Distributed Network Monitoring for Distributed Denial of Service Attacks Detection and Prevention

Thesis submitted in accordance with the requirements of
the University of Liverpool for the degree of Doctor in Philosophy by

Azibanagein Godswill Lucky

November 2021

"Start by doing what is necessary, then do what is possible and suddenly you are doing the impossible."
St Francis of Assisi

Abstract

There are two main categories of Distributed Denial of Service (DDoS) attacks that are capable of disrupting the daily operations of internet users and these are the low and high-rate DDoS attacks. The detection and prevention of DDoS attacks is a very important aspect in network security in ensuring that the operations of businesses, communication, and educational facilities operate efficiently without disruptions. Over the years, many DDoS attacks detection systems have been proposed. These detection systems have focused more on obtaining high accuracy, reduction of false alarm rates and simplification of detection systems. However, less attention has been given to the computational costs of detection systems (processing power requirements and memory consumptions), early detection and flexibility in their deployment to support the different needs of networks and distributed monitoring approaches.

The focus of this thesis is to investigate the use of a robust feature selection approach and machine learning classifiers to develop useful DDoS detection architectures for fast, effective, and efficient DDoS attacks detection to achieve high performance at low computational cost.

To achieve this, a lightweight software architecture which is simple in design using minimal number of network flow features for distinguishing normal from DDoS attack network flows is proposed. The architecture is based on the Decision-Tree (DT) classifier and distinguishes DDoS attack from normal traffic network flows with a detection accuracy of over 99.9% when evaluated with up-to-date DDoS attack datasets. In addition, it can flexibly be deployed in a real-time network environment and at different network nodes to meet the needs of the network being monitored creating an avenue for distributed monitoring. Also, the use of minimal network flow features selected through a robust features selection approach results in a massive reduction in memory requirements when compared to traditional systems. Results from the software implementation of the architecture indicated that it uses just 7% processing power of a core of the detection system's CPU in offline mode and provides no additional overhead to the monitored network.

However, software applications for distinguishing normal from DDoS attack traffic are struggling to cope with the ever-increasing complexity and intensity of DDoS attack traffic. This increased workload ranges from the capturing and processing of millions of packets per second to classification of thousands of network flows per second which is evident in some of the most recent DDoS attacks faced by a variety of companies.

To cope with this workload, a hardware accelerated hybrid network monitoring application is proposed. The proposed application is capable of fast network flows classification by leveraging the hardware parallel processing characteristics of a Field Programmable Gate Array (FPGA) whilst using a software application in the CPU for the network flow pre-processing required for classification. The hybrid system is capable of distinguishing DDoS attacks from normal network traffic flows with a detection accuracy of over 98% when deployed in a real-time environment under different network traffic conditions with detection in 1μs which is over thirty times faster than the software implementation of the architecture.

The hardware accelerated application was implemented in the Zynq-7000 All Programmable SoCs ZedBoard which can monitor up to 1Gbps line rate.

The evaluation results and findings from analysis of the experimental results of the hardware accelerated application provide some important insights in improving the programmability, overall performance, scalability, and flexibility in deployment of the detection system across a network for accurate and early DDoS attack detection.

In the final part of this thesis, the use of distributed network monitoring is explored with the implementation of the lightweight DDoS attacks detection architecture using Network Simulator 3 (NS-3). The systems are distributed at different parts of a network and results from the approach indicated that effective implementation of distributed network monitoring systems dramatically reduces the effect of DDoS attack to a minimal on the target network or network node.

Acknowledgements

First and most importantly, I would like to express my deep gratitude to God Almighty, for seeing me through this journey. I would like to express my deep gratitude to my best friend and partner, Miss Joyce Tiani, for her unconditional support through my journey. Thanks a lot for your prayers, patience and constant words of encouragement; I could not have done it without you.

I would like to express my special appreciation and thanks to my supervisor Prof. Alan Marshall, for his support and for believing in me over these years. He along with the University of Liverpool gave me the opportunity, to begin and finish this journey and his guidance and suggestions have been invaluable. I would like to thank you for encouraging my research and for allowing me to grow.

My sincere thanks go to my secondary supervisors, Dr Junqing Zhang and Dr Zaheer Khan who gave me insightful advice and enhanced my academic skills. I would like to thank you for their valuable time and helpful guideline that have enriched the quality of my work.

I want also to thank all my friends in the Advanced Networks Research Group especially Dr Fred Jjunju and Dr Valerio Selis for their support, guidance and companionship over the years. A special thanks to all the EEE Department staff, including the administrative and technical team, for their advice and help during this journey.

I want to thank the members of Love Assembly Liverpool especiall Pastor Fola Olaoye for the words of prayers, advice and encouragement throughout my stay in Liverpool.

I would also like to acknowledge the Center for Applied Internet Data Analysis (CAIDA) and the Canadian Institute for Cybersecurity (CIC) for their datasets which were used to evaluate the outcomes presented in my research.

I want to thank my beloved family especially my mother Mrs Salome Solomon and uncle Dr Davidson Egirani, for supporting me through every moment of my time at university. Their abiding support has helped me face every challenge that I have come up against. I am eternally thankful for their love and encouragement.

Finally, I want to thank me for believing in me, pushing me beyond my confort zone, the hardwork, dedication, commitment and for not giving up even through the toughest periods.

Contents

Illustrations

List of Figures

List of Tables

Abbreviations

ACC	Accuracy
ACK	Acknowledgement flag
ALU	Arithmetic Logic Unit
ADS	Anomaly Detection System
ANN	Artificial Neural Network
ANRG	Advanced Network Research Group
AS	Autonomous System
ASIC	Application-Specific Integrated Circuit
AUC	Area Under the Curve
AWS	Amazon Web Service
BFE	Backward Feature Elimination
bps	Bits per second
CAGR	Compound Annual Growth Rate
CAIDA	Center for Applied Internet Data Analysis
C&C	Command and Control
CART	Classification And Regression Tree
CIC	Canadian Institute for Cybersecurity
CPU	Central Processing Unit
CRW	Code Red Worm
DARPA	Defence Advanced Research Projects Agency
DDoS	Distributed Denial of Service Attack
DFA	Direct Flooding Attack
DL	Deep Learning
DMA	Direct Memory Access
DNN	Deep Neural Network
DNS	Domain Name Server
DoS	Denial of Service Attack
DPI	Deep Packet Inspection
DT	Decision-Tree

FFS Forward Feature Selection
FIN Finish flag
Fm F-measure
FN False Negative
FP False Positive
FPGA Field-Programmable Gate Array

GA Genetic Algorithm
Gbps Gigabits per second
GPU Graphics Processing Unit

HCF High Correlation Filter
HDL Hardware Description Language
HIDS Host-based Intrusion Detection System
HLS High-Level Synthesis
HPC High Performance Computing
HRDDoS High-Rate DDoS
HTTP Hypertext Transfer Protocol

IAT Inter-Arrival-Time
ICMP Internet Control Message Protocol
ID3 Iterative Dichotomiser 3
IDS Intrusion Detection System
IFA Indirect Flooding Attack
IoT Internet-of-Things
IP Internet Protocol
IPS Intrusion Prevention System
ISP Internet Service Provider
IT Information Technology

KDD Knowledge Discovery and Data mining
KNN K-Nearest Neighbors

LAN Local Area Network
LOIC Low-Orbit Ion Canon
LRDDoS Low-Rate DDoS
LVF Low Variance Filter
LVQ Learning Vector Quantisation

Mbps Megabits per second

MER Misclassification Error Rate
ML Machine Learning
MLP MultiLayer Perceptron
MVR Missing Value Ratio

NB Naïve Bayes
NGN Next Generation Networks
NIDS Network Intrusion Detection System
NN Neural Network
NS Network Simulator

OS Operating System

P Precision
PCA Principal Component Analysis
PL Programmable Logic
pps Packets per second
PS Processing System
PSH Push flag

QoS Quality of Service

RAM Random Access Memory
RBF Radial Basis Function
RF Random Forest
RNN Recurrent Neural Network

SDN Software-Defined Networking
SE Sensitivity
SIDS Signature-based Intrusion Detection Systems
SNMP Simple Network Management Protocol
SoC System on Chip
SP Specificity
SVM Support Vector Machine
SYN Synchronization flag

Tbps Terabits per second
TCP Transmission Control Protocol
TFN Tribe Flood Network
TN True Negative
TP True Positive

TTL	Time to Live
UDP	User Datagram Protocol
URG	Urgent Pointer flag
WAN	Wide Area Network
WEKA	Waikato Environment for Knowledge Analysis

Preface

This thesis is primarily my own work. The sources of other materials are identifed.

Chapter 1

Introduction

There has been a rapid growth in the Internet and the field of Information Technology (IT) over the past decades. In addition, there has been an exponential growth in the number of organizations, companies and individuals using the internet and according to projections, these areas will continue to experience growth due to the offered benefits and advantages to humanity. However, the absence of security from the original design of the internet, shortage of security agreements, absence of international laws, vast availability of attack tools and lack of collaboration between Internet Service Providers (ISP) have led to the internet being a platform that can be used by attackers to perform unlawful activities and launch devastating attacks [1].

Distributed Denial of Service (DDoS) attacks are key security threats that have grown significantly over recent years and are some of the most prominent attacks affecting the use of the Internet [1]. These attacks are achieved by sending large volumes of network traffic from many exploited systems called zombies to either a host or a network targeting the exhaustion of network resources. These attacks exhaust the network resources rendering services unavailable and subsequently leading to loss of confidential and critical information as well as financial loss [1].

These attacks are structured and designed to disable internet services, reduce network performance, generate large volume of traffic and damage internet applications [1], [2]. However, these attacks are not solely designed to compromise or steal data such as passwords or usernames [2]. With the emergence of the Internet-of-Things (IoT) concept, varieties of devices that are connected to the internet are not adequately secured. With this inadequacy in security, attackers exploit these unsecured devices and use them to instigate large-scale sophisticated DDoS attacks [3], [4].

1.1 DDoS Attacks Severity

DDoS attacks have become more complex, rampant and hard to mitigate due to the ever-changing threat landscape [5]. There has been an exponential increase in the size of attacks over the past two decades and continues to increase as shown in Figure 1.1. The attacks sizes

were relatively small-scale initially with an estimate network traffic of 200Mbps in 2001 and has increased to over 2Tbps in 2020 [6], [7].

Small size DDoS attacks were capable of bringing down the network of victims at the early days. For example, a continuous maximum 90Mbps DDoS attack was capable of crippling Estonia's Internet for almost three weeks causing damages to the banking, media and government websites in 2007 and in 2008, the Internet infrastructure of Georgia was crippled for almost one month as a result of a 200Mbps attack [8], [9]. These attacks resulted in disruption of business, government organisations and financial losses [8], [9].

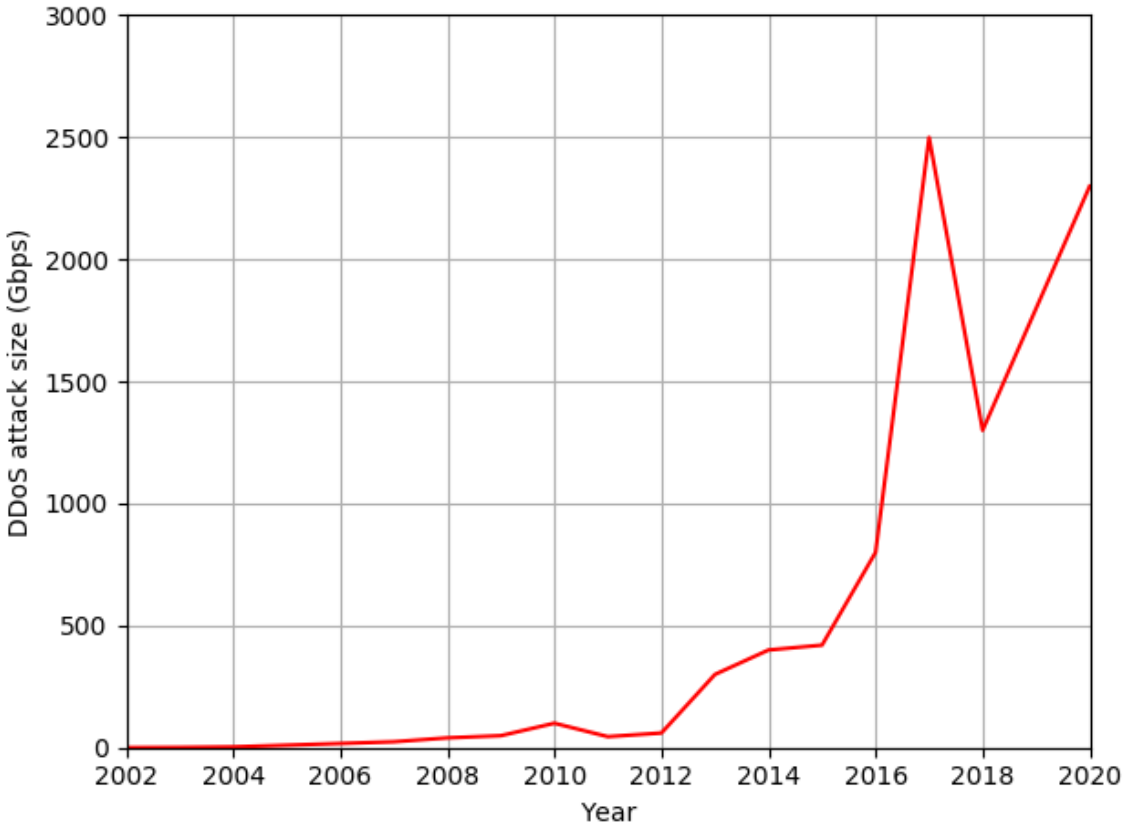

FIGURE 1.1: Largest DDoS attack size per year worldwide [10], [11].

With the massive growth of the Internet and the emergence of IoT, attackers are using large numbers of exploited devices to increase the size and complexity of DDoS attacks with the sole objective of bringing down the networks of a wide range of victims. Earlier forms of DDoS attacks focused on mainly targeting a specific organization or business whereas current attacks have evolved and have the capacity of bringing down the Internet of several organizations and businesses within a selected region.

The whole Internet infrastructure was tested in 2013 when a DDoS attack of around 300Gbps was launched against Spamhaus, which is a not-for-profit anti-spam organization [12]. This attack was successful in taking down the servers of Spamhaus resulting in internet

congestion across Europe and disrupting accessibility to some major websites [12]. The following year, a 400Gbps DDoS attack was launched against OVH which is a web hosting company [13].

Another major DDoS attack peaking over 1Tbps was launched in 2016 using Mirai Botnet which resulted in some high profile websites becoming inaccessible for users [13]. Some of the affected websites included Airbnb, Twitter, GitHub and Reddit [13]. Furthermore, another DDoS attack peaking over 2Tbps was launched in 2017 and 2020 against Google Cloud and Amazon Web Service (AWS) [7].

Due to the disruption caused on the daily operations of individuals, government organizations and business spanning across several sectors by the severity of DDoS attacks, important attention and effort should be given so that these attacks are detected and mitigated more effectively. However, the detection of DDoS attacks is not an easy task. Traditional forms of DDoS attacks involves the sending of high volume and high-intensity network traffic that is directed to the victim's network that leads to a denial of service whereas newer forms of DDoS attacks involve a combination of high and low intensity network traffic directed to a victim with the capability of congesting the network links that are shared by the victims and third-party devices [14], [15].

The combination of both low and high-rate DDoS attacks has led to increased complexity and have made it much more difficult for traditional detection systems to effectively and efficiently detect and mitigate DDoS attacks [15].

The DDoS attacks detection, mitigation and protection market is currently valued at USD 4.6 billion and is projected to experience a Compound Annual Growth Rate (CAGR) of 13.2% between 2021 and 2027 [16], [17]. Some of the keyplayers in the market include Arbor Networks, Juniper Networks, Akamai, A10 Networks, Fortinet, CloudFlare, Huawei, Microsoft, Nexusguard, Imperva, Radware, Corero, Verisign, F5 Networks and Nsfocus [16], [17].

1.2 Problems with Current Detection Solutions

Some of the characteristics of an ideal DDoS detection system include the detection of DDoS attacks network traffic with high detection accuracy and in real-time, mitigation of DDoS attacks closer to the source, which reduces the potential damages to the victim's network and doing this with low computational cost [18]. This means that the primary goals of a DDoS attacks detection system include high detection accuracy, early detection and low computational cost. High detection accuracy is synonymous to low false positive and high true positive rates, low computational cost is synonymous to low complexity in designs with less resource requirements and early detection is synonymous to detection of attacks close of the source of attack.

Due to the distributive nature of DDoS attacks, attack traffic stem from multiple sources and the aggregate of this traffic which results in a huge volume leads to the exhaustion of the victim's network infrastructure. This indicates that the generated network traffic at the

source is usually small compared to the traffic received at the victim's network. Therefore, detecting and mitigating the attack closer to the source will dramatically reduce the potential damages caused by the attacks.

Currently, many proposed DDoS detection systems have focused more on obtaining high accuracy, reduction of false alarm rates and simplification of detection systems [18], [19]. However, less attention has been given to the computational costs of detection systems i.e. processing power requirements and memory consumptions, programmability of systems due to the everchanging attack landscape and flexibility in deployments to support the different needs of networks [18], [19]. The main issues highlighted above that are hindering the effectiveness and efficiency of DDoS attacks detection are:

1.2.1 Flexible Detection

Previous DDoS attacks detection solutions have been effective in effectively detecting attacks however; these solutions are struggling to cope with the ever-increasing size and complexity of attacks [18]. Examples of these detection systems are NetBouncer, D-WARD and LADS [20]–[22]. Some of these detection solutions apply some techniques, which are not very effective in detecting newer DDoS attacks due to the variations and complexity of these attacks.

In addition, these rely on fixed thresholds which result in detecting only specific types of attacks without consideration for the different variations and complexities of newer attacks [18]. There are some promising techniques in detecting newer attacks such as the use of entropy-based approaches however; there is limitation in detection accuracy due to the selected features and the set thresholds used in detecting normal and DDoS attack traffic [23]–[27].

The over-reliance of commonly used network traffic features such as the five tuple of network flows (which are the network protocol used, source and destination port numbers of the packets and source and destination Internet Protocol (IP) addresses of the packets) has led to the neglecting of other attributes such as packet Inter-Arrival-Time (IAT), number of received packets, total size of network flows, etc., that may lead to more accurate detection of DDoS attacks.

Currently, some industry based solutions which have been effective in detecting a wide range of DDoS attacks include FortiDDoS by Fortinet, Prolexic by Akamai, Arbor Cloud by Arbor Networks, Azure DDoS Protection by Microsoft, etc. However, these solutions are quite expensive with some ranging between $6,000 and $80,000 with additional licenses incuring extra charges for hardware based solutions [28]. The cost of cloud based solutions provided my Microsoft start from $2,944 per month with additional licenses incuring extra charges [29]. However, due to the high cost of DDoS protection and mitigation solutions, smaller organisations are often ill equiped and are often the targets of these attacks [30]. Therefore, there is a need for the design and implementation of low-cost solutions that are affordable and flexible in detecting a wide range of DDoS attack types.

1.2.2 Computational Cost

The ever-increasing size of DDoS attacks means that a lot more resources are required to fully process the network traffic [18]. In addition, the processing of huge volume of network traffic especially in a large network can lead to high computational cost or even failure of the monitoring and processing systems especially in the case of a centralised monitoring approach [18].

However, this problem can be overcome by monitoring and processing only the relevant network traffic features instead of capturing and processing all the features. This will drastically reduce the computational burden on the detection system, as only a few features are required in processing network traffic. Additionally, the use of distributed monitoring approach leads to the distribution of network loads across the monitoring and detection system, thereby, reducing the computational burden on each system [31].

1.2.3 Programmability

Network monitoring and DDoS attacks detection systems are programmed to perform the required processing of network traffic to effectively and efficiently detect DDoS attacks. Traditional monitoring systems lack programmability, which leads to them struggling to cope with the ever-increasing attack landscape [31]. To effectively cope with the changing demands of networks, monitoring and detection systems should be programmable. This programmability characteristic is present in some of the current network devices i.e. some switches can be programmed to deal with certain needs of networks.

1.2.4 Scalability

The increasing sizes of network and network traffic presents some challenges in efficiently processing the huge volume of traffic and effectively detecting DDoS attacks. One of the main challenges of scalable monitoring and detection systems is the smooth processing of network traffic without incurring an overhead on the network being monitored.

However, some promising solutions have increased scalability in detecting DDoS attacks. These techniques include NetFlow and sFlow technologies [32], [33]. The use of these techniques along with appropriate network traffic sampling approaches to reduce processing cost. However, these solutions struggle to detect attack traffic that are stealthy in nature since not all types of network traffic are analysed [31].

In this case, scalability of DDoS attacks detection solutions can be increased using lightweight architecture incorporated with a robust feature selection approach that considers a wide range of network features through analysis of up-to-date datasets. These solutions should be low cost to enable distributed monitoring approaches to effectively cope with the needs of the network with the ever-increasing size and complexity of the network traffic.

1.2.5 Flexible Deployment

Flexible deployment of network monitoring and detection systems is important to deal with the demands of current networks. Currently, the need for systems, which are capable of flexible deployment, is crucial in dealing with the different needs of networks. These systems should be capable of being deployed at different network nodes to meet demands.

1.3 Problem Statement

The problems of current network monitoring and DDoS attacks detection systems are summarized in the following problem statement:

The major problem with current DDoS attacks detection solutions is their inability to accurately detect a wide range of DDoS attacks as early as possible in medium and large-scale networks. Traditional DDoS attacks detection solutions are often centralized and are not very effective in accurately detecting DDoS attacks early due to limitations in flexible detection, programmability, scalability, computational cost and flexible deployment.

Therefore, the use of new approaches in effectively and efficiently detecting DDoS attacks accurately and early is important.

1.3.1 Research Questions

Based on the problems facing current DDoS attacks detection systems highlighted in Section 1.2, attempts to address these problems will be covered in this thesis by providing answers to the following research questions:

How can DDoS attack detection systems effectively and efficiently detect attacks early with high detection accuracy and low computational cost?

The above stated question is further broken down into some sub-questions:

1. **How can high detection accuracy be achieved in DDoS attack detection systems with scalability?**

   This question is related to the flexible detection and scalability issues discussed in Section 1.2. This thesis tries to answer this question by examining the use a wide range of datasets that consists of different attack types and considering only relevant features in the detection systems which leads to early detection as very little processing is required. In addition, it examines the use of Machine Learning (ML) algorithms. This leads to improved scalability, flexible detection and low computational cost.

2. **How can fast and early detection be achieved?**

   This question is related to the flexible detection and deployment issues highlighted in Section 1.2. This thesis tries to answer this question by examining the implementation of Machine Learning (ML) algorithms on different high-computing platforms such as Multi-core processors and Field Programmable Gate Array (FPGA) for fast network traffic classification leading to fast detection of DDoS attacks. These high-computing

platforms offer the flexibility of deployment at different network nodes. In addition, it examines the use of distributed monitoring approach across different network nodes for early detection.

3. **What are the most effective feature selection methods?**

   This question is related to computational cost and scalability issues highlighted in Section 1.2. This thesis tries to answer this question by examining several robust feature selection techniques such as Backward Feature Elimination, Low Variance filter, Forward Feature selection, etc. in selecting only the relevant features required in effectively detecting DDoS attacks. In addition, it tries to evaluate the tradeoffs between complexity of detection systems due the number of selected features and detection accuracy. Furthermore, it attempts to evaluate the effects of different number of selected features to computational cost.

1.4 Thesis Structure and Research Contributions

An overview of the thesis and its contributions are highlighted in a tree structure shown in Figure 1.2.

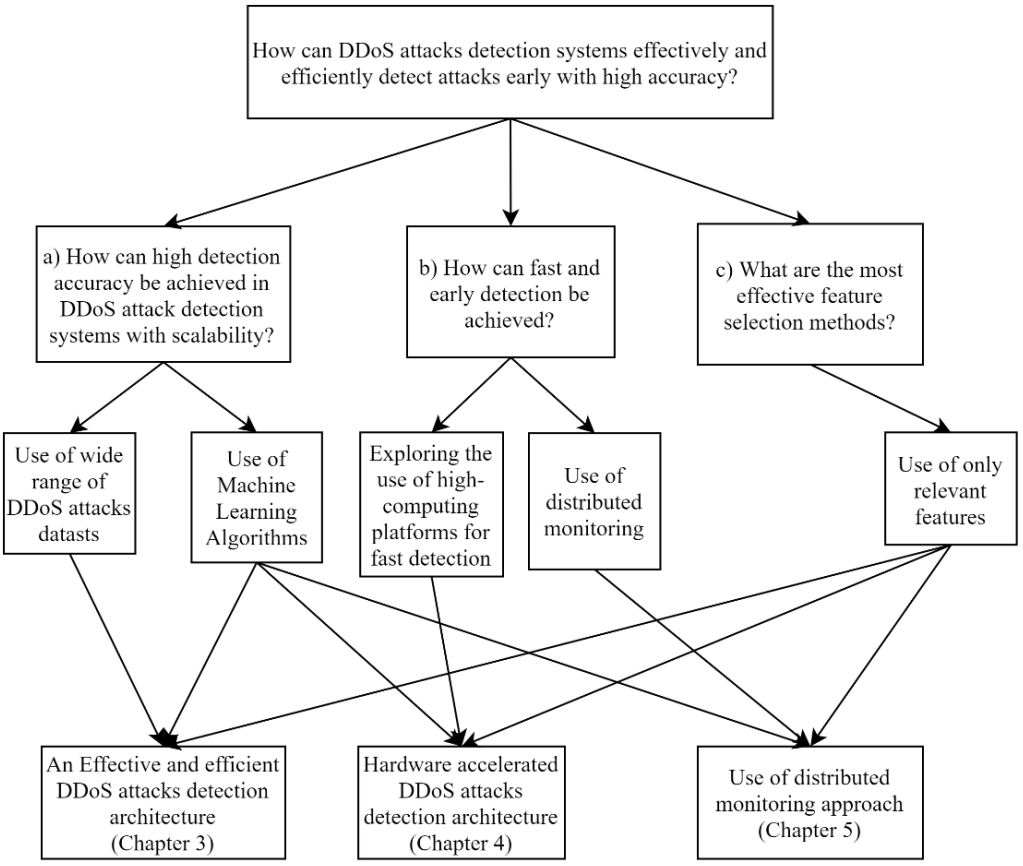

FIGURE 1.2: Thesis Contributions

Below are brief discussions on the contributions of the thesis.

1.4.1 Thesis Structure

The remaining content of this thesis is organised as follows.

- **Chapter 1 – Introduction**

  This chapter introduces Distributed Denial of Service (DDoS) attacks. In addition, discussion of some of the problems faced by current DDoS attacks detection solutions was provided based on extensive reviews conducted on these solutions. Furthermore, a problem statement is provided along with some of the possible answers to address the issues discovered.

- **Chapter 2 - Background and Related Work**

  This chapter outlines information relating to the existing status of DDoS attacks landscape, Machine Learning Classification techniques, feature selection approaches, high performance computing platforms, analysis of existing detection solutions and identification of research gaps in current literature.

- **Chapter 3 - A Lightweight Decision-Tree Algorithm for detecting DDoS flooding attacks**

  In this chapter, a lightweight Decision-Tree (DT) architecture for accurately detecting DDoS flooding attacks is presented. The motivation to designing and implementing a real-time detection system using ML algorithms is presented. The proposed architecture uses a robust feature selection technique to select only the most relevant features and can be deployed on multiple host machines at different network nodes. In addition, an extensive analysis of the results obtained is carried and comparisons are drawn with existing related detection solutions.

- **Chapter 4 - Hardware Accelerated Application for DDoS flooding attacks detection**

  In this chapter, a hardware accelerated DDoS flooding attacks detection solution based on a high performance computing platform is proposed. The solution is based on the implementation of the proposed detection ML model from chapter 3 on an FPGA. Results obtained from multiple experiments conducted are presented and comparisons are drawn with both related work and software implementation of the detection solution.

- **Chapter 5 - Distributed Monitoring for DDoS Flooding Attacks Detection**

  This chapter evaluates the modelling of DDoS flooding attacks using Network Simulator 3 also known as NS3. In addition, the potential damages of a successful DDoS attack without the use of detection systems are presented. Furthermore, an evaluation on the effectives of distributed monitoring approach where detection systems are distributed across a network is conducted. Finally, the impacts of DDoS attacks initiated in the presence of many distributed detection systems are investigated.

- **Chapter 6 - Contributions and Future Work**

  In this chapter, conclusions and evaluation of the applications of the various proposed monitoring and detection solutions are presented. In addition, the contributions and findings from the research conducted are highlighted. Furthermore, future research prospects on the subject are proposed.

1.5 Publication List

Godswill Lucky, Fred Jjunju, and Alan Marshall. "A Lightweight Decision-Tree Algorithm for detecting DDoS flooding attacks." 2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS 2020).

Chapter 2

Background and Related Work

This chapter introduces the concept of Distributed Denial of Service (DDoS) attacks and the current attack landscape. It discusses some important approaches in DDoS attacks detection such as Machine Learning Classification techniques, feature selection techniques, high performance computing platforms and distributed monitoring. In addition, it reviews the literature of existing DDoS attacks detection solutions and identifies some of the well-known DDoS attacks datasets and testbeds used in evaluating the solutions. Current research gaps are highlighted in the summary of this chapter.

2.1 Distributed Denial of Service Attack

Distributed Denial of Service (DDoS) attacks can be referred to as large-scale coordinated versions of Denial of Service (DoS) attacks. The aim of DDoS attacks is to disrupt network services and render them unavailable to legitimate users and are achieved by sending large volumes of network traffic from many exploited systems called zombies to either a host (i.e. end devices, servers) or network links targeting the exhaustion of network resources [34]–[36]. The success of these attacks can lead to loss of confidential and critical information as well as financial loss [36].

DoS attacks involve the use of a single attack source whereas DDoS attacks are conducted using many exploited devices called zombies which could be hundreds of thousands from different networks across the globe [37]. Due to the nature of these attacks involving the use of many devices, their sizes are significantly large especially nowadays, where the largest record size is around 2.5Tbps as discussed in Chapter 1.

