^a	Local controller *a* queue length
λ_{TR}	Average TCP arrival rate in the root controller
λ_{UR}	Average UDP arrival rate in the root controller
μ_R	Average service rate in the root controller
K_R	Root controller maximum queue size
q_R	Root controller queue length
R_{Trans}	Transition rate from a state to another state in a model
D_{Prop}	Propagation delay between any two nodes

The following assumptions are utilized by the developed queuing models.

- Assume the notation *i* is used for numbering an OF switch. *i* = 1, 2, ..., K. Also, the notation *a* is used for numbering a local controller. *a* = 1, 2, ..., M.
- The 7-D state $(N_{TCPS1}^{ia}, N_{TCPS2}^{i}, N_{TCPC1}^{a}, N_{TCPC2}^{a}, q_S^i, q_C^a, q_R)$ is used for the model of TCP flows.
- For the UDP model, the 7-D state $(N_{UDPS1}^{ia}, N_{UDPS2}^{i}, N_{UDPC1}^{a}, N_{UDPC2}^{a}, q_S^i, q_C^a, q_R)$ is used.
- Each OF switch has a queue, which is limited in capacity.

- Each controller (including the root controller) has its independent limited-capacity queue.
- Each queue is represented as M/M/1/m model. Therefore, whether the traffic is normal or DDoS attack, the TCP/UDP flow arrivals besides the packet arrivals within each flow follow a Poisson process. Besides, the TCP/UDP flow duration and packet service time follow an exponential distribution at the controllers and OF switches. The main difference between normal and DDoS attack traffic is the value of average arrival rates, which is very large in the case of an attack.
- For both TCP and UDP flows in both normal and attack traffic, the packet that belongs to a new flow is called an NF (New Flow) packet, but the packet that belongs to an existing flow is called a DF (Defined or Data Flow) packet.
- All TCP flows and UDP flows are considered new at both the switch and the local controller.
- The arrival rates to all switches, as well as requesting rates to all local controllers are balanced.
- No DF (data) packets enter the switch in the case of TCP flows before the first (NF or SYN) packet has been processed. This assumption is legitimate because it matches the three-way handshake process used to create actual TCP connections. A TCP connection is established before sending any data packets. The data packets cannot be sent before sending the SYN packets. Then the new flows in TCP contain SYN packets only and it is impossible to be data packets then the new rules are associated with the SYN packets in the TCP model. Then for every new flow in TCP, one packet (SYN) is directed to the controller because they found no match in the switch table.

- Additional DF packets may reach the switch in the case of UDP flows before the controller has finished processing the first packet of the flow. Since UDP has no three-way handshaking and it is a connectionless protocol with the only purpose of sending data to the destination as soon as possible. In the UDP model, there are no SYN packets. The sender can start the transmission by sending more than one packet back-to-back. The new flows do not contain SYN packets and can contain more than one data packet sent back-to-back. Then all the packets in the new flow which can be more than one packet are directed to the controller because they found no match in the switch table. Therefore, UDP sends packets of the same flow back-to-back without waiting for an acknowledgment.

In the case of a DDoS attack, the value of the total arrival rate at switch i, γ_S^i can be very large. As mentioned before, the compromised end systems flood victims by large numbers of packets sent with small-time durations between them. To compute the total arrival rate, to OF switch *i*, γ_S^i, we should consider the total rates that are sent to switch *i* from other K OF switches in its network during routing operations, α_S^i as shown in

$$\alpha_S^i = \sum_{j=1, j\neq i}^{K} R_{ji}\gamma_S^j \tag{4.1}$$

As well as considering the average arrival rate to switch *i* from end systems, λ_S^i. Therefore, equation (4.2) expresses the value of γ_S^i, as follows:

$$\gamma_S^i = \lambda_S^i + \alpha_S^i \tag{4.2}$$

In the developed models, γ_S^i, may contain γ_P^i, γ_{TCPS}^i, and γ_{UDPS}^i. When the value of γ_S^i is very large such that the traffic intensity value approaches 1, this indicates that the traffic can be a DDoS attack [41][42][62][63].

4.3 SDN TCP Model

In this subsection, a detailed explanation of the developed TCP model is provided. Firstly, transitions of the TCP seven-state are described, and then performance measures of the developed TCP model are proposed.

4.3.1 SDN TCP Model States

Figure (4.2) depicts the nine possible states (labeled 1, 2,..., 9) of state $(N_{TCPS1}^{ia}, N_{TCPS2}^{i}, N_{TCPC1}^{a}, N_{TCPC2}^{a}, q_S^i, q_C^a, q_R)$ in the SDN system model for TCP flows. The various transitions are described in the following.

- First transition: A packet with a known flow rule (DF packet) is only received by switch *i* and stored in the queue but switch *i* has not handled the packet yet. q_S^i is thus increased by 1. The transition rate is calculated as:

  $$R_{Trans} = \gamma_P^i . N_{TCPS2}^i \tag{4.3}$$

- Second transition: Switch *i* handles the stored packet in the first transition and sent it to the destination. We took into consideration the waiting time in the queue which represents the difference between the arrival time and the starting of the serving time. In such a state, the probability of finding such a match between the packet and an entry of the flow table is $(1 - \frac{N_{TCPS1}^{ia} - \frac{q_C^a}{K}}{q_S^i})$. Therefore, switch *i* sends the packet towards its destination. q_S^i is decreased by 1. The transition rate is formulated as follows.

  $$R_{Trans} = \left(1 - \frac{N_{TCPS1}^{ia} - \frac{q_C^a}{K}}{q_S^i}\right) . \mu_S^i \tag{4.4}$$

- Third transition: Switch *i* receives a packet of a new TCP flow. Switch *i* has no flow table entry that matches the received packet. Therefore, N_{TCPS1}^{ia} and q_S^i are incremented by 1. The transition rate is γ_{TCPS}^i.

- Fourth transition: An NF packet is forwarded to local controller *a* by switch *i*. Local controller a can process a packet successfully. q_S^i is thus decreased by 1. q_C^a is increased by 1. The transition rate can be determined as:

$$R_{Trans} = \frac{N_{TCPC2}^a \cdot \gamma_{TCPC2}^a}{K} \tag{4.5}$$

- Fifth transition: Local controller *a* can process a packet received from switch *i*. Hence, local controller *a* sends the packet back along with its suitable flow rules to switch *i*. The probability for local controller *a* to process a packet successfully is $(1 - \frac{N_{TCPC1}^a - \frac{q_R}{M}}{q_C^a})$. Therefore, N_{TCPS1}^{ia} and q_C^a are decreased by 1. N_{TCPS2}^i and q_S^i are increased by 1. The transition rate can be represented as:

$$R_{Trans} = \left(1 - \frac{N_{TCPC1}^a - \frac{q_R}{M}}{q_C^a}\right) \cdot \frac{\mu_C^a}{K} \tag{4.6}$$

- Sixth transition: An NF packet is forwarded to local controller *a* by switch *i*. Local controller *a* fails to find a flow rule for the packet. Therefore, N_{TCPC1}^a and q_C^a are increased by 1. q_S^i is decreased by 1. The transition rate is formulated as:

$$R_{Trans} = \frac{\gamma_{TCPC}^a}{K} \tag{4.7}$$

- Seventh transition: Local controller *a* cannot process a TCP packet. Local controller *a* fails to process the packet successfully with the probability $(\frac{N_{TCPC1}^a - \frac{q_R}{M}}{q_C^a})$. In this case, local controller *a* forwards a packet to the root controller. q_C^a is decreased by 1. q_R is increased by 1. The transition rate can be shown as:

$$R_{Trans} = \left(\frac{N_{TCPC1}^a - \frac{q_R}{M}}{q_C^a}\right) \cdot \frac{\mu_C^a}{K} \tag{4.8}$$

- Eighth transition: The root controller processes a TCP NF packet, and sends it back with the corresponding flow rule to local controller *a*. Therefore, N^{a}_{TCPC1} and q_R are decreased by 1. N^{a}_{TCPC2} and q^{a}_{C} are increased by 1. The transition rate is $\frac{\mu_R}{M}$.
- Ninth transition: Switch *i* has a TCP flow that terminates. N^{i}_{TCPS2}, is decreased by 1. The transition rate is depicted in

$$R_{Trans} = N^{i}_{TCPS2} \cdot \mu^{i}_{TCPS} \tag{4.9}$$

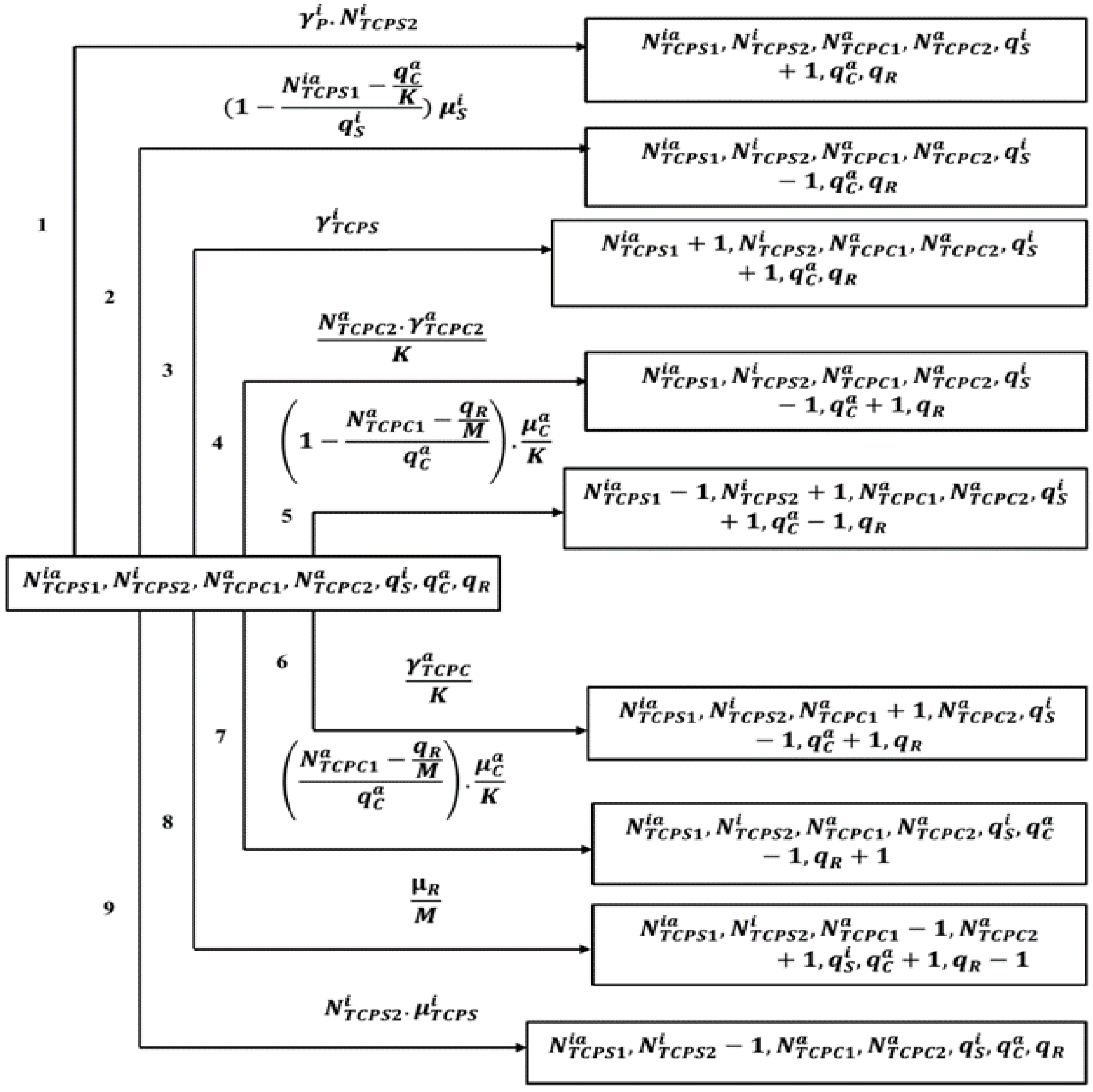

$\gamma^i_P \cdot N^i_{TCPS2}$
$N^{ia}_{TCPS1}, N^{i}_{TCPS2}, N^{a}_{TCPC1}, N^{a}_{TCPC2}, q^{i}_{S} + 1, q^{a}_{C}, q_R$
$(1 - \frac{N^{ia}_{TCPS1} - \frac{q^a_C}{K}}{q^i_S})\, \mu^i_S$
$N^{ia}_{TCPS1}, N^{i}_{TCPS2}, N^{a}_{TCPC1}, N^{a}_{TCPC2}, q^{i}_{S} - 1, q^{a}_{C}, q_R$
1
γ^i_{TCPS}
$N^{ia}_{TCPS1} + 1, N^{i}_{TCPS2}, N^{a}_{TCPC1}, N^{a}_{TCPC2}, q^{i}_{S} + 1, q^{a}_{C}, q_R$
2
$\frac{N^a_{TCPC2} \cdot \gamma^a_{TCPC2}}{K}$
3
$N^{ia}_{TCPS1}, N^{i}_{TCPS2}, N^{a}_{TCPC1}, N^{a}_{TCPC2}, q^{i}_{S} - 1, q^{a}_{C} + 1, q_R$
4
$\left(1 - \frac{N^a_{TCPC1} - \frac{q_R}{M}}{q^a_C}\right) \cdot \frac{\mu^a_C}{K}$
$N^{ia}_{TCPS1} - 1, N^{i}_{TCPS2} + 1, N^{a}_{TCPC1}, N^{a}_{TCPC2}, q^{i}_{S} + 1, q^{a}_{C} - 1, q_R$
5
$N^{ia}_{TCPS1}, N^{i}_{TCPS2}, N^{a}_{TCPC1}, N^{a}_{TCPC2}, q^{i}_{S}, q^{a}_{C}, q_R$
6
$\frac{\gamma^a_{TCPC}}{K}$
$N^{ia}_{TCPS1}, N^{i}_{TCPS2}, N^{a}_{TCPC1} + 1, N^{a}_{TCPC2}, q^{i}_{S} - 1, q^{a}_{C} + 1, q_R$
7
$\left(\frac{N^a_{TCPC1} - \frac{q_R}{M}}{q^a_C}\right) \cdot \frac{\mu^a_C}{K}$
8
$N^{ia}_{TCPS1}, N^{i}_{TCPS2}, N^{a}_{TCPC1}, N^{a}_{TCPC2}, q^{i}_{S}, q^{a}_{C} - 1, q_R + 1$
$\frac{\mu_R}{M}$
9
$N^{ia}_{TCPS1}, N^{i}_{TCPS2}, N^{a}_{TCPC1} - 1, N^{a}_{TCPC2} + 1, q^{i}_{S}, q^{a}_{C} + 1, q_R - 1$
$N^i_{TCPS2} \cdot \mu^i_{TCPS}$
$N^{ia}_{TCPS1}, N^{i}_{TCPS2} - 1, N^{a}_{TCPC1}, N^{a}_{TCPC2}, q^{i}_{S}, q^{a}_{C}, q_R$

Figure 4.2: State transition diagram of the SDN TCP model

4.3.2 Performance Measures of the Developed TCP Model

In this subsection, the related performance measures of the proposed TCP model are calculated to track its behavior. The loss probability and the average packet delay of a TCP packet are used as performance measures and are evaluated. Firstly, TCP traffic intensity in switch i, ρ_{TS}^{i}, TCP traffic intensity in local controller a, ρ_{TC}^{a} , and TCP traffic intensity in the root controller, ρ_{TR}, are calculated to be used in computing the packet loss probability and the average packet delay. Also, the mean queue length for switch i, $\overline{q_{TCPS}^{\iota}}$, and the mean queue length for local controller a, $\overline{q_{TCPC}^{a}}$, are calculated.

In addition, the mean queue length for the root controller, $\overline{q_{TCPR}}$, and the average number of TCP flows in local controller a sending requests to the root controller, $\overline{N_{TCPC1}^{a}}$, are calculated as well.

The traffic intensity in switch i, ρ_{TS}^{i}, is obtained as:

$$\rho_{TS}^{i} = \frac{\gamma_{TS}^{i}}{\mu_{S}^{i}} \tag{4.10}$$

The traffic intensity in local controller a, ρ_{TC}^{a}, is obtained as:

$$\rho_{TC}^{a} = \frac{\gamma_{TC}^{a}}{\mu_{C}^{a}} \tag{4.11}$$

The traffic intensity in the root controller, ρ_{TR}, is obtained as:

$$\rho_{TR} = \frac{\lambda_{TR}}{\mu_{R}} \tag{4.12}$$

The mean length of a queue for switch i, $\overline{q_{TCPS}^{\iota}}$ can be illustrated as:

$$\overline{q_{TCPS}^{\iota}} = \sum_{x=0}^{\infty} x\,.\,P\left(N_{TCPS1}^{ia}, N_{TCPS2}^{i}, N_{TCPC1}^{a}, N_{TCPC2}^{a}, q_{S}^{i} = x, q_{C}^{a}, q_{R}\right) \tag{4.13}$$

Where $P\left(N_{TCPS1}^{ia}, N_{TCPS2}^{i}, N_{TCPC1}^{a}, N_{TCPC2}^{a}, q_{S}^{i}, q_{C}^{a}, q_{R}\right)$ represents the probability for the steady state of the state

$\left(N^{ia}_{TCPS1}, N^{i}_{TCPS2}, N^{a}_{TCPC1}, N^{a}_{TCPC2}, q^{i}_{S}, q^{a}_{C}, q_{R}\right)$ and *x* represents the values that can be assigned to the random variable that represents the queue length of the switch *i*. According to [92][125][126], the previous equation can be calculated as:

$$\overline{q^{i}_{TCPS}} = \frac{\rho^{i}_{TS}}{1-\rho^{i}_{TS}} \tag{4.14}$$

By substitution ρ^{i}_{TS} by its value from equation (4.10) in equation (4.14), $\overline{q^{i}_{TCPS}}$ is obtained by:

$$\overline{q^{i}_{TCPS}} = \frac{\frac{\gamma^{i}_{TS}}{\mu^{i}_{S}}}{1-\frac{\gamma^{i}_{TS}}{\mu^{i}_{S}}} = \frac{\gamma^{i}_{TS}}{\mu^{i}_{S}-\gamma^{i}_{TS}}. \tag{4.15}$$

For local controller *a*, the queue mean length is formulated as:

$$\overline{q^{a}_{TCPC}} = \sum_{y=0}^{\infty} y \,.\, P\left(N^{ia}_{TCPS1}, N^{i}_{TCPS2}, N^{a}_{TCPC1}, N^{a}_{TCPC2}, q^{i}_{S}, q^{a}_{C} = y, q_{R}\right) \tag{4.16}$$

Where *y* represents the values that can be assigned to the random variable that represents the queue length of the local controller *a*.

Using substitution from equation (4.11), $\overline{q^{a}_{TCPC}}$ can be derived as:

$$\overline{q^{a}_{TCPC}} = \frac{\rho^{a}_{TC}}{1-\rho^{a}_{TC}} = \frac{\frac{\gamma^{a}_{TC}}{\mu^{a}_{C}}}{1-\frac{\gamma^{a}_{TC}}{\mu^{a}_{C}}} = \frac{\gamma^{a}_{TC}}{\mu^{a}_{C}-\gamma^{a}_{TC}} \tag{4.17}$$

The root controller has a mean length of its queue as follows:

$$\overline{q_{TCPR}} = \sum_{z=0}^{\infty} z \,.\, P\left(N^{ia}_{TCPS1}, N^{i}_{TCPS2}, N^{a}_{TCPC1}, N^{a}_{TCPC2}, q^{i}_{S}, q^{a}_{C}, q_{R} = z\right) \tag{4.18}$$

Where *z* represents the values that can be assigned to the random variable that represents the queue length of the root controller.