The first recorded large-scale attack occurred in the year 2000, when the success of a high volume flood attack rendered major Internet sites offline for several hours. These sites included ZDNet, CNN, Yahoo and eBay [38]. DDoS attacks have grown both in size and complexity over the years leading to disruption of major Internet services across the globe.

2.1.1 How to Launch a DDoS Attack

There are many strategies and tools used by attackers to launch effective DDoS attacks. An effective attack is an attack that is successful in breaching security defences and causing the required disruption to services. These attacks require the use of a large number of exploited devices in launching the attacks against either an end device or an organization's network link. These attacks are classified under two categories based on the method used in launching the attacks. These categories are direct and indirect flooding.

1. **Direct Flooding**

   This method of launching DDoS attacks involves the sending of commands to all exploited devices (zombies) to send attack traffic straight to the target end device or network link as shown in Figure 2.1.

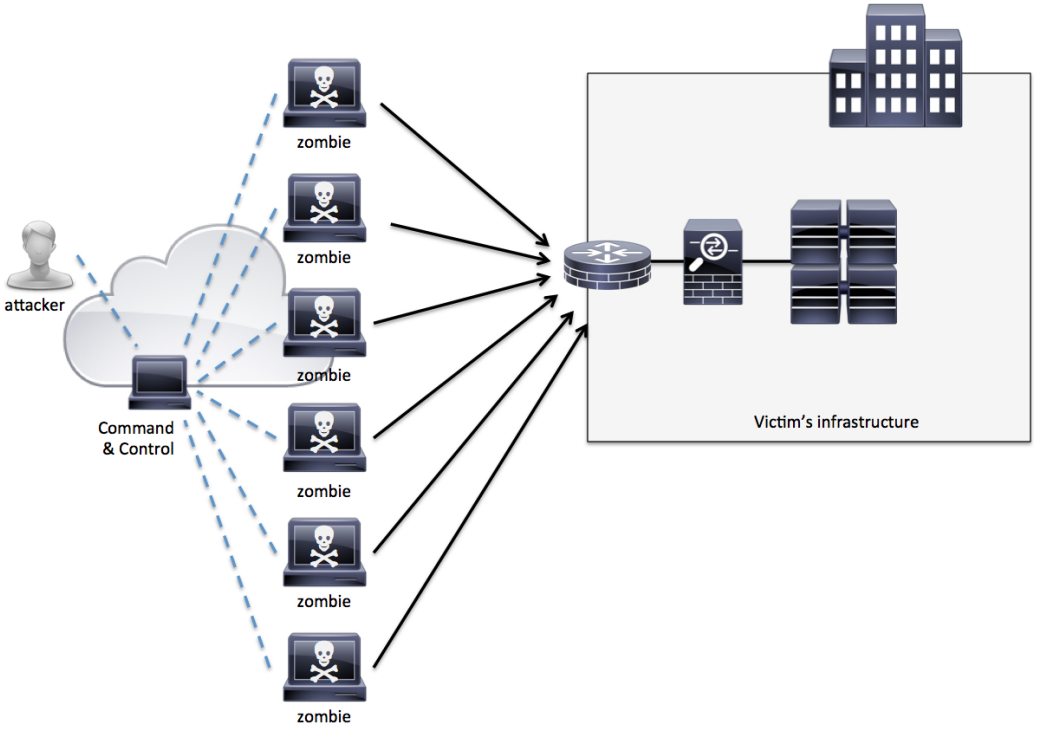

FIGURE 2.1: Direct Flooding Architecture [39].

Some examples of Direct Flooding Attacks (DFA) include:

- **ICMP Flood Attacks**

  The Internet Control Message Protocol (ICMP) Flood attack also referred to as Ping Flood is a type of DDoS attack that involves the sending of a large number of pings to a target so that it becomes overwhelmed and does not respond to legitimate requests [40], [41].

- **UDP Flood Attacks**

  These attacks abuse the User Datagram Protocol (UDP). These attacks are launched by sending a large number of UDP packets to random ports of the target device so that it is overwhelmed and does not respond to legitimate requests [40], [41].

- **SYN Flood Attacks**

  These attacks aim to create many half-open connections by abusing the three-way handshake of the Transmission Control Protocol (TCP) so that it exhausts the resources of the end device and does not respond to legitimate requests [40], [41]. The three-way handshake session comprises of three messages exchanged between a server and a client in order to establish a successful connection as shown in Figure 2.2.

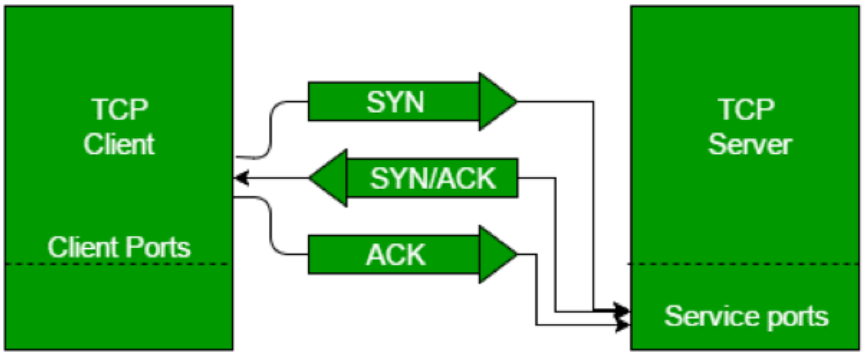

FIGURE 2.2: TCP three-way handshake [42].

The first message towards establishing a successful connection is a SYN message sent from the client to the server and the server responds with the second message SYN-ACK to the client. The connection will be fully established after the client responds with the third message ACK [41]. However, under the SYN flood attacks, many SYN requests are sent to the server without responding with the required ACK message. There will be increase in half-open connections whilst the server is waiting for the ACK message from the client. The server becomes unresponsive to legitimate requests due to the accumulated half-open connections exceeding the server's resources [41].

2. **Indirect Flooding**

   This method of launching DDoS attacks involves the sending of attack traffic to third-party end devices. The third-party end devices in turn route the attack traffic to the target device as shown in Figure 2.3. In some attack scenarios, the third-party devices have the ability to magnify the attack traffic in order to inflict a larger impact on the target devices [31].

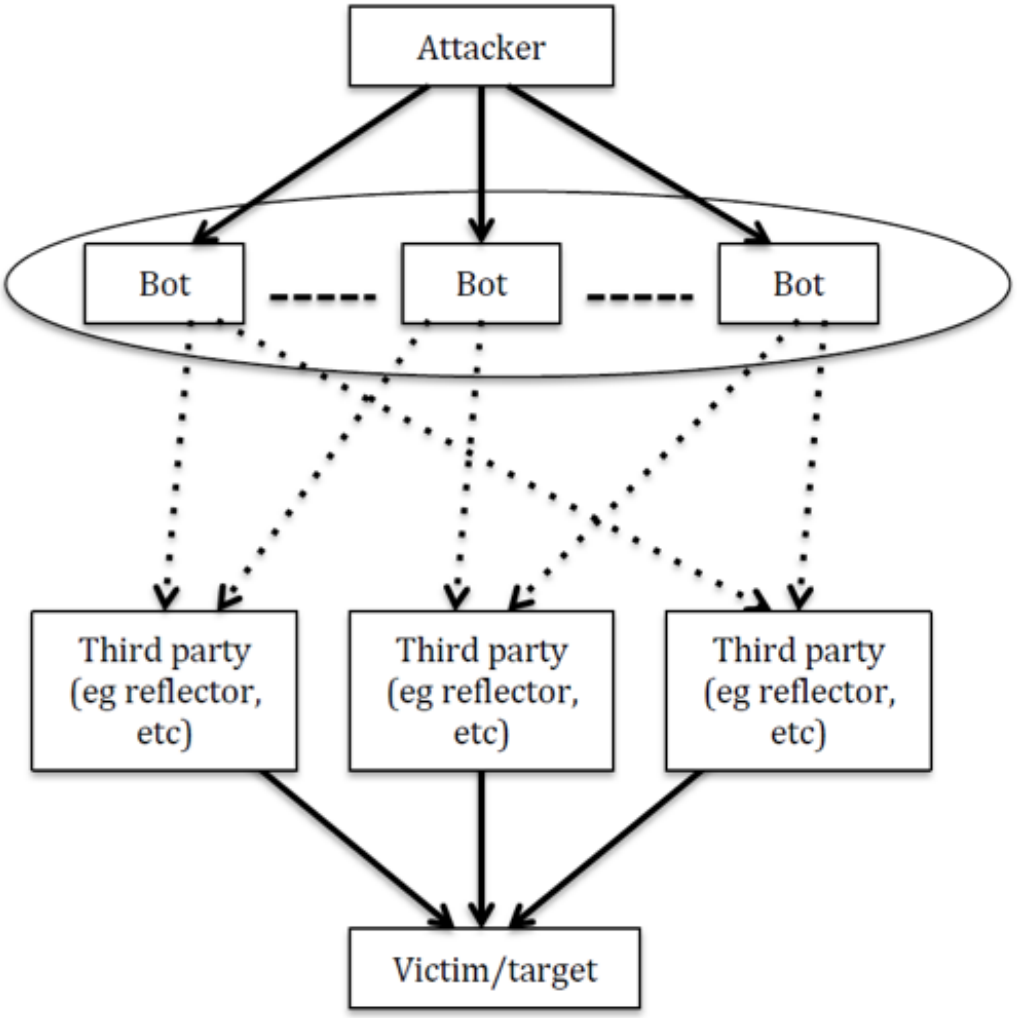

FIGURE 2.3: Indirect Flooding Architecture [31].

In essence, this attack method aims to conceal the original source of the attack to avoid detection. Some examples of Indirect Flooding Attacks (IFAs) include:

- **Coremelt attack**

  This Indirect Flooding Attack was introduced in 2009. Its aim is to congest the network links of the target's network core through the use of many zombies as sources of destinations of attacks [9]. These zombies are capable of communicating with one another legitimately. This attack has proven to be a difficult challenge in terms of detection as the attack traffic generated by the zombies are similar to legitimate network traffic. This attack was demonstrated through a simulation using two network sizes of 4746 and 720 Autonomous Systems (ASes) [14]. The demonstration proved successful with the use of 700,000 to 1,008,000 zombies [14].

- **Crossfire Attack**

  The aim of this attack is to congest the network links of the target's network by sending attack traffic to decoy servers in the target's network [15]. This attack is similar to the Coremelt attack as both attacks make use of zombies for sending the attack traffic however, the intention of the Crossfire attack is not to cause the target servers inaccessible but to act as decoy servers which aids in flooding the network links that are connected to the server [15]. In addition, both attacks are similar as they aim to congest network links between the source and target's network. The Crossfire attack is carefully coordinated to ensure targeted network links are successfully congested in order to cut-off the internet access of the targeted organization's network.

- **Reflection Attack**

  This attack makes use of reflectors on the network in directing attack traffic to the target's network [43]. A reflector is any network device that is capable of returning received packets such as routers, web servers and Domain Name Server (DNS) servers [43]. In order to orchestrate a powerful flooding attack traffic to the target, attackers often combine the use of reflectors and the attacker's machines. Reflectors often act as amplifiers as their reply packets are often larger than the received packets [43]. Therefore, a lot of damage can be caused to the target's network with the use of a large number of reflectors. The use of reflectors can conceal the original attack source thereby resulting in a high level of stealthiness [43]. From the literature, reflectors pose a serious threat to the Internet. These reflectors include TCP based servers that suffer predictable sequence numbers, GNUTELLA servers and DNS servers.

- **Amplification attack**

  This attack makes use of third-party end devices to either broadcast requests with the use of spoofed IP address that is usually the IP address of the target leading to an influx of replies to the target or amplify small requests into larger requests that are directed to the target [44]. A Smurf attack is an example as ICMP echo requests are sent by the attacker to a vulnerable broadcast domain, which causes all the end devices in the domain sending echo reply messages to the target resulting in a huge volume of network traffic which can deplete the network bandwidth of the target [44]. This attack is like reflection attack as spoofed source IP addresses are used. DNS amplification is another example of this attack as the open recursive feature of DNS servers is taken as an advantage by the attacker to send recursive queries to the server, which results in the server sending out large amplified traffic to the target as responses. Other examples include the NTP amplification attack, which is very similar to the DNS amplification attack with the difference being that it exploits the MONLIST command of NTP servers. Therefore, target's bandwidth can easily be depleted by an amplification attack

because there is a significant increase in the attack traffic load directed to the target.

2.1.2 DDoS Attacks Intensities

DDoS attacks can be classified based on their intensities and these are low and high-rate DDoS attacks [31].

- **Low-Rate DDoS Attacks**

  A low-rate DDoS attack is a type of DDoS attack which is stealthy in nature as it makes use of a low-rate or slow network attack traffic to saturate the target's network.

  A Low-Rate DDoS attack such as Slowloris [45] does not rely on network traffic volume in causing denial-of-service but seeks to keep hold of as many opened connection ports as possible to prevent legitimate users from having access to the target server. These attacks are capable of crippling the target's network through significant increase in aggregated attack volume without activating the detection systems. Other low-rate attacks aim to degrade a target service over a long period of time to achieve economic damage instead of completely disrupting service [8].

- **High-Rate DDoS Attacks**

  A high-rate DDoS attack is a type of DDoS attack that makes use of a high network attack traffic to exhaust the target's network resources [46]. An example of this attack is Smurf attacks which involves the transmission of attack packets at a high transmission rate to the target's network that results in a large traffic volume [44]. However, detection systems can be triggered by the rapid increase in network traffic, which can lead to successful mitigation and prevention of the target's network from adverse damages [44].

2.1.3 DDoS Attack Targets

The targets of DDoS attacks are mostly network links and end devices (hosts) in the target's network as these are the core for effectively providing a service.

- **End Device (Host)** In terms of end devices, DDoS attacks target the resources of the host such as memory, CPU, service resources especially in a server or hardware such as switches and edge routers with the aim of bringing down the end device.

- **Network Links** In terms of network links, DDoS attacks aim to congest the network links and exhaust the link bandwidth, which results in bringing down the communication link to the target's network. The attacks on network links are considered more severe as a successful bringing down of a network link results in subsequent bringing down of all hosts in the target network.

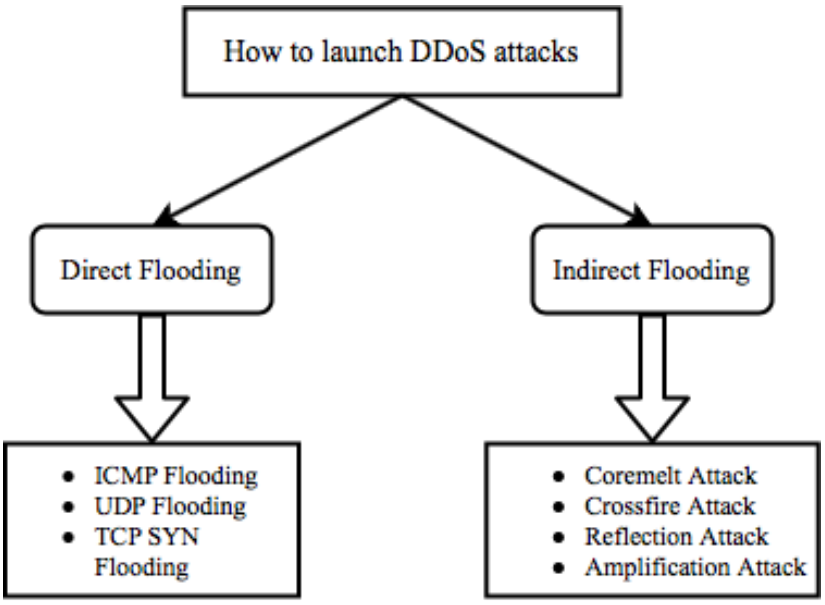

FIGURE 2.4: Methods of launching DDoS attacks

2.1.4 Launching a DDoS Attack

There are different ways of launching DDoS attacks and these are explained below.

- **Botnets**

  A botnet is defined as a collection of connected systems that have been compromised and are are usually controlled by an attacker [47]. A large proportion of the network security incidents on the Internet are based on the use of botnets [47]. These compromised systems are referred to as bots. Due to the fact that these bots are cheap and easy to deploy, many of these are sold at low prices on the dark web [31]. Due to their easy deployability, minimum effort is required by the attacker in initiating the attacks using the bots. The only effort from the attacker is uploading the attack command into Command & Control (C&C) servers which are used to distribute the commands to the individual bots [31].

  A network that comprises of bots that are connected to a single C&C server is referred to as a botnet [31]. These bots are usually distributed globally and are used for launching DDoS attacks directed at a target's network. There has been a rapid increase in the number of bots used in initiating DDoS attacks since 2013 and this is evident in the exponential increase in the size and complexity of DDoS attacks [31]. An example of one of the most recent DDoS attacks botnet is the Mirai botnet. Below are some of the well-known botnets used in initiating DDoS attacks.

  – **Internet-of-Things (IoT) Botnet**

    An IoT botnet is a type of botnet that comprises of IoT devices that are poorly secured but have access to the Internet. These poorly secured IoT devices are

referred to as IoT bots and some of the well-known IoT botnets are the Hajime and Mirai botnets.

- **Internet Chat Relay (IRC) Botnet**

  The IRC is the network protocol that is used by connected devices for real-time text messaging [48]. In order to manage the the busy channels used by the IRC protocol, the IRC bots were introduced. However, these bots are currently used for launching DDoS attacks by attackers targetting IRC servers and users [49]. The attackers make use of a C&C server to initiate the attacks. Some of the well-known IRC botnets are Spybot, Kaiten, Nesebot and Agobot [6].

- **Code Red Worm (CRW) Botnet**

  The CRW botnet makes use of a worm which quickly spreads and infects hundreds of thousands of devices within a short period of time [31]. Each of the compromised devices (bots) has a specific target that has been predefined [31].

- **Peer-to-peer (P2P) Botnet**

  The P2P botnet makes use of P2P compromised devices for launching DDoS attacks but unlike the previous botnets which rely on C&C server, it makes use of communication between the P2P bots [50]. This results in more resilient attacks as any P2P bot can act as C&C server instead of using a special designated server [50]. Some of the well-known P2P botnets include Peacomm, SpamThru, Phatbot and Nugache [51].

2.1.5 DDoS Attack Tools

There are several DDoS attack tools that are readily available to attackers for launching DDoS attacks. Since these attacks target the resources of the target network, attackers make use of a variety of tools for initiating these attacks. This is evident in the increasing size and complexity of DDoS attacks. In addition, the available tools are mostly automated thereby making it easy for attackers to manage and coordinate large-scale DDoS attacks. Below are descriptions of some of the well-known DDoS attack tools and a summary of these tools with respect to the network protocols used in generating these attack types is shown in Table 2.1.

- **Hping3**

  Hping3 is a network security tool that can be used for a wide range of network capability and security testing. It is command-line oriented and can act as both an assembler and analyser of TCP/IP packets. In addition, it can be used to launch a wide range of DDoS attacks as shown in Table 2.1.

- **Tribe Flood Network (TFN)**

  TFN is a DDoS attack tool that provides an attacker the platform for launching a wide range of DDoS attacks which are directed to either random or specified ports of devices in the victim's network [52]. These attacks include UDP flood, Smurf, ICMP flood and TCP SYN flood attacks. This DDoS attack tool comprises of a set of servers for launching the attacks and a master.

  To launch DDoS attacks, attack commands are sent by the master to the servers [52]. Since the communication between the master and the servers is masked using the ICMP Echo reply packet, DDoS attacks initiated with TFN are not very easy to mitigate [52]. In addition, filtering such packets will stop some applications which depend on these packets from functioning appropriately.

- **Low-Orbit Ion Canon (LOIC)**

  LOIC is a DDoS attack tool which is open source and provides an attacker the platform for launching TCP and UDP attacks [53]. A major disadvantage of using LOIC to launch DDoS attacks is that the source of attack can be tracked because the IP addresses of the generated traffic are not masked [53].

- **Stacheldraht**

  Stacheldraht is a DDoS attack tool which provides an attacker the platform for launching ICMP flood, Smurf, UDP flood and TCP SYN flood attacks [52]. A major advantage of using Stacheldraht for launching DDoS attacks is that it is difficult to trace the attack source as this tool makes use of encrypted packets for communication [52].

- **MStream**

  MStream is a DDoS attack tool which provides an attacker the platform for launching rapid DDoS flood attacks such as TCP SYN flood and ICMP flood which deplete the bandwidth of the target's network [54]. A major advantage of using MStream for launching DDoS attacks is that the IP addresses of the source of attacks are masked [54].

- **Trinoo**

  Trinoo also referred to as Trin00 is a DDoS attack tool which provides an attacker the platform for launching UDP flood attacks which deny specific services [52]. It is quite a sophisticated tool as an attacker is capable of modifying the sizes of network traffic packets and specify the duration of a DDoS attack [52].

Table 2.1: DDoS attack tools with protocols for launching attacks

Name of Attack Tool	Protocols for launching attack
HPing3	ICMP, UDP and TCP
LOIC	TCP and UDP
TFN	ICMP, UDP and TCP
MStream	ICMP and TCP
Stacheldraht	ICMP, UDP and TCP
Trinoo	UDP
Shaft	ICMP, UDP and TCP
Knight	TCP and UDP
Trinity	TCP

2.2 How to Detect a DDoS Attack

DDoS attack detection is referred to a process through which attack network traffic are efficiently and accurately distinguished from legitimate network traffic. DDoS attacks detection systems are important in effectively detecting the occurrence of attacks in order to secure the activities of a network. DDoS attacks detection methods are commonly grouped into three categories, which are the anomaly, hybrid and signature-based detection methods.

2.2.1 DDoS Attacks Detection Methods

The primary aim of an Intrusion Detection System (IDS) with respect to DDoS attacks is the detection of the emergence of DDoS attacks, which helps in taking the necessary steps to mitigate and prevent the adverse effects of these attacks. An IDS can either be a software or hardware application that monitors networks and reports any suspicious behaviour especially in the case of attacks to a system administrator for further action to be taken [55]. These detection solutions are widely used in DDoS attacks detection and are classified as either network or host-based depending on the implementation target. It is host based if it is implemented in an end device (host) and network-based if implemented in a network device like firewalls, routers, etc. [55].The signature, hybrid and anomaly-based detection systems are the three categories of detection systems. This is based on the technique that is adopted by an IDS for intrusion detection.

- **Signature-based IDS**

  A signature-based IDS (SIDS) consists of a database used for storing signatures of known attacks. All incoming packets to a target's network are monitored and compared to the stored signatures. Once there is a match in the signatures, the packets are marked as malicious [56]. These systems are easy to develop and implement which are major advantages. However, a major disadvantage is that only known attacks are detected therefore, constant updates of the database with signatures of new attacks are necessary [57]. These detection systems are also known as Knowledge-Based Detection or Misuse Detection [56].

- **Anomaly-based IDS**

  Anomaly-based detection systems also referred to as Anomaly Detection Systems (ADS) are modelled with normal behaviour of the network through the use of knowledge-based, machine learning or statistical-based techniques [57]. The network activity is monitored for any significant deviation of behaviour from the modelled normal behaviour. Once a significant deviation is detected, it is interpreted as an attack scenario [57].

  These detection systems are developed with the assumption that legitimate network traffic behaviours are different from attack traffic behaviours. Therefore, abnormal behaviours are classed as attacks [57]. The ability to detect new/zero-day attacks is a major advantage of these detection systems due to their ability to recognize any abnormal behaviour in the network [58]. This is a major advantage that these detection techniques have over the signature-based detection approach.

  However, the difficulty in defining the perfect thresholds is a major disadvantage as small deviations from the perfect threshold can create a significant number of false positives, which leads to low accuracy in the detection of attacks [59]. Once attacks are detected, response mechanisms, which could either be rate limiting or filtering algorithms, are activated against such network traffic.

- **Hybrid IDS**

  Hybrid Intrusion Detection Systems (HIDSs) are a combination of signature-based and anomaly-based solutions. It is regarded as one of the most effective detection strategy as it can detect a wide range of DDoS attacks due to constant updates of DDoS patterns in a set database [60].

2.2.2 Packet Classification Strategies for DDoS Attacks Detection

DDoS attack detection systems make use of two main network traffic classification strategies as part of the detection process. These network traffic classification stategies are flow and packet classification. Below are descriptions of these two classification stategies.

- **Packet Classification**

  Packet classification incolves the analysis of individual network traffic packet using the Deep Packet Inspection (DPI) technique to distinguish legitimate from attack network packets [61]. This approach makes use of some predefined characteristics of a network packet in determining if a packet is an attack or legitimate i.e. in a traditional TCP SYN flooding DDoS attack scenario, the attack is initiated using TCP packets with the SYN-flag set, therefore, this characteristic could be used along with other properties of the packet to determine if it is a legitimate or attack packet [61]. This approach has been evaluated not to be the best approach in detecting recent variations of DDoS attacks as the packets in these attacks are well coordinated and synchronised to behave more like legitimate packets [31].

- **Flow Classification**

  To overcome the limitations of the network traffic packet classification approach, the flow classification approach is used instead. A network flow can be defined as a stream of network traffic packets with the same network protocol, destination IP address, source IP address, destination port and source port within a period of time [31]. The network traffic flow classification approach can easily be deployed at a nework switch node for the detection of attacks and filtering.

  To detect DDoS attack network flows, a variety of distribution analysis are used by the flow classification approach. However, the pre-processing of network traffic flows can be a time consuming task especially when it involves a large volume of network traffic therefore, the use of lightweight applications are required for the collection of statistics [31]. The flow classification approach can be subdivided into two types based on the number of flow samples considered within the chosen time interval and these are the sampling and complete detection.

2.2.3 DDoS Attack Detection Architectures

There are two major architectures used for the detection of DDoS attacks and these are the distributed and centralised architectures. Below are descriptions of these two architectures.

- **Centralised Architecture**

  In the centralised architecture, all network traffic received are transferred to a central location of the network to be processed [62]. This architecture is better suited for small enterprise networks [62]. The detection of DDoS attacks using this architecture has been evaluated to be ineffective in larger-scale networks due to the huge volume of network traffic involved which can result in a single point of failure and high computational requirements on the centralised system [62].

- **Distributed Architecture**

  In the distributed architecture, multiple devices are used as part of a wider detection ecosystem which results in a distribution of the computational requirements across the devices instead of a single device [62]. A major advantage of this architecture is that the single point of failure present in the centralised architecture is completely eliminated [63]. In addition, the network traffic received are processed by the different devices distributed across the network for the detection of DDoS attacks [62]. However, devices is this architecture must be strategically deployed to achieve better detection accuracy and scalability results [62].

2.3 Major Global DDoS Attack Incidents

There has been a significant increase in the size, complexity and number of DDoS attacks over the years. Personal, financial and political gains have motivated these attacks [64].

Some of the well-known DDoS attacks which have occurred over the past years are outlined below.

- **The Spamhaus Attack**

  Spamhaus is an organisation that is responsible for online filtering of malware and spam transmitted through the Internet. In 2013, it was hit with a DDoS attack peaking at 300Gbps [65]. The attack was successful in overloading servers used by Spamhaus and resulted in huge Internet congestion [65]. The attacker made use of the Open Resolver Domain Name Server in transmitting a huge volume of network attack traffic directed at the servers of Spamhaus and the attack successfully crippled the network of Spamhaus for the duration of more than a week [65].

- **The ProtonMail Attack**

  ProtonMail is an organisation that provides free, secure and encrypted email services across Europe. In 2015, it was hit with a DDoS attack that brought down its main data centre. All the providers of upstream Internet Services connected to the ProtonMail's data center were unable to access it due to the congestion experienced from the attack [66]. The attack continued for over a week with over one hundred affected companies [66].

- **The BBC Attack**

  In December 2015, the BBC was hit with a then record breaking DDoS attack of 600 Gbps [67]. The attack that was carried out by the group called "New World Hacking" using its BangStresser application tool. BBC services including the BBC's website and its iPlayer on-demand service were successfully taken down for about three hours by the attack [67].

- **The Dyn Attack**

  Dyn is an organisation that is responsible for servicing a major proportion of the Domain Name Server infrastructures of the Internet. In 2016, it was hit with a DDoS attack peaking at over 1.2 Tbps through the use of a Mirai botnet [68]. The attack make use of over one hundred thousand compromised systems (bots) which successfuly took down major websites including Twitter, Github and Netflix [68].

- **The GitHub Attack**

  GitHub is a well-known online code management service which is utilized by millions of developers across the globe, GitHub was hit with a then record breaking 1.3 Tbps of network traffic which flooded the company's servers at a packet rate of 126.9 million packets per second in 2018 [69]. This attack successfully took down GitHub's servers for around twenty minutes.

- **The Amazon Web Services (AWS) Attack**

  In February 2020, Amazon Web Services (AWS) was hit with a then record breaking network attack traffic which peaked at 2.3 Tbps [70]. Some Connectionless web servers making use of the Lightweight Directory Access Protocol (CLDAP) were hijacked by the attacker and used for conducting the attack [70].

2.4 Network Traffic Analysis Tools

A network traffic analysis tool is an application that can be used to record and analyze specified network traffic within a network [71]. These tools are capable of dissecting network traffic based on IP addresses, applications, network protocols, etc. and can produce visual representations of data flows through tables or diagrams. Furthermore, these tools aid in the identification of problems within a network insfrastructure with possible solutions [71]. Some of the most widely used network traffic analysis tools are described below:

2.4.1 Wireshark

This is an open source network analyzer that can be used to display packets/flows passing through a network using a Graphic User Interface (GUI) [72]. It comprises of a wide range of properties i.e. live packet capturing, filtering of packets using either protocol type or other methods. The filters are of two types i.e. capture filter for capturing specific type of data and display filter for the display of specific data type i.e. TCP, UDP, etc. [72]. Wireshark can be installed and executed on Windows, Linux or Mac Operating Systems.

2.4.2 TShark

This is a network protocol analyser that can be used to capture live network traffic or read packets from previously captured files [73]. It makes use of the native capture file format pcapng which is widely used by wireshark and other network capture and analysis tools. TShark can be installed and executed on Linux and Mac Operating Systems via the Command Line Interface (CLI).

2.4.3 Scapy

This is a python based powerful interactive packet manipulation program with the capabilities of forging and decoding packets of a wide range of network protocols [74]. In addition, it can send packets over the wire, capture network traffic and other security tasks such as scanning, probing, tracerouting, etc. [74]. Scapy can be executed and run on Windows, Linux or Mac Operating Systems.

2.5 Feature Selection and Machine Learning Classification

This section provides a discussion on feature selection and the importance of using feature selection techniques in large datasets. Furthermore, it provides on overview into Machine Learning (ML) and provides a review on some of the related works conducted in the field of DDoS attacks detection using machine learning classifiers.

2.5.1 Feature Selection Techniques

Feature selection methods are a subset of dimensionality reduction techniques, which is a critical aspect in building Machine Learning classification models. Dimensionality reduction is the process whereby the number of available attributes in a dataset are reduced to a set of principal attributes which are deemed effective in solving the specified problem [75], [76]. Some benefits of dimensionality reduction include:

- Reduction in the number of attributes leads to lower memory requirement for data storage.
- Less computation/training time for ML algorithms
- It helps some ML algorithms to perform better as the use of a large number of attributes can affect performance.
- It resolves the problem of multicollinearity as redundant features are removed.
- It helps in data visualization.

Dimensionality reduction consists of aspects such as feature selection, projection based and component/factors-based reduction as shown in the Figure 2.5.

Selection of the most salient features to be used in detecting attacks is one of the most frequent issue faced by researchers. The selection of features is a very important step to determine the performance of the system. For example, the use of too many features may lead to high detection accuracy, however, it may also lead to an overly complex system which will require more processing power while the use of too little features may lead to low detection accuracy [76], [77] albeit using less processing power. Finding the balance between high detection accuracy and low complexity in design is important. This is because although high level of accuracy is very important, available resources in implementing the detection systems may be limited especially in terms of memory and processing power requirements.

Some of the most widely used feature selection methods in this research area include the Backward Feature Elimination, High Correlation filter, Forward Feature selection, Random Forest and Chi-squared feature selection. A discussion on some of the well-known feature selection techniques has been provided in Chapter 3.

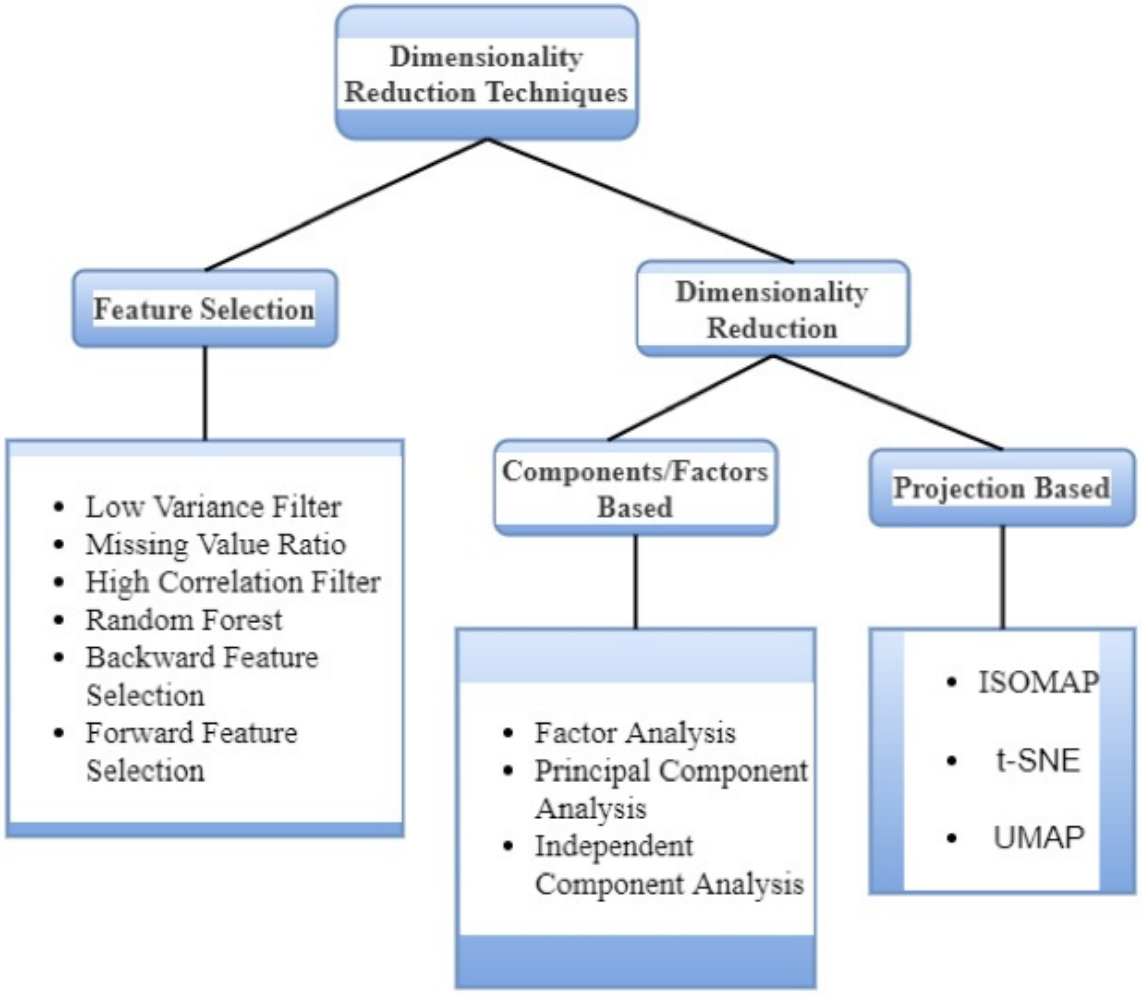

FIGURE 2.5: Dimensionality Reduction Techniques

2.5.2 Machine Leaning Classification

Machine learning (ML) is regarded as an approach through which different machine learning models gain up-to-date knowledge using already existing knowledge with the aim of improving performance [78], [79]. Over the past years, ML approaches have been implemented for the purpose of detecting and preventing more sophisticated DDoS attacks that traditional DDoS detection solutions struggle to cope with [79]. Due to the ever-increasing size, number and complexity of network traffic anomalies, the analaysis of network tarffic for detecting these anomalies were carried out using different ML approaches since 1994 [80]. These ML models are also referred to as ML classifiers.

Some of the most widely used ML classifiers for solving classification problems include Multilayer Perceptron, Random Forest, Decision-Tree and Support Vector Machine algorithms. Detailed explanations of these classifiers have been provided in Section 3.3 of Chapter 3.

2.5.3 Machine Learning DDoS Attacks Detection Solutions

Based on recent literature, some proposed machine learning solutions used for DDoS attacks detection are reviewed in the next paragraphs.

The work presented in [81] proposed a multi-ANN classifier for efficient and accurate DDoS attacks detection using an enhanced Genetic Algorithm (GA) and Principal Component Analysis (PCA) for both the optimisation of parameters and improvement in feature selection. The proposed classifier was trained using samples from both old and up-to-date datasets. From the results presented, the approach was effective and accurate in detecting unknown and known DDoS attack samples with similar patterns to those used in the training set.

DDoS attacks detection based on feature selection (i.e. 27 features) using an ensemble ML consisting of Multilayer Perceptron, Random forest, and Naïve Bayes was proposed to improve the detection accuracy of DDoS attacks [82].

Work proposed in [83] makes use of statistical features selected from a Simple Network Management Protocol (SNMP) Management Information Base (MIB) instead of using real DDoS attack packets. Their implementation using the Support Vector Machine (SVM) algorithm was effective in detecting a real attack after being tested with real DDoS attack traces.

An intelligent DDoS attack detection system based on packet analysis and the SVM algorithm was implemented in [84]. The detection system made use of packet properties such as difference in the mean-time interval between the attacker and victim, the mean-time interval frequency of the attackers IP addresses, etc. and achieved a detection rate of over 85% using two different datasets.

A deep learning approach was proposed for the DDoS attacks detection for the Internet of Things (IoT) infrastructures [85] with higher accuracy. A distributed approach was implemented, and tests were carried out using the KDDCUP99 and NSL-KDD datasets with 41 selected features. The results indicated a high detection rate for intrusions such as Probe and Denial of Service attacks and highlighted the advantages of using distributed over centralized monitoring systems and deep models over shallow [85], [86].

A Feature Feature Score (FFSc) technique for multivariate data analysis was proposed in [87] to distinguish normal network traffic from DDoS attack traffic. The approach was tested on a normalised Center for Applied Internet Data Analysis (CAIDA) 2007 DDoS dataset and it achieved a detection rate of over 98%.

A hybrid machine learning technique DDoS attack detection system is proposed in [88]. The approach made use of the Genetic Algorithm for features selection based on 43 features generated using Network Measurement and Accounting System (Netmate) Software [89]. The Multilayer Perceptron algorithm was used for attack detection using a ten-fold cross validation technique. The approach achieved a high detection rate of over 99% however, a single dataset was used throughout the paper therefore its performance on a wider and more recent range of datasets would need to be validated.

On the other hand, an approach proposed in [90] makes use of an ANN classifier to flag both known and unknown DDoS attacks (which could be TCP, ICMP and UDP) from legitimate network traffic. The detection results obtained were compared with training conducted using both old and up-to-date datasets and it was observed that poor detection results were obtained with improper training of old patterns. In addition, based on the results presented, the proposed approach showed better precision, accuracy, specificity and sensitivity output when compared to other traditional detection systems such as Probabilistic Neural Network (PNN), Snort and Back-Propagation (BP) approaches [90].

In another study in [91], the authors proposed the use of Learning Vector Quantisation (LVQ) neural network classifier which is a supervised form of quantisation used for classification, data compression and pattern recognition tasks for the purpose of detecting DDoS attacks. The experiment conducted made use of dataset samples that were converted to numerical values which were used as inputs to the neural network for DDoS attacks detection [91].

The use of a SVM based detection system was proposed in [92]. The authors made use of a hierarchical clustering algorithm for data pre-processing before training and testing the classifier. The use of the hierarchical clustering algorithm resulted in the reduction of dataset size with sustained quality of the original dataset. This application resulted in a reduced training time and better detection accuracy by the classifier [92].

A real-time DDoS attacks detection system using an enhanced Seupport Vector Machine (eSVM) string kernel was proposed in [93] for classifying incoming network traffic flows as either legitimate or attack flows. In the authors approach, a normal network traffic profile was constructed using the behaviour of the user's access which was then used to build a model file by the eSVM for classifying network traffic as either attack or legitimate [93].

The authors in [94] proposed a DDoS attacks detection system using the Decision-Tree (DT) classifier based on the C4.5 algorithm. In addition, the c4.5 algorithm was used as part of the feature selection technique for splitting the datasets into further smaller subsets. The datasets were repeatedly splitted until all data samples in the subsets belong to the same class [94].

2.6 High Performance Computing Platforms

Nowadays, most computers are used as general-purpose computers as a variety of tasks are to be performed and for these tasks to be accomplished, multiple programs are to be executed. Apart from general-purpose computers, there is an availability of special-purpose computers. These special-purpose computers are mostly used to solve specific problems and are designed based on specific requirements. However, the capability of problem solving is dependent on the program stored and the computer system's design as stated in [95].

Both special-purpose and embedded systems are designed and dedicated to specific tasks. These tasks are performed repeatedly to meet the needs of the task assigned. The design of these systems consists of both hardware architecture and software methodology, which can

be controlled by the embedded system designer. The flexibility of hardware design, software development, embedded system platforms including memory design, processor architecture and operating systems lead to improved design performance and reduced costs [96].

However, real-time constraints are often faced by embedded systems for example; embedded systems have the traditional problem of maintaining efficiency during the real-time execution of complex algorithms [96]. To address the traditional problem of embedded systems, parallel computing platforms and parallel programming methods can be used. This parallelism is often employed in High Performance Computing (HPC) environments. To build HPC systems using parallel computing platforms, different options are available. General-purpose computing can be found on Graphics Processing Unit (GPU), Multi-core processing systems, Field Programmable Gate Array (FPGA) and Application-Specific Integrated Circuits (ASICs) devices. A detailed discussion of these parallel processing high-performance computing platforms that can be used to obtain optimized real-time performance in embedded systems can be found in Chapter 4.

2.7 Available DDoS Attacks Datasets

There are some well-known DDoS attacks datasets that have commonly been used for developing DDoS attacks detection solutions over the years. Some of these datasets include the 1998 DARPA Intrusion Detection dataset generated and managed by MIT Lincoln Labs, the 1999 KDD intrusion detection, the CAIDA 2007 DDoS dataset, the Information Security Centre of Excellence (ISCX) 2012 daaset, the Canadian Institute for Cybersecurity Intrusion Detection System 2017 (CICIDS2017) dataset and the the Canadian Institute for Cybersecurity 2019 DDoS (CICDDoS2019) dataset just to name a few [97]–[99].

However, for the experiments carried out in thesis, the CAIDA 2007, CICIDS2017 and the CICDDoS2019 datasets were selected as these consists of relevant and up-to-date DDoS attack types spanning over just over a decade. The three main datasets used in this thesis are described below:

- **The CAIDA 2007**

  This is a well-known DDoS attack dataset which is widely used by researchers around the world and one of the most prominent datasets currently widely available [100]. It consists of approximately one hour of network traffic traces with over 350 million normal and attack packets captured on the 4th of August 2007. It consists of ICMP and TCP SYN flooding DDoS attacks. The results of a comprehensive analysis conducted on the dataset can be found under Appendix A and the analysis were carried out using the CICFlowMeter tool [101]. From the data analysis carried out, this dataset consists of over 300,000 network flow records.

- **CICIDS2017**

  This dataset was generated through a 5 days simulation of different network attacks in a real network environment by the Canadian Institute for Cybersecurity [98]. It

contains both legitimate and the most common up-to-date DDoS attacks [102]. The DDoS attacks data samples were extracted from the dataset and used in the experiments carried out in this research [98], [102]. From the data analysis carried out, the CICIDS2017 dataset consists of 225,745 network flow records of which around 57% were classed as attack and around 43% as legitimate network traffic flows.

- **CICDDoS2019**

  This dataset is provided by the Canadian Institute for Cybersecurity and consists of both legitimate and the most common up-to-date DDoS attacks in Packet Capture (PCAP) format that were generated in a real network environment [5]. The dataset contains different up-to-date DDoS attacks such as SYN flooding, LDAP, Port Map, UDP flooding, NetBIOS, UDP-Lag, SNMP, MSSQL, DNS and NTP attacks [5], [99]. From the data analysis carried out, the CICDDoS2019 dataset consists of 294,627 network flow records of which around 58% were classed as attack and around 42% as legitimate network traffic flows.

2.8 Summary

In this chapter, a thorough review has been conducted on the concept of DDoS attacks including the different classification of attacks and the ever-increasing severity of these attacks to network infrastructure. In addition, the different DDoS attacks detection methods, strategies and architectures have been studied and reviewed. Related works in the areas of feature selection, use of machine learning models and the use of high-performance computing platform for fast and early detection of DDoS attacks have been reviewed and presented.

The main observations made from reviewing several DDoS attack detection approaches can be summarised:

- All DDoS attack detection approaches concentrate mostly on improving the accuracy of detecting host-based DDoS attacks. Based on the review conducted on existing DDoS attacks detection solutions, less attention has been given to the computational costs of detection systems (processing power requirements and memory consumptions), flexibility in deployments to support the different needs of networks and accurate detection of network link based DDoS attacks.

  Therefore, the investigation on the development and implementation of a generalised lightweight and accurate DDoS attacks detection system (capable of detecting a wide range of DDoS attacks both host-based and network link-based attacks) with low computational cost and high flexibility in deployment is paramount to meet the different needs of current networks.

- Existing DDoS attacks detection solutions lack the capability of early detection of a wide range of DDoS attacks with high accuracy. Most traditional DDoS attacks detection systems are centralised and as such, focus on detecting DDoS attacks within the target's network from a single monitoring point.

With the ever-increasing size, complexity and severity of DDoS attacks, distributed monitoring techniques should be adopted to detect the occurrence of DDoS attacks early and mitigate the adverse effect that these attacks may have if successful. The use of distributed monitoring approach will lead to early detection and closer to the sources of attacks so that they are adequately mitigated. Furthermore, the use high-performance computing platforms would aid fast classification of network traffic flows (between hundreds of nanoseconds to a few milliseconds as shown in the literature) leading to early detection of attacks.

- Current DDoS attack detection solutions lack flexible deployment, programmability, scalability for detecting a wide range DDoS attacks attack types. Existing detection solutions are only deployed in specific locations of a network for example, most traditional solutions are deployed centrally to meet the needs of the network being monitored however, the growing sizes of networks and the increasing demands mean that the use of detection solutions that can be flexibly deployed at different network nodes is crucial in order to meet the different needs of the network and aid the early detection of these attacks.

  The programmability of detection solutions is very important due to the increasing complexity of modern attacks. Programmability offers the possibility of detection models to be retrained with new datasets with new attacks trends to be able to effectively cope with new attacks.

Chapter 3

A Lightweight Decision-Tree Algorithm for detecting DDoS flooding attacks

In Chapter 2, different network DDoS attack types were presented. These attacks are achieved by sending large volumes of network traffic from many exploited systems called zombies to either a host or a network targeting the exhaustion of network resources. Several proposed DDoS attacks detection solutions were reviewed in the literature but a lot of these are software based and focus more on detection rates and accuracy with very little or no focus on system's resource utilization.

In this chapter, the limitations of the detection methods presented in the literature are presented and a lightweight Decision-Tree algorithm called DT3 based on a classification approach is presented. A lightweight DDoS attacks detection application as explained in [103] is an application with low processing overhead, low resource utilization and fast detection (in milliseconds) of DDoS attacks. The proposed lightweight architecture distinguishes attacks from normal network traffic flows with a detection accuracy of over 99.6% across all selected datasets. The architecture presented is optimised for deployment in low-cost environments for efficient, rapid detection and prevention of DDoS attacks. To achieve a computationally efficient architecture, the model was trained with a minimal number of features using a robust feature selection approach and validated against the CAIDA 2007, Canadian Institute for Cybersecurity (CIC) CICIDS2017 and CICDDoS2019 datasets. Analysis of the design is presented along with system resource utilization under different network traffic conditions and results shows that the new architecture provides no additional overhead to the monitored network.

3.1 Background and Motivation

In Chapter 2, it was outlined that DDoS detection methods are categorised as:

- Signature-based;

- Anomaly-based and
- Hybrid

To tackle DDoS attacks, many researchers have proposed different detection systems [19]. Some make use of data mining techniques as part of the implementations. However, the ever-increasing variations of DDoS attacks have rendered some of the proposed detection systems ineffective as these systems struggle to cope with the current network line rates and the complexity of current attacks. Furthermore, some systems are very complex, leading to high computational cost, and incur a high overhead to the network being monitored.

Previous research into DDoS detection have focused more on obtaining high accuracy, reduction of false alarm rates and simplification of detection systems [18], [19]. However, less attention has been given to the computational costs of detection systems (processing power requirements and memory consumptions), and flexibility in deployments to support the different needs of networks and distributed monitoring [18], [19].

This suggests there remains scope for new and more efficient approaches to detect DDoS attacks. For these reasons, a lightweight architecture for detecting DDoS attacks using a robust feature selection technique is proposed in this chapter. The proposed approach comprises of the design and implementation of a DDoS attack detection solution that is based on a Machine Learning (ML) classifier. The main advantage of the proposed solution is its ability to be distributed throughout the network being monitored, leading to a lightweight and scalable architecture that retains the ability to obtain high detection accuracy using simple designs with low computational cost without affecting the performance of the network.

3.2 Proposed Approach

The use of ML classifiers provides an avenue through which data samples are assigned to known class labels. In the proposed approach presented in this chapter, there are only two known class labels of network traffic flows: legitimate and attack. The type of classification involving two class labels is known as a binary classification issue. Each class label in a classification method consists of a series of extracted features from past observations in one or more datasets.

The process of choosing a classifier for distinguishing legitimate from DDoS attack network traffic flows is a non-trivial task as it is not possible to know apriori which one is the best in solving the stated problem. Therefore, several supervised learning classification methods were evaluated, and it was discovered that the Decision-Tree classification algorithm performed the best. This chapter focuses on the implementation using the Decision-Tree classification algorithm. However, comparisons will be drawn with respect to related work in areas which made use of other supervised classification methods. The subsections below will outline the steps that are required in effectively choosing the ML classifier which is capable of efficiently and accurately distinguishing legitimate from DDoS attacks network traffic flows as shown in Figure 3.1.

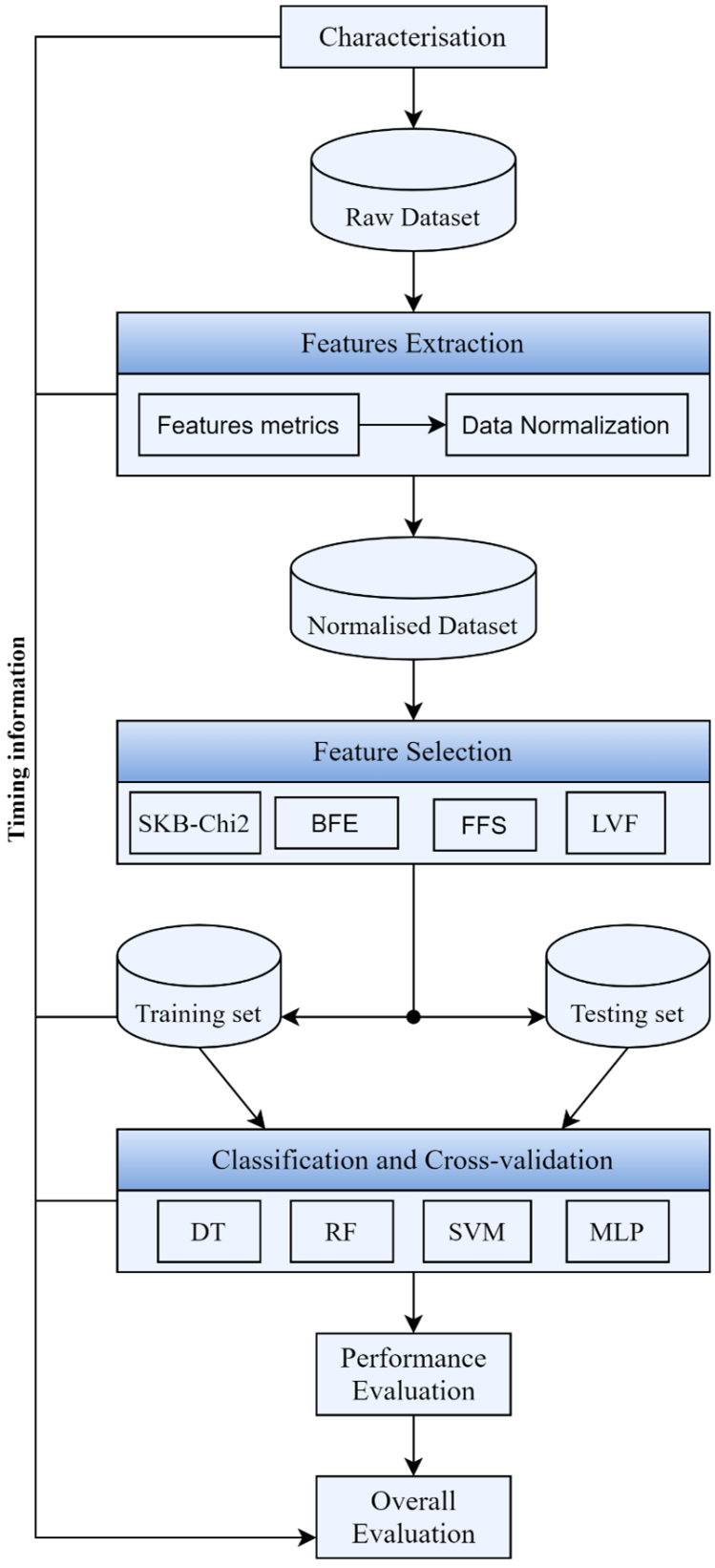

FIGURE 3.1: Proposed approach.

3.2.1 Datasets

The primary dataset used is the CAIDA 2007 DDoS attack dataset. In addition, the CICIDS2017 and CICDDoS2019 datasets were used to test the validity and robustness of the selected Machine Learning (ML) models. The data samples (network flows) were randomly selected from the datasets described in Chapter 2 to create a balanced dataset of equal number of legitimate and attack network traffic flows from all selected datasets as shown in Table 3.1. This was done to overcome the problems of bias and overfitting which can occur as a result of using unbalanced datasets to train classifiers [104]. These were used for training and testing the selected ML classifiers.

TABLE 3.1: Number of network flows extracted from the selected datasets

Dataset	Total Number of samples	Legitimate samples	DDoS Attack samples
CAIDA 2007	172,800	86,400	86,400
CICIDS 2017	195,400	97,700	97,700
CICDDoS 2019	101,000	50,500	50,500

3.2.2 Pre-processing

To effectively and efficiently implement the DT models, important attributes relevant to the DDoS attack flows were extracted from the datasets. The approach focuses on distinguishing normal from attack traffic flows and not on individual packets. Over 40 network flow features were extracted from the datasets using the Tshark Library and CICFlowMeter used in [5], [73], [101], [102]. Not all features are highly correlated in distinguishing flow records, in this regard a robust technique; Low Variance Filter (LVF) feature selection to highlight only the most relevant DDoS features was utilized.

3.2.3 Feature Selection Methods

Below are descriptions of the most widely used feature selection techniques in selecting the top ranked features that are deemed effective in distinguishing normal from DDoS attacks network flows using ML algorithms along with the approach explored in this chapter:

- **Missing Value Ratio (MVR)**

  This method is used to reduce the features with significant missing values. If the dataset consists of attributes with too many missing values based on a set threshold, then these attributes are dropped.

- **High Correlation Filter (HCF)**

  This technique is used to remove highly correlated attributes from a dataset. This is because data attributes that are correlated are likely to carry very similar information. To achieve this, the dataset must be normalised as correlation is scale sensitive [77]. In

addition, the correlation coefficient between attributes that are numerical or nominal in nature are calculated using the Pearson's chi square value and the Pearson's Product Moment Coefficient [77]. Then, pairs of attributes are reduced to a single attribute when a high correlation coefficient value which is above a set threshold is attained. For example, a correlation coefficient that is equal to means that there is no correlation between selected attributes however, if it is equal to 1, it indicates that there exists a full correlation so one of the attributes can be dropped.

- **Random Forest (RF)**

  Apart from being effective classifiers, random forests are useful techniques for feature selection. This technique presents the importance of each attribute in the dataset. It achieves this by generating a set number of constructed trees against a specific attribute and uses the statistics obtained to find the most informative subset of features. An attribute that is often selected is likely to be an informative attribute and it is retained. The calculated score of the attribute's usage statistics shows its predictive power with respect to the other attributes in the dataset. The top features are retained thereby leading to dimensionality reduction in the dataset.

- **Backward Feature Elimination (BFE)**

  This technique aims to eliminate the attributes that offer the least predictive power towards the overall accuracy of the ML model in use through multiple iterations of training. To achieve this, at the first iteration, the chosen classification algorithm is trained with the total number of attributes (n) in the dataset. Then the attribute whose removal produces the smallest increase in the error rate of the algorithm is removed from the dataset and the model is trained with $n - 1$ attributes n times [105]. This process is then repeated using $n - 2$ attributes until a set threshold is met. For each iteration i, a model is trained with $n - i$ attributes and an error rate $e(i)$. By setting a maximum tolerable error rate, a set number of attributes necessary to achieve a classification performance are selected [105].

- **Forward Feature Selection (FFS)**

  This technique is the inverse to the Backward Feature Elimination technique. It starts with a single attribute and progressively, one attribute is added after each iteration and the attributes that contributes the highest performance increase are reserved [105]. Both the Forward Feature Selection and Backward Feature Elimination techniques are time and computationally expensive [105]. Therefore, these techniques are practically applied to small datasets with low number of attributes [106].

- **Chi-Squared**

  This statistical test technique is applied to a group of categorical attributes to evaluate the likelihood that there is a correlation or association between the attributes based on the frequency distribution of the attributes [107].

- **Low Variance Filter (LVF) Feature Selection**

  In this technique, attributes with very little or no changes in the dataset carry very little information and are therefore ignored. This technique is very effective in datasets that are numerical in nature [77]. The technique is used to filter out attributes with low variances which might contribute little or nothing towards the overall efficiency of the model and it involves calculating the variance (using Equation 3.1) of each attribute from a balanced dataset and filtering out the attributes with low variances below a set threshold. Also, variance is range dependent therefore, normalization of the dataset is required before the application of this technique.

TABLE 3.2: Characterisation metrics.

Metrics	Equation
Basic summary statistics	
Minimum	$\min = \min(x_i)$
Maximum	$\max = \max(x_i)$
Sum	$\text{sum} = \sum_{i=1}^{N} x_i$
Range	$\text{range} = \min - \max$
Central tendency	
Mean	$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$
Mode	$\text{mode} = \text{freq}(X)$
Median	$\text{median} = \frac{1}{2}\left(x_{\frac{N}{2}} + x_{\frac{N}{2}+1}\right)$ of $\text{sort}(X)$

  X represents all samples; X_i is the $i-th$ sample; N represents the total number of samples; freq is the most frequent value; sort represents values in sorted order.

$$Variance\ (\sigma^2) = \frac{\sum_{i=1}^{N}(X_i - \mu)^2}{N} \tag{3.1}$$

  where μ is the mean of all the values of the selected attribute and is calculated using the equation highlighted in Table 3.2. X_i represents individual sample values for the specific attribute and N is the total number of samples.

Some of the extracted network flow features and their meanings are shown in Table 3.3. However, tables containing all the extracted network flow features and their meanings can be found in Appendix E.