Using substitution from equation (4.12), $\overline{q_{TCPR}}$ can be derived as:

$$\overline{q_{TCPR}} = \frac{\rho_{TR}}{1-\rho_{TR}} = \frac{\frac{\lambda_{TR}}{\mu_{R}}}{1-\frac{\lambda_{TR}}{\mu_{R}}} = \frac{\lambda_{TR}}{\mu_{R}-\lambda_{TR}} \tag{4.19}$$

The average number of TCP flows in local controller *a* sending requests to the root controller, $\overline{N_{TCPC1}^{a}}$, is calculated as:

$$\overline{N_{TCPC1}^{a}} = \frac{\gamma_{TCPC1}^{a}}{\mu_R - \gamma_{TCPC1}^{a}} \tag{4.20}$$

4.3.2.1 Loss Probability

In the developed SDN TCP model, if the full capacity of a queue of switch *i* is used (i.e., $q_S^i = K_S^i$), the new arriving packet is discarded and the packet is lost. A packet loss at switch *i*, L_{TCPS}^{i}, has the probability illustrated as:

$$L_{TCPS}^{i} = P\left(\mathrm{N_{TCPS1}^{ia}}, \mathrm{N_{TCPS2}^{i}}, \mathrm{N_{TCPC1}^{a}}, \mathrm{N_{TCPC2}^{a}}, q_S^i = K_S^i, q_C^a, q_R\right) \tag{4.21}$$

According to [92][126], L_{TCPS}^{i} can be formulated as:

$$L_{TCPS}^{i} = \left(\frac{\gamma_{TS}^{i}}{\mu_S^i}\right)^{K_S^i} \tag{4.22}$$

Similarly, at local controller *a*, the new arriving packet is discarded when the full capacity of its queue is used (i.e., $q_C^a = K_C^a$). Therefore, local controller *a* losses a packet with the probability, L_{TCPC}^{a}, can be illustrated as:

$$L_{TCPC}^{a} = P\left(N_{TCPS1}^{ia}, N_{TCPS2}^{i}, N_{TCPC1}^{a}, N_{TCPC2}^{a}, q_S^i, q_C^a = K_C^a, q_R\right) \tag{4.23}$$

According to [92], L_{TCPC}^{a} can be derived as:

$$L_{TCPC}^{a} = \left(\frac{\gamma_{TC}^{a}}{\mu_C^a}\right)^{K_C^a} \tag{4.24}$$

Also, the root controller discards the new arriving packet when it has a full queue (i.e., $q_R = K_R$). Thus, a packet discarding probability at the root controller, L_{TCPR}, can be calculated as:

$$L_{TCPR} = P\left(N_{TCPS1}^{ia}, N_{TCPS2}^{i}, N_{TCPC1}^{a}, N_{TCPC2}^{a}, q_S^i, q_C^a, q_R = K_R\right) \tag{4.25}$$

Similarly, L_{TCPR} can be derived as:

$$L_{TCPR} = \left(\frac{\lambda_{TR}}{\mu_R}\right)^{K_R} \tag{4.26}$$

The loss probability of a TCP NF packet, L_{TCP}^{NF} and the loss probability of a TCP DF packet, L_{TCP}^{DF} can be derived as follows.

(I) Loss probability of a TCP NF packet, L_{TCP}^{NF}

Within the developed TCP model, when an NF packet is received by switch *i*, an NF packet is stored firstly at the switch, then it is queued in a local controller such as controller *a*. Then, the packet is forwarded to the root controller with $\frac{\overline{N_{TCPC1}^{a}} - \frac{\overline{q_{TCPR}}}{M}}{\overline{q_C^a}}$ probability. In such a case, the packet is stored in the root controller as well. Hence, the packet is reforwarded to local controller *a*, which sends it back to switch *i*. Hence, switch *i* stores a packet twice, and local controller *a* stores it once. Also, it is probable to be stored once again in local controller *a* in addition to the root. Therefore, the packet loss for an NF packet in the TCP flow model has the probability formulated as:

$$L_{TCP}^{NF} = 2.\, L_{TCPS}^{i} + L_{TCPC}^{a} + \frac{\overline{N_{TCPC1}^{a}} - \frac{\overline{q_{TCPR}}}{M}}{\overline{q_{TCPC}^{a}}}.\left(L_{TCPC}^{a} + L_{TCPR}\right) \tag{4.27}$$

By substitution from equation (4.17), equation (4.19), equation (4.20), equation (4.22), and equation (4.24) in equation (4.27), L_{TCP}^{NF} is obtained by:

$$L_{TCP}^{NF} = 2.\left(\frac{\gamma_{TS}^{i}}{\mu_S^{i}}\right)^{K_S^i} + \left(\frac{\gamma_{TC}^{a}}{\mu_C^{a}}\right)^{K_C^a} + \frac{\frac{\gamma_{TCPC1}^{a}}{\mu_R - \gamma_{TCPC1}^{a}} - \frac{\lambda_{TR}}{M(\mu_R - \lambda_{TR})}}{\frac{\gamma_{TC}^{a}}{\mu_C^{a} - \gamma_{TC}^{a}}}.\left(\left(\frac{\gamma_{TC}^{a}}{\mu_C^{a}}\right)^{K_C^a} + \left(\frac{\lambda_{TR}}{\mu_R}\right)^{K_R}\right)$$

(4.28)

(II) Loss probability of a TCP DF packet, L_{TCP}^{DF}:

For a DF packet, when it arrives at switch *i*, the packet is queued and served directly by the switch. This is because the DF packet belongs to a familiar flow for switch *i*, so it does not redirect such a packet to the local controller. Hence, the packet may be lost with probability, L_{TCP}^{DF}, which is equal to the probability of losing a packet at switch i, L_{TSPS}^{i}.

$$L_{TCP}^{DF} = L_{TSPS}^{i} = \left(\frac{\gamma_{TS}^{i}}{\mu_{S}^{i}}\right)^{K_{S}^{i}} \tag{4.29}$$

4.3.2.2 Delay Analysis

The delay of a packet at switch i, D_{TCPS}^{i}, is formulated as:

$$D_{TCPS}^{i} = \frac{\overline{q_{TCPS}^{i}}+1}{\mu_{S}^{i}} \tag{4.30}$$

By substitution from equation (4.14) in equation (4.30), D_{TCPS}^{i} is obtained by:

$$D_{TCPS}^{i} = \frac{\frac{\gamma_{TS}^{i}}{\mu_{S}^{i}-\gamma_{TS}^{i}}+1}{\mu_{S}^{i}} = \frac{1}{\mu_{S}^{i}-\gamma_{TS}^{i}} \tag{4.31}$$

Local controller a introduces a delay for the packet, D_{TCPC}^{a}, which can be obtained by:

$$D_{TCPC}^{a} = \frac{\overline{q_{TCPC}^{a}}+1}{\mu_{C}^{a}} \tag{4.32}$$

By substitution from equation (4.17) in equation (4.32), D_{TCPC}^{a} is obtained by:

$$D_{TCPC}^{a} = \frac{\frac{\gamma_{TC}^{a}}{\mu_{C}^{a}-\gamma_{TC}^{a}}+1}{\mu_{C}^{a}} = \frac{1}{\mu_{C}^{a}-\gamma_{TC}^{a}} \tag{4.33}$$

Similarly, the packet suffers a delay in the root controller, D_{TCPR}, can be expressed as:

$$D_{TCPR} = \frac{\overline{q_{TCPR}}+1}{\mu_{R}} \tag{4.34}$$

By substitution from equation (4.19) in equation (4.34), D_{TCPR} is obtained by:

$$D_{TCPR} = \frac{\frac{\lambda_{TR}}{\mu_R - \lambda_{TR}} + 1}{\mu_R} = \frac{1}{\mu_R - \lambda_{TR}} \tag{4.35}$$

We consider the two types of a TCP flow packet as follows.

(I) Average packet delay for a TCP NF packet, D_{TCP}^{NF}:

First, the packet suffers a delay at switch *i* equals D_{TCPS}^{i}. Then, switch *i* redirects the packet to local controller a, since the packet belongs to a new flow, taking D_{Prop}. Local controller a receives the NF packet and queues it D_{TCPC}^{a} unit time. Here, local controller *a* may fail to process it, so it forwards it to the root controller with $(\frac{\overline{N_{TCPC1}^{a}} - \frac{\overline{q_{TCPR}}}{M}}{\overline{q_C^a}})$ probability. The packet takes D_{Prop} to arrive at the root controller. Then, it suffers from queuing in the root controller equals D_{TCPR}, and needs more D_{Prop} to be sent back to controller *a*, which queues it another D_{TCPC}^{a}. Therefore, the amount of time added to the average packet delay when the packet is forwarded to the root controller, D_{TCP}^{CR}, is formulated as:

$$D_{TCP}^{CR} = 2D_{Prop} + D_{TCPR} + D_{TCPC}^{a} \tag{4.36}$$

By substitution from equation (4.32) and equation (4.34) in equation (5.36), D_{TCP}^{CR} is obtained by:

$$D_{TCP}^{CR} = 2D_{Prop} + \frac{1}{\mu_R - \lambda_{TR}} + \frac{1}{\mu_C^a - \gamma_{TC}^a} \tag{4.37}$$

Then, local controller a sends the packet back to switch *i* taking D_{Prop}. The packet suffers one more D_{TCPS}^{i} at switch *i*. Hence, the average transfer delay for a TCP NF packet, D_{TCP}^{NF}, is expressed as:

$$D_{TCP}^{NF} = 2D_{TCPS}^{i} + 2D_{Prop} + D_{TCPC}^{a} + \frac{\overline{N_{TCPC1}^{a}} - \frac{\overline{q_{TCPR}}}{M}}{\overline{q_{TCPC}^{a}}}\left(2D_{Prop} + D_{TCPR} + D_{TCPC}^{a}\right) \tag{4.38}$$

By substitution from equation (4.17), equation (4.19), equation (4.20), equation (4.31), equation (4.33) and equation (4.35) in equation (4.38), D_{TCP}^{NF} is obtained by:

$$D_{TCP}^{NF} = 2\frac{1}{\mu_S^i - \gamma_{TS}^i} + 2D_{Prop} + \frac{1}{\mu_C^a - \gamma_{TC}^a} + \frac{\frac{\gamma_{TCPC1}^a}{\mu_R - \gamma_{TCPC1}^a} - \frac{\lambda_{TR}}{M(\mu_R - \lambda_{TR})}}{\frac{\gamma_{TC}^a}{\mu_C^a - \gamma_{TC}^a}}\left(2D_{Prop} + \frac{1}{\mu_R - \lambda_{TR}} + \frac{1}{\mu_C^a - \gamma_{TC}^a}\right) \tag{4.39}$$

(II) Average packet delay of a TCP DF packet, D_{TCP}^{DF}:

As stated before, a DF packet does not require the switch to redirect it to the local controller. Therefore, the packet is stored only in switch *i*. The delay is like the packet delay in switch *i*, as shown:

$$D_{TCP}^{DF} = D_{TCPS}^{i} \tag{4.40}$$

By substitution from equation (4.31) in equation (4.40), D_{TCP}^{DF} is obtained by:

$$D_{TCP}^{DF} = \frac{1}{\mu_S^i - \gamma_{TS}^i} \tag{4.41}$$

4.4 SDN UDP Model

A detailed explanation of the developed model for UDP flows is provided. Firstly, transitions of the UDP seven-state are described, and then performance measures of the developed UDP model are proposed.

4.4.1 SDN UDP Model's States

Figure (4.3) depicts the nine possible states (labeled 1, 2,..., 9) of state $(N_{UDPS1}^{ia}, N_{UDPS2}^{i}, N_{UDPC1}^{a}, N_{UDPC2}^{a}, q_S^i, q_C^a, q_R)$ in the SDN system model for UDP flows. States 2, 5, and 7 are explained in the incoming points. The other states are similar to their corresponding states in the TCP model, so they are not explained here.

- Second transition: switch *i* finds a matched flow entry with a received UDP DF packet. In such a state, the probability of finding such a match between the packet and an entry of the flow table is $(\frac{N_{UDPS2}^{i}}{N_{UDPS1}^{ia}+N_{UDPS2}^{i}})$. Therefore, switch *i* sends the packet towards its destination. q_S^i is decreased by 1. The transition rate is formulated as follows.

$$R_{Trans} = \left(\frac{N_{UDPS2}^{i}}{N_{UDPS1}^{ia}+N_{UDPS2}^{i}}\right).\mu_S^i \tag{4.42}$$

- Fifth transition: local controller *a* can process a packet received from switch *i*. Hence, local controller *a* sends the packet back along with its suitable flow rules to switch *i*. The probability for local controller *a* to process a packet successfully is $(\frac{N_{UDPC2}^{a}}{N_{UDPC1}^{a}+N_{UDPC2}^{a}})$. Therefore, N_{UDPS1}^{ia} , and q_C^a are decreased by 1. N_{UDPS2}^{i} , and q_S^i are increased by 1. The transition rate can be represented as:

$$R_{Trans} = \left(\frac{N_{UDPC2}^{a}}{N_{UDPC1}^{a}+N_{UDPC2}^{a}}\right).\frac{\mu_C^a}{K} \tag{4.43}$$

- Seventh transition: local controller *a* cannot process a TCP packet. Local controller *a* fails to process the packet successfully with the probability $(\frac{N_{UDPC1}^{a}}{N_{UDPC1}^{a}+N_{UDPC2}^{a}})$. In this state, local controller *a* forwards a packet to the root controller. q_C^a is decreased by 1. q_R is increased by 1. The transition rate can be shown as:

$$R_{Trans} = \left(\frac{N_{UDPC1}^a}{N_{UDPC1}^a + N_{UDPC2}^a}\right) . \frac{\mu_C^a}{K} \quad (4.44)$$

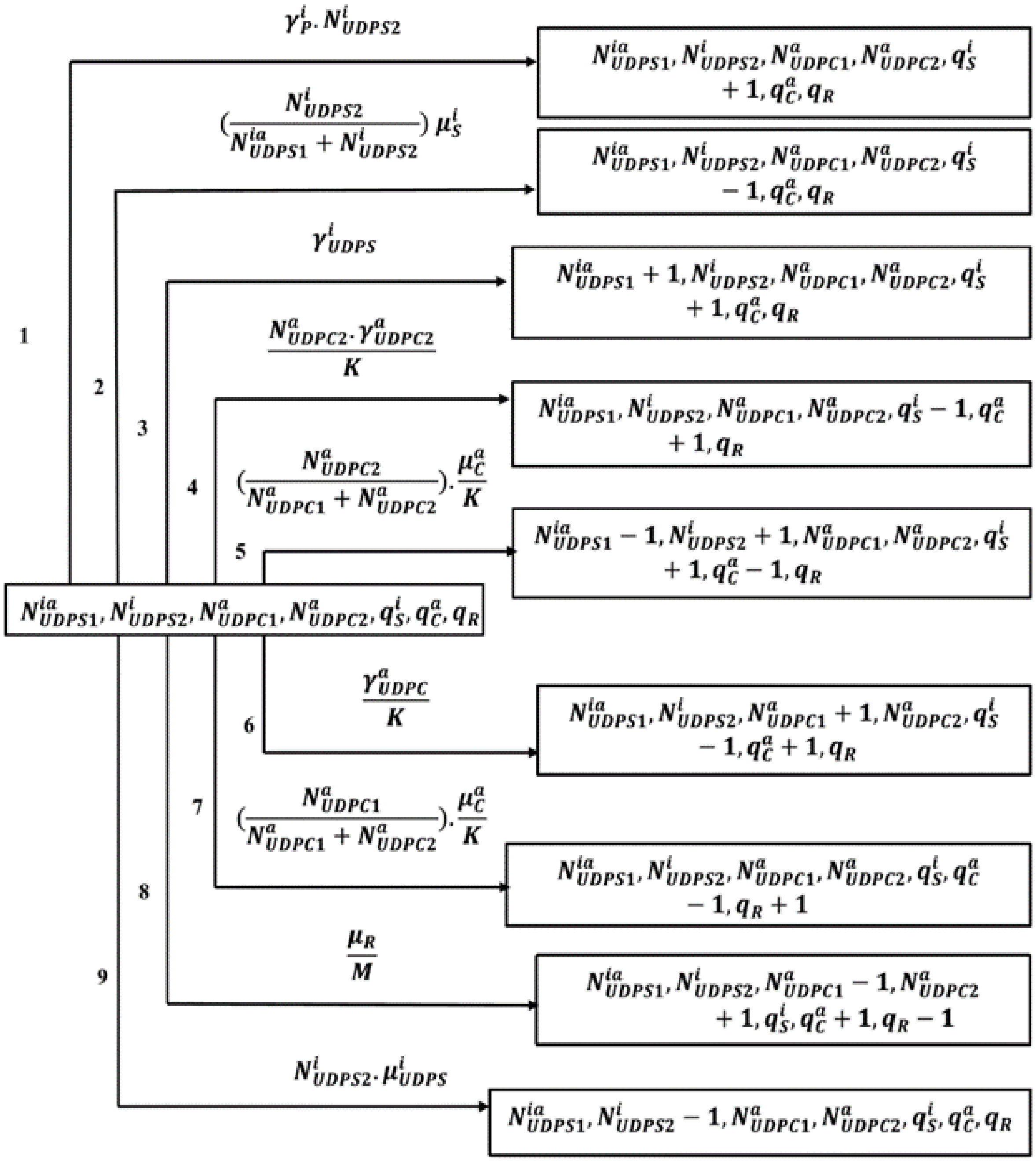

$\gamma_P^i . N_{UDPS2}^i$
$N_{UDPS1}^{ia}, N_{UDPS2}^i, N_{UDPC1}^a, N_{UDPC2}^a, q_S^i + 1, q_C^a, q_R$
$(\frac{N_{UDPS2}^i}{N_{UDPS1}^{ia} + N_{UDPS2}^i}) \mu_S^i$
$N_{UDPS1}^{ia}, N_{UDPS2}^i, N_{UDPC1}^a, N_{UDPC2}^a, q_S^i - 1, q_C^a, q_R$
γ_{UDPS}^i
$N_{UDPS1}^{ia} + 1, N_{UDPS2}^i, N_{UDPC1}^a, N_{UDPC2}^a, q_S^i + 1, q_C^a, q_R$
1
2
3
$\frac{N_{UDPC2}^a . \gamma_{UDPC2}^a}{K}$
$N_{UDPS1}^{ia}, N_{UDPS2}^i, N_{UDPC1}^a, N_{UDPC2}^a, q_S^i - 1, q_C^a + 1, q_R$
4
$(\frac{N_{UDPC2}^a}{N_{UDPC1}^a + N_{UDPC2}^a}) . \frac{\mu_C^a}{K}$
5
$N_{UDPS1}^{ia} - 1, N_{UDPS2}^i + 1, N_{UDPC1}^a, N_{UDPC2}^a, q_S^i + 1, q_C^a - 1, q_R$
$N_{UDPS1}^{ia}, N_{UDPS2}^i, N_{UDPC1}^a, N_{UDPC2}^a, q_S^i, q_C^a, q_R$
$\frac{\gamma_{UDPC}^a}{K}$
6
$N_{UDPS1}^{ia}, N_{UDPS2}^i, N_{UDPC1}^a + 1, N_{UDPC2}^a, q_S^i - 1, q_C^a + 1, q_R$
7
$(\frac{N_{UDPC1}^a}{N_{UDPC1}^a + N_{UDPC2}^a}) . \frac{\mu_C^a}{K}$
$N_{UDPS1}^{ia}, N_{UDPS2}^i, N_{UDPC1}^a, N_{UDPC2}^a, q_S^i, q_C^a - 1, q_R + 1$
8
$\frac{\mu_R}{M}$
9
$N_{UDPS1}^{ia}, N_{UDPS2}^i, N_{UDPC1}^a - 1, N_{UDPC2}^a + 1, q_S^i, q_C^a + 1, q_R - 1$
$N_{UDPS2}^i . \mu_{UDPS}^i$
$N_{UDPS1}^{ia}, N_{UDPS2}^i - 1, N_{UDPC1}^a, N_{UDPC2}^a, q_S^i, q_C^a, q_R$

Figure 4.3: State transition diagram of the SDN UDP model

4.4.2 Performance Measures of the Developed UDP Model

The related performance measures of the proposed UDP model are calculated in this subsection to track its behavior. The loss probability and the average packet delay of a UDP packet are calculated. Firstly, traffic intensity in switch *i*, ρ_{US}^{i}, traffic intensity in local controller a, ρ_{UC}^{a}, and traffic intensity in the root controller, ρ_{UR}, are calculated to help in computing the packet loss probability and the average packet delay. Also, the mean queue length for switch *i*, $\overline{q_{UDPS}^{i}}$, the mean queue length for local controller a, $\overline{q_{UDPC}^{a}}$, and the mean queue length for the root controller, $\overline{q_{UDPR}}$, are calculated. In addition to calculating the average number of UDP flows in local controller *a* sending requests to the root controller, $\overline{N_{UDPC1}^{a}}$, and the average number of UDP flows that are served locally by local controller a, $\overline{N_{UDPC2}^{a}}$.