Also, Table 3.4 shows some typical network flow records with some of the features. However, since variance is range dependent, normalisation of the dataset is therefore necessary before applying the Low Variance filter. In this regard, the min-max normalisation technique (using

Table 3.3: Some Extracted Features and their meanings

Features	**Meaning**
Fwd IAT Mean	Mean time between two packets sent in the forward direction received by target
Fwd Bytes	Number of bytes in the forward direction received by target
Avg Packets/s	Average size of packet of flow
Fwd IAT Std	Standard deviation time between two packets received by target
Fwd Packets/s	Number of forward packets per second received by target
Total Fwd Packets	Total packets in the forward direction received by target
PSH Flag Count	Number of packets with PUSH flag set
Bwd Bytes	Number of bytes in the backward direction sent by target
Bwd IAT Mean	Mean time between two packets sent in the backward direction sent by target
Bwd IAT Std	Standard deviation time between two packets sent by target
ACK Flag Count	Number of packets with ACK
Fwd PSH Flags	Number of times the PSH flag was set in packets received by target
SYN Flag Count	Number of packets with SYN flag set
Total Bwd Packets	Total packets in the backward direction sent by target
Bwd Packets/s	Number of backward packets per second sent by target
Bwd PSH Flags	Number of times the PSH flag was set in packets travelling from target
Fwd IAT Tot	Total time between two packets sent in the forward direction

Equation 3.2) was applied where each data value is normalised to a value between 0 and 1 where 0 is minimum and 1 the maximum using the equation below and some examples of the normalised samples are shown in Table 3.5.

Table 3.4: Some extracted network flow features from the CICDDoS2019 dataset

Flow Duration (microseconds)	**Flow Packets/s**	**Fwd IAT Mean (microseconds)**	**Bwd IAT Mean (microseconds)**	**Label**
47549279.00	0.42	5283241.56	5283241.67	DDoS
46000372.00	0.43	5111141.33	5111140.56	DDoS
48494766.00	0.29	5388299.00	6507682.00	DDoS
43082212.00	0.37	4786902.78	5586661.60	DDoS
47986607.00	0.50	3691269.77	5331839.56	DDoS
30489331.00	1.97	1218439.72	923060.06	BENIGN
225244.00	159.83	9889.94	10830.29	BENIGN
464754.00	234.53	8529.71	6833.42	BENIGN
140318.00	228.05	8243.47	7644.54	BENIGN
10096655.00	4.16	5430.06	457639.68	BENIGN

$$\tilde{x}_j^k = \frac{x_j^k - \min^k}{\max^k - \min^k} \tag{3.2}$$

where x_j^k is the $j-th$ value of the $k-th$ feature and the minimum and maximum values of the $k-th$ feature are $\min^k$ and $\max^k$ respectively.

TABLE 3.5: Records of some normalised network flow samples from the CICDDoS2019 dataset.

Flow Duration (microseconds)	**Flow Packets/s**	**Fwd IAT Mean (microseconds)**	**Bwd IAT Mean (microseconds)**	**Label**
0.953101804	0.000000008	0.105899967	0.105899969	DDoS
0.957746177	0.000000009	0.106416010	0.106415994	DDoS
0.942219682	0.000000006	0.104690914	0.126439749	DDoS
0.929872761	0.000000008	0.103318987	0.120580727	DDoS
0.963597766	0.000000010	0.074122751	0.107066305	DDoS
0.985743927	0.000000064	0.039393110	0.029843254	BENIGN
0.971636388	0.000689460	0.042662293	0.046718687	BENIGN
0.947657197	0.000478219	0.017392515	0.013933693	BENIGN
0.915818579	0.001488422	0.053802955	0.049893896	BENIGN
0.966234623	0.000000398	0.000519649	0.043795426	BENIGN

To evaluate possible changes in variance of the extracted features, the variance of each attribute was calculated from a pool of 6,000, 12,000, 24,000 and 48,000 balanced and normalised flow records. In addition, the average variance value from the different pools of normalised flow records was recorded for each attribute.

From the 40+ extracted features, 15 features were observed to have variance values between 0.00032 and 0.37544 with other features having zero variance as shown in Table 3.6. Commencement of training and testing were conducted using attributes with conservative (i.e. lower) threshold values but discovered that higher detection accuracy stems from attributes with variance values above 0.025. Therefore, this was set as the threshold value.

As can be observed in Table 3.6, the first three features contribute most value to the overall sum of the variance values from all attributes. Accordingly, the top 3, 5 and 7 features were selected respectively due to high variance values of over 0.025 compared to the other attributes. The results from using the top 3, 5 and 7 features from the LVF are compared with the results from using the top 3, 5 and 7 features from different feature selection techniques to evaluate the effectiveness of the selected features.

Table 3.7 and Table 3.8 summarise some of the related work carried in the field highlighting the feature selection techniques applied, the number of selected features and the different Machine Learning algorithms implemented. From the tables, it can be observed that most of the feature selection techniques applied have been discussed in Section 3.2.3 of this chapter.

Table 3.6: Variance of Extracted Network Flow Features

Extracted Flow Features	**Variance values**
Fwd IAT Mean	0.37544
Fwd Bytes	0.22656
Avg Packets/s	0.20336
Fwd IAT Std	0.03251
Fwd Packets/s	0.02879
Total Fwd Packets	0.02795
Fwd PSH Flag Count	0.02542
Bwd Bytes	0.01942
Bwd IAT Mean	0.01878
Bwd IAT Std	0.01767
ACK Flag Count	0.01480
Fwd PSH Flags	0.00624
SYN Flag Count	0.00284
Total Bwd Packets	0.00148
Bwd Packets/s	0.00032
Bwd PSH Flags	0.00000
Fwd IAT Tot	0.00000

Table 3.7: Summary of feature selection for ML-based DDoS detection systems

Work	Feature Selection	Selected Features	ML Detection Model
[83]	SNMP MIB	ip.ipInReceives, ip.ipInDelivers, ip. ipOutRequests, ip.ipOutDiscards, tcp.tcpAttemptFails, tcp.tcpOutRsts, udp.udpInErrors, icmp.icmpInMsgs, icmp.icmpInErrors, Icmp.icmpInDestUnreachs, icmp.icmpOutMsgs, icmp.icmpOutErrors, Icmp.icmpOutDestUnreachs	SVM
[84]	Characteristic-based	total number of packets, mean time interval, mean packet size, number of packets per second, number of bytes per second & total amount of bytes for each IP address	SVM
[85]	-	123 encoded features	Deep Learning
[88]	Genetic Algorithm	total_fpackets, max_biat, std_biat, mean_idle, burg_cnt	MLP
[108]	Chi-squared, Symmetrical Uncertainty	ip.srccountry, frame.len, ip.proto, syn flag, ack flag, rst flag, ip.ttl, frame.deltatime	DT: C4.5

TABLE 3.8: Summary of feature selection for ML-based DDoS detection systems cont.

Work	Feature Selection	Selected Features	ML Detection Model
[109]	-	-	Naive Bayes
[110]	Characteristic-based	-	SVM
[111]	Forward feature selection	Over 11 selected features	Hybrid supervised learning algorithm
[112]	-	Number of Packets, Average of Packet Size, Time Interval Variance, Packet Size Variance, Number of Bytes, Packet Rate, Bite Rate	Neural Network
[113]	Information Gain, Gain Ratio, Chi-squared, RelifF	13 selected features	DT: J48
[114]	Characteristic-based	-	Dendritic Cell Algorithm
[115]	Chi-squared, Information Gain	Ratioicmp, Land, Ratioudp, OnewayRatio, Ratiotcp, Protocoltype, AverageLengthIPFlow, RatioofInOut Packets	Naïve Bayes (NB), DT: C4.5, K-means, SVM, k-NN, Fuzzy c-means
[116]	Random Forest, ID3	Over 30 selected features	SVM
[117]	Consistency-based Subset Evaluation, Characteristic-based	Over 10 selected features	MLP

Table 3.9 summarises the characteristics of some feature selection techniques used in this field of research.

3.3 Machine Learning Algorithms

Machine Learning (ML) algorithms have been used to solve multiple real world problems over the years and this continues to be the case [78]. These algorithms are also being applied in the area of network monitoring and security to detect abnormal behaviours and attacks in networks [18]. As noted previously, the process of choosing a classifier for distinguishing legitimate from DDoS attack network traffic flows is a non-trivial task as it is not possible to know a priori which one is the best in solving the stated problem. Therefore, several supervised learning algorithms such as Multilayer Perceptron (MLP), Random Forest (RF),

TABLE 3.9: Summary of feature selection techniques and their characteristics

Feature Selection Technique	Numerical input Data	Categorical input Data	Numerical output	Categorical output	High Computation Requirement
High Correlation Filter	Y	N	Y	Y	N
Chi-squared	N	Y	N	Y	N
Forward feature selection	Y	N	Y	N	Y
Backward Feature Elimination	Y	N	Y	N	Y
Random Forest	Y	Y	Y	Y	Y
Low Variance Filter	Y	N	Y	Y	N

Decision-Tree (DT) and Support Vector Machine (SVM) are used as these are the most widely used ML algorithms for solving classification problems.

3.3.1 Types of Classifiers

There are hundreds of machine learning classifiers available for classification, however, different classification problems require different types of classifiers to produce the best classification results [118]. According to the survey paper, there are a total of 17 families of classifiers in machine learning: discriminant analysis, bayesian, neural networks, support vector machines, decision trees, rule-based classifiers, boosting, bagging, stacking, random forests and other ensembles, generalised linear models, nearest-neighbours, partial least squares and principal component regression, logistic and multinomial regression, multiple adaptive regression splines and other methods [118].

The paper evaluated the performance of 179 classifiers on 121 datasets and found that there is no single classifier that can be used to solve all classification problems [118]. It was discovered that the classifiers with the overall best performance across different datasets are Decision-Tree, Random Forest (ensemble classifiers built from decision trees), Support Vector Machines and Neural Networks. These classifiers are described as follows:

- **Decision-Tree Algorithm**

  It is an active learning algorithm that makes use of a predictive modelling technique based on a structure similar to a flow chart, with classification rules contained in the internal nodes and the class labels in the leaf nodes [119]–[121]. Once constructed, the DT uses the classification rules which were generated from the supplied features at the training stage to predict a test sample into a class label. These classification rules are

used as operating guidelines for distinguishing legitimate from DDoS attack network traffic flows. The root node is the attribute with the best conditions for splitting the samples into different internal or terminal (leaf) nodes using the characteristics of the samples as shown in Figure 3.2.

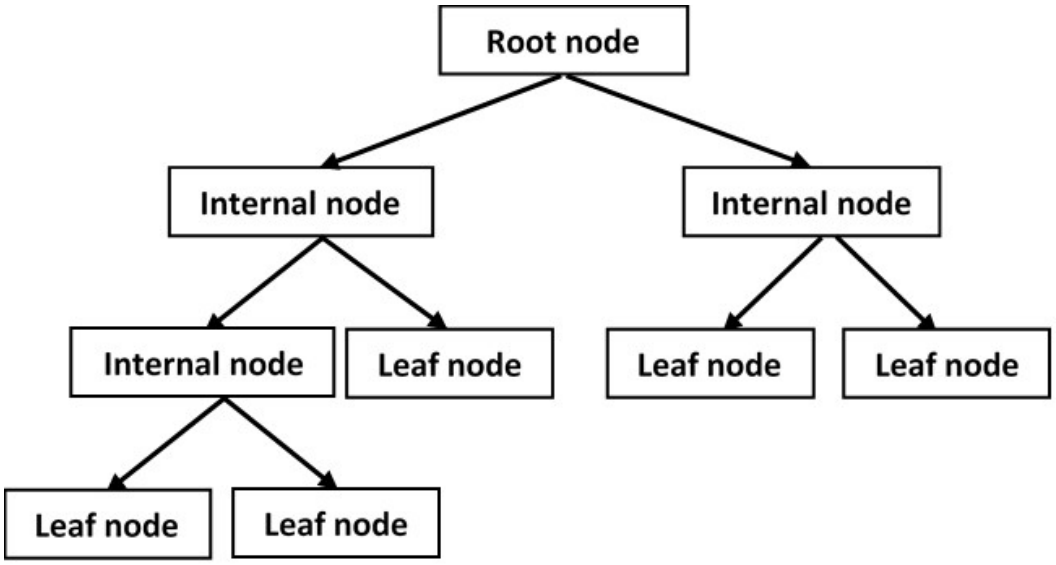

FIGURE 3.2: Decision-Tree structure.

The most frequently used DT classifiers are the Classification And Regression Tree (CART) which include the Iterative Dichotomiser 3 (ID3) and C4.5 algorithms [122]–[125]. The C4.5 algorithm is a revised version of the ID3 algorithm, which utilizes the information gain ratio for selection of the attributes with the best split. This algorithm possesses a series of advantages over the CART and ID3 algorithms. It effectively handles continuous attributes compared to CART, using the information gain ratio to handle the problem of overfitting which exists in ID3 and it can efficiently handle missing values in a dataset [104]. In this chapter, the DT model is constructed using C4.5 following the construction guidelines from [104], [125]–[127]. The processes involved in the development and implementation of the Decision-Tree algorithm can be found in Appendix B.

- **Random Forest (RF)**

  It is one of the classification trees algorithms and it consists of decision tree classifiers, termed a forest. To implement it, individual trees from the selected number of trees in the forest predict the expected output and a voting technique is used to select the output that obtains the highest number of votes [128]. Also, to overcome over-fitting, the average of the probabilistic predictions is used [128]. A structure of the Random Forest classifier with N number of trees is shown in Figure 3.3.

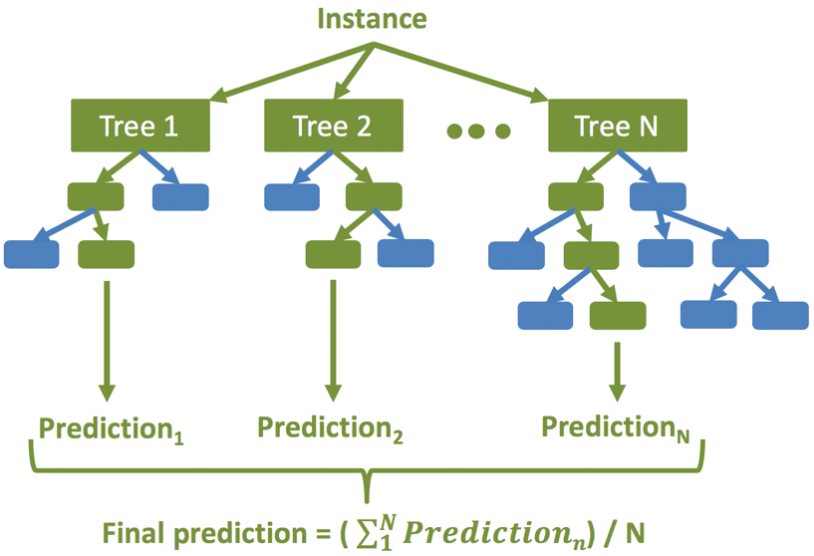

FIGURE 3.3: Random Forest structure with N number of trees.

- **Support Vector Machine (SVM)**

  A Support Vector Machine (SVM) is a type of supervised machine learning approach which is capable of performing pattern recognition and network traffic analysis through the mapping of input attribute vectors to a higher dimensional attribute space [127]. The SVM classifier transforms the initial dataset (normalised dataset) to a higher dimension using non-linear mapping. In order to properly separate the class labels, it searches the hyperplane and the closest data to the hyperplane are known as support vectors [127], [129]. There are three main kernel approaches in collaboration with the SVM classifier. These are the Radial kernel SVM (R-SVM), Polynomial kernel SVM (P-SVM) and Linear kernel SVM (L-SVM)- [130]. The SVM classifier supports the γ and C parameters, which are the deviation of the kernel and the penalty for misclassification respectively.

- **Multilayer Perceptron (MLP)**

  The Multilayer Perceptron is a type of deep artificial neural network-based classifier [131], [132]. It consists of more than a single perceptron. It composes of an arbitrary number of hidden layers between the input and output layers [91], [133]. The hidden layers act as the computational engines of the MLP. Each attribute acts as an input and the outputs are the class labels. The MLP may be linear if a single layer of nodes is used. However, it can be non-linear when several hidden layers with multiple layers of nodes are used in its application [133]. Figure 3.4 shows the structure of an MLP with a single hidden layer.

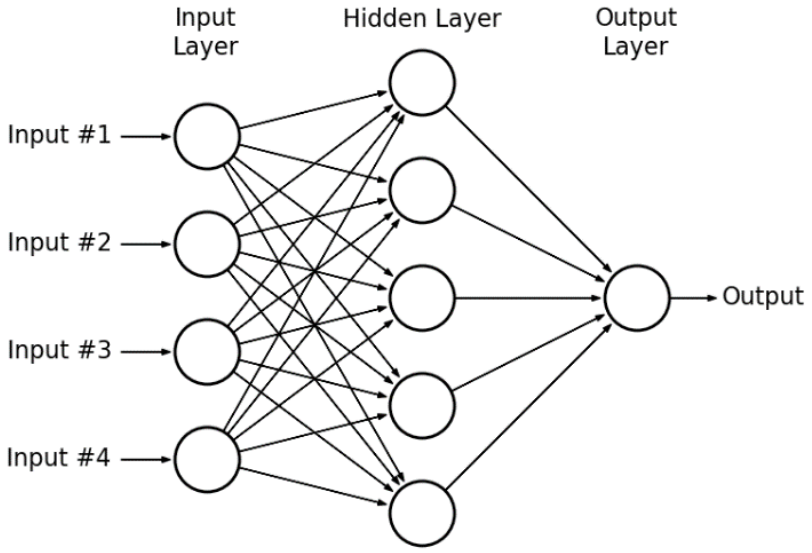

FIGURE 3.4: Multi-layer Perceptron structure.

The ML classifiers discussed above are implemented using the Scikit-learn Python module explained in [127] for classification of samples. The tuning parameters used by the selected ML classifiers for the classification of samples using the Scikit-learn Python module are highlighted in Table 3.10. The top 3, 5 and 7 features were selected due to their high variance values of over 0.025 which contributed more towards high accuracy in classification compared to the other features. Therefore, the results obtained based on the use of the Low Variance Filter are compared with the results from other feature selection techniques explored in this thesis using the threshold of 3, 5 and 7 features.

TABLE 3.10: Summary of the ML Classifiers and the parameters used in Scikit-learn

Classifier	Parameters	Value
Decision-Tree	criterion	gini, entropy
	max_features	3 , 5, 7
Random Forest	n_estimators	$1, 5, 10, 50, 100, 200, 500$
	max_features	3 , 5, 7
Support Vector Machine	C	$10^{-2}, 10^{-1}, \cdots, 10^{3}$
	γ	$10^{-9}, 10^{-8}, \cdots, 10^{3}$
	max_features	3 , 5, 7
Multi-layer Perceptron	max_iter	$10, 50, 100, 200, 500, 1000$
	activation_functions	relu, sigmoid
	max_features	3 , 5, 7

The maximum number of selected features ($max\_features$) used across the classifiers are three, five and seven based on the top selected features by the different feature selection techniques employed. The number of hidded layers explored for the Multi-layer Perceptron

classifier are between 1 and 4 layers as recommended in [134] to obtain optimum performance in classification accuracy and other performance measures used to evaluate the classifiers.

3.3.2 Classification and Performance Evaluation

Well-known open-source data mining tools such as the Waikato Environment for Knowledge Analysis (WEKA) and Scikit-Learn which is a Python module were used to implement the ML classifiers. These data mining tools consists of different algorithms and their functions include clustering, classification, regression, etc. Also, they provide several metrics i.e. True Positives (TP), True Negative (TN), False Positives (FP), False Negatives (FN), Accuracy, Sensitivity, Precision, F-measure, etc. which can be used to evaluate the performance of the classifiers [127], [134]. Below are the performance measures used in evaluating the performance of the classifiers:

- **Accuracy**: This is the proportion of correctly classified records over the total number of classification attempts.
- **Precision**: This is the ratio of the correctly classified records to the number of positive results predicted by the classifier.
- **Misclassification Error Rate**: This is the proportion of legitimate traffic flows that are classified as attack.
- **Sensitivity**: This is a measure of the proportion of normal flow records that are predicted as legitimate flows. It is also known as Recall.
- **Specificity**: This is the proportion of attack records which are predicted as attack.
- **F–Measure**: This is the harmonic mean of the sensitivity and precision. It shows how precise the classifier model is (how many instances it correctly classifies). Also, it shows how robust the classifier is (i.e. if it does not miss a significant number of instances).

The results of the performance measures discussed above range between 0 (worst outcome) and 1 (best outcome) based on the overall classification of all samples. These measures are calculated using the formulas highlighted in Table 3.11. In addition, these measures are computed for both legitimate and DDoS attacks samples in order to evaluate how well each label is classified.
However, before evaluating the performance of the classifiers, classification needs to take place. At the core of classification algorithms is the classification phase, where the network flow samples are classed as either legitimate or attack. Two stages are involved in the classification of network flow samples and these stages are training and testing.

In the training stage, the classifiers are trained to learn how to distinguish attack from legitimate network traffic flows. This is achieved with the use of a training set obtained from the selected datasets. The training set comprises of several network flow samples in

Table 3.11: Evaluation of Classification Measures.

Classification Measure	Equation
Accuracy	$ACC = \frac{TP+TN}{TP+TN+FP+FN}$
Precision	$P = \frac{TP}{TP+FP}$
Sensitivity	$SE = \frac{TP}{TP+FN}$
Specificity	$SP = \frac{TN}{TN+FP}$
F-Measure	$Fm = 2\frac{P \times R}{P+R}$

which the class labels are well-known. The classification performance of the classifiers can be improved by tuning specific parameters associated with them. These parameters are selected by estimating the performances of the classifiers which is achieved through the application of the cross-validation technique on the training set. In this chapter the stratified K-fold cross-validation technique is used with ten as the value of K.

The supplied sample data set is subdivided into 10 subsets with each part containing an equal number of network traffic flow records. A subset is chosen for testing while the rest are used as training sets. The process runs repeatedly for 10 times. The advantage of this approach is that the whole supplied sample data set are used as training and testing sets. Each subset from the whole sample is tested for one time [104] and this ensures that all the samples are evenly represented.

In the testing stage, the classifiers use the training received to classify new network flow samples present in the test set (a new collected data set) to the different classes. However, in order to effectively evaluate how accurate the classifiers perform in assigning the new samples to the correct class labels, the class labels of the samples in the test set are unknown to the classifiers.
The overall normalised dataset was subdivided in two parts: 75% of the samples represented the training set and 25% represented the test set.

3.4 Experimental Results

The experiments were performed using a system consisting of an Intel core i7 Central Processing Unit (CPU) with 3.6GHz Quad core processor, 500GB of memory and 16GB of Random Access Memory (RAM) running a Windows 10 Operating System. In this section, the results of the experiments conducted in this chapter are presented. This involves the performance evaluation of the selected ML classifiers using the different number of selected features from the feature selection techniques across the datasets described in Section 3.2.1.

3.4.1 Performance of the selected ML Classifiers

In this section, the performance of the ML classifiers are evaluated with the use of different numbers of selected features from different feature selection techniques.

- **Use of Chi-Squared Feature Selection Technique**

  The Select-K-Best with Chi-squared (SKB-Chi2) feature selection approach was used to select the top three, five and seven ranked features from the datasets. These features were used in training and testing the different ML classifiers. Figure 3.5 shows the classification accuracy of the ML classifiers using the different selected number of features.

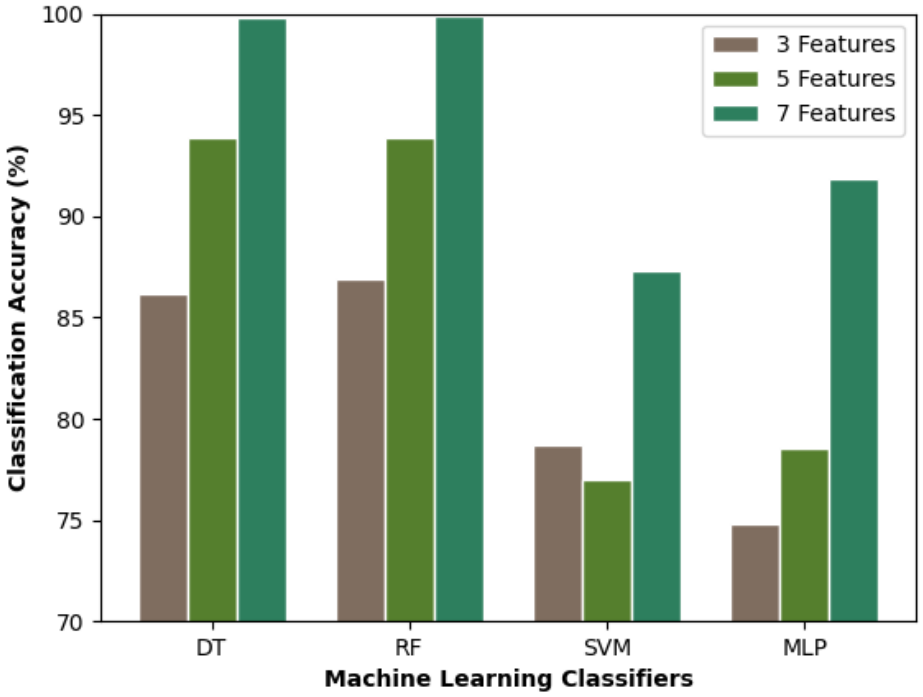

FIGURE 3.5: Classification accuracy of the ML classifiers with selected features using Chi-squared.

From the figure, it can be seen that the DT and RF classifiers performed the best with accuracies of over 86%, 93% and 99% with 3, 5 and 7 selected features respectively. In addition, the MLP classifier performed the least with just over 74% classification accuracy. Overall, it can be observed that there is an increase in classification accuracy with increase in the number of selected features across the classifiers apart from the SVM classifier which experienced a slight drop in accuracy when 5 features were used.

- **Use of BFE and FFS Feature Selection Techniques**

  The application of the Backward Feature Elimination (BFE) and Forward Feature Selection (FFS) feature selection techniques in selecting the top three, five and seven ranked features from the datasets yielded the same features and these were used in training and testing the different ML classifiers. Figure 3.6 shows the classification accuracy of the ML classifiers using the different number of selected features. From the figure, it can be seen that all the classifiers achieved similar classification accuracy of over 93% on average across the selected number of features.

  Overall, it can be observed that there is an increase in classification accuracy with increase in the number of selected features across all classifiers.

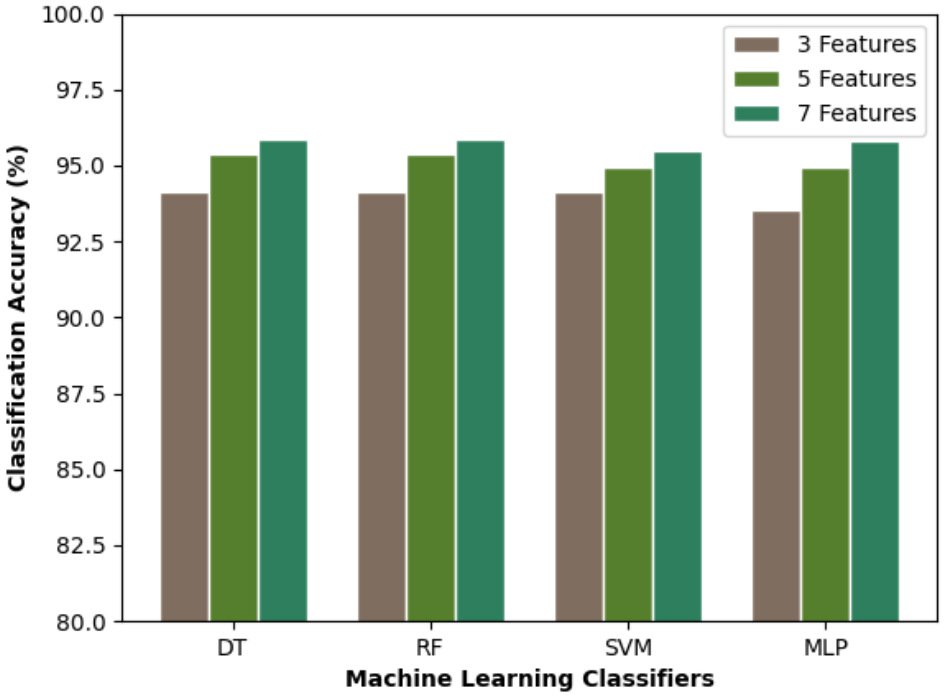

FIGURE 3.6: Classification accuracy of the ML classifiers with selected features using BFE and FFS.

- **Use of LVF Feature Selection Technique**

  The Low Variance Filter feature selection approach is the proposed feature selection technique which has been explained in Section 3.2.3. Figure 3.7 shows the classification accuracy of the ML classifiers using the different number of selected features. From the figure, it can be seen that the DT and RF classifiers achieved very high and similar classification accuracy of over 99% across the selected number of features. This displayed the highest classification accuracy obtained for the two classifiers when compared to results from the other feature selection approaches explored above. Overall,

it can be observed that there is an increase in classification accuracy with an increase in the number of selected features across all classifiers.

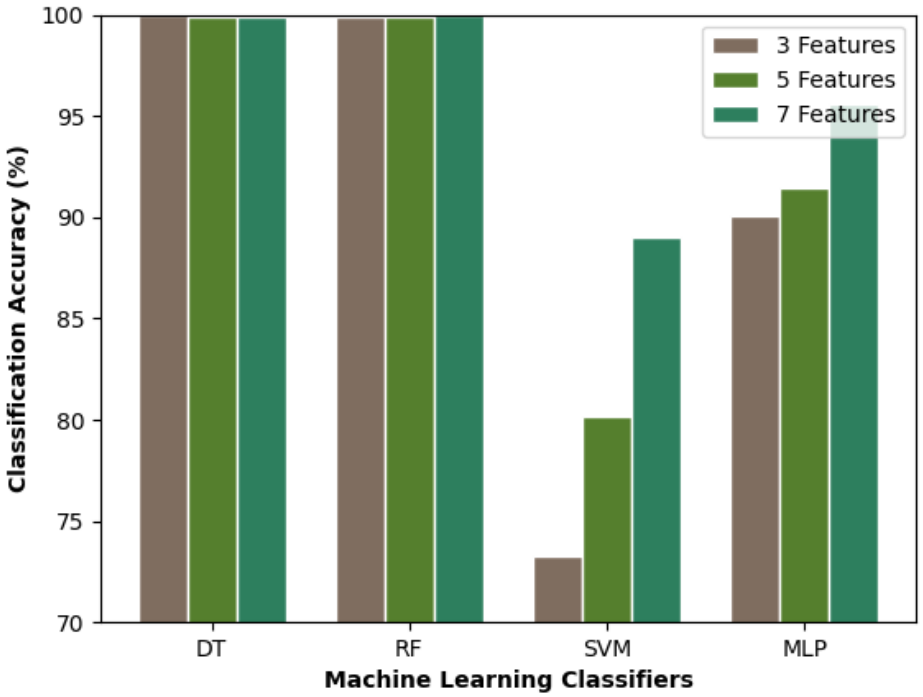

FIGURE 3.7: Classification accuracy of the ML classifiers with selected features using LVF.

Overall, the DT classifier outperformed the other classifiers as it achieved a significant 99.93% classification accuracy in the CAIDA 2007 and CICDDoS2019 DDoS attacks datasets respectively. This classifier was selected and explored further in the subsequent sections. For further detailed information regarding the performance of these classifiers (i.e. Accuracy, Precision, Sensitivity and F-measure), please refer to Appendix B.

3.4.2 Performance of the DT Classifiers

Figure 3.8 shows a partial DT classifier/model which was constructed using 3 selected features. As shown, the split at the root node (node 1) is based on the attribute (Avg Packets/s). Based on the structure of the DT model, the classification rules can be extracted. For example, if the Avg Packets/s at the root node is less than or equal to 60.5 and the magnitude of Fwd Bytes at node 2 is less than or equal to 112 then the network flow is classified as benign. Another classification rule is that if the Avg Packets/s at the root node is less than or equal to 60.5 and magnitude of Fwd Bytes at node 2 is greater than 112 then the network flow is classified as a DDoS attack. The use of dashed lines is to indicate that there are further internal nodes and splits between node 5 and the leaf nodes connected to it.

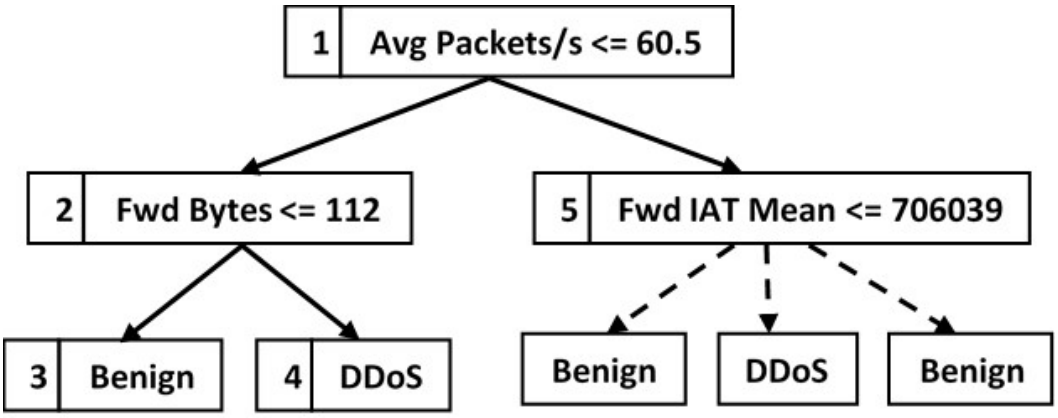

FIGURE 3.8: Partial Constructed DT model with 3 selected features.

3.4.3 Performance of the DT Classifiers on selected datasets

Different DT models are constructed using different numbers of features which were selected using the Low Variance Filter technique to investigate the optimal number of selected features for best efficiency and effectiveness. Table 3.12 shows the performances of DT models with 3, 5 and 7 features.

TABLE 3.12: Performance of DT Models with Different Number of Selected Features using LVF

Datasets	**DT model (based on number of features)**	**Modelling time (seconds)**	**Classification accuracy (%)**
CAIDA 2007	DT3	0.20	99.93
	DT5	0.26	99.93
	DT7	0.31	99.93
CIC 2017	DT3	0.16	99.69
	DT5	0.24	99.61
	DT7	0.34	98.98
CIC 2019	DT3	0.09	99.93
	DT5	0.20	99.92
	DT7	0.23	99.88

From Table 3.12, it can be seen that the DT constructed with the top 3 selected features based on the Low Variance filter technique produced high classification accuracy across all three datasets when compared with the DT models using 5 and 7 selected features. The effectiveness of using the DT model with the top 3 selected features based on the application of the Low Variance filter technique was evaluated using the performance measures. As shown in Figure 3.9, the DT model achieved high results with respect to sensitivity and specificity. However, these classification accuracy results are based on Decision-Tree classifiers with a

depth of 3. An increase in the number of selected features led to an increase in classification accuracy when the tree depth is increased from 3 to 4 from the experiments carried out however, it is worth noting that increase in the tree depth will lead to an increase in complexity and additional computational requirement.

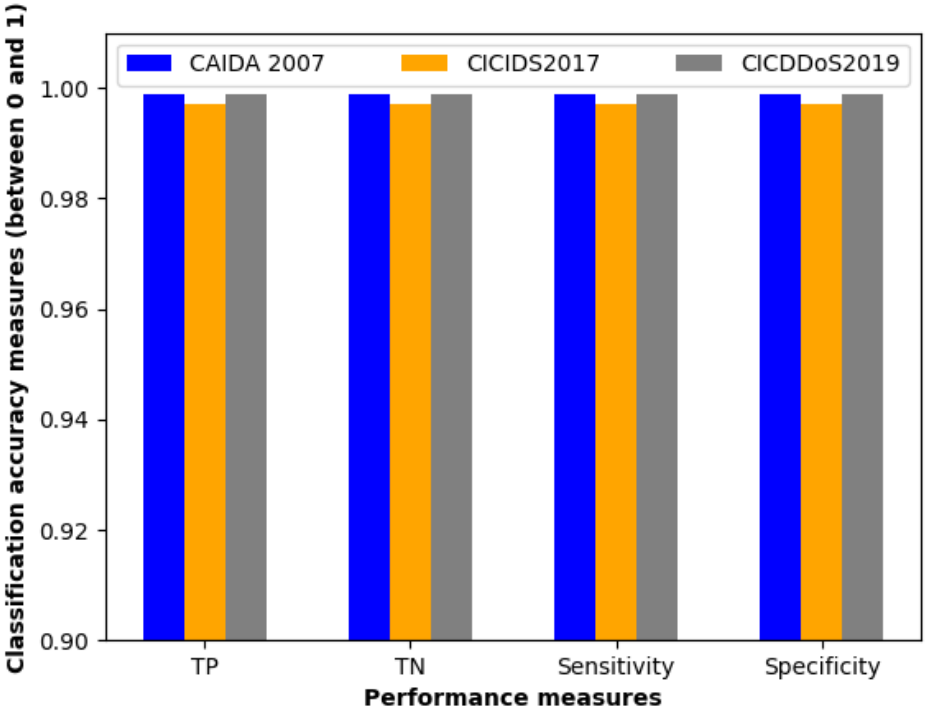

FIGURE 3.9: Effectiveness of the DT model with 3 selected features across the 3 datasets.

This shows that the model achieved high accuracy in predicting legitimate and attack network flow samples from the number of predicted samples. It achieved high TP results, showing it is effective in predicting benign samples that belonged to the benign class label. Also, it achieved high TN results showing that it can predict most of the attack samples from the total number of attack predictions made.

Furthermore, the model achieved very low false alarms. The FP and FN rates achieved is 0.1% in both the CAIDA 2007 and CIC 2019 datasets showing that misclassifications were very low as displayed in Figure 3.10. The highest false positive and false negative rates of 0.3% occurred on the CICIDS2017 dataset. However, the FP and FN rates can vary depending on the number of samples that are incorrectly classified with respect to the total number of samples used in the experiment. The values obtained are based on the experiments conducted in this chapter.

Overall, this shows that the proposed lightweight DT model with just 3 selected features is effective and efficient in distinguishing legitimate from DDoS attack network traffic flows when deployed in a real network environment.

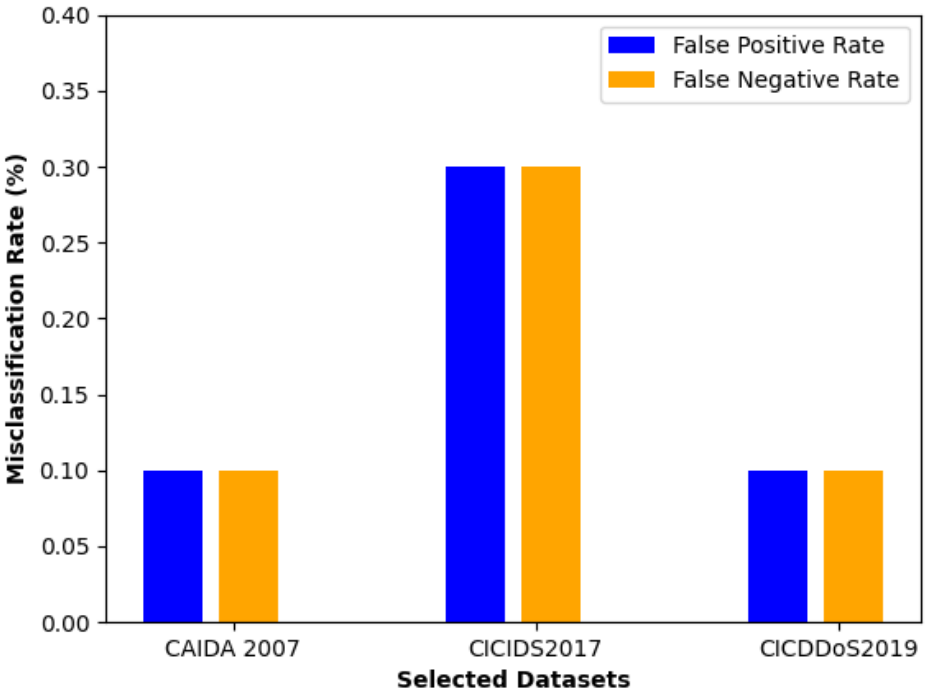

FIGURE 3.10: False positive and False negative of the DT model.

3.4.4 Real-time network monitoring using DT3 model

The DT model was deployed at different monitoring points of the Advanced Network Research Group (ANRG) private network at the University of Liverpool to tests its effectiveness and reliability in a real-time environment. The different points of the network where the models were deployed are indicated with the monitoring nodes 1-3 shown in Figure 3.11.

The attacker node shown in the Figure 3.11 was used to generate the attack traffic targeting server 1 and workstations 1 and 3. The attack and normal network traffic (around 500 network flows per second of which 400 were attack and the rest were normal) in the CAIDA 2007, CIC 2017 and 2019 DDoS datasets were replayed using the attacker node directed at the different target nodes in the network. As shown in Figure 3.12, the red and the green small rectangles in the real-time network monitoring application represent attack and legitimate network packets respectively moving through the network link.

The port mirroring technique is applied whereby all packets going into each network switch are mirrored to the monitoring node attached to each switch. Once packets are captured in the monitoring nodes, relevant features are extracted from each network flow and fed to the DT models for classification. Results from preliminary tests show that the DT model with three selected features was effective in distinguishing normal from attack network traffic flows.

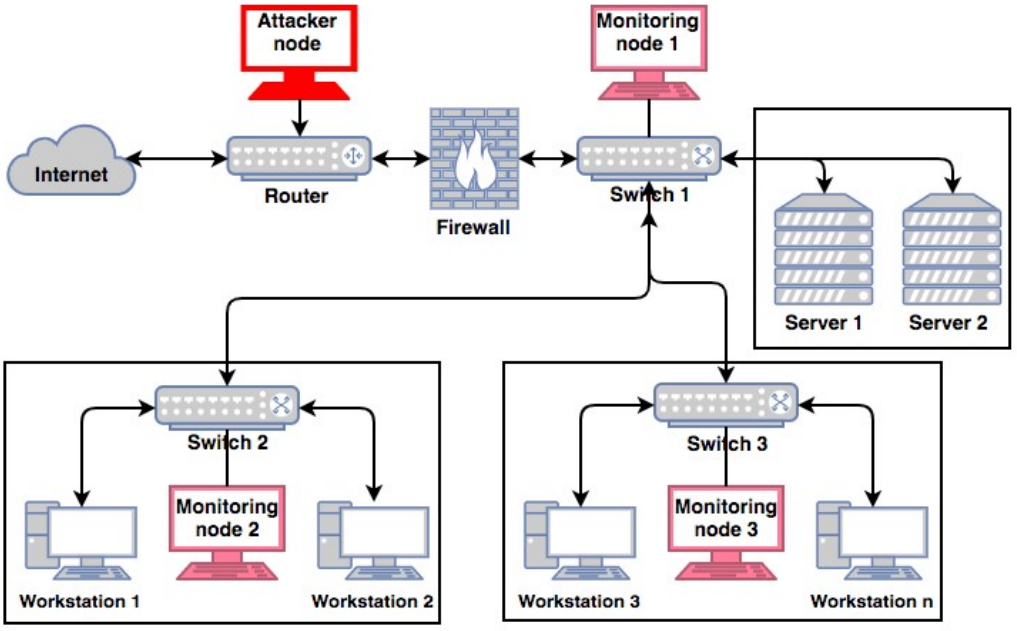

FIGURE 3.11: Real-time deployment of DT model in the ANRG network.

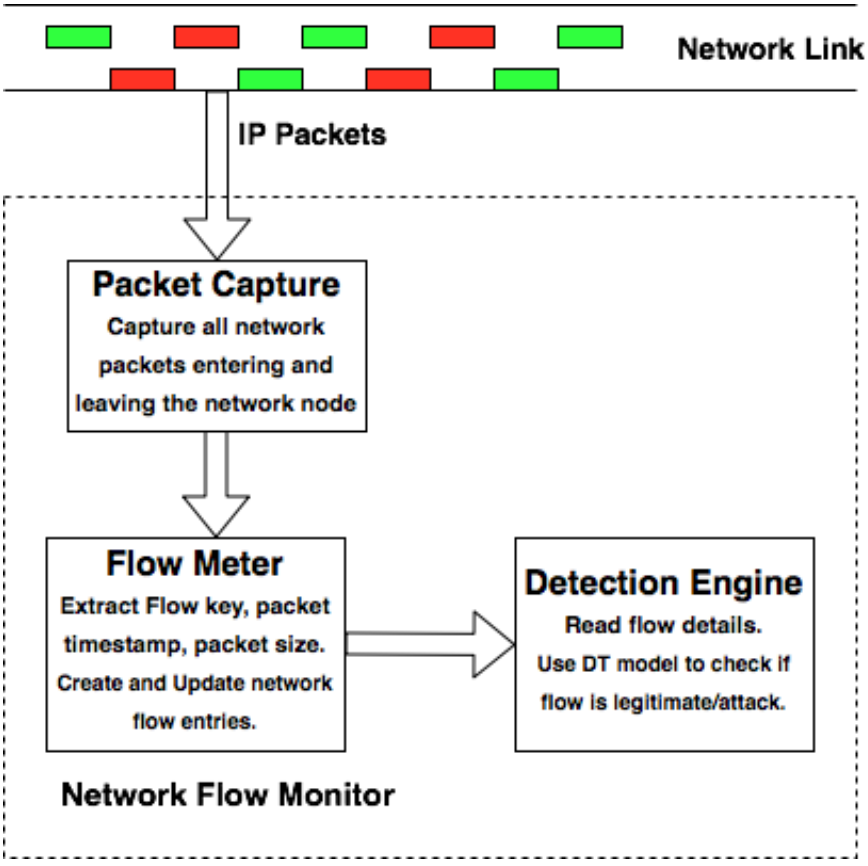

FIGURE 3.12: Real-time network monitoring application

3.4.5 Real-time Monitoring Implementation System Resource Utilization

The resource utilization (CPU processing power) was examined for the DT3 model under the different network traffic conditions highlighted in Table 3.13. These network traffic scenarios were modelled based on the network traffic scenario in the CAIDA 2007 dataset whereby the ratio of attack to legitimate network traffic flows was approximately 5:1. However, in other real attack scenarios, this ratio could be larger in favour of attack traffic flows.

TABLE 3.13: Different network traffic conditions

Traffic conditions	Number of legitimate network traffic flows	Number of attack network traffic flows
Scenario 1	10	0
Scenario 2	10	50
Scenario 3	20	100
Scenario 4	100	Over 1000

It was discovered that an increase in the network traffic directed to server 1 from the attacker node leads to an increase in the CPU resource utilization of the monitoring node 1 shown in Figure 3.13. In scenario 1, ten legitimate with no attack network traffic flows were directed to server 1 and around 2% of a core's processing power was used to analyse the network flows and produce results based on the DT3 model. Increasing the network traffic flows to sixty comprising of fifty attack and ten legitimate from scenario 2 led to an increase in the core's processing power requirement to around 10% as shown in Figure 3.13.

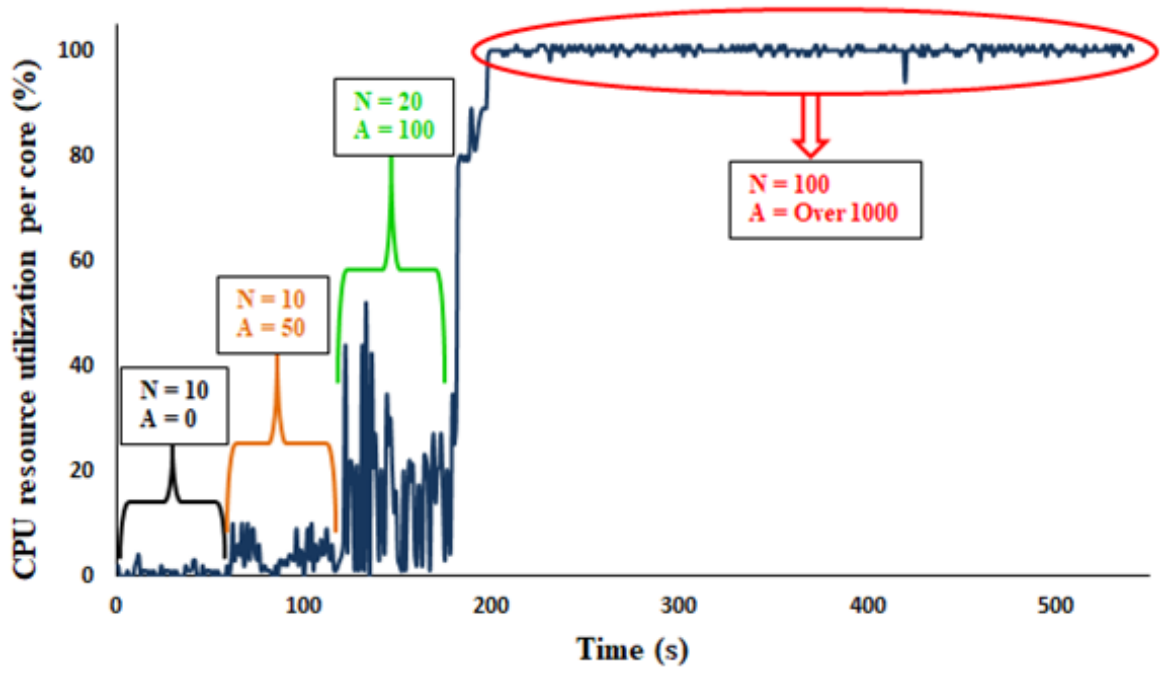

FIGURE 3.13: Process CPU resource utilization

In addition, increasing the traffic to scenario 3 led to a further increase in the core's processing power requirement. Finally, introducing over a thousand attack traffic flows directed to server 1 led to the monitoring system utilizing close to 100% of the core's processing power. This shows that increase in network traffic directly leads to an increase in the CPU's processing power requirement as shown in Figure 3.13.

3.4.6 Comparison with related works

The effectiveness of the proposed DT3 model is compared with other related works in terms of feature selection techniques used, number of selected features and the overall classification accuracy of the models. Based on the related works and the different approaches taken by other researchers, it can be seen in Table 3.14 and Table 3.15 that although the proposed model uses the least number of features in this field, results show that it is effective in distinguishing legitimate from attack samples when compared to related works. It competes with other models in classification accuracy whilst validating its effectiveness with multiple up-to-date DDoS attacks across multiple datasets.

Table 3.14: Comparison of approach with related work

Work	DDoS Attacks	Feature Selection Technique	Number of Selected Features	Dataset	ML Detection Model	Accuracy (%)
[83]	TCP SYN, UDP and ICMP	SNMP MIB	13	-	SVM	99.53
[84]	TCP SYN, and ICMP	Characteristic-based	6	CAIDA	SVM	98.7
[85]	-	-	123 encoded features	NSL-KDD	Deep Learning	99.27
[88]	TCP SYN, and ICMP	Genetic Algorithm	5	CAIDA	MLP	99.98
[108]	TCP SYN and ICMP	Chi-squared, Symmetrical Uncertainty	8	CAIDA	DT: C4.5	95
[109]	HTTP flooding	-	-	Generated dataset	Naive Bayes	95
[110]	TCP SYN, UDP and ICMP	Characteristic-based	-	Generated dataset	SVM	98.99
[111]	-	Forward feature selection	Over 11 selected features	NSL-KDD	Hybrid supervised learning algorithm	98.9
[112]	-	-	7	DARPA LLDOS 2000	Neural Network	94

TABLE 3.15: Comparison of approach with related work cont.

Work	DDoS Attacks	Feature Selection Technique	Number of Selected Features	Dataset	ML Detection Model	Accuracy (%)
[113]	-	Info. Gain, Gain Ratio, Chi-squared, ReliefF	13	NSL-KDD	DT: J48	99.67
[114]	TCP SYN	Characteristic-based	-	Generated dataset	Dendritic Cell Algorithm	-
[115]	TCP SYN and ICMP	Chi-squared and Information Gain	8	CAIDA	Naïve Bayes (NB), DT: C4.5, K-means, SVM, k-NN, Fuzzy c-means	98.7
[116]	-	Random Forest, ID3	Over 30 selected features	KDD99	SVM	82.99
[117]	-	Consistency-based Subset Evaluation, Characteristic-based	Over 10 selected features	NSL-KDD	MLP	91.7
DT3	TCP SYN, UDP and ICMP	Low Variance Filter	3	CAIDA 2007, CIC 2017 and CIC2019	DT	Over 99.69 across all datasets

3.5 Summary

The detection of DDoS attacks is a very important aspect in the area of network security and traditional monitoring and detection systems are either very complex which may lead to a higher computational cost in operation, or are less effective in attack detection. In this chapter, a lightweight DT3 model based on the C4.5 algorithm has been proposed using a robust and effective feature selection technique. The DT3 model was constructed based on the number of selected features using the Low Variance filter technique with a variance threshold of 0.025 (from normalised datasets using the min-max normalisation technique) obtained after a series of tests starting from conservative threshold values. Analysis of the design shows that it can maintain high detection accuracy using only 3 features. Also, it consumes minimal processing power of the detection system's CPU resources whilst processing less

than a thousand network traffic flows per second per core of the system used. However, high processing power is required for processing over a thousand attack traffic flows per second.

In the next chapter, the DT3 classification model is implemented in a Field Programmable Gate Array (FPGA) device for classification as it offers fast processing power compared to traditional CPUs. This property of the FPGA will help reduce the workload on the CPU of the system. Finally, the applicabilition of the proposed solution to network monitoring for detecting DDoS attacks will be discussed.

Chapter 4

Hardware Accelerated Application for DDoS Flooding Attacks Detection

The use of different Machine Learning models for DDoS attack detection were presented in Chapter 3. From the selected ML models, the Decision-Tree (DT) model called DT3 was found to outperform the other models and was implemented as part of a software-based network monitoring system for detecting DDoS attacks. In addition, the performance of the monitoring system was evaluated under different network traffic conditions with respect to the system's resource utilization.

In this chapter, an insight is provided into different parallel processing platforms that can be used to obtain optimized real-time performance in embedded systems when implementing classification algorithms. The limitations of the network monitoring system for distinguishing legitimate from DDoS attacks network flows presented in Chapter 3 are presented and a hybrid network monitoring system using both FPGA and multi-core processor having multiple Central Processing Units (CPUs) is developed.

Some of the main characteristics of FPGAs include high performance, reconfigurability and low power consumption, however, FPGA logic resources will be greatly wasted with increase in program development cost if it is used to completely replace a traditional CPU [135]. Therefore, a more practical and beneficial approach is the combined use of CPU and FPGA where instructions that require fast execution are handled by the FPGA and the other instructions are performed in the CPU [135].

The hybrid network monitoring system developed leverages the parallel processing capabilities of the FPGA for fast network flow classification, whilst using a software application in the CPU for network flow pre-processing required for classification.

The effectiveness of the system is evaluated using the three network datasets mentioned in Chapter 3 which are the CAIDA 2007, CICIDS2017 and CICDDoS2019 and also a real-time environment with varying network traffic. It is shown to be capable of distinguishing attacks from normal network traffic flows with a detection accuracy of over 98% when deployed in

the real-time environment under different network traffic conditions with detection within 1μs, which is over thirty times faster than the software implementation of the architecture when larger sample sizes are used. The hybrid system presented is optimised for flexible deployment in low-cost platforms (e.g. in line cards in switches and routers) for the efficient, rapid detection and prevention of DDoS attacks.

4.1 Parallel Processing Platforms

Below are different parallel processing platforms that can be used to obtain optimized real-time performance in embedded systems.

4.1.1 Graphical Processing Unit (GPU)

Graphics Processing Units (GPUs) have been used in systems ranging from embedded devices to supercomputers for High Performance Computing (HPC) applications. The rapid growth of GPUs resulted in the inception of the concept of general-purpose processing on Graphical Processing Unit (GPGPU) [136].

A GPU consists of an expanded Arithmetic Logic Units (ALUs) silicon area. The latency of GPUs are hidden using a thread-level parallelism with the capabilities of executing hundreds of threads at once per GPU. These threads are executed in batches and each batch comprises of 32 threads called warp. Different parts of a program can be executed through the flexibility of enabling and disabling threads independently within a warp. The number of active threads within a warp determine the number of executed instructions per cycle. However, the use of batching induces an important cost in the aspect of minimizing thread divergence as all threads execute loops the same number of times and take up the same conditional statements [137]. The use of multiple GPUs present a number of advantages as highlighted in [138]:

- Memory can be accessed in parallel as each graphic card consists of its own RAM leading an overall increase in memory bandwidth
- There is no contention of GPUs for memory access to a shared memory

GPU cores are regarded as native hardware floating-point processors and a single core is capable of executing hundreds of floating-point math's operations per clock cycle, which makes it a good fit for intensive signal and image processing applications [139].

Furthermore, GPUs provide backward compatibility so software with modified algorithms can still be executed on older chips [139]. However, a major disadvantage of GPUs is their power consumption as they are regarded as "power hogs" compared to hardware architectures based on FPGAs [136]. In addition, algorithms are executed in software by GPUs so the fetch, decode and execute cycle which involves movement of data to and from memory, and so it will lead to additional latency [136].

4.1.2 Application-Specific Integrated Circuit (ASIC)

An Application-Specific Integrated Circuit (ASIC) is a customized microchip designed for a specific application instead of general-purpose use. Both analogue and logic functions can be incorporated in mixed signal ASIC designs, which are particularly important in building System-on-Chip (SoC) devices. ASIC chips are designed following some conceptual stages in [140]. These stages involved are highlighted below although some stages could overlap:

- Design idea with specifications
- Production of structural and functional description which involves the decision on the best architecture for the design
- Implementation of subsystems through schematics, finite state machine, logic representation and combinatorial and sequential logic.
- Use of synthesis tool i.e. Design Compiler (Synopsys) and Blast Create (Magma) for Logic/RTL synthesis. These tools make use of standard cell libraries and RTL hardware description to generate a Gate-Level Netlist
- Physical implementation of the design, which involves the conversion of the Gate-Level Netlist to geometric representation.
- Generation of the GDSII file used to fabricate the chip by the foundry

With ASIC technology, product cost can be reduced however; the prices of ASIC solutions are not efficient when low volume production is involved. This is why ASIC designs are dedicated for special application requirements such as high-performance. Furthermore, ASIC design involves a more complex design cycle when compared to the design flow of FPGAs and requires more time to market. Finally, the hardware in ASICs is generally not reprogrammable, and the design flow requires a time consuming planning, place and route and time analysis when compared to FPGAs [136], [140].

4.1.3 Multi-core Processors

Multi-core processors are processors that consist of multiple assembled processing units on the same die for efficient processing of multiple tasks simultaneously, enhanced performance and reduced power consumption [136]. The main difference between multi-core processors and a single core processor is that the multi-core processors are capable of executing multiple instructions at any given time as opposed to the single core processor, which is can execute a single instruction stream at a time [136].

However, the operation of the multi-core CPUs is the same as the operation of a single processor in that the cores communicate via the shared memory and a cache memory is used for performing the synchronization during communication as shown in Figure 4.1. In addition, each core executes a thread and each thread has a dedicated ALU consisting of a number of functional units. Also, the status of the thread is stored in a set of registers

[137]. Furthermore, a large unit handles the scheduling of tasks such as instruction ordering, speculative execution and branch prediction.