The traffic intensity in switch *i*, ρ_{US}^{i}, is obtained as:

$$\rho_{US}^{i} = \frac{\gamma_{US}^{i}}{\mu_{S}^{i}} \tag{4.45}$$

The traffic intensity in local controller a, ρ_{C}^{a}, is obtained as:

$$\rho_{UC}^{a} = \frac{\gamma_{UC}^{a}}{\mu_{C}^{a}} \tag{4.46}$$

The traffic intensity in the root controller, ρ_{UR} is obtained as:

$$\rho_{UR} = \frac{\lambda_{UR}}{\mu_{R}} \tag{4.47}$$

The mean length of a queue for switch i, $\overline{q_{UDPS}^{i}}$ can be illustrated as:

$$\overline{q_{UDPS}^{i}} = \sum_{x=0}^{\infty} x \,.\, P\left(N_{UDPS1}^{ia}, N_{UDPS2}^{i}, N_{UDPC1}^{a}, N_{UDPC2}^{a}, q_{S}^{i} = x, q_{C}^{a}, q_{R}\right) \tag{4.48}$$

Where $P\left(N_{UDPS1}^{ia}, N_{UDPS2}^{i}, N_{UDPC1}^{a}, N_{UDPC2}^{a}, q_{S}^{i} = x, q_{C}^{a}, q_{R}\right)$ represents the probability for the steady state of the state $\left(\mathrm{N}_{UDPS1}^{\mathrm{ia}}, \mathrm{N}_{UDPS2}^{\mathrm{i}}, \mathrm{N}_{UDPC1}^{\mathrm{a}}, \mathrm{N}_{UDPC2}^{\mathrm{a}}, q_{S}^{i}, q_{C}^{a}, q_{R}\right)$.

According to [92], the previous equation can be calculated as:

$$\overline{q_{UDPS}^{i}} = \frac{\rho_{US}^{i}}{1-\rho_{US}^{i}} \tag{4.49}$$

By substitution from equation (4.45) in equation (4.49), $\overline{q_{UDPS}^{i}}$ is obtained by:

$$\overline{q_{UDPS}^{i}} = \frac{\frac{\gamma_{US}^{i}}{\mu_{S}^{i}}}{1-\frac{\gamma_{US}^{i}}{\mu_{S}^{i}}} = \frac{\gamma_{US}^{i}}{\mu_{S}^{i}-\gamma_{US}^{i}}. \tag{4.50}$$

For local controller a, the queue mean length, $\overline{q_{UDPC}^{a}}$ is formulated as:

$$\overline{q_{UDPC}^{a}} = \sum_{y=0}^{\infty} y\,.\,P\left(N_{UDPS1}^{ia}, N_{UDPS2}^{i}, N_{UDPC1}^{a}, N_{UDPC2}^{a}, q_{S}^{i}, q_{C}^{a} = y, q_{R}\right) \tag{4.51}$$

Similarly, $\overline{q_{UDPC}^{a}}$ can be derived as:

$$\overline{q_{UDPC}^{a}} = \frac{\rho_{UC}^{a}}{1-\rho_{UC}^{a}} \tag{4.52}$$

By substitution from equation (4.46) in equation (4.52), $\overline{q_{UDPC}^{a}}$ is obtained by:

$$\overline{q_{UDPC}^{a}} = \frac{\frac{\gamma_{UC}^{a}}{\mu_{C}^{a}}}{1-\frac{\gamma_{UC}^{a}}{\mu_{C}^{a}}} = \frac{\gamma_{UC}^{a}}{\mu_{C}^{a}-\gamma_{UC}^{a}} \tag{4.53}$$

The root controller has a mean length of its queue as shown:

$$\overline{q_{UDPR}} = \sum_{z=0}^{\infty} z\,.\,P\left(N_{UDPS1}^{ia}, N_{UDPS2}^{i}, N_{UDPC1}^{a}, N_{UDPC2}^{a}, q_{S}^{i}, q_{C}^{a}, q_{R} = z\right) \tag{4.54}$$

Similarly, $\overline{q_{UDPR}}$ can be derived as:

$$\overline{q_{UDPR}} = \frac{\rho_{UR}}{1-\rho_{UR}} \tag{4.55}$$

By substitution from equation (4.47) in equation (4.55), $\overline{q_{UDPR}}$ is obtained by:

$$\overline{q_{UDPR}} = \frac{\frac{\lambda_{UR}}{\mu_{R}}}{1-\frac{\lambda_{UR}}{\mu_{R}}} = \frac{\lambda_{UR}}{\mu_{R}-\lambda_{UR}} \tag{4.56}$$

The average number of UDP flows in local controller *a* sending requests to the root controller, $\overline{N_{UDPC1}^{a}}$, is calculated as:

$$\overline{N_{UDPC1}^{a}} = \frac{\frac{\gamma_{UDPC1}^{a}}{\frac{\mu_R}{M}}}{1-\frac{\gamma_{UDPC1}^{a}}{\frac{\mu_R}{M}}} = \frac{\gamma_{UDPC1}^{a}}{\frac{\mu_R}{M}-\gamma_{UDPC1}^{a}} \tag{4.57}$$

Similarly, the average number of UDP flows in local controller *a* that are served locally without being forwarded to the root controller, $\overline{N_{UDPC2}^{a}}$, is calculated as:

$$\overline{N_{UDPC2}^{a}} = \frac{\frac{\gamma_{UDPC2}^{a}}{\mu_C^a}}{1-\frac{\gamma_{UDPC2}^{a}}{\mu_C^a}} = \frac{\gamma_{UDPC2}^{a}}{\mu_C^a-\gamma_{UDPC2}^{a}} \tag{4.58}$$

4.4.2.1 Loss Probability

In the developed SDN UDP model, if the full capacity of a queue of switch *i* is used (i.e., $q_S^i = K_S^i$), the new arriving packet is discarded. A packet discarding at switch *i*, L_{UDPS}^{i}, has the probability illustrated as:

$$L_{UDPS}^{i} = P\left(N_{UDPS1}^{ia}, N_{UDPS2}^{i}, N_{UDPC1}^{a}, N_{UDPC2}^{a}, q_S^i = k_S^i, q_C^a, q_R\right) \tag{4.59}$$

According to [92], L_{UDPs}^{i} can be derived as:

$$L_{UDPs}^{i} = \left(\frac{\gamma_{US}^{i}}{\mu_S^i}\right)^{K_S^i} \tag{4.60}$$

The loss of a packet at local controller *a*, L_{UDPC}^{a}, is shown as:

$$L_{UDPC}^{a} = P(N_{UDPS1}^{ia}, N_{UDPS2}^{i}, N_{UDPC1}^{a}, N_{UDPC2}^{a}, q_S^i, q_C^a = k_C^a, q_R) \tag{4.61}$$

Similarly, L_{UDPC}^{a} can be derived as:

$$L_{UDPC}^{a} = \left(\frac{\gamma_{UC}^{a}}{\mu_C^a}\right)^{K_C^a} \tag{4.62}$$

A packet loss at the root controller, L_{UDPR}, can be calculated as:

$$L_{UDPR} = P\left(N_{UDPS1}^{ia}, N_{UDPS2}^{i}, N_{UDPC1}^{a}, N_{UDPC2}^{a},\ q_S^i, q_C^a, q_R = K_R\right) \tag{4.63}$$

Similarly, L_{UDPR} can be derived as:

$$L_{UDPR} = \left(\frac{\lambda_{UR}}{\mu_R}\right)^{K_R} \tag{4.64}$$

Therefore, the loss probability of a UDP NF packet, L_{UDP}^{NF} and the loss probability of a UDP DF packet, L_{UDP}^{DF} are derived as follows.

(I) Loss probability of a UDP NF packet, L_{UDP}^{NF}:

As in the TCP model, when an NF packet is received by switch i, an NF packet is stored firstly at the switch, then it is queued in a local controller a. Then, the packet is forwarded to the root controller with $\frac{\overline{N_{UDPC1}^{a}}}{\overline{N_{UDPC1}^{a}}+\overline{N_{UDPC2}^{a}}}$ probability. In such a case, the packet is stored in the root controller as well. Hence, the packet is reforwarded to local controller a, which sends it back to switch i. Hence, switch i stores a packet twice, and local controller a stores it once. Also, it is probable to be stored once again in local controller a and the root. Therefore, the packet loss for an NF packet in the UDP flow model has the probability formulated as:

$$L_{UDP}^{NF} = 2L_{UDPS}^{i} + L_{UDPC}^{a} + \frac{\overline{N_{UDPC1}^{a}}}{\overline{N_{UDPC1}^{a}}+\overline{N_{UDPC2}^{a}}}\left(L_{UDPC}^{a} + L_{UDPR}\right) \tag{4.65}$$

By substitution from equation (4.57), equation (4.58), equation (4.60), equation (4.62) and equation (4.64) in equation (4.65), L_{TCP}^{NF} is obtained by:

$$L_{UDP}^{NF} = 2\left(\frac{\gamma_{US}^{i}}{\mu_S^{i}}\right)^{K_S^i} + \left(\frac{\gamma_{UC}^{a}}{\mu_C^{a}}\right)^{K_C^a} + \frac{\frac{\gamma_{UDPC1}^{a}}{\frac{\mu_R}{M}-\gamma_{UDPC1}^{a}}}{\frac{\gamma_{UDPC1}^{a}}{\frac{\mu_R}{M}-\gamma_{UDPC1}^{a}}+\frac{\gamma_{UDPC2}^{a}}{\mu_C^{a}-\gamma_{UDPC2}^{a}}}\left(\left(\frac{\gamma_{UC}^{a}}{\mu_C^{a}}\right)^{K_C^a} + \left(\frac{\lambda_{UR}}{\mu_R}\right)^{K_R}\right)$$

(4.66)

(II) Loss probability of a UDP DF packet, L_{UDP}^{DF}:

UDP DF packets are served locally by switch i without needing to redirect them to local controller a. The discarding probability for such DF packets is the same as L_{UDPS}^{i}.

$$L_{UDP}^{DF} = \left(\frac{\gamma_{US}^{i}}{\mu_S^{i}}\right)^{K_S^i} \tag{4.67}$$

4.4.2.2 Delay Analysis

As in the TCP model, the delay of a packet at switch i, D_{UDPS}^{i}, is formulated as:

$$D_{UDPS}^{i} = \frac{\overline{q_{UDPS}^{i}}+1}{\mu_S^{i}} \tag{4.68}$$

By substitution from equation (4.50) in equation (4.68), D_{UDPS}^{i} is obtained by:

$$D_{UDPS}^{i} = \frac{\frac{\gamma_{US}^{i}}{\mu_S^{i}-\gamma_{US}^{i}}+1}{\mu_S^{i}} = \frac{1}{\mu_S^{i}-\gamma_{US}^{i}} \tag{4.69}$$

Local controller a introduces a delay for the packet, D_{UDPC}^{a}, which is obtained by:

$$D_{UDPC}^{a} = \frac{\overline{q_{UDPC}^{a}}+1}{\mu_C^{a}} \tag{4.70}$$

By substitution from equation (4.53) in equation (4.70), D_{UDPC}^{a} is obtained by:

$$D_{UDPC}^{a} = \frac{\frac{\gamma_{UC}^{a}}{\mu_C^{a}-\gamma_{UC}^{a}}+1}{\mu_C^{a}} = \frac{1}{\mu_C^{a}-\gamma_{UC}^{a}} \tag{4.71}$$

Similarly, the packet suffers a delay in the root controller, D_{UDPR}, which is expressed by:

$$D_{UDPR} = \frac{\overline{q_{UDPR}}+1}{\mu_R} \tag{4.72}$$

By substitution from equation (4.56) in equation (4.72), D_{UDPR} is obtained by:

$$D_{UDPR} = \frac{\frac{\lambda_{UR}}{\mu_R - \lambda_{UR}} + 1}{\mu_R} = \frac{1}{\mu_R - \lambda_{UR}} \tag{4.73}$$

We consider the two types of a UDP flow packet as follows.

(I) Average transfer delay of a UDP NF packet, D_{UDP}^{NF}:

It has similarities to the corresponding case in the TCP model. But, in the UDP model, local controller *a* may fail to process an NF packet and need to forward it to the root controller with $\left(\frac{\overline{N_{UDPC1}^{a}}}{\overline{N_{UDPC1}^{a}} + \overline{N_{UDPC2}^{a}}}\right)$ probability. Hence, the average transfer delay of a UDP NF packet, D_{UDP}^{NF}, can be formulated as:

$$D_{UDP}^{NF} = 2D_{UDPS}^{i} + 2D_{Prop} + D_{UDPC}^{a} + \frac{\overline{N_{UDPC1}^{a}}}{\overline{N_{UDPC1}^{a}} + \overline{N_{UDPC2}^{a}}}\left(2D_{Prop} + D_{UDPR} + D_{UDPC}^{a}\right) \tag{4.74}$$

By substitution from equation (4.57), equation (4.58), equation (4.69), equation (4.71), and equation (4.73) in equation (4.74), D_{UDP}^{NF} is obtained by:

$$D_{UDP}^{NF} = 2\frac{1}{\mu_S^{i} - \gamma_{US}^{i}} + 2D_{Prop} + \frac{1}{\mu_C^{a} - \gamma_{UC}^{a}} + \frac{\frac{\gamma_{UDPC1}^{a}}{\frac{\mu_R}{M} - \gamma_{UDPC1}^{a}}}{\frac{\gamma_{UDPC1}^{a}}{\frac{\mu_R}{M} - \gamma_{UDPC1}^{a}} + \frac{\gamma_{UDPC2}^{a}}{\mu_C^{a} - \gamma_{UDPC2}^{a}}}\left(2D_{Prop} + \frac{1}{\mu_R - \lambda_{UR}} + \frac{1}{\mu_C^{a} - \gamma_{UC}^{a}}\right) \tag{4.75}$$

(II) Average transfer delay of a UDP DF packet, D_{UDP}^{DF}:

A DF packet is served locally by switch *i*, hence the average transfer delay is the same as the delay of switch *i*, D_{UDPS}^{i}, which is obtained by:

$$D_{UDP}^{DF} = \frac{1}{\mu_S^{i} - \gamma_{US}^{i}} \tag{4.76}$$

Chapter 5

Simulation Results

5 Simulation Results

5.1 Introduction

This chapter presents the simulation results of the proposed techniques and models in two sections: Section 5.2 presents the simulation results of the performance of the proposed ML-based DDoS attack detection techniques against the control plane and data plane in SDN, compared with other related techniques and approaches. Section 5.3 presents the simulation results of the performance of the developed 7-D queuing analysis models which can be used roughly to differentiate normal traffic from DDoS attack traffic in SDN, compared with other related models.

5.2 Simulation Results for ML-based DDoS Attack Detection Techniques Performance

In this section, the simulation, experiments, and result analysis for the proposed ML-based techniques to detect DDoS attacks against the control and data planes in SDN are described as follows.

5.2.1 Experimental and Simulation Setup for ML Detection

Simulation experiments of the proposed ML technique were executed on a PC running Ubuntu 18.04.2 LTS OS. The system has a CPU of Intel@ Core i7 CPU E7500 @ 3.4GHZ x 2 and a 7.9 GiB memory. We used SDN architecture with a Pox controller [79] as the control plane. The Pox controller is executed mainly using python. POX is a popular choice among SDN researchers because it is quick, light, and can be tailored to a specific purpose. The network emulator used for this research is Mininet [127], which is considered the standard network emulator for the SDN environment.

For the control plane, using Mininet, a tree-type network with depth 2, which has 9 switches, and 64 hosts was created as illustrated in figure (5.1). For forwarding elements, OVS [74] was used.

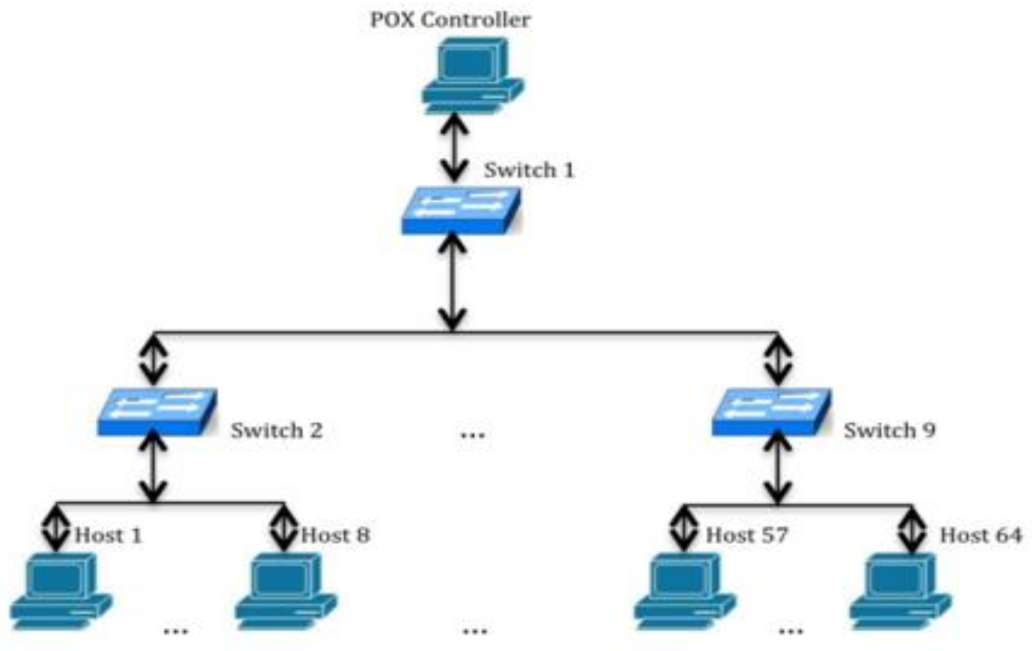


Figure 5.1: The network topology for control plane DDoS detection

For the data plane, as shown in figure (5.2), we built a tree-type SDN topology with depth 3, which has 13 switches, and 27 hosts using Mininet.

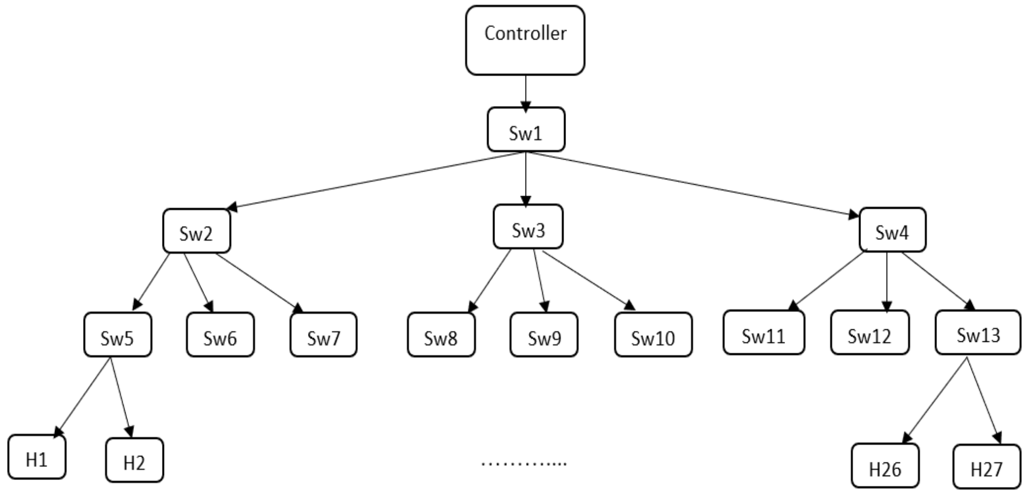


Figure 5.2: The network topology for data plane DDoS detection

For packet generation in both techniques in the control plane and data plane, Scapy [128] is used. Scapy is a very effective packet generation, search, sniff, strike, and packet forging tool [128]. There were two kinds of traffic generated: normal and attack traffic. Scapy is used also for spoofing the source IP address.

5.2.2 Result Analysis for ML Detection

The work began by launching 70,000 packets originating from hosts in the simulated SDN network. Different packet headers and valuable traffic statistics were extracted and used in various calculations to get features. Therefore, the dataset has been created consisting of the obtained features. The dataset is used as an input into ML-based algorithms to create classification models. The detection classifier is created using SVM. Moreover, other ML-based algorithms are also utilized and compared with the SVM model to improve the detection operation. The proposed technique is compared with other ML-based, entropy-based, and traffic pattern analysis-based techniques.

5.2.2.1 Comparison with other ML Algorithms' Models

For both the control plane and data plane detection techniques, ML-based techniques such as Naive Bayes, KNN, Decision Tree, and Random Forest are utilized and compared to create detection models. The generated models are compared with the created SVM model to choose the model that gives the highest detection accuracy. Based on experimental results, SVM gives the best accuracy among the other ML-based technologies used. For the control plane, table (5.1) and figure (5.3) depict the accuracy of various algorithms.

Table 5.1: Accuracy of various ML algorithms for control plane detection

Algorithm	Accuracy (%)
SVM	99.84
KNN	98.96
Decision Tree	99.26
Naive Bayes	77.64
Random Forest	99.19

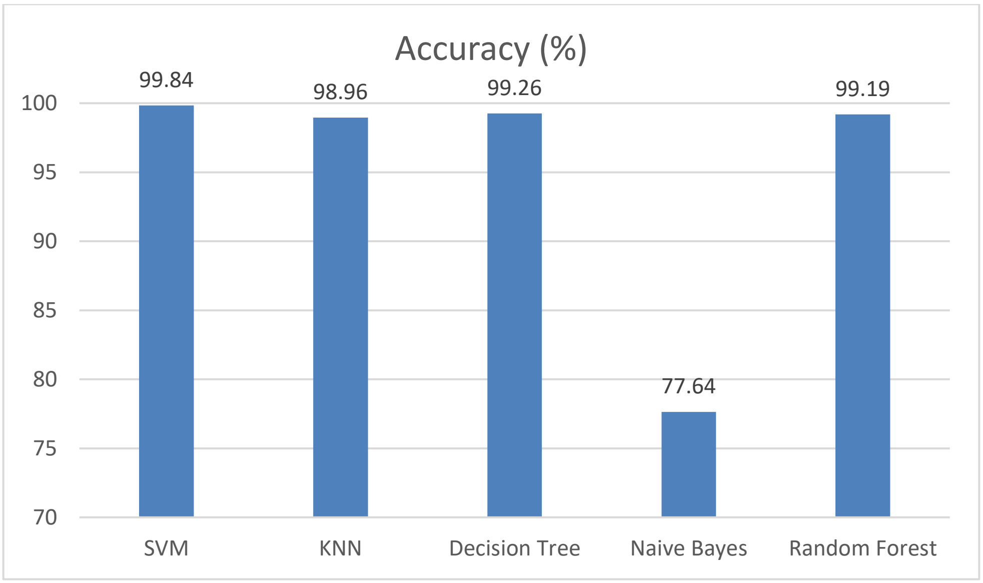


Figure 5.3: Accuracy of various ML algorithms for control plane detection

For the data plane, table (5.2) and figure (5.4) depict the accuracy of various algorithms.

Table 5.2: Accuracy of various ML algorithms for data plane detection

Algorithm	Accuracy (%)
SVM	99.88
KNN	97.35
Decision Tree	99.38
Naive Bayes	85.22
Random Forest	99.12

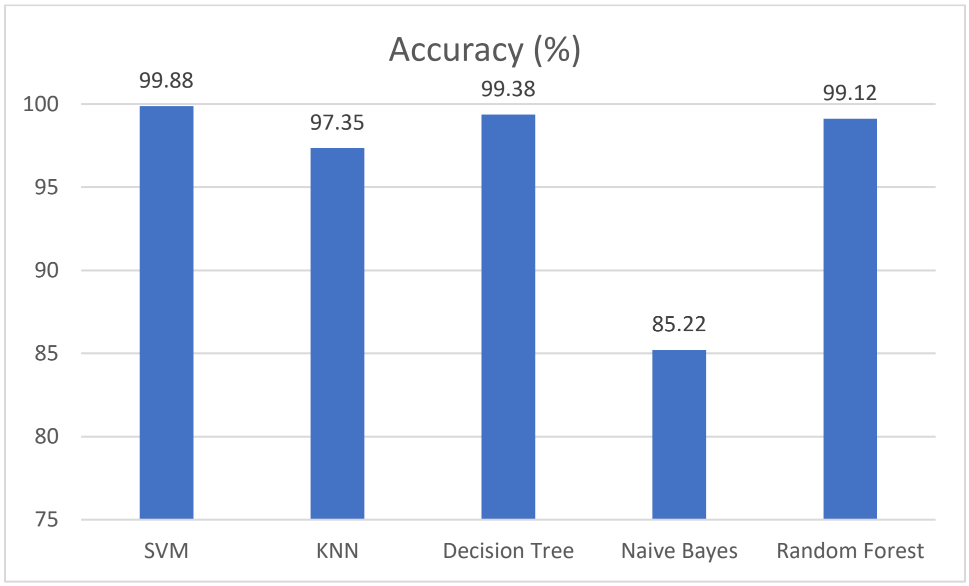


Figure 5.4: Accuracy of various ML algorithms for data plane detection

5.2.2.2 Comparison with other techniques

For the control plane, the proposed technique is compared with other DDoS detection techniques which can be based on ML, entropy, or traffic analysis pattern.

Through the detection operation, 30,000 packets were launched from hosts in the network such that 14987 DDoS attack packets were sent from 9 compromised end systems and 15,013 normal packets from the remaining hosts. Based on experimental results, there are 14982 TN packets. Additionally, there are 17 FN packets, 31 FP packets, and 14970 TP packets. Hence, various performance measures such as detection accuracy, detection rate, false alarm rate, and f_measure are calculated to compare the performance of the introduced technique against other techniques in related works. To complete achieving the research objectives, the compromised end systems are blocked, and their information is logged by the SDN controller as a mitigation mechanism to decrease the harmful effects of the DDoS attack.

Also, we use our environment to implement techniques of SOM [26], Two-stage approach [23], DPTCM-KNN [24], Query-based detection [129], and LSTM-FUZZY [28].

Table (5.3) as well as figure (5.5), figure (5.6), and figure (5.7) show comparisons of accuracy, detection rate, and FPR for SOM [26], Early detection entropy [55], time and space-efficient technique [110], Sguard [27], Two-stage approach [23], Entropy-based model [107], DPTCM-KNN [24], RL-RF [32], K-FKNN [33], Query-based detection [129], Feature selection method [34], LSTM-FUZZY [28], Chaos theory detection technique [35], Polling detection technique [36], and the technique introduced in this thesis. It can be shown from the results that the proposed technique in this thesis gives the best results compared with other techniques. Also, it can be shown that LSTM-FUZZY gives results that are close to the proposed technique in this thesis. However, LSTM-FUZZY which is an artificial neural network algorithm may suffer from many drawbacks. LSTM-FUZZY takes a large training time and memory. Also, it is prone to overfitting problems. In addition, LSTM-FUZZY is sensitive to different random weight initializations. On the other side, the SVM algorithm is used in the proposed technique. SVM is fast and preferable for classification into two labels, complex problems, and data with large dimensions. Also, it requires less memory space to train the classifier and is less prone to the overfitting problem.

Table 5.3: Accuracy, detection rate, and FPR comparisons of different algorithms for control plane detection

Algorithm	Accuracy (%)	Detection rate (%)	FPR (%)
SOM	96.4	98.75	0.55
Early detection entropy [55]	96	NA	NA
Time and space-efficient technique [110]	NA	96	0.7
Sguard	NA	99.77	NA
Two-stage approach [23]	99	98.5	0.27
Entropy-based model [107]	94	NA	6
DPTCM-KNN	97.85	96.65	4.5

RL-RF	99.54	NA	0.7
K-FKNN	NA	97.5	NA
Query-based detection [129]	99.2	98.43	0.3
Feature selection method [34]	98.3	97.73	NA
LSTM-FUZZY	99.2	99.45	0.24
Chaos theory technique [35]	NA	94	0.25
Polling detection technique [36]	91	83	4
The proposed technique	99.84	99.88	0.21

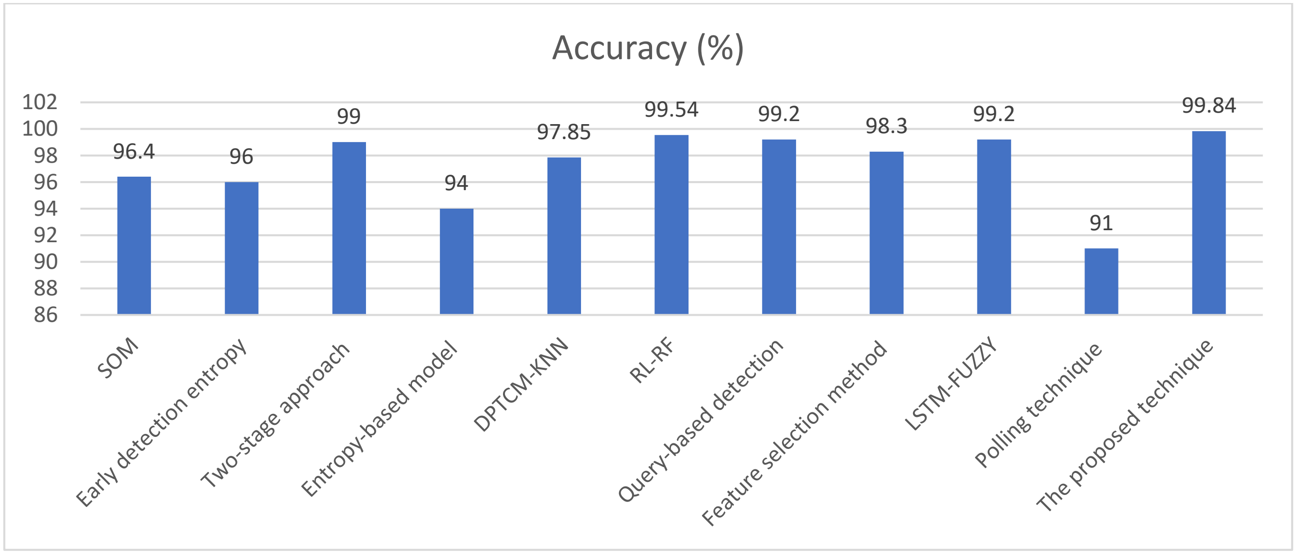


Figure 5.5: Accuracy comparison of different methods for control plane detection

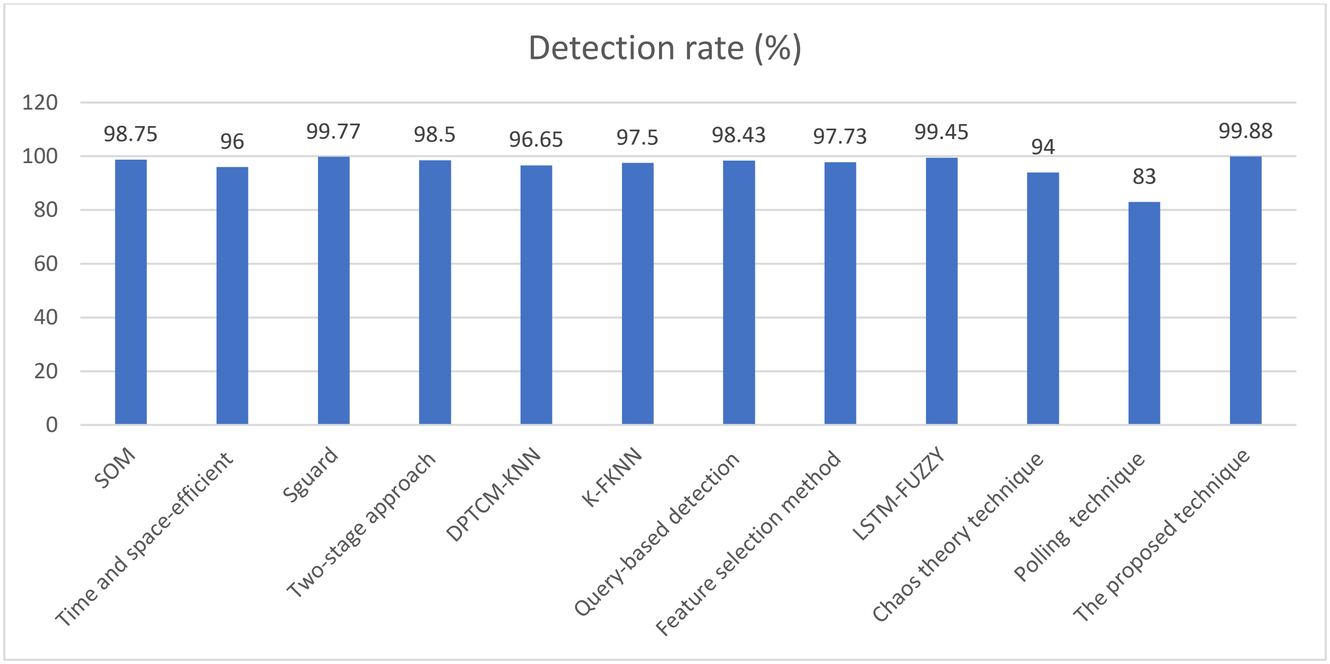


Figure 5.6: Detection rate comparison of different methods for control plane detection

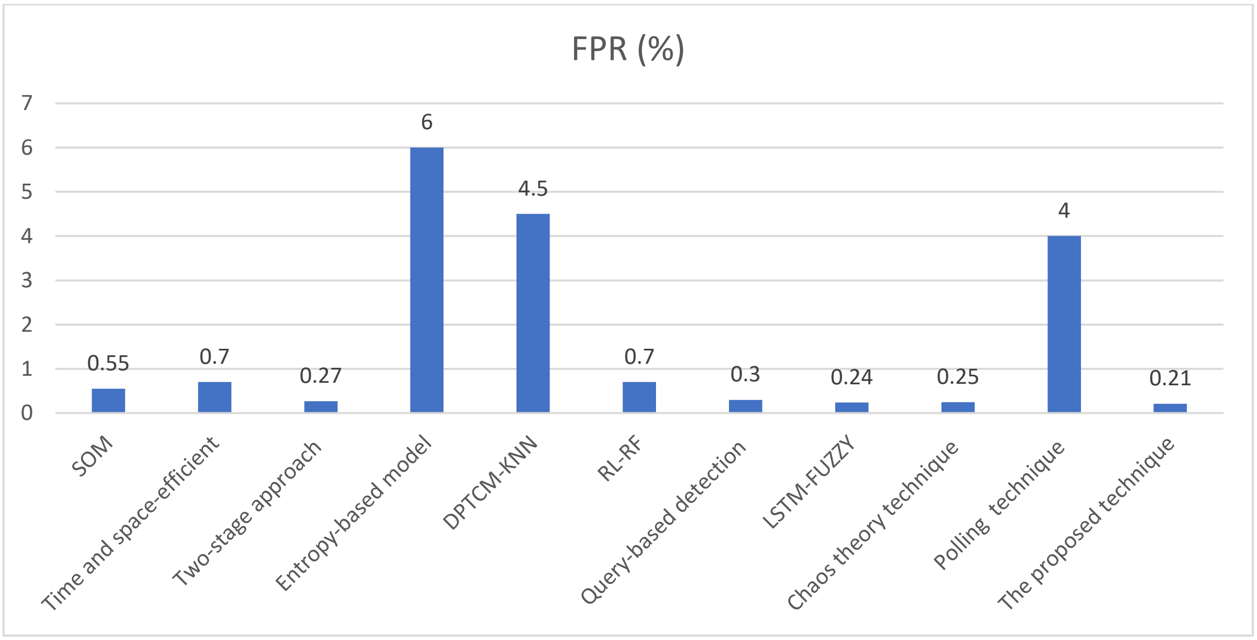


Figure 5.7: FPR comparison of different methods for control plane detection

Table (5.4) and figure (5.8) show f_measure comparisons among K-FKNN, Feature selection method [34], Polling detection technique [36], and the proposed technique. It can be shown from the results that the proposed technique in this paper gives the best results compared with other techniques.

Table 5.4: F_measure comparison of different algorithms for control plane detection

Algorithm	F_measure (%)
SOM	96.4
K-FKNN	97.65
Two-stage approach [23]	98
Feature selection method [34]	97.7
DPTCM-KNN	97.3
Query-based detection	96.8
LSTM-FUZZY	99.8
Polling detection technique [36]	88.1
The proposed technique	99.83

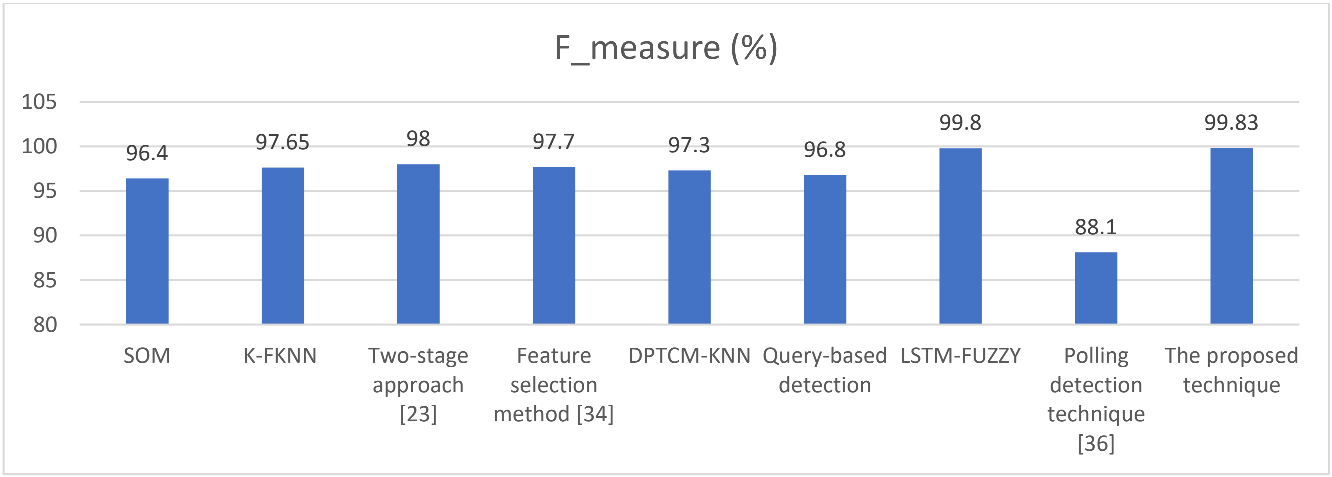


Figure 5.8: F_measure comparison of different algorithms for control plane detection

Table 5.5 and figure (5.9) express detection time for SOM [26], Two-stage approach [23], DPTCM-KNN [24], Query-based detection [129], and LSTM-FUZZY [28], and the introduced technique as a result of implementing all of these techniques in our environment.

Table 5.5: Detection time comparison of different algorithms for control plane detection

Algorithm	Detection time (Sec)
SOM	0.125
Two-stage approach [23]	0.082
DPTCM-KNN	0.050
Query-based detection	0.048
LSTM-FUZZY	0.095
The proposed technique	0.042

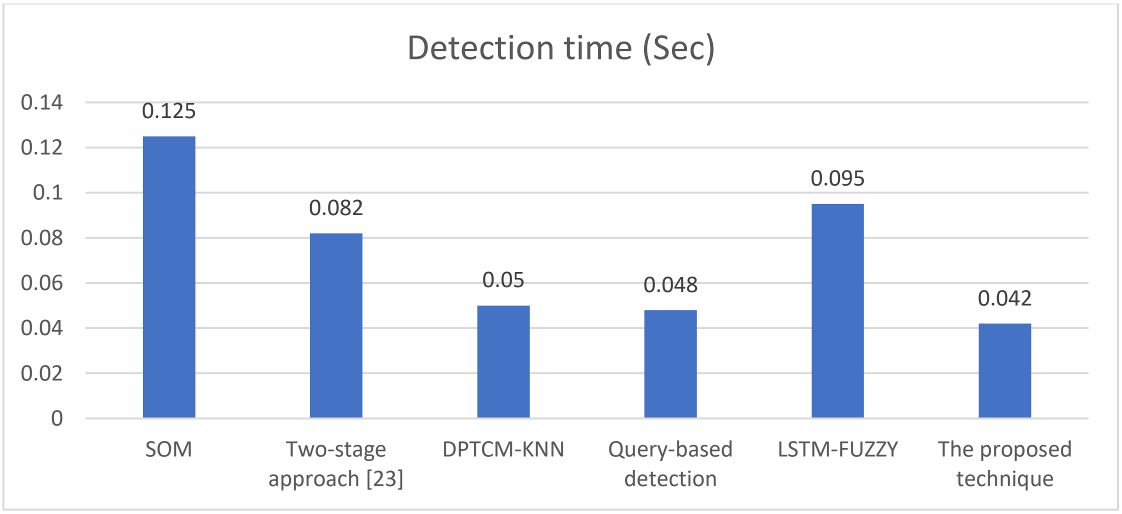


Figure 5.9: Detection time comparison of different algorithms for control plane detection

For the data plane, through the experiments, a threshold was used for the switch's average arrival bit rate with an unknown destination address. 30,000 packets were sent from network hosts during the detection process, with 14200 DDoS attack packets sent from 5 compromised end systems and 15800 regular packets sent from the other hosts. There are 15778 TN packets, according to experimental results. There are also 14 FN packets, 22 FP packets, and 14186 TP packets. As a result, numerous performance indicators were employed to assess how well the introduced technique was performed concerning the research objectives. As a result, we calculated the detection accuracy, detection rate, false alarm rate, and f measure. As a mitigation technique, the trust value is updated, the compromised end systems were blocked when their trust reaches 0, and their information was logged to complete the research objectives.

Table (5.6) as well as figure (5.10), figure (5.11), and figure (5.12) show comparisons of accuracy, detection rate, and FPR for DDoS detection framework [29], automated DDoS detection [30], ML-based detection [37], StateSec [38], FloodDefender [25], and OverWatch [31] as well as the technique introduced in this paper. It can be shown from the results that the proposed technique in this paper gives the best results compared with other techniques.