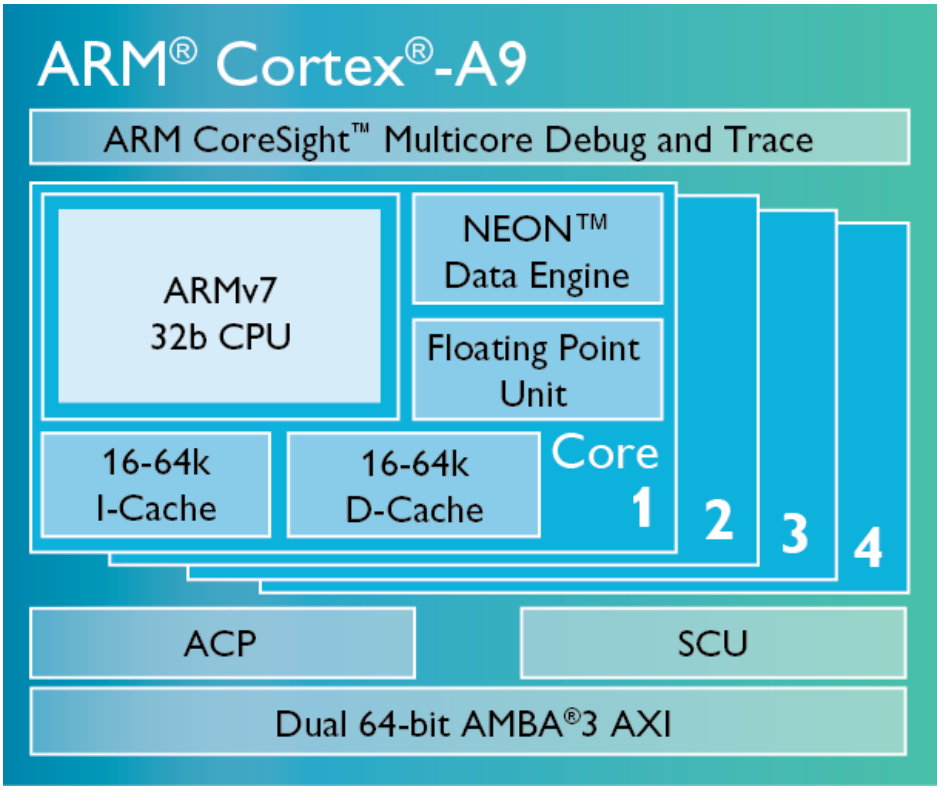

FIGURE 4.1: ARM Cortex-A9 Architecture [141]

Wide ranges of applications in embedded systems widely make use of multi-core processors. The performance improvement derived from using multi-core processors is dependent on the software algorithm to be implemented and its implementation. Performance improvement are obtained in areas of the software that can be simultaneously executed in parallel by multiple cores and the best achievable performance is a speedup factor of the number of available cores [136].

The ARM Cortex-A9 is a popular multi-core processor used for implementing low-power and cost-sensitive applications. It has proven to offer effective solutions in embedded systems and is flexible for implementing multi-core designs that enhance overall system performance. It offers a wide range of advantages as highlighted in [136], [141] including:

- Power efficiency and scalable performance across a wide range of applications.
- It is based on ARMv7-A architecture which benefits from:
    1. Dynamic length pipeline of 8-11 stages
    2. Highly configurable Level-1 (L1) caches
    3. Opportunity for NEON technology implementation

4. A scalable multi-core configuration with up to four coherent cores

- Support for 32-bit software ecosystem

4.1.4 Field Programmable Gate Array (FPGA)

A Field programmable gate array (FPGA) device is an Integrated Circuit (IC) that consists of configurable logic blocks connected through configurable interconnects as shown in Figure 4.2. FPGAs are generally used for solving computational problems. The most widely known FPGA vendors include Altera (now part of Intel), Xilinx, Microchip Technology, Lattice Semiconductor, QuickLogic and Achronix. The major advantage of using FPGAs is that they can be significantly faster in specific applications compared to traditional multi-core processors due to parallel processing nature as multiple instructions can be executed in parallel.

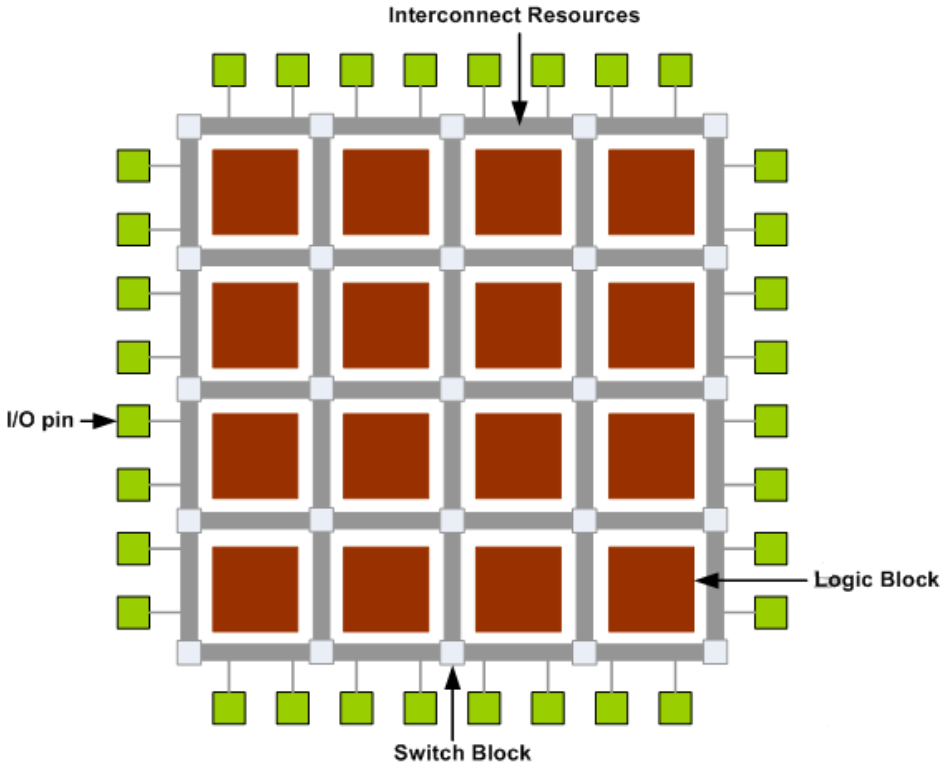

FIGURE 4.2: FPGA architecture

The configurable logic blocks are also known as Logic Cells (LCs) and each early LC comprises of a Look-Up Table (LUT), Digital Flip-Flop (DFF) and a Multiplexer [142]. LUTs are used in the design of combinatorial functions, as this is its main role. However, some FPGA vendors employ the use of LUT as block RAM also referred to as distributed RAM. Additionally, additional functions such as memory blocks are embedded in most FPGAs.

Nowadays, FPGAs contain large distributed chunks of embedded block RAM, as applications possess wide range memory requirements for effective and efficient performance.

These distributed memory blocks are located in different parts of the chip i.e. they may be located across the face of the chip or at the periphery or even organised in columns. In addition, multiple blocks of memory can be combined to form a larger memory block and each individual block can be used independently.

Hardware Description Languages (HDLs) are primarily used for FPGA configuration. The hardware application goes through the process of synthesis which involves the translation of description hardware application to a set of connected logic elements which may include LUT, Multiplexer and DFF and are placed on the FPGA structure and then routed using the FPGA interconnect matrix. The final output of this process is the generated bitstream that can be uploaded into the FPGA configuration memory. FPGA designs comprise of a set of design properties ranging from pipelining, latency, cost and power consumption, which are discussed below.

FPGA Designs Properties

There are some design characteristics associated with the implementation of a circuit on an FPGA. These characteristics are power consumption, latency, area and frequency however; designs in FPGA are compared to other designs in terms of characteristics such as pipelining, latency, power consumption, cost and size. These characteristics are explained below:

- **Pipelining**

  Pipelining is a technique in which multiple processes overlap during execution. A pipeline has the structure of a pipe as data enters from one end and exits on another end [143]. A pipeline consists of stages that are connected sequentially; the input of the current stage is the output from the previous stage when implemented in signal processing. For a hardware design to achieve maximum speed, pipelined code should be written. In addition, larger hardware applications require pipelining of optimized blocks that are considered bottlenecks to achieve the desired performance.

- **Latency**

  Latency is the time it takes for an output to be generated when an input data sample is provided. In electronic devices, latency is a measure of the clock cycles taken for a data sample to navigate its way through a system. In FPGA applications, it is a product of the number of pipelines and the cycle time. This makes FPGAs a popular choice for the design and implementation of applications that require reduced latency [144]. In traditional software applications, the latency ranges between microseconds and many seconds whilst in the hardware fabric, it is in the range of hundreds of nanoseconds [31].

- **Power consumption**

  Power consumption has become a very important variable and has always been a consideration in hardware designs. In terms of reducing power consumptions in hardware designs, FPGAs are a very popular choice as they aid in reducing power consumption

associated challenges. There are four power components associated with an FPGA design as stated in [145]. These power components are pre-programmed quiescent, in-rush programming current, post-programmed static and dynamic power consumption [136]. Software tools have been provided by FPGA vendors that are used by hardware designers for estimating power consumptions in designs.

- **Size and Cost**

  Currently, the overall size of the die and package costs have been driven down by FPGAs to implement convenient custom designs. The generic nature of FPGAs coupled with the advantages of short time-to-market and reduced material costs lead to cost effective designs [146].

4.2 Contributions

In this chapter, a hybrid network monitoring system for DDoS attacks detection for both offline and real-time deployment is proposed using the DT3 model presented in [147]. The major contributions of this chapter are mentioned below.

1. Selection of just three network traffic flow features for DDoS detection without compromising detection accuracy.
2. A hybrid network monitoring system implementation for fast DDoS attack detection. The hardware implementation is based on a Field Programmable Gate Arrays (FPGA) device and requires just microsecond to classify a network traffic flow sample as attack or normal.
3. Evaluation of the proposed hybrid monitoring system and DDoS attacks detection method in terms of detection time and accuracy in offline mode across three benchmark network datasets.
4. Evaluation of the proposed hybrid monitoring system and DDoS attacks detection method in terms of detection time and accuracy in a real-time deployment.

4.3 Proposed Approach

In the field of network monitoring and security, the most challenging difficulty faced is the running of network monitoring applications in real-time [148]. This is because running software applications in real-time using general purpose computers is not always possible as the resources contained in the computers such as memory, CPU processing power and peripheral devices are limited leading to fast exhaustion [149].

Network monitoring involves dozens of operations that are performed on each packet received in a network as information about the packets are extracted and used for flow classification. In addition, further operations are required for network flows to be classified as

either legitimate or attack. General purpose processors perform these operations sequentially which impacts negatively on both resource consumption and performance [150].

However, the FPGA devices possess the capability of executing operations in parallel, which leads to multiple operations running simultaneously. The proposed approach in this chapter presents the use of FPGA for fast classification of network flows. It features a hardware implementation of the DT3 model combined with a software application running in a general-purpose CPU for packet capturing and pre-processing. As shown in Figure 4.3, the pre-processing phase involves the capturing of network packets through a network link, extraction of packet information, classification of packets to flows and temporary storage in cache.

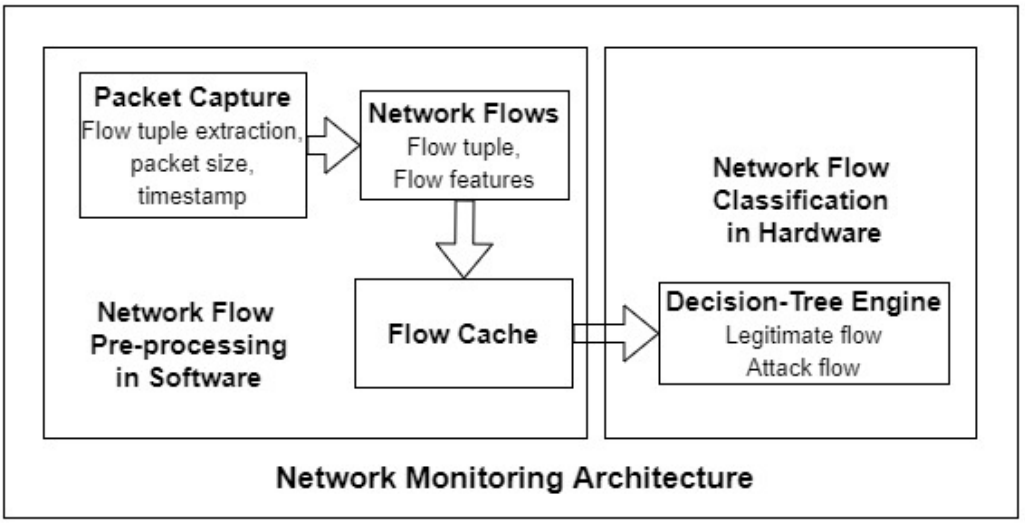

FIGURE 4.3: Proposed Network Monitoring Architecture for DDoS attacks detection

Furthermore, the network flow classification phase involves the classification of all network flows available in the cache into either legitimate or attack. The proposed architecture is implemented in a System-on-Chip (SoC) device that consists of both a general-purpose CPU and Programmable Logic (PL) also known as FPGA. The following subsections highlight the steps required to efficiently implement and evaluate the performance of the proposed system for effective distinguishing of legitimate from DDoS attacks network flows as shown in Figure 4.3.

4.3.1 Proposed Device

The device used to carry out this experiment is the ZedBoard Zynq-7000 FPGA development kit [151]. The Zynq-7000 ZedBoard is different from traditional FPGAs as it contains both the 7 series FPGA provided by Xilinx and a dual core ARM Cortex-A9 processor on the same chip.

Figure 4.4 shows the internal architecture of Zynq-7000 ZedBoard. The ZedBoard consists of two main parts which are the dual core ARM Cortex-A9 Processing System (PS) and 85,000 series-7 Programmable Logic (PL).

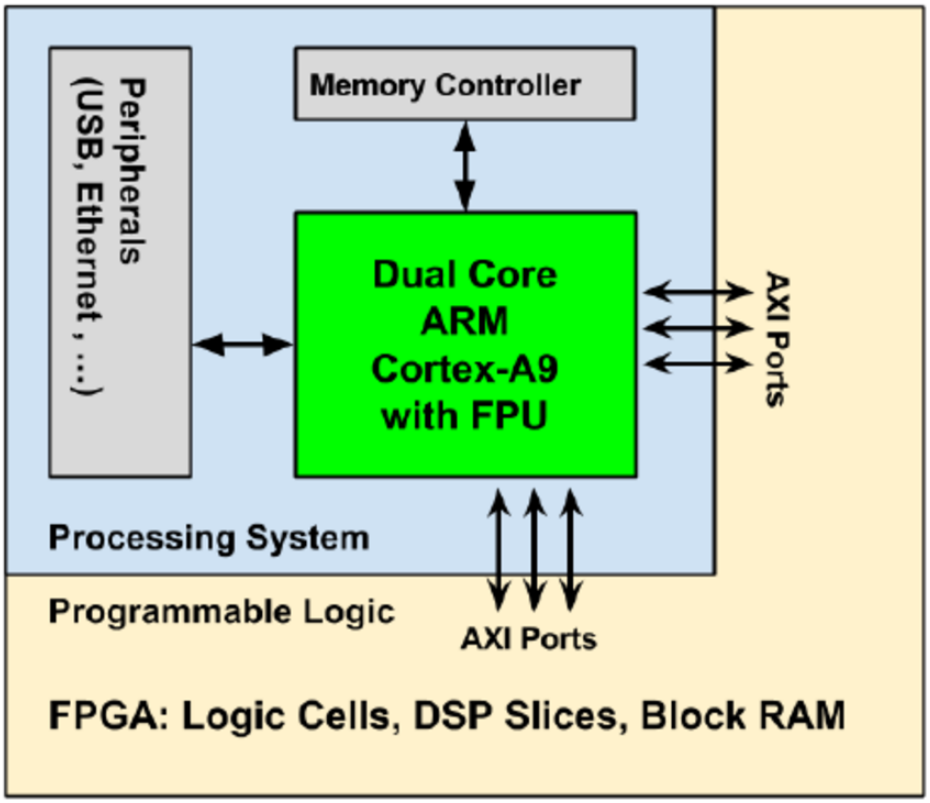

FIGURE 4.4: Internal architecture of the ZedBoard

The Processing System functions like a traditional processor as it consists of components such as Memory Controller, USB controller, Gigabit Ethernet Controller, and the ARM Cortex-A9 whereas, the Programmable Logic consists of all the components present in standard FPGAs [151]. In addition, the communication between the two components of the ZedBoard is provided by high performance data-paths as shown in Figure 4.4.

In a FPGA card having this system, those parts containing intense computations are performed on FPGA and the control parts containing very little or no computations can be performed on the processor by using software applications.

4.3.2 Host Network Flow Pre-processing Application

The host is running an Ubuntu Operating System implemented in the Dual core ARM Cortex-A9 PS of the ZedBoard. The network flows pre-processing application is developed and implemented in the Ubuntu OS. The application captures all network packets through the link connected to a switch. Only relevant information like packet size, source and destination Internet Protocol (IP) addresses, timestamp of packet arrival, etc. are extracted from each packet arriving in the specified network link to be monitored as shown in Figure 4.5.

Some of the extracted information are used to create a Flow Key (tuple) for identifying individual network flows. The tuple information along with other relevant network flow features are stored and constantly updated in cache as highlighted in Figure 4.3. The stored flows are then sent to the classification model in the Programmable Logic (PL) for effective and efficient classification as either legitimate or DDoS attack.

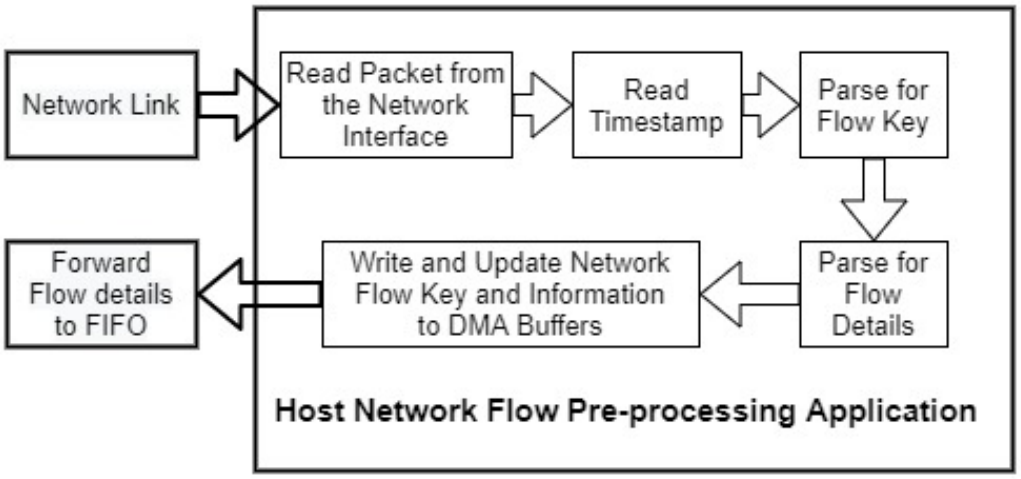

FIGURE 4.5: Processes in Host Network Monitoring Application

4.3.3 Hardware Design and Implementation

The hardware design and implementation phase consist of several stages. These stages range from the design and implementation of the DT3 model to the implementation of a communication link between the PS and PL of the ZedBoard for effective and efficient transmission of data between them. Figure 4.6 shows the hardware design flow for the obtaining the required bitstream file for the overall design.

4.3.4 DT3 Design and Implementation in FPGA

In this chapter, the IP core of the proposed DT3 model is designed using the Vivado High-Level Synthesis (HLS) tool [152]. The HLS tool is used to transform an application written in C, C++ and SystemC language into a RTL implementation that can be used as part of an overall system hardware design.

In addition, it offers the pipelining of functions through a GUI interface. The input and output ports to the DT3 block are specified in Figure 4.7 using the AXI interface available in HLS for data transfer in and out of the block to other components of the design. The third port shown in the figure is for control signals involve with indicating when the block is available to receive data and when it is busy with data processing. The data inputs a, b and c represent the network flow features that are required for effectively distinguishing legitimate form DDoS attack flows.

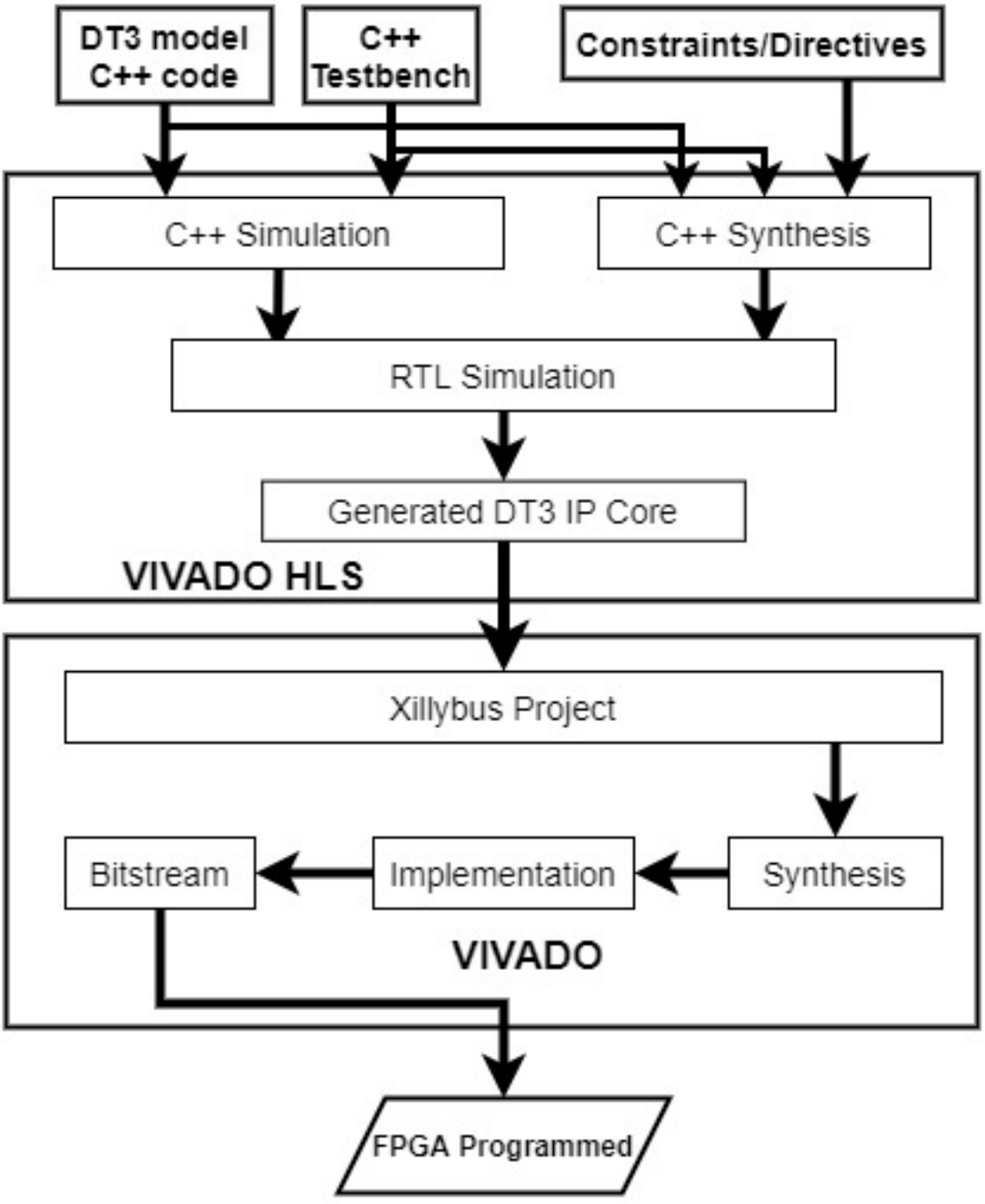

FIGURE 4.6: Hardware design flow

```
void Decision_Tree(int *data_in, int *data_out)
{
#pragma AP interface axis port=data_in
#pragma AP interface axis port=data_out
#pragma AP interface ap_ctrl_none port=return


  uint32_t a, b, c;
  float x1,x2, x3, x4,x5, y1, y2, y3;
  int v,r;

  // Handle input data
  a = *data_in++;
  b = *data_in++;
  c = *data_in++;


  // Convert uint32_t to float
```

FIGURE 4.7: Handling input data to the Decision-Tree block in HLS

Figure 4.8 shows the generated IP core of the DT3 design with the specified input, output, clock, and control ports. This generated block is used as part of the overall system design but it is responsible of network flow classification as legitimate or DDoS attacks.

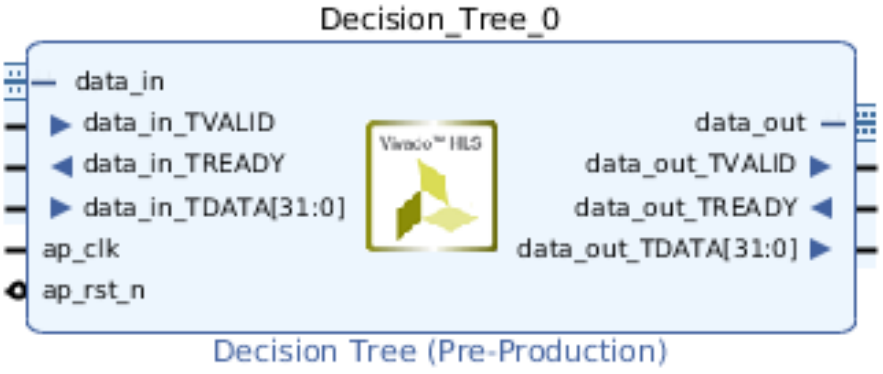

FIGURE 4.8: Decision-Tree block design from Vivado HLS

4.3.5 Communication between host application and FPGA

Effective communication between the software application running in the PS (host) and the DT3 classification model in the PL (FPGA) is paramount for the effective functionality of the overall network monitoring system. In addition, to create an eligible hardware accelerated application, an efficient data transmission flow implementation is crucial. Data samples are

to be transferred from the PS to the PL and the result from the classification is transferred back to the PS for action to be taken.

To implement a high-speed data transmission in the proposed approach, communication between the host application and the DT3 block in the FPGA fabric is achieved using the AXI bus, which interconnects the application on the host to the AXI ports on the PL as shown in Figure 4.4. The Xillybus IP core is adopted for fast data transmission between the PS and PL of the ZedBoard [153]. The Xillybus IP core is an Intellectual Property (IP) core developed by Xillybus Ltd. It contains the implementation of the necessary logic required for fast and efficient transmission of data between the processor and the FPGA via either PCIe bus or AXI4 interfaces [153]. The PCIe bus logic is for implementation on a PC whereas the AXI4 bus is for implementation on SoC devices such as the ZedBoard.

The Xillybus project consists of the necessary software kernel driver for interfacing the software application to the FPGA and Direct Memory Access (DMA) are used to move data between applications in the processor and FPGA with minimal processor overhead. In addition, communication across the buses are transparent leading to design simplicity and decrease in development time [153]. Furthermore, its core presents a range of bandwidth from 8 bits to 32 bits and runs at 250 MHz. For the proposed hardware design, the bandwidth is set at 32 bits which is due the size of the data samples. The installation of the Xillybus driver enables the recognition of the FPGA by the host Operating System that can be Windows or Linux.

The motivation for hardware acceleration implies that many data samples are required to be effectively handled. The implemented approach makes use of two threads. The first thread is responsible for the transmission of data samples from the host application to the FPGA logic for network flow classification and the second thread is responsible for receiving the classification results from the FPGA as depicted in Figure 4.9. The host shown in Figure 4.9 is running an Ubuntu Operating System.

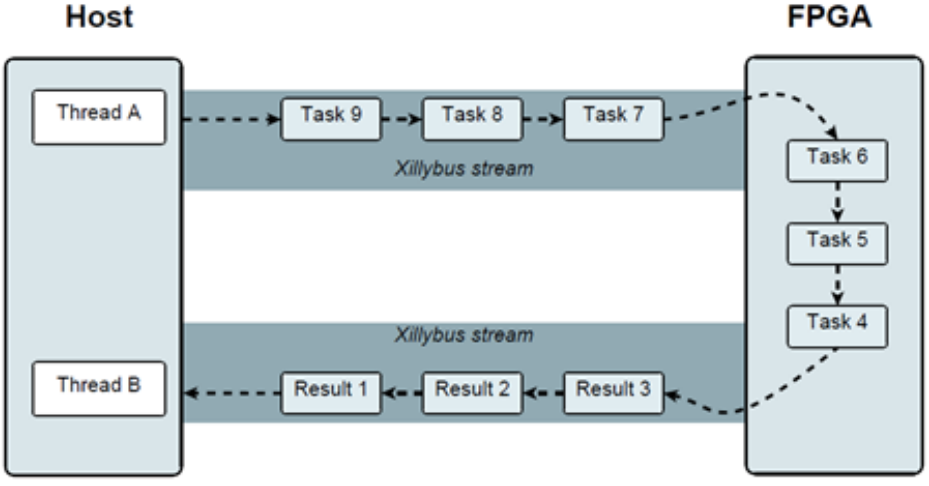

FIGURE 4.9: Xillybus data flow between host and FPGA [153].

The use of the pipelined technique leads to reduction in hardware latencies as latency is influenced by the processing time of data samples between the two threads and the throughput is dependent on the processing capabilities of both the two threads and the FPGA logic.

- **Host Application to FPGA Data Transfer**

  Xillybus IP core uses a continuous stream transport of data samples to the user interface on either the host application or the FPGA side. It uses the traditional round robin technique with a set of DMA buffers under the hood [153].

  Figure 4.10 presents the flow of data from the host application to the FPGA (downstream) direction. The shaded portions in the image represent the unconsumed data in the storage elements. As shown in Figure 4.10, four DMA buffers are used; however, this could be modified depending on the requirements of the application.

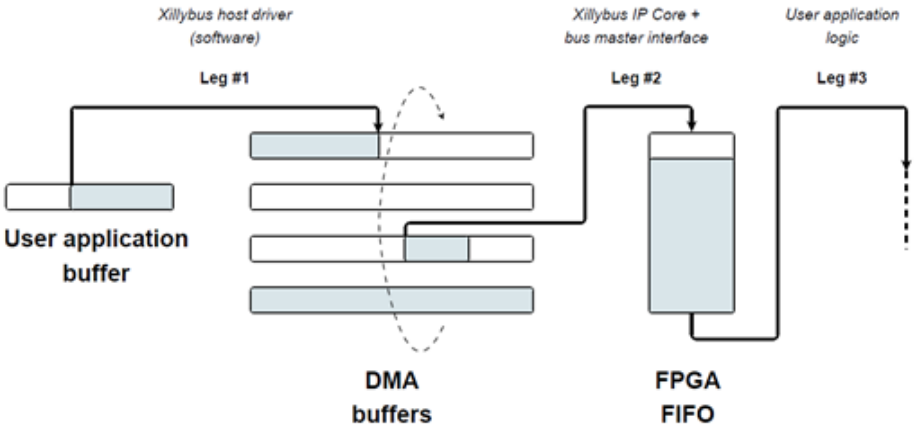

FIGURE 4.10: Xillybus data flow from Host to FPGA [153].