Table 5.6: Accuracy, detection rate, and FPR comparisons of different algorithms for data plane detection

Algorithm	Accuracy (%)	Detection rate (%)	FPR (%)
DDoS detection framework	98.85	98.47	0.97
Automated DDoS detection	98	98	NA
ML-based detection	98.5	NA	NA
StateSec	NA	85	0.1
FloodDefender	NA	98	2
OverWatch	NA	94	NA
Proposed technique	99.88	99.9	0.1

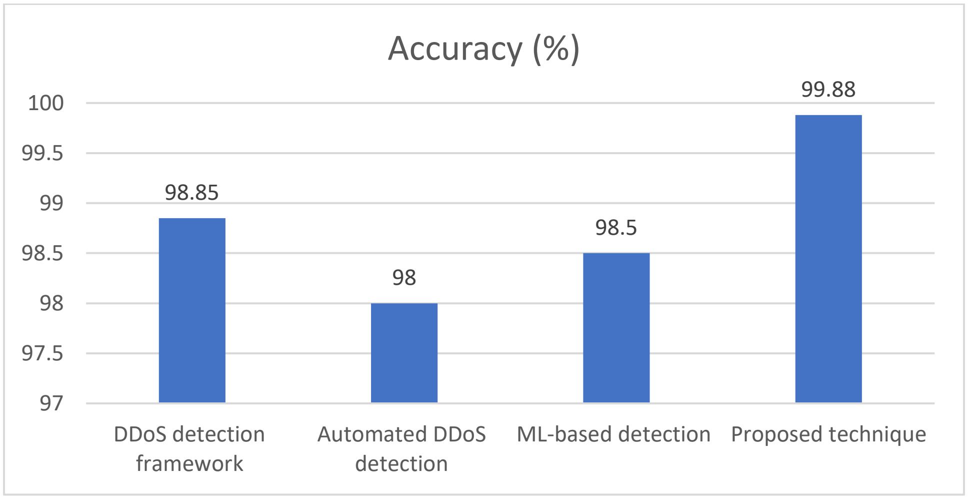


Figure 5.10: Accuracy comparison of different algorithms for data plane detection

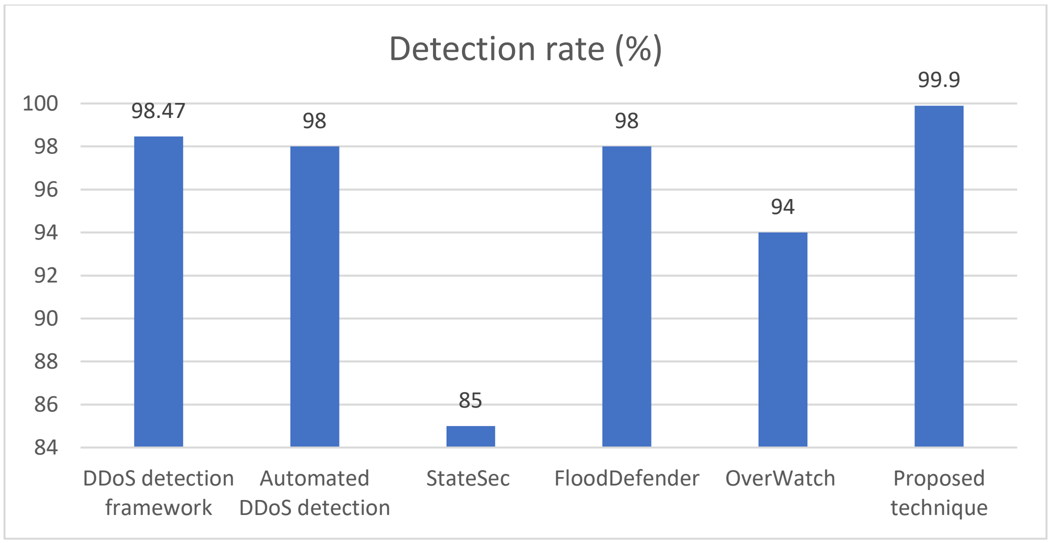


Figure 5.9: Detection rate comparison of different algorithms for data plane detection

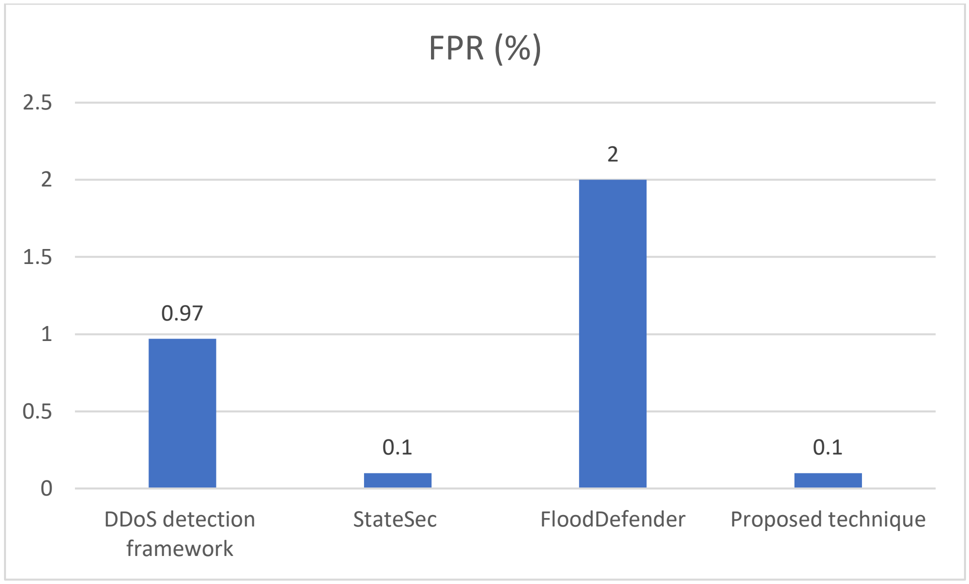


Figure 5.10: FPR comparison of different algorithms for data plane detection

Table (5.7) and figure (5.13) show f_measure comparisons among different algorithms and the proposed technique. The results suggest that the proposed technique in this study produces the greatest outcomes when compared to other techniques.

Table 5.7: F_measure comparison of different algorithms for data plane detection

Algorithm	F_measure (%)
Automated DDoS detection	98
OverWatch	93
Proposed technique	99.87

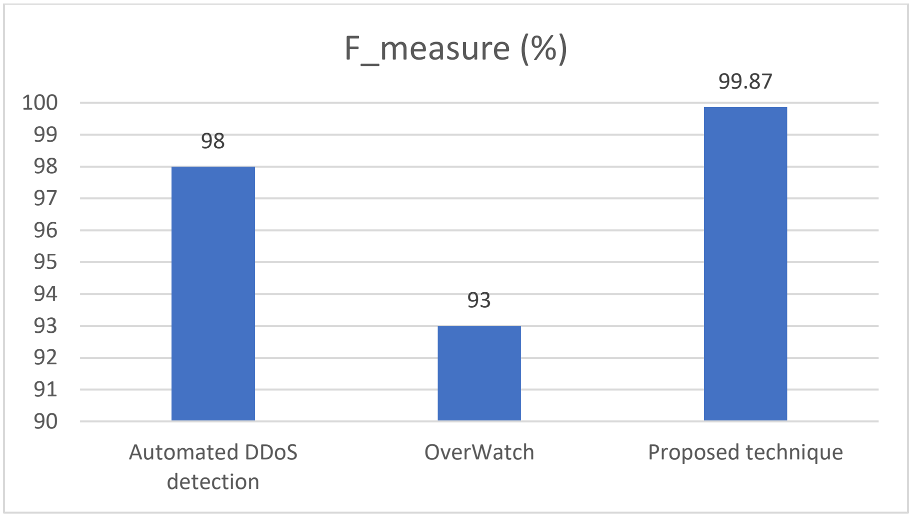


Figure 5.11: F_measure comparison of different algorithms for data plane detection

5.3 Simulation Results for Performance of Queuing Analysis Models

The simulation setup, experiments, and result analysis for the proposed 7-D queueing models which can be used roughly to differentiate normal traffic from DDoS attack traffic in SDN are explained.

5.3.1 Experimental and Simulation Setup for Queueing Models

We use a Java simulator to build the required SDN network with its parameters, on a system running 64-bit Windows 10 with Intel(R) Core (TM) i7-9700 CPU @ 3.00GHz CPU and memory of 7.9 GB. The network contains a root controller, in addition to 3 local controllers and 9 switches. We launch

10,000 simulation runs, with each run taking 3s. The results are obtained as the average values of the 10,000 simulation runs. Also, analytical results are calculated using the developed equations in chapter (4). The results are compared against the model developed in [40]. Through the simulation and analytical analysis, we denoted γ_{TCPS}^{i} and γ_{UDPS}^{i} as γ_{FS}^{i} because they are both assigned the same value. Similarly, γ_{TC}^{a} and γ_{UC}^{a} as γ_{BC}^{a}, and γ_{TCPC}^{a} and γ_{UDPC}^{a} as γ_{NC}^{a}. Also, λ_{TR} and λ_{UR} as λ_{BR}. Table (5.8) lists the basic parameters used in the simulations and analytical models.

Table 5.8: Basic parameters in the simulation and the analysis

Parameter	Value
Flow arrival rate, γ_{FS}^{i}	950 (946:955) flows/sec
Packet arrival rate per flow, γ_{P}^{i}	1,000 (950:1,040) packets/sec
Switch average service rate	1,000,000 (980,000:1,016,000) packets/sec
Local controller average service rate	10,000,000 (8,000,000:11,600,000) packets/sec
Root controller average service rate, μ_R	25,000,000 packets/sec
Total arrival rate to local controller, γ_{BC}^{a}	8,400,000 packets/sec
Average new flow arrival rate to local controller, γ_{NC}^{a}	8,000,000 packets/sec
Root controller average arrival rate,	9,000,000 packets/sec

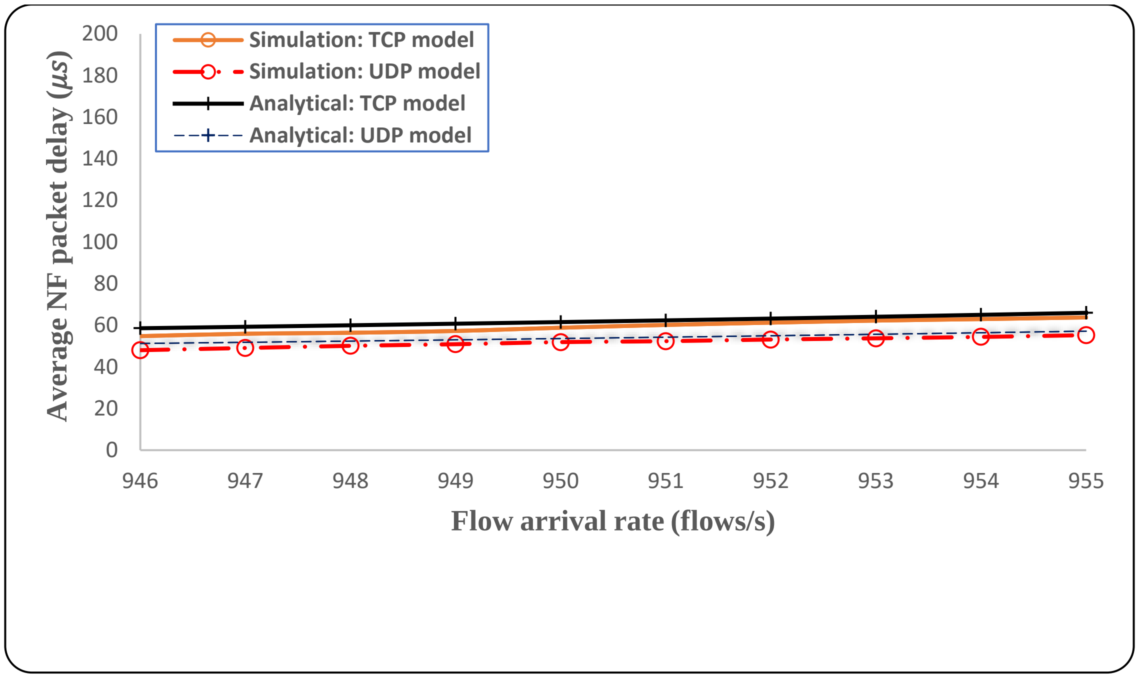
λ_{BR}	
Propagation delay, D_{Prop}	10 μs
Switch queue size	200 packets
Local controller queue size	600 packets
Root controller queue size	1,200 packets

5.3.2 Result Analysis for Queueing Models

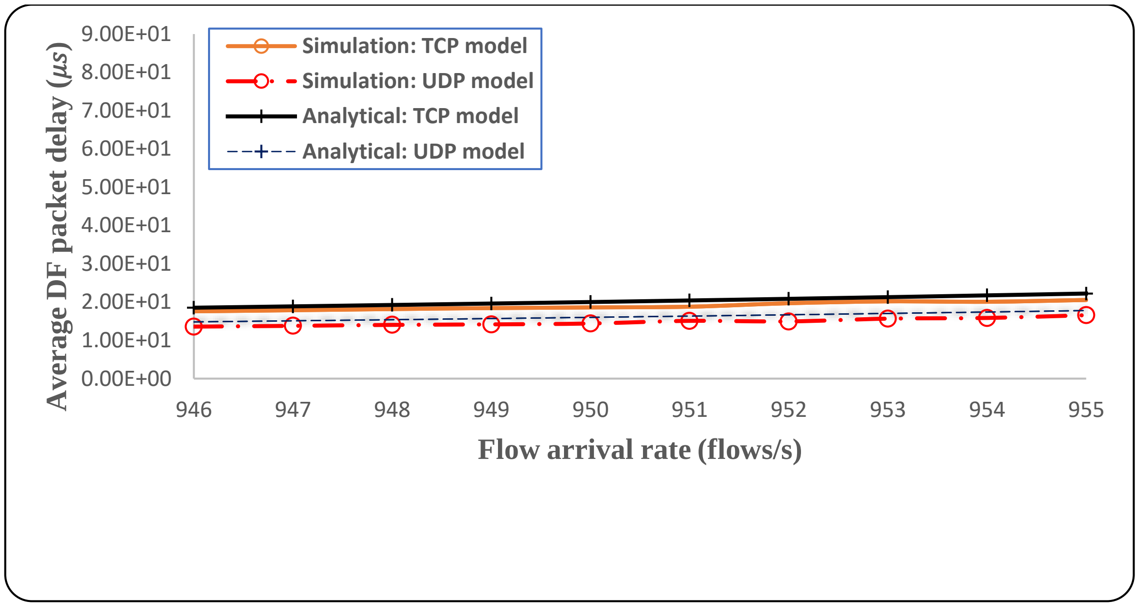
The obtained results from simulating the developed models over 10,000 simulation runs are expressed and compared against the analytical results for the packet's mean delay and loss probability for the models. Effects of different parameters such as average flow arrival rate, packet average arrival rate per flow, switch average service rate, and controller average service rate on performance measures are illustrated.

5.3.2.1 Effect of Average Flow Arrival Rate, γ_{FS}^{i}

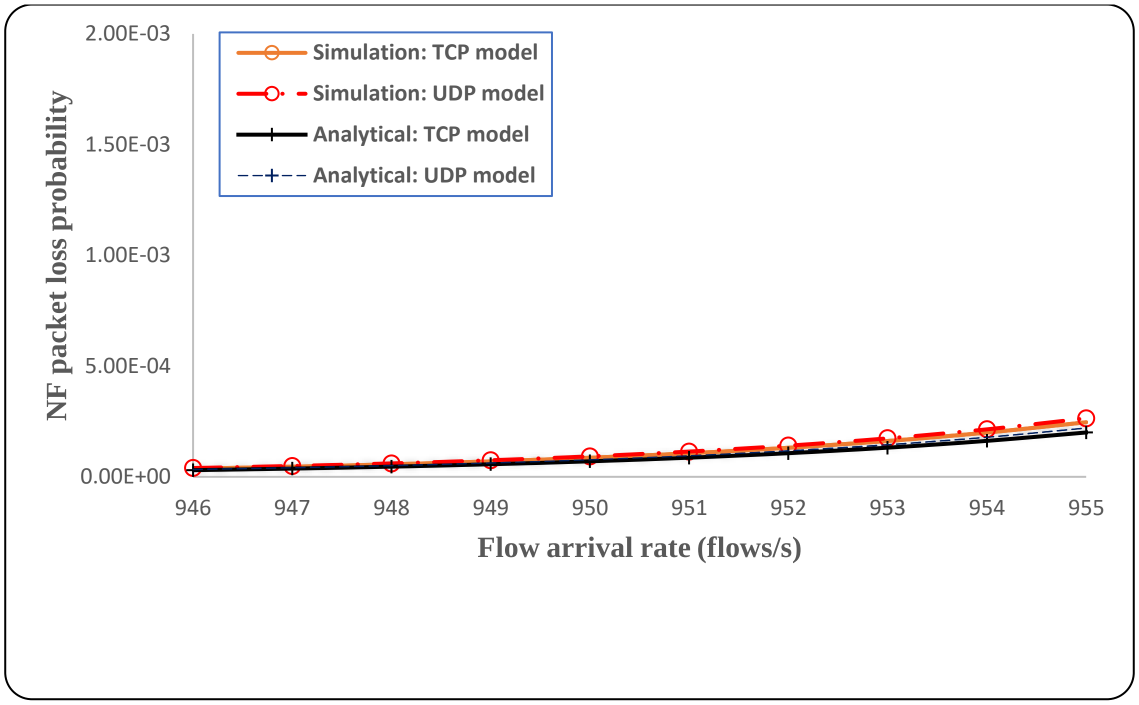
The effects of γ_{FS}^{i} on the mean packet delay and the loss probability are described in figure 5.13(a-d). Simulation and analytical results for both TCP and UDP models are compared. It can be shown from the figure that when γ_{FS}^{i} increases, the mean transfer and loss probability of either NF or data (DF) packet increase. This is because in both models, when γ_{FS}^{i} increases more new flows arrive at the switch. This results in forwarding more packets to the controller. Normally, such packets suffer more delays because of being forwarded to the controller. Also, increasing γ_{FS}^{i} causes large numbers of packets to be stored in the buffers of the switch and the controller which causes an overflow in these buffers. Also, the packet may need to be stored in multiple queues in such cases. Hence, packet loss probability increases accordingly. It can be noted that the loss probability for both DF and NF packets is close to 0 when the traffic intensity is small (less than 0.5). However, as soon as the traffic intensity increases and approaches 1 due to the increase of γ_{FS}^{i}, the loss probability starts increasing as well. Furthermore, when the traffic intensity in the switch or the controller approaches or reaches 1, this may indicate DDoS attacks. In such a case, the network is flooded with attack packets, which may consume and exhaust different network resources. Hence, network resources cannot provide network services to legitimate users. Also, the average packet delay and loss probability increase greatly accordingly. It can be noted that simulation results have values larger than analytical results, which are improved using more simulation runs to approach analytical ones. Also, the TCP model has average packet delays larger than the UDP model. This is because TCP includes initiating TCP connections, flow and congestion control, and larger packet headers length. The UDP model has a larger loss probability than TCP because as expressed earlier in section (4.2), UDP enables sending many packets back-to-back at the start of transmission. This causes numerous packets to be forwarded to the controller which means more packets must be stored in the queues which increases the loss probability accordingly.



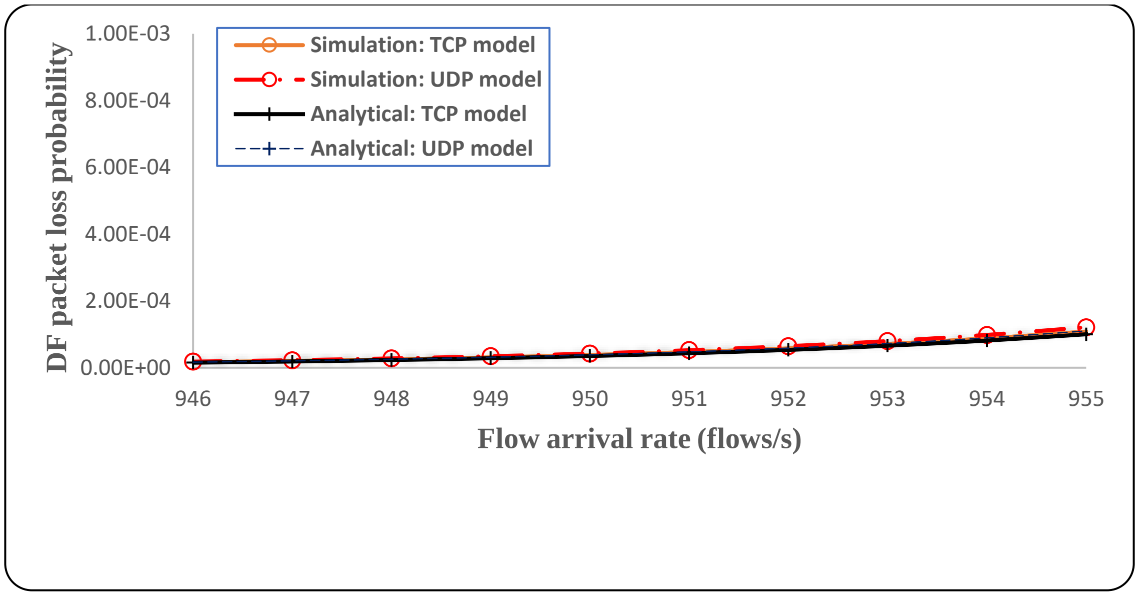
(a) Effect of γ_{FS}^{i} on NF packet delay



(b) Effect of γ_{FS}^{i} on DF packet delay



(c) Effect of γ_{FS}^{i} on NF packet loss probability



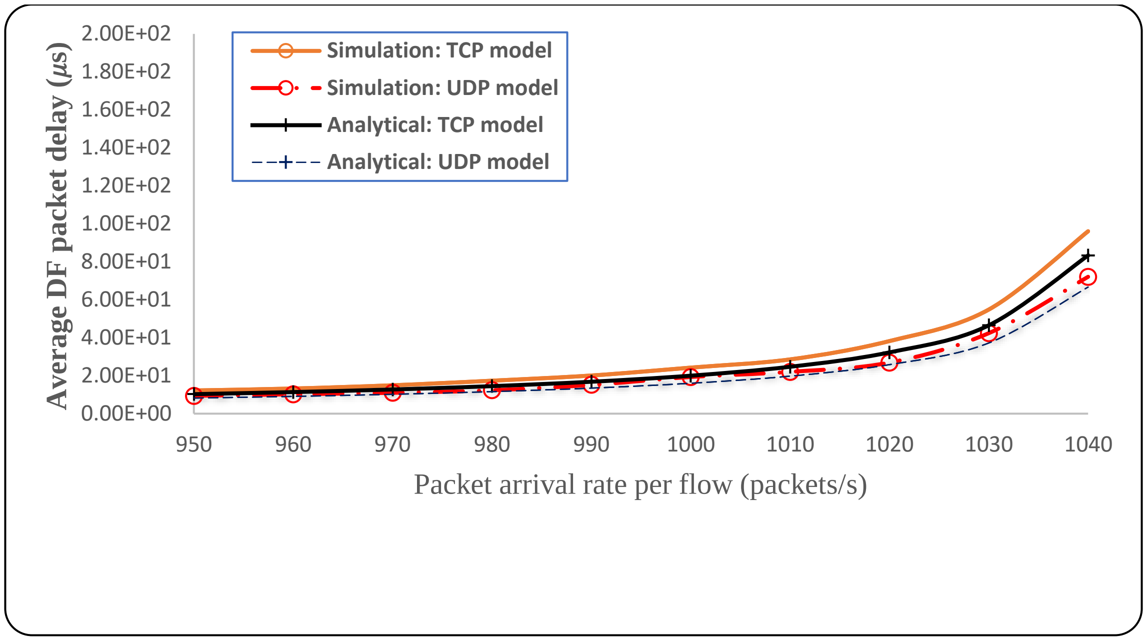
(d) Effect of γ_{FS}^{i} on DF packet loss probability

Figure 5.14: Effect of average flow arrival rate

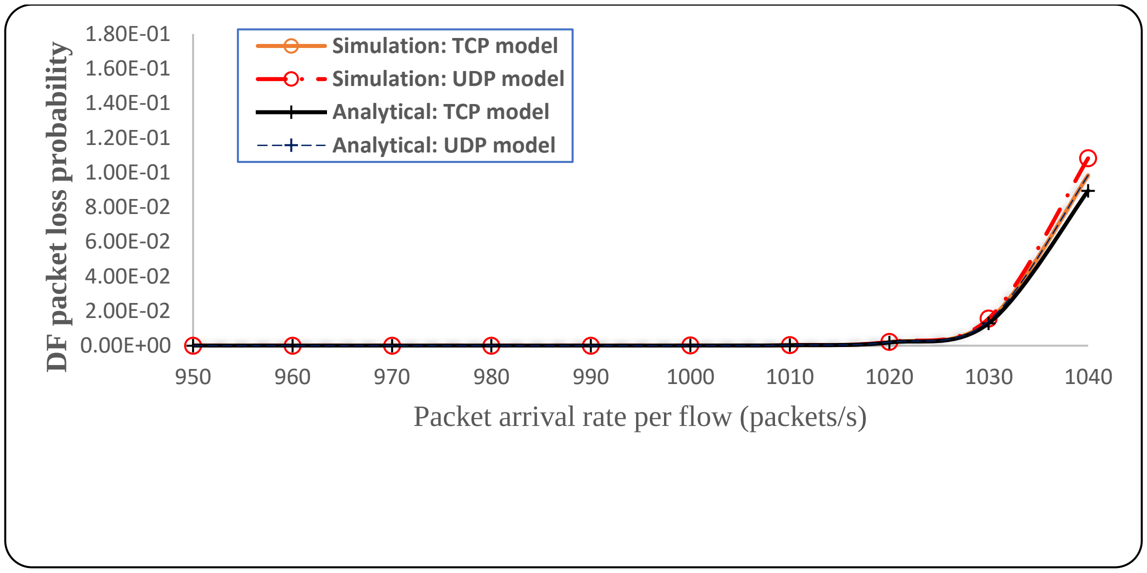
5.3.2.2 Effect of Packet Average Arrival Rate per Flow, γ_P^i

Figure5.14(a, b) shows how γ_p^i influences the mean TCP DF packet delay, D_{TCP}^{DF}, and the mean UDP DF packet delay, D_{UDP}^{DF} in addition to the probability of TCP DF packet loss, L_{TCP}^{DF}, and the probability of UDP DF packet loss, L_{UDP}^{DF}. Simulation and analytical results for both TCP and UDP models are compared. When γ_p^i takes large values, such that the traffic intensity approaches 1, this can indicate that it is DDoS attack traffic. Otherwise, it is usually considered normal traffic.

It can be shown from the figure that as γ_p^i increases, the mean transfer delay of a DF packet increases for both TCP and UDP models increase as well. Increasing γ_p^i means larger numbers of packets in the flow which increases the total arrival rate of packets. This means that there are large numbers of packets that arrive in switch *i* and require the service. Hence, a DF packet's queuing delay would increase since it must wait for more packets that precede it and require the service. Also, these packets need more space to be queued which exhausts the buffer and may increases packet loss. In addition, simulation results have values larger than analytical results, which are improved using more numbers of simulation runs to approach analytical ones. Additionally, average packet delays in the TCP model are longer than in the UDP model. This is due to TCP's support for TCP connection establishment, flow control, congestion control, and longer packet headers. Because UDP allows for the sending of numerous packets back-to-back at the commencement of transmission, the UDP model has a higher loss probability than TCP. Due to the numerous packets being forwarded to the controller, more packets must be kept in the queue, which raises the loss probability.



(a) Effect of γ_p^i on DF packet delay

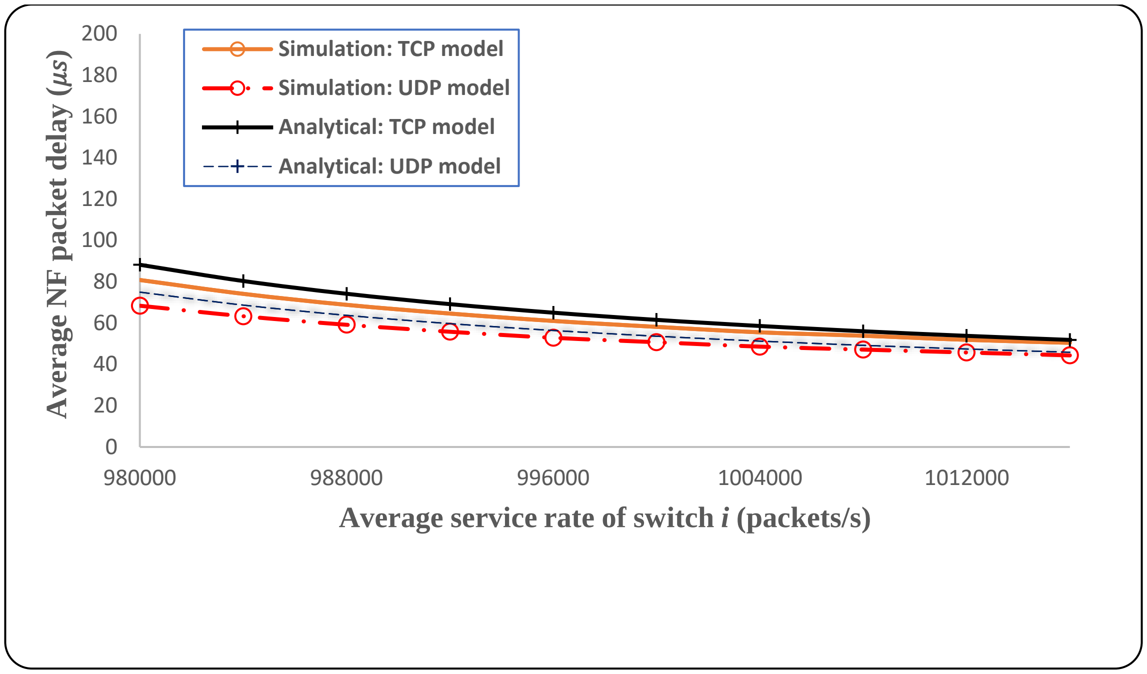


(b) Effect of γ_p^i on DF packet loss probability

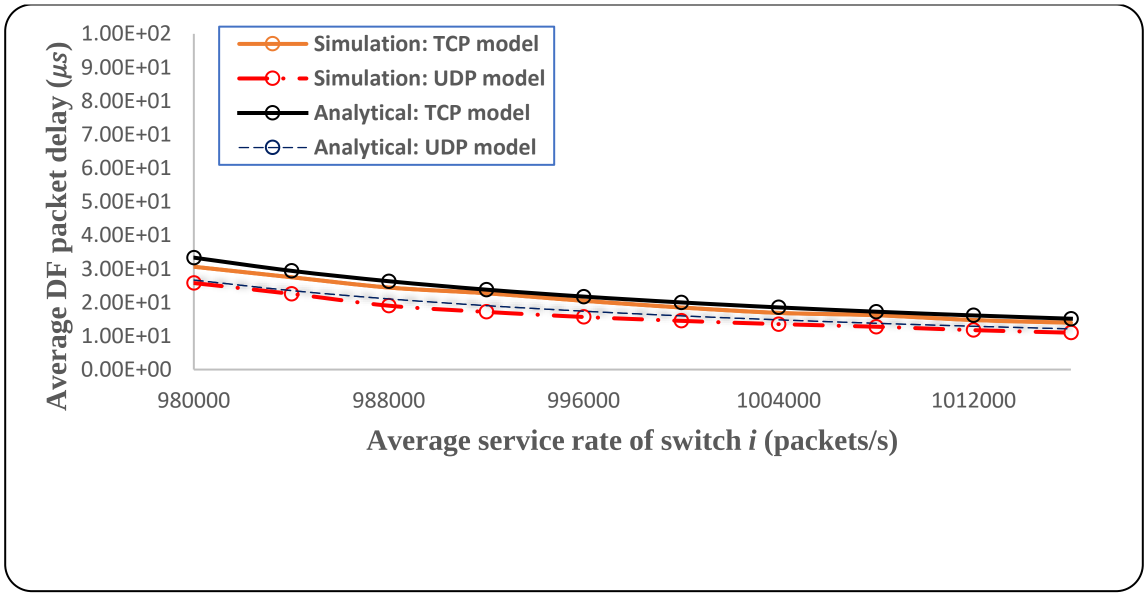
Figure 5.15: Effect of packet average arrival rate per flow

5.3.2.3 Effect of Switch Average Service Rate, μ_S^i

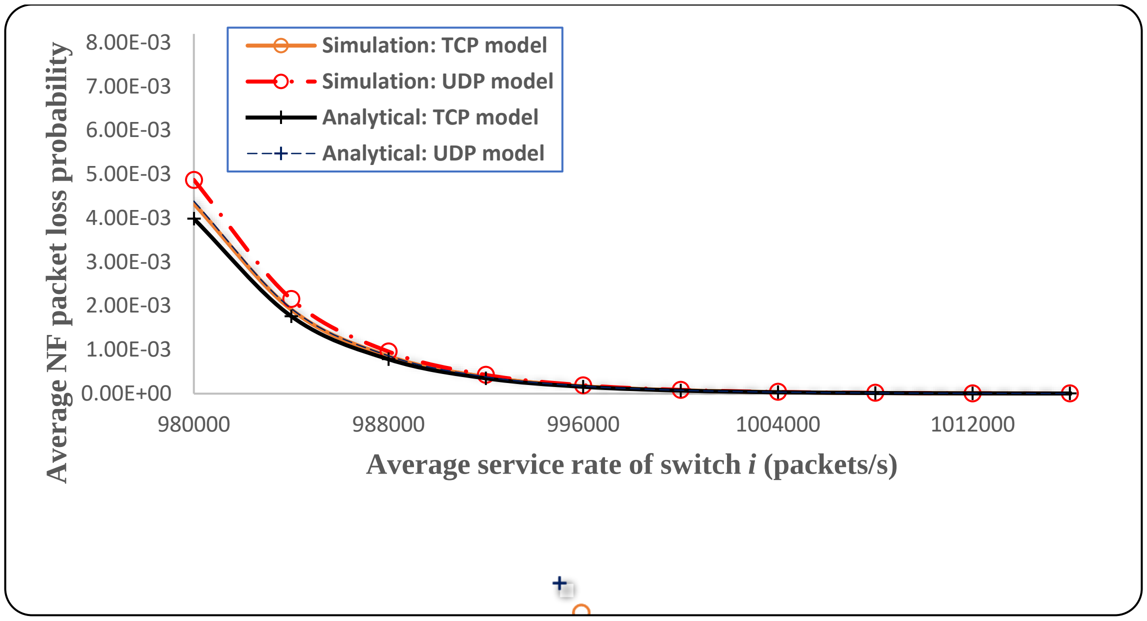
Figure5.15(a-d) shows how μ_S^i influences the mean packet delay and loss probability for both NF and DF packets. Simulation and analytical results for both TCP and UDP models are compared. It can be shown from the figure that as μ_S^i increases, the mean transfer delay of a DF packet decreases for both TCP and UDP models. This decrease occurs since the power of switch i increases, which results in processing more packets. Therefore, the delay of a packet at switch i, D_{TCPS}^i (or D_{UDPS}^i) decreases. Also, the loss probability is reduced since switch i can serve more packets, decreasing the amount of time needed by each packet to be kept in the queue of switch i. Hence, the mean length of a queue for switch i, $\overline{q_{TCPS}^i}$ (or $\overline{q_{UDPS}^i}$) decreases. This decreases the discarding probability of a packet in switch i, L_{TCPS}^i (or L_{UDPS}^i). Moreover, increasing μ_S^i means reducing the traffic intensity which means the network is stable and it is normal traffic. From the obtained results, simulation curves have values that are slightly different from analytical ones, which are improved using more simulation runs to approach analytical curves. Additionally, compared to the UDP model, the TCP model has longer average packet delays. This is because TCP supports opening new TCP connections, flow control, congestion control, and longer packet headers. Due to UDP's capacity to send numerous packets back-to-back at the beginning of transmission, the UDP model has a higher loss probability than TCP. This results in additional packets being forwarded to the controller, which necessitates the storage of more packets in the queues and consequently raises the loss probability.



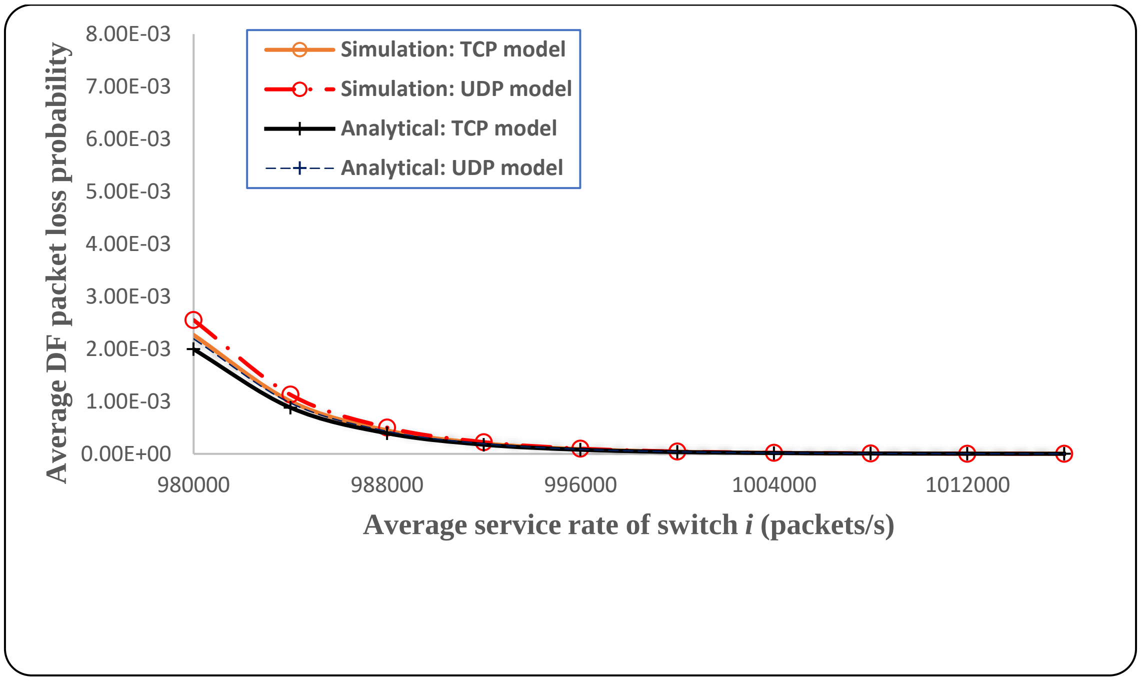
(a) Effect of μ_S^i on NF packet delay



(b) Effect of μ_S^i on DF packet delay



(c) Effect of μ_S^i on NF packet loss probability



(d) Effect of μ_S^i on DF packet loss probability

Figure 5.16: Effect of switch average service rate

5.3.2.4 Effect of Controller Average Service Rate, μ_C^a

Figure(5.16) shows how μ_C^a influences the mean TCP NF packet delay, D_{TCP}^{NF} and the mean UDP NF packet delay, D_{UDP}^{NF}. Simulation and analytical results for both TCP and UDP models are compared. It can be shown from the figure that when μ_C^a increases, the mean transfer delay of an NF packet decreases for both TCP and UDP models. This reduction occurs since the power of a controller increases. This results in serving more packets and decreasing the time needed for each packet to be kept in the controller's queue, D_{TCPC}^a (or D_{UDPC}^a). Hence, the average packet delay is decreased accordingly. Furthermore, increasing μ_C^a can reduce the traffic intensity which means the traffic is not a DDoS attack. Simulation results have smaller values than analytical results, which are improved using more simulation runs to approach analytical ones. Also, the TCP model has average packet delays larger than the UDP model. This is because TCP includes initiating TCP connections, flow and congestion control, and larger packet headers length. The UDP model has a larger loss probability than TCP because UDP enables sending many packets back-to-back at the start of transmission. This causes multiple packets to be forwarded to the controller which means more packets must be stored in the queues which increases the loss probability accordingly.