As shown in Figure 4.10, data flows from the host to the FPGA logic in three stages called legs. The first stage also called leg #1 shown in Figure 4.10 is implemented in the Xillybus driver running in the PS of the ZedBoard by responding to write() system calls. The selected pool of DMA buffers is allocated in the host's RAM memory. The DMA buffers start off empty and belong to the host. The host application then writes data into the buffers in a round robin matter. In the event of a full buffer, the hardware is informed that the buffer is ready for use and guarantees not to write into it again until it is returned to the host by the hardware [153].

In stage 2, the data in the DMA buffers in the host's RAM are copied into the FIFO in the PL of the ZedBoard. This is achieved by using the AXI bus interconnect to read the data directly from the host's DMA buffers without the intervention of the host's processor. The FIFO's "full" signal is used to control data flow. In the event of a full FIFO, the internal state machine of the IP core momentarily stops the fetching of data and then continues from where it stopped in the DMA buffers [153].

Finally, in stage three, data is fetched from the FIFO by the user application logic in the PL of the ZedBoard for processing. The FIFO's "empty" signal is used to indicate when the FIFO becomes empty to avoid underflow.

- **FPGA to Host Application Data Transfer**

  Figure 4.11 displays the flow of data from the FPGA to the host (upstream) direction. The shaded portions in the image represent the unconsumed data in the storage elements. This involves the movement of classification results in the opposite direction to what is depicted in Figure 4.10. In addition, it involves three stages known as legs.

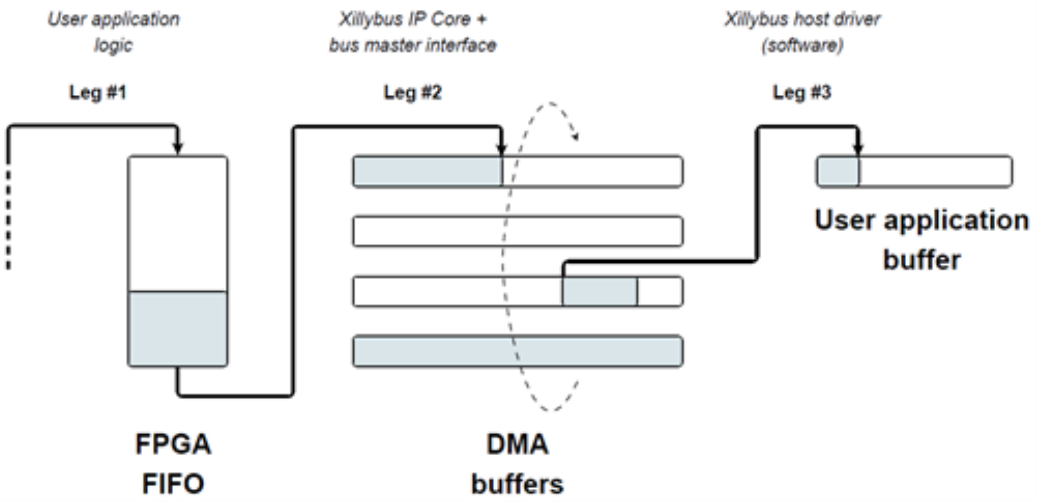

FIGURE 4.11: Xillybus data flow from FPGA to Host [153].

In stage one as shown in Figure 4.11, data is pushed by the user application in the PL into the FIFO, which is connecting the IP core to the user application logic. The FIFO's "full" signal is used to indicate when the FIFO becomes full to avoid overflow.

In stage two, data is copied by the Xillybus IP core from the FIFO of the PL to a DMA buffer in the host's RAM. To achieve this, the master AXI bus master interface is used to write data directly to the DMA buffer of the host's memory without the intervention of the host's processor [153].

The third stage is implemented in the Xillybus driver running in the PS of the ZedBoard by responding to read() system call. The DMA buffers handed to the driver are checked to determine the presence of unconsumed data. If data is present, the data is copied into the user buffer and the empty DMA buffers are returned to the hardware [153].

4.3.6 Overall Hardware Design

Figure 4.12 shows most part of the overall system design and the overall design is depicted in the Appendix C. As shown in the figure, the Decision-Tree (DT) core uses a single AXI Stream (AXIS) interface configured as slave port on the blockdesign to receive data and

instructions as well as a single AXIS interface configured as a master port (*m_axi*) to return the results of the computations by the DT.

In addition, the software application on the host processor (ARM Cortex A9 core) communicates with the DT application on the FPGA using an AXI4 interface and this connection is implemented based on the AXI interconnect block which consist of dedicated AXI4 to AXI Stream converters and FIFO blocks for data buffering and storage. The use of the AXI interconnect block enables continuous flow of data samples to and from the DT application.

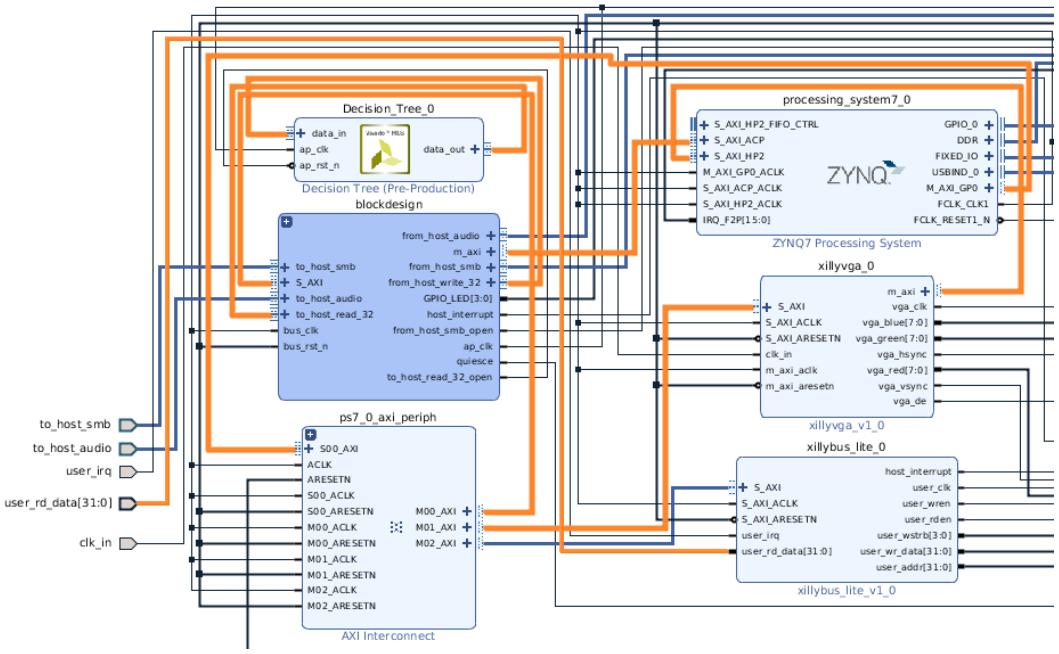

FIGURE 4.12: Hardware design connectivity between PL and PS of the ZedBoard

4.4 Experimental Results

The testbed for the experiment is presented in Figure 4.13. As shown in the figure, the ZedBoard is connected to a monitor through the VGA port for display of results and user experience. In addition, an Ethernet cable is used to connect the ZedBoard to a TP-LINK 16-port smart switch and the USB OTG port on the board is used to connect to both a keyboard and mouse for interacting with the different elements on the board.

4.4.1 ZedBoard Resource Utilization

The available resources of the ZedBoard required for the effective implementation of the hardware design of the proposed solution are shown in Table 4.1. These resources include Flip-Flops (FF), Look-Up Tables (LUT), LUTRAM, Block RAM (BRAM), Input/output (IO), Buffer Global Clock (BUFG), Mixed-Mode Clock Manager (MMCM) and Phased-Locked Loop (PLL).

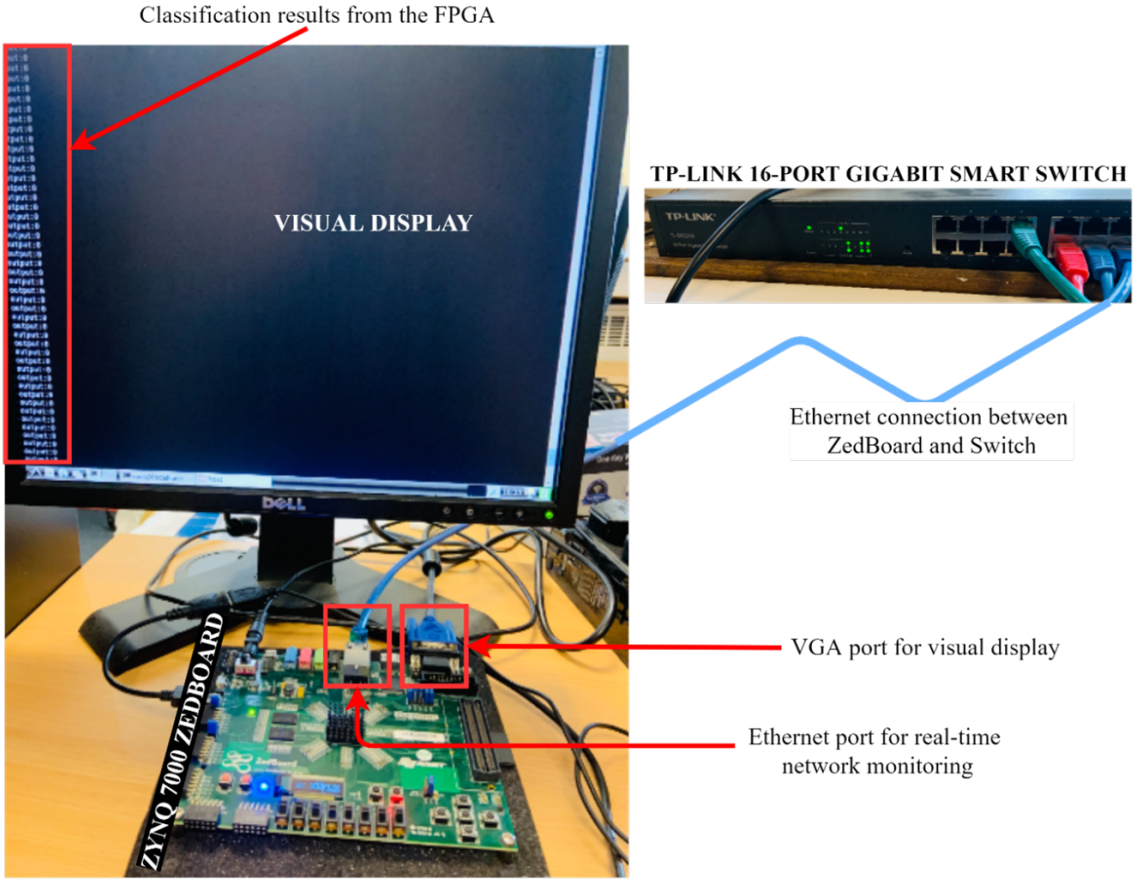

FIGURE 4.13: Testbed setup

TABLE 4.1: Available resources in the ZedBoard

Resource	Available Resources
LUT	53200
LUTRAM	17400
FF	106400
BRAM	140
IO	200
BUFG	32
MMCM	4
PLL	4

Figure 4.14 highlights the resource utilization of the hardware design implementation of the proposed approach on the ZedBoard. As shown in Figure 4.14, the design utilizes around 10% of the available LUT which is partly used in the design and implementation of the DT3 model. The lowest resource utilized is the LUTRAM where only around 2% are utilized. Over 42% of the IO resources were utilized which is the highest in the design. The on-chip power utilization of the hardware application is estimated to be approximately 1.997 Watts which is significantly less than what modern CPUs utilize.

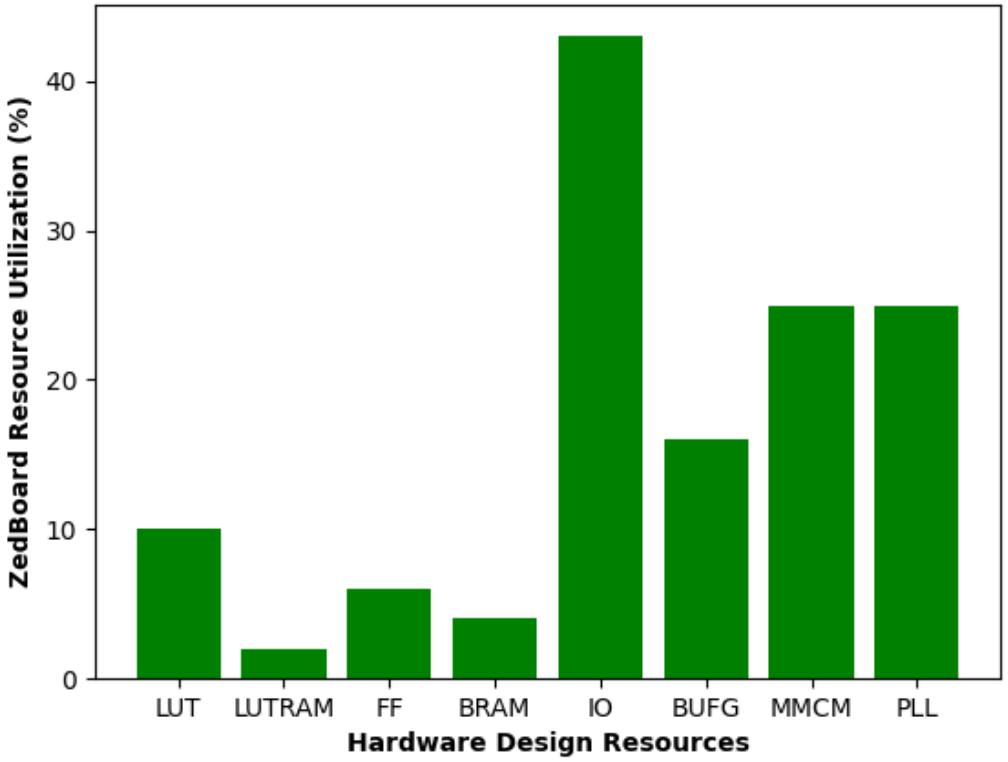

FIGURE 4.14: ZedBoard overall hardware design resource utilization

4.4.2 Hardware Timing Diagram

The timing diagram of the implemented DT3 model during the simulation and synthesis phases is shown in Figure 4.15 using the Vivado Logic Analyzer.

As depicted in the Figure 4.15, the DT3 block starts off (ap_idle) in idle state being 1 which is active and changes to 0 when ap_start is high (becomes active) through the introduction of data samples. The data samples, which are the network flow features are represented with A, B and C, which are 32-bits values.

The result of the classification carried out is represented as outcome, which is a 32-bit value, and as shown in the figure, the outcomes of the first few samples are zeros, which

FIGURE 4.15: Hardware design timing diagram

represent normal network traffic flows. The state of the ap_done signal is high when a result is produced and low when the block is active in the process of classification.

4.4.3 Comparison between Hardware and Software Classification speed

Table 4.2 shows the estimated classification time for both the implementation of the DT3 (optimised classifier) software application on the Dual core Arm processor of the ZedBoard and the hardware implementation on the same board. Different data sample sizes (number of randomly selected network traffic flows) were presented to the different applications for classification and the time required to classify all presented samples were measured. As shown in Table 4.2, classification using the hardware application is significantly faster than the software application i.e. when 200 samples were presented to the applications, it took the hardware application 0.332 milliseconds to classify all samples whereas the software application took 4.399 milliseconds. The difference in the classification speed is also highlighted when 500, 1000 and 2000 network flows were presented for classification.

TABLE 4.2: Software vs Hardware classification speed of the DT3 classifier.

Network flows sample size	**Software classification time (milliseconds)**	**Hardware classification time (milliseconds)**
200	4.399	0.332
500	22.456	0.945
1000	53.807	1.671
2000	115.253	3.857

Further evaluation and comparison regarding the difference in classification speed between between the hardware and software applications were carried out and the results obtained indicated that the hardware application performs better than the software application with around 14 to 32 times faster classification time when tested with 200, 500, 1000 and 2000 network flows data samples. This can be attributed to the pipelining property of PL of the ZedBoard as some instructions were performed in parallel unlike the sequencial approach taken by the software application.

In addition, the use of 1000 samples tend to yield the highest hardware speed-up result compared to the use of 200, 500 and 2000 data samples. With 1000 data samples presented, the optimised hardware classification model yields a classification result approximately every 1μs which is one of the fastest classification rates achieved in this field. This is due to the full utilization of the available link bandwidth for data transfer between the host running on the dual core Arm processor of the ZedBoard and the Programmable Logic (FPGA).

The performance of using 200 and 500 data samples is less due to inefficient use of the data transfer bus as the bandwidth of data bus is not fully utilized. However, when 2000 samples were presented, the data bus is unable to transfer all samples at once so, data samples are transfered in batch leading to an extra latency.

In addition, a classification accuracy of over 98% is achieved when tested with different data sample sizes across the multiple datasets used in Chapter 3. The classification accuracy of the hardware application is slightly less than the one achieved using the software application due to conversion of the data samples from floating point numbers to integers for efficient data transfer and classification in the FPGA. This is because the use of floating point numbers in the hardware application introduced some anomaly in the bheaviour of the system i.e. classification results were presented incorrectly. However, the use of integers in place of floating point numbers resolved the abnormal behaviour of the hardware application.

4.4.4 Comparison with Related Research Work

The results obtained from the experiment carried out in this chapter are compared with results from related works specifically focusing on detection accuracy, speed and power consumption, as these are the key areas in effective and efficient detection of DDoS attacks using hardware systems.

Table 4.3: Comparison of proposed approach with related work

Work	Number of selected features	Hardware and Software test	Hardware only test	Classification time (s)	Detection accuracy (%)
[154]	3	No	Yes	1μs	100%
[155]	5	Yes	No	milliseconds	96%
[156]	-	Yes	No	milliseconds	96%
Proposed Approach	3	Yes	No	1μs	98%

From Table 4.3 it can be seen that the proposed approach covered in this chapter achieves high detection accuracy similar to related works using the least number of selected features carried out in this field. The classification time for work in [155] and [156] involve packet pre-processing and classification however, the number of packets required for classification was not stated and it stated that the classification is within milliseconds. Work in [154] only considered classification by the FPGA block whereas the work proposed in this chapter considered the transfer of data samples between the host application and the FPGA application and the classification of data samples in the FPGA in addition to the transfer of classification results from the harware application to the software application for actions to be taken.

4.4.5 Applications of Monitoring Architecture

The proposed network monitoring architecture implemented on the ZedBoard presents the potential of a wide range of applications across different network nodes. The ZedBoard implements a 10/100/1000 Ethernet port which presents a wide of network connection speeds using the Marvell 88E1518 PHY. This offers the flexibility of monitoring network speeds from 10Mb/s to 1Gb/s links of which applies to a lot of small to medium size networks [151]. The implementation of the proposed network monitoring solution on the ZedBoard could be deployed in a network to monitor the network traffic behaviour at a switch or router. In addition, it could work in collaboration with routers and firewalls to effectively detect, mitigate and prevent DDoS attacks.

The network monitoring architecture permits a flexible approach to securing networks from DDoS attacks as it can be deployed into different nodes (connected to switchs or routers) throughout a network. This is especially important in scenarios where the use of low cost and efficient distributed network monitoring is to be implemented. This means that different parts of the network can be monitored at the same time thereby distributing the network monitoring load across multiple devices instead of using just a single centralised monitoring node. The proposed architecture has been tested in a real-time network environment with deployment at the different monitoring nodes highlighted in the ANRG LAN network shown in Chapter 3.

4.5 Summary

The most prominent processing platforms used in embedded systems for fast processing have been introduced. Although many embedded systems are Multi-core based, performance improvement is limited by the fraction of the instructions that can be executed in parallel simultaneously by the available cores. In addition, GPU based architectures provide a good pipeline performance, but power requirement is an issue. However, FPGA designs provide the best trade-off between pipeline performance and power requirements.

In this chapter, a hybrid network monitoring architecture based on the Zynq 7000 ZedBoard has been proposed. The proposed architecture has advantages for power consumption and less expensive compared to other systems that exist in the state of the art. In addition,

the classification application on the FPGA achieved fast classification of 1μs per data sample. This is over 30 times faster than a tradition core with over 98% classification accuracy. Furthermore, the proposed design offers the flexibility for both deployment and classification as data samples can be processed in real-time and offline. The prototype proposed in this chapter can easily be implemented in other high computing devices like the NetFPGA and NetFPGA Sume to achieve line rate processing of over 40Gbps [157], [158].

In the next chapter, the DT3 classification model is implemented as part of a distributed monitoring approach comprising of a predictive model in Network Simulator 3 (NS3) to investigate the influence of distributed monitoring over centralized monitoring approach used in traditional DDoS attacks detection solutions. Further investigation will be conducted on how distributed monitoring approach leads to early detection of DDoS attacks which leads to reduction in packet loss and network link delays.

Chapter 5

Distributed Monitoring for DDoS Flooding Attacks Detection

In Chapter 4, a hybrid network monitoring architecture based on the Zynq 7000 ZedBoard was implemented for DDoS attacks detection. The proposed architecture comprises of a software application for capturing network packets and an FPGA application for classifying network flows to either attack or legitimate. The architecture achieved fast classification of 1μs per data sample with over 98% classification accuracy. The classification rate achieved is over 30 times faster than a traditional core and the classification time includes the movement of data samples between the host software application and the hardware classification application in the FPGA of the board.

In this chapter, the proposed work includes the development and implementation of a DDoS attacks simulation model for studying and analysing the defensive approaches against DDoS flooding attacks. The most commonly implemented DDoS flooding attacks will be modelled in Network Simulator 3 (NS3) as these attacks account for over 90% of the DDoS attacks performed across the world [7]. However, to model a real DDoS attack scenario, several well-known DDoS attacks datasets were analysed, and the information and characteristics of attack network flows are used. Further investigations will be conducted into the effects of the attacks on the victim's network infrastructure.

In addition, the use of network monitoring and detection systems will be implemented to investigate the impact of these systems in the proposed network. The effectiveness of the systems was evaluated using the distributed monitoring approach.

5.1 Contributions

- Design of a DDoS flooding attacks testbed using NS3
- Implementation of a DDoS flooding attacks predictor model based on statistical behaviour of network traffic flows
- Detection of common DDoS flooding attacks such as ICMP and UDP flooding DDoS attacks

- Analysis of the detection approach and its effectiveness in the rapid detection of the onset of a DDoS attack and effectively stopping the attack.

5.2 Modelling DDoS Flooding Attacks

The modelling of DDoS attacks is not an easy task in real life [159]. However, these attacks can be modelled through a specialized network simulator or testbed. In this chapter, the most affordable and effective option for modelling DDoS flooding attacks is explored. This section will discuss the choice of network simulator along with the network design considerations.

5.2.1 Choice of Network Simulator

There are a few well-recognised network simulators used by researchers for simulating different network behaviours under different conditions. Each of the available network simulators possess its own advantages and disadvantages therefore, it is crucial that the selected simulator can meet the design specifications based on the aims of the research. After extensive research on the well-recognised network simulators, the Network Simulatorn 3 (NS3) was chosen as the tool for conducting the experiment.

Although NS3 is unable to simulate the behaviour of a network server in terms of CPU utilization, which is important in a DDoS attack scenario, other very important network parameters (link bandwidth, transmission delay, link bandwidth consumption and packet loss) which are affected by these attacks, can be simulated in NS3. The highlighted network performance measures above are the targets of DDoS attacks as it aims to exhaust network resources and disrupt the performance of networks [50].

For results obtained from simulation to be validated, they must be compared to results from real networks. However, there are some difficulties in obtaining a 100% accurate comparison rate as there could be differences between the real network topology to the one designed in NS3. In addition, the real network link bandwidth could be higher compared to ones specified in NS3. These are some of the considerations that should be taken into account when analysing such simulation results. NS3 however, provides a behaviour that is close enough to that of real networks [160].

In other network simulators like Riverbed Opnet modeler [159], a DDoS attack scenario can be simulated by setting some network traffic parameters but server applications, network links speed, and other important parameters would be missing which makes it difficult to obtain results that can be compared to those of real networks. The best alternative is NS3 which is a free network software simulator with a high level of reputation through its collaboration with the University of Washington and the Georgia Institute of Technology along with thousands of reseach publications across reputable journals [160], [161]. NS3 allows users to create different network topologies with the added flexibility of automating network nodes generation thereby spending less time is this design phase. NS3 is very dynamic as it can support some visualizers which provide a graphical interface where users can interact with their designs during simulation. Furthermore, it allows users to set network

traffic parameters (transmission rate, packet sizes, link bandwidth, etc.) and monitor the behaviour of the network using different network trace objects which are provided.

5.2.2 Network Design Consideration

Below are the network design considerations made in modelling the proposed network designs presented in this chapter. These considerations include:

- **Distributive nature of DDoS attacks**

  The distributive nature of these attacks mean that the exploited systems could be from different networks in different geographical areas around the world [162]. This is evident in real attack scenarios in the past and present which have been captured and provided in form of empirical datasets which are widely used by researchers in the field of security. For example, the CAIDA 2007 DDoS attack dataset consists of over 5,000 unique IP addresses from different classes of networks, which indicates that the zombies used in the attacks, were from different networks [100]. Based to the distributive nature of DDoS attacks which is evident across the multiple real-life attack scenarios, the designed network comprised of multiple connected LANs.

- **Number of attack devices**

  The implementation of DDoS attacks involves the use of many exploited devices known as zombies to send huge volumes of network traffic to a victim's network infrastructure. In the CAIDA DDoS attacks dataset, the approximate ratio of attack to legitimate network flows was 5:1 meaning there were five times the number of attack devices compared to legitimate devices [100]. Therefore, this ratio of attack to legitimate nodes will be used in the modelling of these attacks' scenarios.

- **Network Link Bandwidth**

  The bandwidth of network links is very important as it determines how much data is transmitted at a given time and it can range from a few Mbps to hundreds of Gbps. Furthermore, the higher the link bandwidth, the lower the transmission delay as more data can be transmitted at a given time period [163]. Also, the transportation of data is faster at a LAN compared to a Wide Area Network (WAN) and this leads to a significantly higher transmission delay in WAN links [163]. Due to these reasons, the designed network comprises of LAN links having higher bandwidth with lower transmission delay compared to the WAN links.

5.2.3 Emulated Testbed Architecture

In the experiments carried out in this chapter, the testbed architecture used in [5], [102] shown in Figure 5.1 for generating up-to-date DDoS attack datasets that are widely used by researchers was emulated and extended in NS3 to form multiple test scenarios.

Figure 5.2 shows the overall proposed network design, which comprises of 16 individual LANs including that of the victim. The LANs highlighted in green are the legitimate LANs

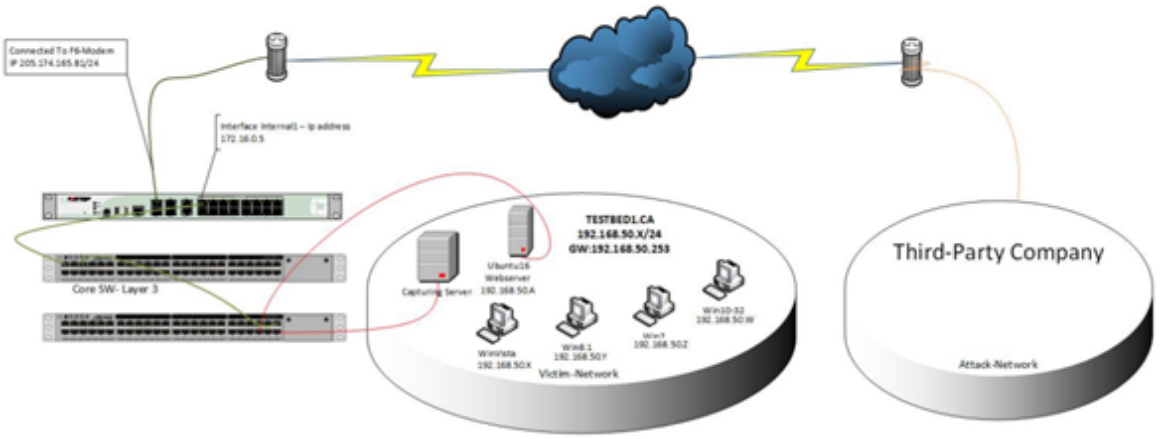

FIGURE 5.1: Testbed Architecture for CICIDS2017 and CIC2019 DDoS Datasets [5], [102]

that consist of only legitimate end devices for generating legitimate network traffic whereas those highlighted in red are the attacker LANs used for generating high volume DDoS flooding attack network traffic. R1 to R6 represent the routers used in connecting the different LANs together.

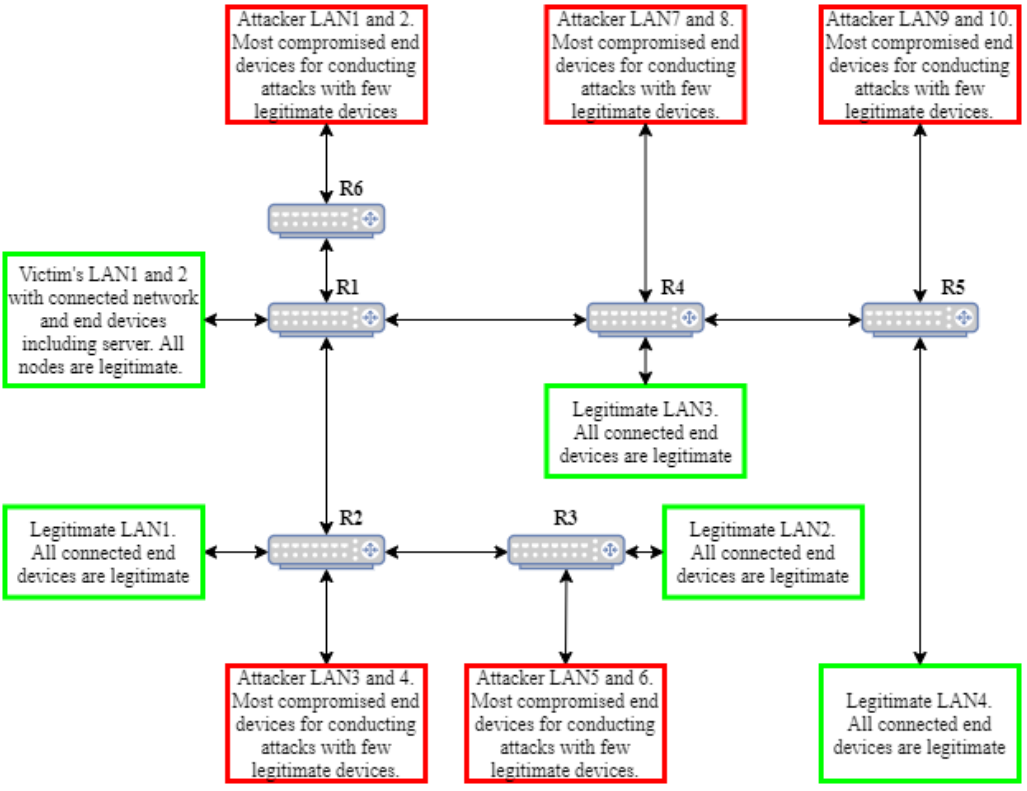

FIGURE 5.2: Proposed network design

Figure 5.3 depicts the LAN network layout for the victim's network infrastructure and how it is connected to other external network LANs. As shown in the figure, the victim's LAN consists of a few network and end devices such as router, switches, workstations, and servers. The core switch labelled as SW1 is directly connected to the gateway router (R1), Server 1 (S1), Server 2 (S2), switches SW1 and SW2 within the victim's LAN. In addition, several workstations along with monitoring and detection systems are directly connected to SW2 and SW3 respectively as shown in Figure 5.3. All the generated network DDoS attack and some legitimate network traffic are directed to the victim's S1, Workstations 1 (W1) and 21 (W21) located in the victim's network infrastructure.