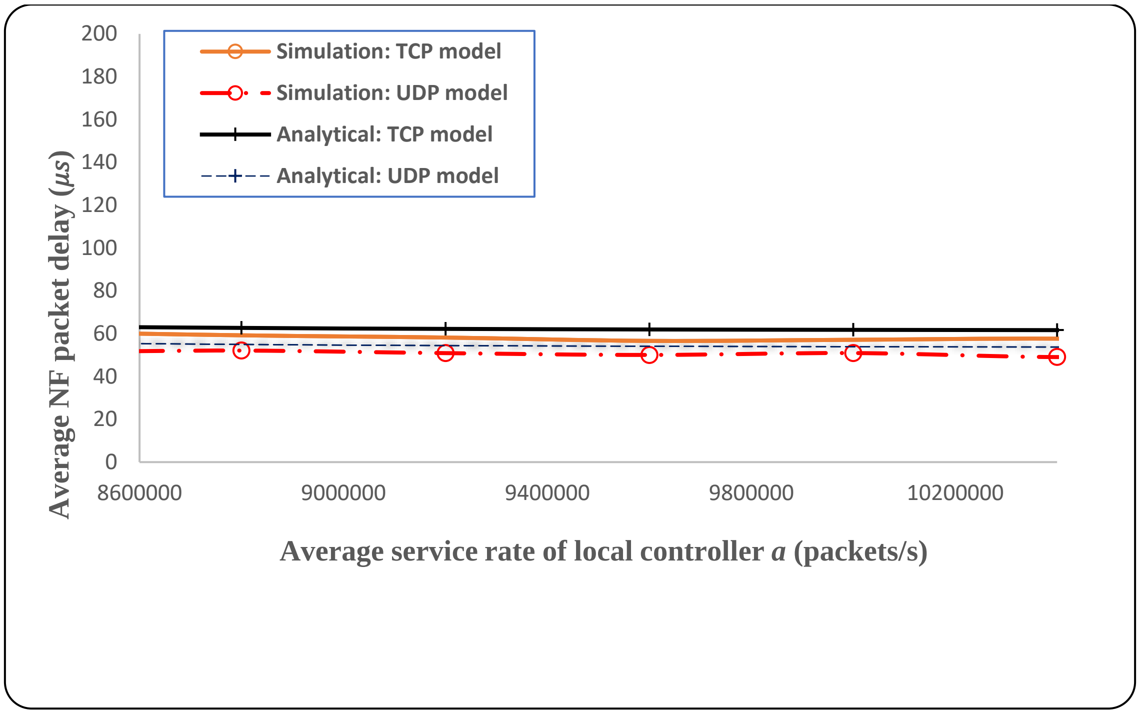


Figure 5.127: Effect of controller average service rate

5.3.2.5 Comparison with other Methods

The obtained results show that the developed TCP and UDP models give an accurate analysis of traffic in SDN architectures and can be roughly used to detect DDoS attacks. Accuracy is calculated to measure the difference between simulation and analytical results in the model. The obtained results prove that the simulated models can behave efficiently and match analytical ones. Over 10,000 simulation runs, the average packet delay accuracy is 91 % and the average loss probability accuracy is 91.5 %. However, the model introduced in [40] gives an accuracy of 90 % for the average packet delay and 90.5 % for the average loss probability accuracy. Constructing and adopting 7-D states prove to give effective and accurate results in network traffic modeling and analysis in the SDN. In addition, the use of multiple OF switches can better reflect the actual use and the real case of the SDN architecture rather than if the network has only one switch. Using multiple controllers and multiple OF switches can support network scalability and allow more hosts to connect to the network. In addition, the use of multiple controllers can also improve network security, since using multiple controllers may support the backup capability.

More simulation runs are performed which can improve the accuracy of the proposed analytical model. A comparison between the developed model and the model introduced in [40] is depicted in table (5.8). The obtained results prove that the accuracy of the proposed model increases as the number of simulation runs increases. Hence, the simulation results can better match the analytical ones.

Table 5.9: Comparison between the introduced model and related model over simulation runs

Simulation runs	10,000	15,000	20,000
Average packet delay accuracy in [40]	90%	90.3%	90.8%
Average loss probability accuracy in [40]	90.5%	91%	91.5%
Average packet delay accuracy in the introduced model	91%	91.5%	92%
Average loss probability accuracy in the introduced model	91.5%	92%	92.7%

Chapter 6
Conclusion and Future Work

6 Conclusion and Future Work

The conclusion of the proposed thesis in addition to the future and further work for the developed research works are expressed in this section.

6.1 Conclusion

DDoS attacks can be classified as the most severe attacks that can threaten the functionality and security of SDN architectures. It may target dropping the SDN controller which is the main component in SDN infrastructure since it represents the centralized control plane of the network. DDoS attacks may target exhausting the forwarding devices in the data plane as well. In this thesis, ML-based detection techniques are used to detect DDoS attacks against the SDN controller and the data plane by utilizing proposed features. The proposed features are extracted from packet headers in addition to the traffic statistics. Advanced features such as unknown destination addresses, packets inter-arrival time, TLP header, and ToS header are used in the detection technique in the control plane. The utilized features in the data plane technique include the switch's stored capacity, the average rate of packets with unknown destination IP addresses, the IP Options header field, and the average number of flows in the switch. SVM is preferred to be used in the proposed work for creating the detection model due to its obtained high accuracy compared with other ML-based algorithms. The proposed techniques prove to detect SDN DDoS attacks with high accuracy, detection rate, f_measure, and low FPR.

This thesis can also propose 7-D state models with transitions to model SDN TCP and UDP flows, unlike other models that are used in previous work which use 3-D or 4-D states. The developed models in the current work have multiple controllers and multiple OF switches to better reflect the actual case of using SDN networks, unlike the related models which use only one controller or one switch. The network security and scalability are improved by using multiple controllers and multiple switches. The developed models consider flow-level arrivals to get more accurate knowledge about modeling the SDN traffic. All controllers and switches have finite-capacity buffers unlike models developed in related works, which may use queues with infinite capacities. The proposed work uses the mean packet transfer delay and packet loss probability to measure the performance of the developed models. Also, the developed models could be used roughly to differentiate normal traffic from DDoS attack traffic by considering that the attack traffic has a Poisson input distribution with high average arrival rates. The constructed simulation experiments prove that their results are close to the results obtained analytically from the developed theoretical models. The results prove that the proposed modeling is accurate compared to the recent work.

6.2 Future Work

As a future development and enhancement plan of the proposed work, some points are suggested as follows:

1. Working on preventing a DDoS attack occurence in SDN architectures in the control plane.
2. Detecting other types of attacks that make use of different SDN vulnerabilities.
3. Differentiating packets that are sent back by the controller to the switch from packets that arrive at the switch as new arrivals from end systems in the queueing models.
4. Developing a more accurate mathematical model for SDN DDoS attack traffic.

List of Publications

List of Publications

1. Waheed G. Gadallah, Nagwa M. Omar, and Hosny M. Ibrahim, "Machine Learning-based Distributed Denial of Service Attacks Detection Technique using New Features in Software-defined Networks", International Journal of Computer Network and Information Security(IJCNIS), Vol.13, No.3, pp.15-27, 2021.
2. Waheed G. Gadallah, Hosny M. Ibrahim, and Nagwa M. Omar, "A Seven-Dimensional State Flow Traffic Modelling for Multi-controller Software-Defined Networks Considering Multiple Switches", Elsevier's Journal of Computer Communications, 2022.* (Accepted).
3. Waheed G. Gadallah, Hosny M. Ibrahim, and Nagwa M. Omar, "A Hybrid Three-Phases Machine Learning Technique to Detect Distributed Denial of Service Attacks against Data Plane in Software-Defined Networks", Journal of Computer Security, 2022.* (Under review).

References

References

[1] M. Priyadarsini and P. Bera, “Software defined networking architecture, traffic management, security, and placement: A survey,” Comput. Networks, vol. 192, p. 108047, 2021.

[2] S. K. Keshari, V. Kansal, and S. Kumar, “A systematic review of quality of services (QoS) in software defined networking (SDN),” Wirel. Pers. Commun., vol. 116, no. 3, pp. 2593–2614, 2021.

[3] E. R. Jimson, K. Nisar, and M. H. A. Hijazi, “The state of the art of software defined networking (SDN) issues in current network architecture and a solution for network management using the SDN,” Int. J. Technol. Diffus., vol. 10, no. 3, pp. 33–48, 2019.

[4] A. P. Fajar and T. W. Purboyo, “A Survey Paper of Distributed Denial-of-Service Attack in Software Defined Networking (SDN),” Int. J. Appl. Eng. Res. ISSN, vol. 13, no. 1, pp. 973–4562, 2018, [Online]. Available: http://www.ripublication.com.

[5] W. Zhijun, L. Wenjing, L. Liang, and Y. Meng, “Low-rate DoS attacks, detection, defense, and challenges: A survey,” IEEE access, vol. 8, pp. 43920–43943, 2020.

[6] T. Ubale and A. K. Jain, “Survey on DDoS attack techniques and solutions in software-defined network,” in Handbook of computer networks and cyber security, Springer, 2020, pp. 389–419.

[7] P. J. Beslin Pajila and E. Golden Julie, “Detection of DDoS attack using SDN in IoT: A survey,” in Intelligent Communication Technologies and Virtual Mobile Networks, pp. 438–452, 2019.

[8] M. A. Alarqan, Z. F. Zaaba, and A. Almomani, “Detection mechanisms of DDoS attack in cloud computing environment: A survey,” in International Conference on Advances in Cyber Security, pp. 138–152, 2019.

[9] A. S. Shukla and R. Maurya, “Entropy-based anomaly detection in a network,” Wirel. Pers. Commun., vol. 99, no. 4, pp. 1487–1501, 2018.

[10] G. Li and J. J. Jung, “Entropy-based dynamic graph embedding for anomaly detection on multiple climate time series,” Sci. Rep., vol. 11, no. 1, pp. 1–10, 2021.

[11] A. Howedi, A. Lotfi, and A. Pourabdollah, "An entropy-based approach for anomaly detection in activities of daily living in the presence of a visitor," Entropy, vol. 22, no. 8, p. 845, 2020.

[12] M. Ahmed and A. N. Mahmood, "Network traffic pattern analysis using improved information theoretic co-clustering based collective anomaly detection," in International conference on security and privacy in communication networks, pp. 204–219, 2014.

[13] S. A. Medjahed, M. Ouali, T. A. Saadi, and A. Benyettou, "An Optimization-Based Framework for Feature Selection and Parameters Determination of SVMs," Int. J. Inf. Technol. Comput. Sci., vol. 7, pp. 1–9, 2015.

[14] L. Zhao and G. Shi, "A trajectory clustering method based on Douglas-Peucker compression and density for marine traffic pattern recognition," Ocean Eng., vol. 172, pp. 456–467, 2019.

[15] S. Omar and H. H. Jebur, "Machine Learning Techniques for Anomaly Detection : An Overview," vol. 79, no. 2, pp. 33–41, 2013.

[16] I. G. A. Poornima and B. Paramasivan, "Anomaly detection in wireless sensor network using machine learning algorithm," Comput. Commun., vol. 151, pp. 331–337, 2020.

[17] M. Nawir, A. Amir, N. Yaakob, and O. B. Lynn, "Effective and efficient network anomaly detection system using machine learning algorithm," Bull. Electr. Eng. Informatics, vol. 8, no. 1, pp. 46–51, 2019.

[18] R. Wang, Z. Jia, and L. Ju, "An entropy-based distributed DDoS detection mechanism in software-defined networking," Proc. - 14th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. Trust, vol. 1, pp. 310–317, 2015.

[19] N. Z. Bawany, J. A. Shamsi, and K. Salah, "DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions," Arab. J. Sci. Eng., vol. 42, no. 2, pp. 425–441, 2017.

[20] "Classification Algorithms in Machine Learning." [Online]. Available: https://medium.com/datadriveninvestor/classification-algorithms-in-machine-learning-85c0ab65ff4, 2019.

[21] M. Mohammed, M. B. Khan, and E. B. M. Bashier, Machine learning: algorithms and applications. Crc Press, 2016.

[22] A. Singh, N. Thakur, and A. Sharma, “A review of supervised machine learning algorithms,” 3rd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 1310–1315, 2016.

[23] M. Latah and L. Toker, “A novel intelligent approach for detecting DoS flooding attacks in software-defined networks,” Int. J. Adv. Intell. Informatics, vol. 4, no. 1, pp. 11–20, 2018.

[24] H. Peng, Z. Sun, X. Zhao, S. Tan, and Z. Sun, “A detection method for anomaly flow in software defined network,” IEEE Access, vol. 6, pp. 27809–27817, 2018.

[25] G. A. O. Shang, P. Zhe, X. Bin, H. U. Aiqun, and R. E. N. Kui, “FloodDefender : Protecting Data and Control Plane Resources under SDN-aimed DoS Attacks,” 2017.

[26] J. Ashraf and S. Latif, “Handling intrusion and DDoS attacks in Software Defined Networks using machine learning techniques,” in 2014 National Software Engineering Conference, pp. 55–60, 2014.

[27] T. Wang and H. Chen, “SGuard: A lightweight SDN safe-guard architecture for DoS attacks,” China Commun., vol. 14, no. 6, pp. 113–125, 2017.

[28] M. P. Novaes, L. F. Carvalho, J. Lloret, and M. L. Proença, “Long Short-Term Memory and Fuzzy Logic for Anomaly Detection and Mitigation in Software-Defined Network Environment,” IEEE Access, vol. 8, pp. 83765–83781, 2020.

[29] A. Alshamrani, A. Chowdhary, S. Pisharody, D. Lu, and D. Huang, “A defense system for defeating DDoS attacks in SDN based networks,” MobiWac 2017 - Proc. 15th ACM Int. Symp. Mobil. Manag. Wirel. Access, Co-located with MSWiM, no. July 2018, pp. 83–92, 2017.

[30] F. Musumeci, A. C. Fidanci, F. Paolucci, F. Cugini, and M. Tornatore, “Machine-Learning-enabled DDoS attacks detection in P4 programmable networks,” J. Netw. Syst. Manag., vol. 30, no. 1, pp. 1–27, 2022.

[31] B. Han, X. Yang, Z. Sun, J. Huang, and J. Su, "OverWatch: A cross-plane DDoS attack defense framework with collaborative intelligence in SDN," Secur. Commun. Networks, vol. 2018, 2018.

[32] C. Xu, H. Lin, Y. Wu, X. Guo, and W. Lin, "An SDNFV-Based DDoS Defense Technology for Smart Cities," IEEE Access, vol. 7, pp. 137856–137874, 2019.

[33] Y. Xu, H. Sun, F. Xiang, and Z. Sun, "Efficient DDoS Detection Based on K-FKNN in Software Defined Networks," IEEE Access, vol. 7, pp. 160536–160545, 2019.

[34] H. Polat, O. Polat, and A. Cetin, "Detecting DDoS Attacks in Software-Defined Networks Through Feature Selection Methods and Machine Learning Models," Sustainability, vol. 12, no. 3, p. 1035, 2020.

[35] A. Chonka, J. Singh, and W. Zhou, "Chaos theory based detection against network mimicking DDoS attacks," IEEE Commun. Lett., vol. 13, no. 9, pp. 717–719, 2009.

[36] R. M. A. Ujjan, Z. Pervez, K. Dahal, A. K. Bashir, R. Mumtaz, and J. González, "Towards sFlow and adaptive polling sampling for deep learning based DDoS detection in SDN," Futur. Gener. Comput. Syst., vol. 111, pp. 763–779, 2020.

[37] F. Musumeci, V. Ionata, F. Paolucci, F. Cugini, and M. Tornatore, "Machine-learning-assisted DDoS attack detection with P4 language," in ICC 2020-2020 IEEE International Conference on Communications (ICC), pp. 1–6, 2020.

[38] J. Boite, P. A. Nardin, F. Rebecchi, M. Bouet, and V. Conan, "Statesec: Stateful monitoring for DDoS protection in software defined networks," IEEE Conf. Netw. Softwarization Softwarization Sustain. a Hyper-Connected World en Route to 5G, NetSoft 2017, 2017.

[39] Y. Goto, B. Ng, W. K. G. Seah, and Y. Takahashi, "Queueing analysis of software defined network with realistic openflow–based switch model," Comput. Networks, vol. 164, p. 106892, 2019.

[40] Y.-C. Lai, A. Ali, M. S. Hossain, and Y.-D. Lin, "Performance modeling and analysis of TCP and UDP flows over software defined networks," J. Netw. Comput. Appl., vol. 130, pp. 76–88, 2019.

[41] R. Swami, M. Dave, and V. Ranga, “Detection and analysis of TCP-SYN DDoS attack in software-defined networking,” Wirel. Pers. Commun., vol. 118, no. 4, pp. 2295–2317, 2021.

[42] S. Yu, Y. Tian, S. Guo, and D. O. Wu, “Can we beat DDoS attacks in clouds?,” IEEE Trans. parallel Distrib. Syst., vol. 25, no. 9, pp. 2245–2254, 2013.

[43] G. Wang, J. Li, and X. Chang, “Modeling and performance analysis of the multiple controllers’ approach in software defined networking,” in 2015 IEEE 23rd International Symposium on Quality of Service (IWQoS), pp. 73–74, 2015.

[44] W. Miao, G. Min, Y. Wu, and H. Wang, “Performance modelling of preemption-based packet scheduling for data plane in software defined networks,” in 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity), pp. 60–65, 2015.

[45] B. Xiong, K. Yang, J. Zhao, W. Li, and K. Li, “Performance evaluation of OpenFlow-based software-defined networks based on queueing model,” Comput. Networks, vol. 102, pp. 172–185, 2016.

[46] A. Fahmin, Y.-C. Lai, M. S. Hossain, and Y.-D. Lin, “Performance modeling and comparison of NFV integrated with SDN: Under or aside?,” J. Netw. Comput. Appl., vol. 113, pp. 119–129, 2018.

[47] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, “Modeling and performance evaluation of an OpenFlow architecture,” in 2011 23rd International Teletraffic Congress (ITC), pp. 1–7, 2011.

[48] K. Mahmood, A. Chilwan, O. N. Østerbø, and M. Jarschel, “On the modeling of openflow-based sdns: The single node case,” arXiv Prepr. arXiv1411.4733, 2014.

[49] K. Mahmood, A. Chilwan, O. Østerbø, and M. Jarschel, “Modelling of OpenFlow-based software-defined networks: the multiple node case,” IET Networks, vol. 4, no. 5, pp. 278–284, 2015.

[50] S. Azodolmolky, R. Nejabati, M. Pazouki, P. Wieder, R. Yahyapour, and D. Simeonidou, “An analytical model for software defined networking: A network calculus-

based approach," in 2013 IEEE Global Communications Conference (GLOBECOM), pp. 1397–1402, 2013.

[51] S. Azodolmolky, P. Wieder, and R. Yahyapour, "Performance evaluation of a scalable software-defined networking deployment," in 2013 Second European Workshop on Software Defined Networks, pp. 68–74, 2013.

[52] Z. Bozakov and A. Rizk, "Taming SDN controllers in heterogeneous hardware environments," in 2013 Second European Workshop on Software Defined Networks, pp. 50–55, 2013.

[53] M. Munir, I. Ardiansyah, J. D. Santoso, A. Mustopa, and S. Mulyatun, "DETECTION AND MITIGATION OF DISTRIBUTED DENIAL OF SERVICE ATTACKS ON NETWORK ARCHITECTURE SOFTWARE DEFINED NETWORKING USING THE NAIVE BAYES ALGORITHM," J. Inf. Syst. Manag., vol. 3, no. 2, pp. 51–55, 2022.

[54] R. Amin, M. Reisslein, and N. Shah, "Hybrid SDN Networks : A Survey of Existing Approaches," IEEE Commun. Surv. Tutorials, vol. 20, no. 4, pp. 3259–3306, 2018, doi: 10.1109/COMST.2018.2837161.

[55] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," 2015 Int. Conf. Comput. Netw. Commun. ICNC 2015, pp. 77–81, 2015.

[56] L. Tan, Y. Pan, J. Wu, J. Zhou, H. Jiang, and Y. Deng, "A new framework for DDoS attack detection and defense in SDN environment," IEEE Access, vol. 8, pp. 161908–161919, 2020.

[57] A. S. Arkan and M. Ahmadi, "Entropy-Based Anomaly Detection Using Observation Points Relations in Wireless Sensor Networks," Wirel. Pers. Commun., vol. 119, no. 2, pp. 1783–1798, 2021.

[58] Y. Sun, J. Yu, J. Tian, Z. Chen, W. Wang, and S. Zhang, "IoT-IE: An Information-Entropy-Based Approach to Traffic Anomaly Detection in Internet of Things," Secur. Commun. Networks, vol. 2021, 2021.

[59] J. Ibrahim and S. Gajin, "Entropy-based network traffic anomaly classification method resilient to deception," Comput. Sci. Inf. Syst., no. 00, p. 45, 2021.

[60] "What Is Network Traff Analysis? Definition, Importance, Implementation, and Best Practices." [Online]. Available: https://www.spiceworks.com/tech/networking/articles/network-traffic-analysis/, 2019.

[61] "5 Critical Reasons for Network Traff Analysis." [Online]. Available: https://www.netmon.com/5-critical-reasons-network-traffic-analysis/, 2019.

[62] T. R. Bikbulatov and I. I. Kurochkin, "Simulation of DDoS attack on software defined networks," in AIP Conference Proceedings, vol. 2181, Nov. 2019, doi: 10.1063/1.5135682.

[63] J. D. Markovic-Petrovic and M. D. Stojanovic, "Analysis of SCADA system vulnerabilities to DDoS attacks," in 2013 11th international conference on telecommunications in modern satellite, cable and broadcasting services (telsiks), vol. 2, pp. 591–594, 2013.

[64] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," Comput. Networks, vol. 72, pp. 74–98, 2014.

[65] Y. Djeldjeli and M. Zoubir, "CP-SDN: A New Approach for the Control Operation of 5G Mobile Networks to Improve QoS," Eng. Technol. Appl. Sci. Res., vol. 11, no. 2, pp. 6857–6863, 2021.

[66] B. Almadani, A. Beg, and A. Mahmoud, "Dsf: A distributed sdn control plane framework for the east/west interface," IEEE Access, vol. 9, pp. 26735–26754, 2021.

[67] A. M. Abdelrahman et al., "Software-defined networking security for private data center networks and clouds: Vulnerabilities, attacks, countermeasures, and solutions," Int. J. Commun. Syst., vol. 34, no. 4, p. e4706, 2021.

[68] S. Kaur, K. Kumar, and N. Aggarwal, "A review on P4-Programmable data planes: architecture, research efforts, and future directions," Comput. Commun., vol. 170, pp. 109–129, 2021.

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

[70] N. Feamster, J. Rexford, and E. Zegura, “The road to SDN: An intellectual history of programmable networks,” Comput. Commun. Rev., vol. 44, no. 2, pp. 87–98, 2014, doi: 10.1145/2602204.2602219.

[71] A. Doria et al., “Forwarding and Control Element Separation (ForCES) Protocol Specification.,” RFC, vol. 5810, pp. 1–124, 2010.

[72] P. H. Isolani, J. A. Wickboldt, C. B. Both, J. Rochol, and L. Z. Granville, “Interactive monitoring, visualization, and configuration of OpenFlow-based SDN,” in 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 207–215, 2015.

[73] N. Anerousis, P. Chemouil, A. A. Lazar, N. Mihai, and S. B. Weinstein, “The origin and evolution of open programmable networks and sdn,” IEEE Commun. Surv. Tutorials, vol. 23, no. 3, pp. 1956–1971, 2021.

[74] “Open Vswitch. 2018, [Online]. Available at: http://openvswitch.org.”, 2018.

[75] A. R. Abdou, P. C. Van Oorschot, and T. Wan, “Comparative analysis of control plane security of SDN and conventional networks,” IEEE Commun. Surv. Tutorials, vol. 20, no. 4, pp. 3542–3559, 2018, doi: 10.1109/COMST.2018.2839348.

[76] V. Bhavsar, C. Sahrial, and B. Goswami, “Security and Performance Evaluation of Software Defined Network Controllers Against Distributed Denial of Service Attack,” in 2021 Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT), pp. 1–8, 2021.

[77] H. Zhang, Z. Cai, Q. Liu, Q. Xiao, Y. Li, and C. F. Cheang, “A Survey on Security-Aware Measurement in SDN,” Secur. Commun. Networks, vol. 2018, 2018, doi: 10.1155/2018/2459154.

[78] Z. Latif, K. Sharif, F. Li, M. M. Karim, S. Biswas, and Y. Wang, “A comprehensive survey of interface protocols for software defined networks,” J. Netw. Comput. Appl., vol. 156, pp. 1–30, 2020, doi: 10.1016/j.jnca.2020.102563.

[79] M. N. A. Sheikh, “SDN-Based approach to evaluate the best controller: internal controller NOX and external controllers POX, ONOS, RYU,” Glob. J. Comput. Sci. Technol., 2019.

[80] A. V. Priya and N. Radhika, “Performance comparison of SDN OpenFlow controllers,” Int. J. Comput. Aided Eng. Technol., vol. 11, no. 4–5, pp. 467–479, 2019.

[81] P. T. Duy, H. Do Hoang, A. G.-T. Nguyen, and V.-H. Pham, “B-DAC: A decentralized access control framework on Northbound interface for securing SDN using blockchain,” J. Inf. Secur. Appl., vol. 64, p. 103080, 2022.

[82] L. Xiao, H. Zhu, S. Xiang, and P. Cong Vinh, “Modeling and verifying SDN under Multi-controller architectures using CSP,” Concurr. Comput. Pract. Exp., vol. 33, no. 2, p. e5334, 2021.

[83] M. A. Togou, D. A. Chekired, L. Khoukhi, and G.-M. Muntean, “A hierarchical distributed control plane for path computation scalability in large scale software-defined networks,” IEEE Trans. Netw. Serv. Manag., vol. 16, no. 3, pp. 1019–1031, 2019.

[84] “Software-Defined Networking – SDN Guide.” [Online]. Available: https://www.comparitech.com/net-admin/software-defined-networking/, 2022.

[85] “SDN benefits and advantages.” [Online]. Available: https://opennetworking.org/sdn-resources/whitepapers/software-defined-networking-the-new-norm-for-networks/, 2022.

[86] “SDN advantages and benefits.” [Online]. Available: https://www.ibm.com/services/network/sdn-versus-traditional-networking, 2022.

[87] “Software-Defined Networking (SDN): An Overview.” [Online]. Available: https://www.analyticssteps.com/blogs/software-defined-networking-sdn-overview, 2022.

[88] “5 Advantages and Disadvantages of SDN | Drawbacks & Benefits of SDN.” [Online]. Available: https://www.hitechwhizz.com/2021/06/5-advantages-and-disadvantages-drawbacks-benefits-of-sdn.html, 2021.

[89] S. Bhardwaj and S. N. Panda, “Performance Evaluation Using RYU SDN Controller in Software-Defined Networking Environment,” Wirel. Pers. Commun., vol. 122, no. 1, pp. 701–723, 2022.

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

[91] J. F. Balarezo, S. Wang, K. G. Chavez, A. Al-Hourani, and S. Kandeepan, "A survey on DoS/DDoS attacks mathematical modelling for traditional, SDN and virtual networks," Eng. Sci. Technol. an Int. J., 2021.

[92] Kleinrock, "L. Queuing systems, vol 1: theory." New York, NY, USA: Wiley Interscience, 1975.

[93] T. D. Nadeau and P. Pan, "Software Driven Networks Problem Statement,", Network Working Group Internet-Draft, Sep, vol. 30, 2011.

[94] D. Kreutz, F. M. V Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, pp. 55–60, 2013.

[95] V. G. H. Xie, T. Tsou, D. Lopez, H. Yin, "Use cases for ALTO with software defined networks." [Online]. Available: https://tools.ietf.org/html/draft-xie-alto-sdn-use-cases-01, 2019.

[96] I. Ahmad, S. Namal, M. Ylianttila, S. Member, A. Gurtov, and S. Member, "Security in Software Defined Networks : A Survey," vol. 17, no. 4, pp. 2317–2346, 2015.

[97] 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), pp. 1–2, 2013, doi: 10.1109/ICNP.2013.6733624.

[98] M. Iqbal, F. Iqbal, F. Mohsin, M. Rizwan, and F. Ahmad, "Security Issues in Software Defined Networking (SDN): Risks, Challenges and Potential Solutions," Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 10, 2019, doi: 10.14569/IJACSA.2019.0101042.

[99] Z. Shu, J. Wan, D. Li, J. Lin, A. V. Vasilakos, and M. Imran, "Security in Software-Defined Networking: Threats and Countermeasures," Mob. Networks Appl., vol. 21, no. 5, pp. 764–776, 2016, doi: 10.1007/s11036-016-0676-x.

[100] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDOS) flooding attacks," IEEE Commun. Surv. Tutorials, vol. 15, no. 4, pp. 2046–2069, 2013, doi: 10.1109/SURV.2013.031413.00127.

[101] C. Douligeris and A. Mitrokotsa, “DDoS attacks and defense mechanisms: classification and state-of-the-art,” Comput. Networks, vol. 44, no. 5, pp. 643–666, 2004.

[102] K. Kalkan, G. Gur, and F. Alagoz, “Defense Mechanisms against DDoS Attacks in SDN Environment,” IEEE Commun. Mag., vol. 55, no. 9, pp. 175–179, 2017, doi: 10.1109/MCOM.2017.1600970.

[103] L. Barki, A. Shidling, N. Meti, D. G. Narayan, and M. M. Mulla, “Detection of distributed denial of service attacks in software defined networks,” 2016 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI, pp. 2576–2581, 2016, doi: 10.1109/ICACCI.2016.7732445.

[104] H. Zenati, M. Romain, C.-S. Foo, B. Lecouat, and V. Chandrasekhar, “Adversarially learned anomaly detection,” in 2018 IEEE International conference on data mining (ICDM), pp. 727–736, 2018.

[105] M. Ahmed, A. N. Mahmood, and J. Hu, “A survey of network anomaly detection techniques,” J. Netw. Comput. Appl., vol. 60, pp. 19–31, 2016.

[106] M. H. H. Khairi, S. H. S. Ariffin, N. M. Abdul Latiff, A. S. Abdullah, and M. K. Hassan, “A Review of Anomaly Detection Techniques and Distributed Denial of Service (DDoS) on Software Defined Network (SDN),” Eng. Technol. Appl. Sci. Res., vol. 8, no. 2, pp. 2724–2730, 2018, doi: 10.48084/etasr.1840.

[107] R. M. A. Ujjan, Z. Pervez, K. Dahal, W. A. Khan, A. M. Khattak, and B. Hayat, “Entropy based features distribution for anti-DDoS model in SDN,” Sustainability, vol. 13, no. 3, p. 1522, 2021.

[108] M. Javed, A. B. Ashfaq, M. Z. Shafiq, and S. A. Khayam, “On the Inefficient Use of Entropy for Anomaly Detection,” in Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection, pp. 369–370, 2009, doi: 10.1007/978-3-642-04342-0_28.

[109] T.-F. Yen and M. K. Reiter, “Traffic Aggregation for Malware Detection,” in Proceedings of the 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 207–227, 2008, doi: 10.1007/978-3-540-70542-0_11.

[110] S. Ali, M. K. Alvi, S. Faizullah, M. A. Khan, A. Alshanqiti, and I. Khan, “Detecting DDoS attack on SDN Due to vulnerabilities in OpenFlow,” 2019 Int. Conf. Adv. Emerg. Comput. Technol. AECT, 2020, doi: 10.1109/AECT47998.2020.9194211.

[111] R. Sommer and V. Paxson, “Outside the Closed World: On Using Machine Learning for Network Intrusion Detection,” in 2010 IEEE Symposium on Security and Privacy, pp. 305–316, 2010, doi: 10.1109/SP.2010.25.

[112] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, “Intrusion detection by machine learning: A review,” Expert Syst. Appl., vol. 36, no. 10, pp. 11994–12000, 2009, doi: https://doi.org/10.1016/j.eswa.2009.05.029.

[113] A. Abduvaliyev, A. K. Pathan, J. Zhou, R. Roman, and W. Wong, “On the Vital Areas of Intrusion Detection Systems in Wireless Sensor Networks,” IEEE Commun. Surv. Tutorials, vol. 15, no. 3, pp. 1223–1237, 2013, doi: 10.1109/SURV.2012.121912.00006.

[114] M. H. H. Khairi, S. H. S. Ariffin, N. M. Abdul Latiff, A. S. Abdullah, and M. K. Hassan, “A Review of Anomaly Detection Techniques and Distributed Denial of Service (DDoS) on Software Defined Network (SDN),” Eng. Technol. Appl. Sci. Res., vol. 8, no. 2 SE-, pp. 2724–2730, Apr. 2018, doi: 10.48084/etasr.1840.

[115] B. Mahesh, “Machine learning algorithms-a review,” Int. J. Sci. Res. (IJSR).[Internet], vol. 9, pp. 381–386, 2020.

[116] Bonaccorso, Giuseppe, “Machine learning algorithms,” Packet Publishing Ltd, 2017.

[117] “35 Types of DDoS Attacks Explained.” [Online]. Available: https://javapipe.com/blog/ddos-types/, 2018.

[118] J. Ashraf and S. Latif, “Handling intrusion and DDoS attacks in Software Defined Networks using machine learning techniques,” Natl. Softw. Eng. Conf. NSEC 2014, pp. 55–60, 2014, doi: 10.1109/NSEC.2014.6998241.

[119] “The Mathematics Behind Support Vector Machine Algorithm (SVM). 2018, [online]. Available at: https://www.analyticsvidhya.com/blog/2020/10/the-mathematics-behind-svm/.”, 2018.

[120] T. N. T. Thu and V. D. Xuan, “Supervised support vector machine in predicting foreign exchange trading,” Int. J. Intell. Syst. Appl., vol. 10, no. 9, pp. 48–56, 2018, doi: 10.5815/ijisa.2018.09.06.

[121] G. A. Montazer, D. Giveki, M. Karami, and H. Rastegar, “Radial basis function neural networks: A review,” Comput. Rev. J, vol. 1, no. 1, pp. 52–74, 2018.

[122] M. A. Al-Adaileh, M. Anbar, Y.-W. Chong, and A. Al-Ani, “Proposed statistical-based approach for detecting distribute denial of service against the controller of software defined network (SADDCS),” in MATEC Web of Conferences, vol. 218, p. 2012, 2018.

[123] V. Moorthy, R. Venkataraman, and R. Gururajan, “Bayesian trust analysis of flooding attacks in distributed software defined networking nodes,” J. Ambient Intell. Humaniz. Comput., vol. 12, no. 7, pp. 7489–7498, 2021.

[124] D. Marek, A. Domański, J. Domańska, J. Szyguła, T. Czachórski, and J. Klamka, “Diffusion Model of a Non-Integer Order PIγ Controller with TCP/UDP Streams,” Entropy, vol. 23, no. 5, p. 619, 2021.

[125] P. Leguesdron, J. Pellaumail, G. Rubino, and B. Sericola, “Transient analysis of the M/M/1 queue,” Adv. Appl. Probab., vol. 25, no. 3, pp. 702–713, 1993.

[126] Y.-C. Lai, A. Ali, M. M. Hassan, M. S. Hossain, and Y.-D. Lin, “Performance modeling and analysis of tcp connections over software defined networks,” in GLOBECOM 2017-2017 IEEE global communications conference, pp. 1–6, 2017.

[127] “Mininet. 2018, [Online]. Available at: http://mininet.org.”, 2018.

[128] “Scapy 2018, [Online]. Available at: http://www.secdev.org/projects/scapy.”, 2018.

[129] M. E. Ahmed, H. Kim, and M. Park, “Mitigating DNS query-based DDoS attacks with machine learning on software-defined networking,” Proc. - IEEE Mil. Commun. Conf. MILCOM, vol. 2017-Octob, pp. 11–16, 2017, doi: 10.1109/MILCOM.2017.8170802.

تقنية مقترحة لاكتشاف هجوم انكار الخدمة الموزع في الشبكات المعرفة بالبرمجيات

الملخص العربي

إن الشبكات المعرفة بالبرمجيات Software-Defined Networking (SDN) هي بنية للشبكة جديدة تفصل مستوى التحكم عن مستوى البيانات [3]–[1]. إن SDN لديها التحكم المركزي في الشبكة وإمكانيات البرمجة، لذلك يمكن لـ SDN تحسين مرونة الشبكة وإدارتها وأدائها وقابليتها للتوسع. وعلى الرغم من أن SDN لها العديد من المميزات إلا أنها تواجه بعض التحديات الأمنية التي تهدد الأداء المناسب للشبكة مثل هجوم إنكار الخدمة الموزع Distributed Denial of Service (DDoS) attacks والذى يستهدف الهجوم على مستوى التحكم وكذلك مستوى البيانات [8]–[4].

هناك الكثير من الأساليب الأمنية التي تحمي ضد هجمات DDoS في SDN والتي تشمل الطرق المعتمدة على entropy وتحليل نمط المرور والتعلم الآلى Machine Learning (ML) [17]–[15][14]–[12] [11]–[9]. وتتسم تقنيات entropy وتحليل نمط المرور بنفقات منخفضة ولكنها تعاني بشكل أساسي من صعوبة التكيف مع تغييرات الشبكة ومع سلوكيات الشبكة المختلفة [18]. وتُفضل تقنيات ML لاكتشاف النشاط الضار لأنها تتعلم ذاتيًا ويمكن أن تتكيف مع العديد من حالات الشبكة [19] .

إن هذه الرسالة تقترح تقنية دقيقة وهامة لإكتشاف هجمات DDoS ضد مستوى التحكم ومستوى البيانات في SDN. وتثبت التقنية المقدمة أنها قادرة على اكتشاف الهجوم بدقة عالية باستخدام خوارزمية قائمة على التعلم الآلي وتستخدم هذه التقنية ملامح (Features) متقدمة جديدة تم الحصول عليها من معلومات وإحصاءات تدفق حركة المرور. يتم تدريب النموذج المطورة باستخدام kernel radial basis function. وتستخدم التقنية ملامح متقدمة مثل عنوان IP للوجهة غير المعروفة ، والفترة بين وصول الحزم ، والمعلومة المضافة من بروتوكول transport layer ، ومعلومة نوع الخدمة لمستوى التحكم. أيضًا، يتم استخدام الملامح المتقدمة مثل السعة التخزينية ل switch ، ومتوسط معدل الحزم ذات عناوين IP للوجهة غير المعروفة، وحقل معلومة خيارات IP، ومتوسط عدد تدفقات الحزم لمستوى البيانات. لم يتم استخدام الملامح المقترحة من قبل. تُستخدم الملامح لإنشاء مجموعة بيانات تُستخدم كمدخلات لـLinear Support Vector Machine (SVM) Classifier

[22]–[20] ,[13]. ويتم حظر الأنظمة المخترقة التي ترسل حركة مرور هجومية، ويتم تخزين معلوماتها.

ولقد أثبتت النتائج التجريبية أن التقنية المقترحة القائمة على ML باستخدام SVM يمكنها اكتشاف الهجوم بدقة عالية، ومعدل اكتشاف مرتفع، وقياس f_measure مرتفع، وإنذار كاذب منخفض، مقارنة بالتقنيات الأخرى ذات الصلة [38]–[23].

أيضًا، تقدم هذه الرسالة نماذج تحليلية ومحاكاة لتدفق حزم TCP و UDP في SDN. ويعد تحليل الحركة والنمذجة في SDN أمرًا مهمًا لتتبع نشاط الشبكة وتوافرها لاكتشاف التشوهات، مثل مشكلات الأمان والتشغيل. لذلك، تم تطوير نماذج الحالات ذي الأبعاد السبعة Seven-Dimensional (7-D) في هذه الرسالة لنمذجة تدفقات حزم TCP وUDP فى SDN، على عكس النماذج الأخرى المستخدمة في الأعمال السابقة التي تستخدم حالات ثلاثية الأبعاد أو رباعية الأبعاد [39][40]. ويستخدم العمل المقترح وحدات تحكم متعددة وأجهزة switches متعددة ذات سعات تخزينية محدودة على عكس النماذج الأخرى في الأعمال ذات الصلة والتي قد تستخدم وحدة تحكم واحدة أو جهاز switch واحد مع مخزن سعته لانهائية [52]–[39]. ويؤدي استخدام وحدات تحكم متعددة إلى تحسين أمان الشبكة نظرًا لإمكانية النسخ الاحتياطي. أيضًا، يتم تحسين قابلية توسيع الشبكة نظرًا لوجود وحدات تحكم كافية لخدمة المزيد من أجهزة ال switches. يمكن أن يوفر استخدام switches متعددة نتائج أكثر دقة لتعكس حالات مرور الحزم فى الشبكة بشكل أفضل. وتم فى الرساله تتبع متوسط زمن تأخر وصول الحزمة إلى وجهتها واحتمال فقد الحزمة لقياس أداء النماذج المطورة. أيضًا، يمكن استخدام النماذج المقترحة تقريبًيا للتمييز بين حركة مرور الحزم العادية وحركة هجمات DDoS من خلال اعتبار أن حركة الهجوم لها شكل وصول للحزم يتبع توزيع Poisson بمعدلات وصول عالية.

وتم فى الرسالة صياغة نتائج المحاكاة ومقارنتها بنتائج التحليل الرياضى. ولقد أثبتت النتائج أن النمذجة المقترحة دقيقة مقارنة بالأعمال المقدمة حديثا ذات الصلة.

جامعة أسيوط

كلية الحاسبات والمعلومات
قسم تكنولوجيا المعلومات

تقنية مقترحة لاكتشاف هجوم انكار الخدمة الموزع في الشبكات المعرفة بالبرمجيات

رسالة

لإستيفاء الحصول علي درجة الماجستير في

الحاسبات والمعلومات (تكنولوجيا المعلومات)

إعداد

وحيد جميل جادالله غالى
معيد بقسم تكنولوجيا المعلومات
كلية الحاسبات والمعلومات
جامعة أسيوط

لجنة الإشراف:

أ.د. حســنى محمد ابراهيم
أستاذ متفرغ بقسم تكنولوجيا المعلومات،
كلية الحاسبات والمعلومات، جامعة أسيوط

أ.م.د. نجـوى محـمـد عمر
أستاذ مساعد بقسم تكنولوجيا المعلومات،
كلية الحاسبات والمعلومات، جامعة أسيوط

لجنة التحكيم:

أ.د. عبدالمجيد أمين علي
عميد كلية الحاسبات والمعلومات، جامعة المنيا

أ.د. مؤمن طه حنفي أحمد
أستاذ ووكيل كلية الهندسة لشئون الدراسات العليا والبحوث، جامعة أسيوط

أ.د. حســنى محمد ابراهيم
أستاذ متفرغ بقسم تكنولوجيا المعلومات،
كلية الحاسبات والمعلومات، جامعة أسيوط

أ.م.د. نجـوى محـمـد عمر
أستاذ مساعد بقسم تكنولوجيا المعلومات،
كلية الحاسبات والمعلومات، جامعة أسيوط

أسيوط، 2022