Furthermore, the monitoring and attack detection systems denoted with M1, M2 and M3 are distributed across strategic network points in the victim's LAN to investigate the effect of distributed monitoring in safeguarding networks from the adverse effect of DDoS flooding attacks. These monitoring and attack detection systems will be activated during simulation to measure the effects of active monitoring in the effective and efficient detection and prevention of DDoS flooding attacks. The victim's LAN network layout is similar to the layout shown in Figure 5.1 used in [5], [102]. The attacker LANs which are external to the victim's network infrastructure shown in Figure 5.2 consist of several exploited devices called zombies, which are used to generate the DDoS flooding attack network traffic that are directed to the victim's network.

Finally, there are other legitimate LANs that are external to the victim's network. These LANs consist of several legitimate network nodes used 1for generating legitimate network traffic directed to the victim's servers (S1 and S2).

5.3 Network Traffic Generation

To generate the appropriate network traffic that will emulate the occurrence of DDoS flooding attacks, multiple factors are considered. The first factor considered involves the use of the properties of the DDoS attack flows based on the well-known available datasets [5], [100], [102]. The second factor involves the use of attack to legitimate network flows ratio, which is important in obtaining a close estimation of real-life DDoS attacks scenarios [5].

5.3.1 Characteristics of Network Flows

Based on the analysis carried out on the characteristics of network flows, which provide enough information for effectively distinguishing legitimate from attack network flows across the different DDoS attacks datasets considered in this chapter, the three main features ranked highest based on the Low Variance filter technique discussed in Chapter 3, were selected. A few network flows with the selected attributes are presented in Table 5.1.

As shown in Table 5.1, it can be evaluated that both the average packet size and average packet Inter-Arrival-Time (IAT) for DDoS flooding attacks of selected samples are significantly less than those of legitimate network flows. This indicates that these network flow properties can be used to model both legitimate and DDoS attack scenarios in NS3.

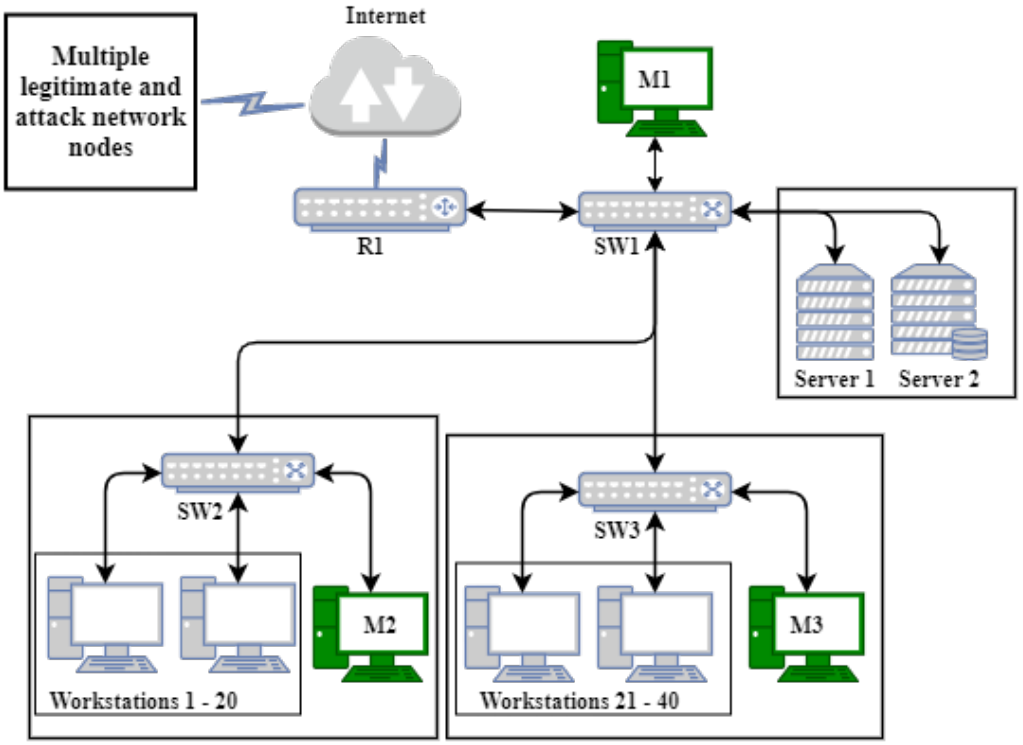

FIGURE 5.3: Victim's LAN network architecture

TABLE 5.1: Some network flow samples with features and labels from the CIADA 2007 dataset

Flow ID	Average packet size (Bytes)	Total bytes	Average IAT (milliseconds)	class
1	124	1364	1841	Normal
2	84	1596	3136	Normal
3	84	1092	3785	Normal
4	60	6000	24	Attack
5	60	6000	27	Attack
6	60	6000	17	Attack

5.3.2 Ratio of Attack to Legitimate Network Flows

Based on the analysis carried out on the CAIDA 2007 DDoS dataset, the ratio of attack to legitimate network traffic flows was evaluated to be around 5:1 which means that for every legitimate network node, there are five attack nodes. However, the ratio of attack packets to legitimate network packets was significantly different as there were many more attack compared to legitimate network packets. Based on this, different network traffic loads were introduced at different phases of the simulation. The different network traffic simulation scenarios are shown in Table 5.2. These are based on the attack to legitimate network flows ratio.

As shown in the Table 5.2, these different scenarios involve the use of several network nodes in generating the network traffic flows. The behaviour and performance of the victim's network under normal and attack traffic conditions are examined under the scenarios highlighted in Table 5.2.

TABLE 5.2: NS3 network simulation scenarios

Network Traffic Scenarios	Number of Attack Nodes	Number of Legitimate Nodes
SC1	50	10
SC2	100	20
SC3	500	100

In Scenario 1 (SC1), the network traffic generated involves the use of fifty attack and ten legitimate network nodes. In addition, one hundred attack and twenty legitimate network nodes were used in generating the network traffic conditions in Scenario 2 (SC2). Finally, the Scenario 3 (SC3) network traffic involves the use of five hundred attack and one hundred normal network nodes. Figure D.1 in Appendix D.1 shows the network architecture implemented in NS3 under which the simulation of the different DDoS attacks scenarios stated were conducted. It consists of multiple LANs with multiple nodes and WAN connections. Following the design considerations, all WAN and LAN network links were assigned 100Mbps with 1ms and 5ms as minimum and maximum delay within the LAN. In addition, 10ms and 50ms as minimum and maximum delay within WAN connections [164].

5.4 Monitoring, Detection and Mitigation of DDoS attacks

The monitoring and detection approach comprises of mainly two phases. The first phase involves the use of the flowmonitor model in NS3 to monitor and gather statistics of all network flows passing through a specified network node/link. In addition, the predictor model (**algorithm 1**) is used to predict the emergence of possible DDoS attack network flows using the network flow statistics gathered by the flow monitor model.

In phase two, if the predicted outcome from the predictor model is attack, the network flow details are forwarded to the DT3 detection model for validation. If a network flow is validated is an attack, the IP address of the attacker is used by the gateway router denoted with R1 in Figure 5.3 to create a rule and all subsequent packets of the attack flow arriving at the victim's network are dropped/discarded by the router. Multiple network rules are created and used by the router to drop all packets that match the rules. This is to mitigate the adverse effect of DDoS attacks on the victim's network by dropping the packets at the gateway router. Also, all subsequent packets of network flows validated as legitimate are allowed into the network.

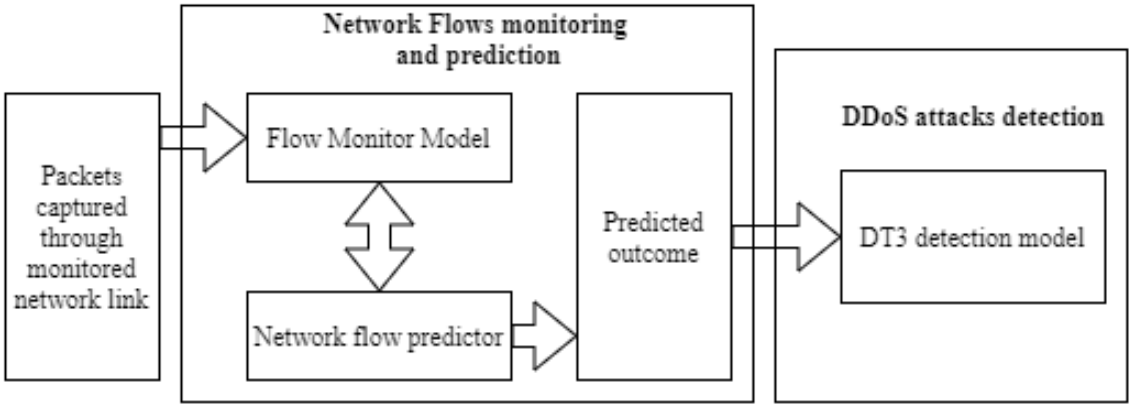

FIGURE 5.4: Network monitoring and DDoS attacks detection approach

5.4.1 Network Flow Predictor Model

To effectively maintain the performance of a network under DDoS flooding attacks, early and accurate detection is paramount. A lot of research in the field of DDoS attacks focus on detection and prevention using various detection techniques. However, if these attacks are not detected early then the network performance will be greatly affected [18]. Having a predictor model can be useful for early detection and prevention of attacks to maintain the effectiveness and efficiency of a network.

A DDoS flooding attacks predictor model is designed and implemented based on the network behaviour under such attacks. Since, a network under DDoS flooding attacks receives a high volume of network traffic; the predictor model can predict the possible future network traffic based on the current condition of the network. To achieve this, network packets entering a network are captured and the packets are classified into network flows through the flowmonitor model in NS3, the attributes and details are passed on to the predictor to predict the characteristic (either normal or a possible DDoS flooding attack) of the network flow.

The predictor model predicts only network flows with three or more packets within a time window of t =1s. The chosen packet threshold is because on average, DDoS flooding attack nodes transmit a high number of packets per second which has been obtained from analysis carried out on some of the well-known empirical datasets used by researchers and the DDoS attack tools freely available such as Hping3 [5], [100], [102].

Algorithm 1: Predictor algorithm

Input: number of packets ($num\_packets$), average packet inter-arrival time (avg_iat)

Output: prediction (attack/legitimate)

Threshold: $iat\_thresh = 43ms; pps\_thresh = 100$

Data: network flow sample x

```
initialization
while num_packets(x) ≥ 5 do
    Predict the future packet per second (pps) of the next one second for sample x
     based on current avg_iat using the equation below:
     pps(x) = 1 / avg_iat
    if avg_iat(x) ≥ iat_thresh and pps(x) ≤ pps_thresh then
        prediction = legitimate
    else
        prediction = attack
    end
    return prediction
    forward details of network flow sample x to DT3 for validation
end
```

If a DDoS attack flow is the result of the prediction, the required network flow attributes are passed to the DT3 model for validation. Once the network flow is validated by the DT3 as attack, a network rule is created in the gateway router and subsequent packets that match the rule are dropped by the router.

5.5 Results and Discussion

The experiments were conducted using NS3 installed on a system consisting of an Intel core i7 Central Processing Unit (CPU) with 3.6GHz Quad core processor, 500GB of storage and 16GB of RAM running a Linux Operating System. The results of the experiments and the performance evaluation of using the predictor and DT3 models for predicting and detecting DDoS flooding attacks are presented in this section. Furthermore, the effects of the DDoS flooding attacks on the behaviour and performance of the victim's network are analysed.

5.5.1 Network Traffic Load at the Victim's network

The different network traffic workload as experienced by the victim's network infrastructure are presented in this section based on the different scenarios shown in Table 5.2.

- **SC1 simulation**

  In the normal network traffic condition where only the legitimate nodes were used in generating the network traffic directed to S1, the workload experienced by the victim's

network at SW1 was estimated to be around 55 packets per second on average which is significantly low compared to the bandwidth of the network link. A combination of both legitimate and attack network traffic is generated using ten and fifty legitimate and DDoS attack nodes respectively and the result of this is shown in Figure 5.5.

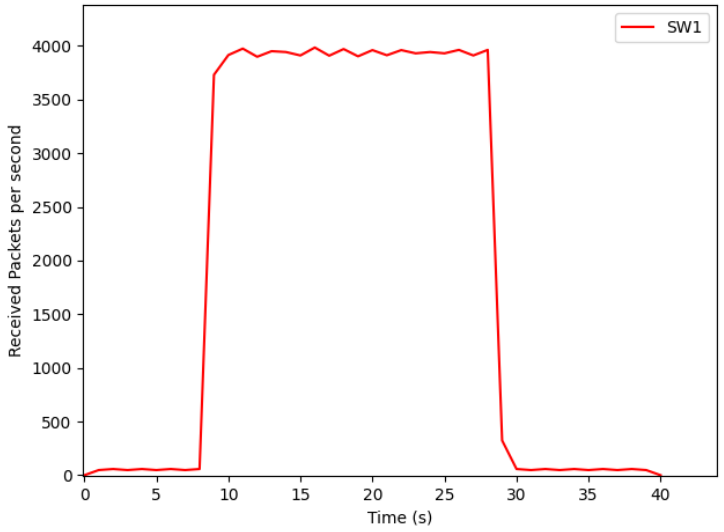

FIGURE 5.5: SC1 network traffic conditions from legitimate and attack nodes at SW1

Under the network traffic conditions shown in the Figure 5.5, the peak operational link bandwidth during the simulation was evaluated to be around 3.2Mbps, which is significantly less than the assigned bandwidth of 100Mbps as shown in Figure 5.6.

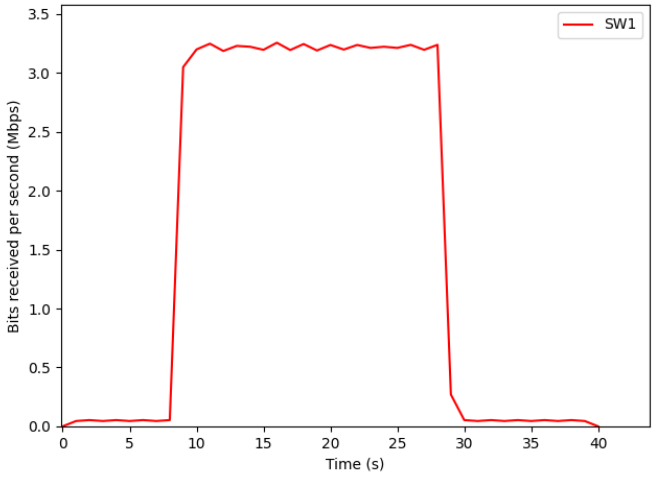

FIGURE 5.6: SC1 network link operational bandwidth between R1 and SW1

This low operational bandwidth resulted in little or no link delays resulting from congestion and 0% packet loss for the monitored legitimate network flows across the network.

- **SC2 simulation**

  In the normal network traffic condition simulated, the workload experienced at the victim's network at node SW1 was low with an average of 78 packets per second.

  A combination of both legitimate and attack network traffic is generated using twenty and one hundred legitimate and DDoS attack nodes respectively and the result of the workload is shown in Figure 5.7.

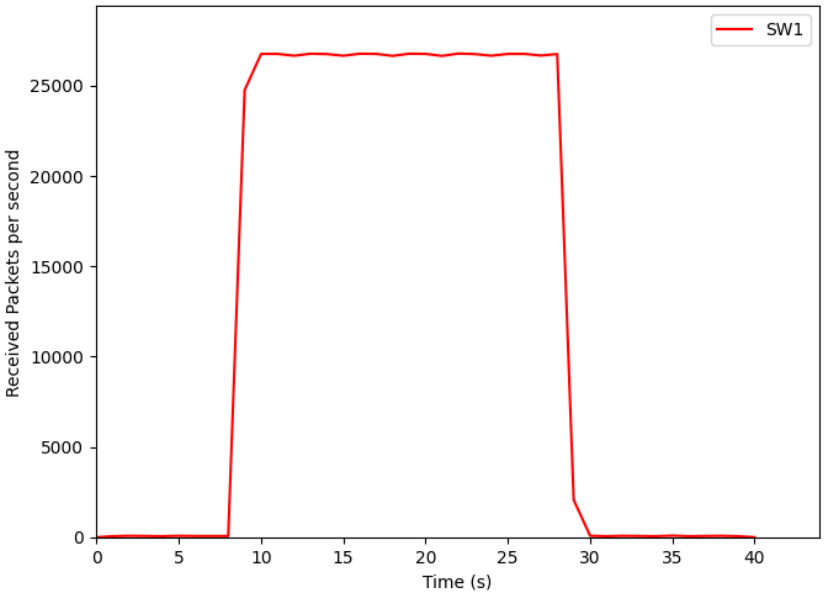

FIGURE 5.7: SC2 network traffic conditions from legitimate and attack nodes at SW1

Under the network traffic conditions shown in the Figure 5.7, the peak operational link bandwidth during the simulation was evaluated to be over 21Mbps, which is less than the assigned bandwidth of 100Mbps as shown in Figure 5.8.

This low operational bandwidth resulted in little or no link delays resulting from congestion and 0% packet loss for the monitored legitimate network flows across the proposed network.

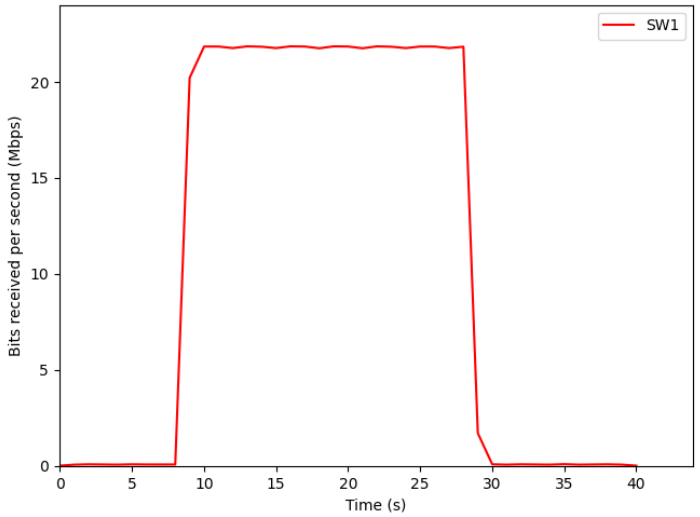

FIGURE 5.8: SC2 network link operational bandwidth between R1 and SW1

- **SC3 simulation**

  In the normal network traffic condition simulated, the result of the workload across selected network nodes is shown in Figure 5.9.

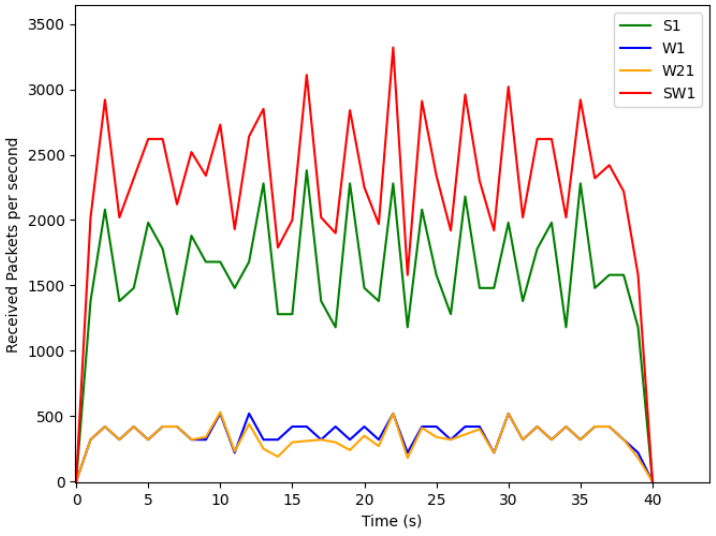

FIGURE 5.9: Legitimate network traffic load are different network nodes in SC3

As shown in the figure, the network traffic loads for Server 1 (S1), Workstations 1 and 21 denoted with W1 and W21 and SW1 are displayed. The traffic load at SW1 represents the overall traffic load experienced by the victim's network infrastructure.

A combination of both legitimate and attack network traffic is generated using one hundred and five hundred legitimate and attack nodes respectively and the result of the workload experienced by different nodes in the victim's network is shown in Figure 5.10.

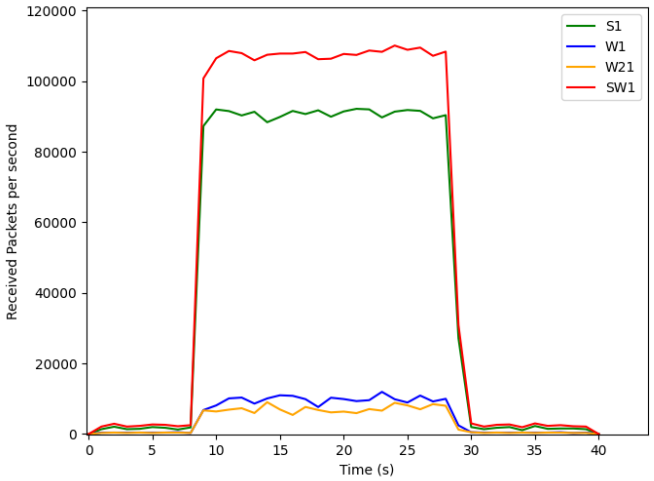

FIGURE 5.10: Legitimate and attack network traffic load at different network nodes in SC3

As shown in Figure 5.10, the network traffic workload is significantly higher than the workload experienced by the victim's network in SC1 and SC2 which is evident with the higher number of network nodes used in generating the network traffic.

Under the traffic conditions shown in the Figure 5.10, the peak operational link bandwidth was evaluated to be above 85Mbps, which means that the assigned bandwidth was consumed during the attack period as shown in Figure 5.11.

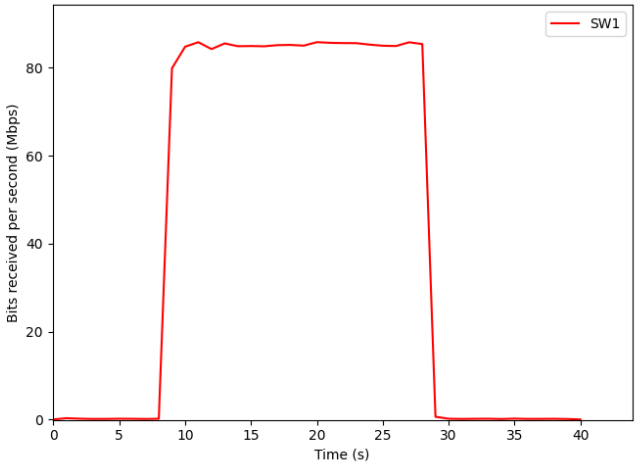

FIGURE 5.11: SC3 network link operational bandwidth between R1 and SW1

The high operational bandwidth resulted in congestion of the network link leading to significant level of delays and packet loss across the monitored legitimate network flows.

5.5.2 Effect of DDoS flooding attacks on victim's network

The results of the victim's network behaviour without the use of any monitoring and detection system are presented in this section. The analysed behaviours include network flow mean delays and packet loss ratio.

- **Mean Delays**

  In Figure 5.12, the effect of DDoS attacks on the victim's network is expressed with respect to link delays experienced in the communication between legitimate nodes and Server 1 (S1). As shown in the figure, the link mean delays across all monitored legitimate network nodes are equivalent to the theoretical delays in the link used in communication in SC1 represented with the green bars due to the very low network traffic generated.

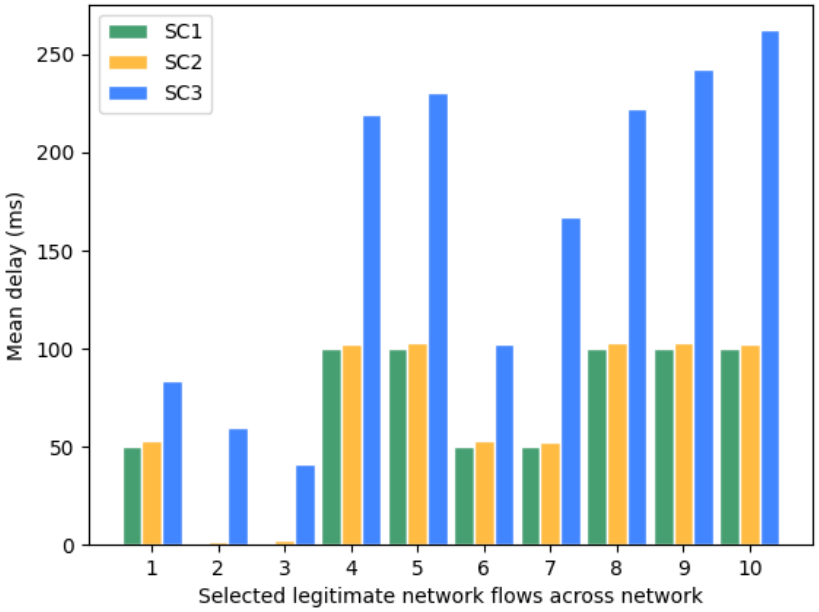

FIGURE 5.12: Network flows mean delays across the selected scenarios

However, the introduction of more attacking and legitimate nodes in SC2 resulted in a slight increase in the link delays represented with the orange bars. The introduced delays were approximately 2ms on average across all the monitored network flows.

In SC3, the use of more attacking nodes with legitimate nodes resulted in the network link between R1 and SW1 operating at maximum capacity. This led to the introduction of significant delays, which on average were over twice the original delays experienced

in SC1. These delays are represented with the blue bars. The minimum and maximum mean delays experienced in SC3 were 32ms and 162ms when compared to mean delays in SC1.

- **Packet Loss Ratio**

  Packet loss is another indicator of a successful DDoS attack. This occurs because of exhaustion of server resources or link bandwidth. This exhaustion leads to congestion, and received packets are randomly dropped, as the link/server is unable to process all the received requests.

  Across the three scenarios discussed in this chapter, the network traffic conditions in both SC1 and SC2 resulted in a 0% packet loss as the server and network links were not tested at maximum operating capacity. However, the network traffic workload in SC3 resulted in packet loss across all monitored flows due to the large volume of traffic introduced. In SC3, the network link bandwidth connecting R1 to SW1 and SWI to S1 were exhausted between t = 8s and t = 30s. This exhaustion resulted in packet loss shown in Figure 5.13.

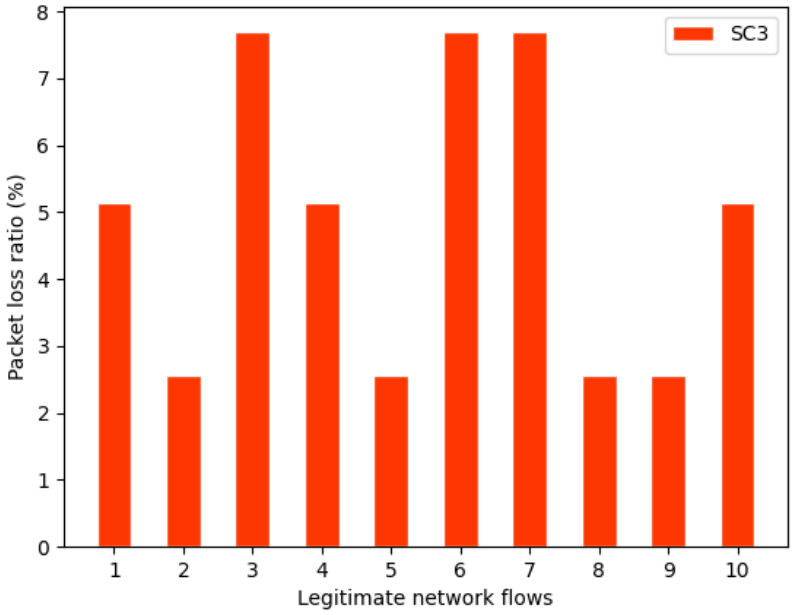

FIGURE 5.13: Network flows packet loss ratio in SC3

As shown in the figure, some network flows experienced a maximum of over 7.5% packet loss whereas other network flows experienced between 2.5% to over 5% loss. This level of packet loss can cause significant network performance issues and affect the Quality of Service (QoS) rendered in a real network environment [165].

5.5.3 Distributed Monitoring and DDoS attacks Detection Systems

The effects of active distributed monitoring and attack detections on the behaviour of the victim's network infrastructure are presented in this section. The analysed behaviours include network flow mean delays and packet loss ratio.

- **Detection Time**

  The average DDoS attacks network flow detection time was evaluated across the monitoring and attacks detection systems across the different network traffic scenarios discussed above. Figure 5.14 highlights the different monitoring and detection nodes denoted by M1, M2 and M3 which were shown in the victim's LAN network architecture in Figure 5.3.

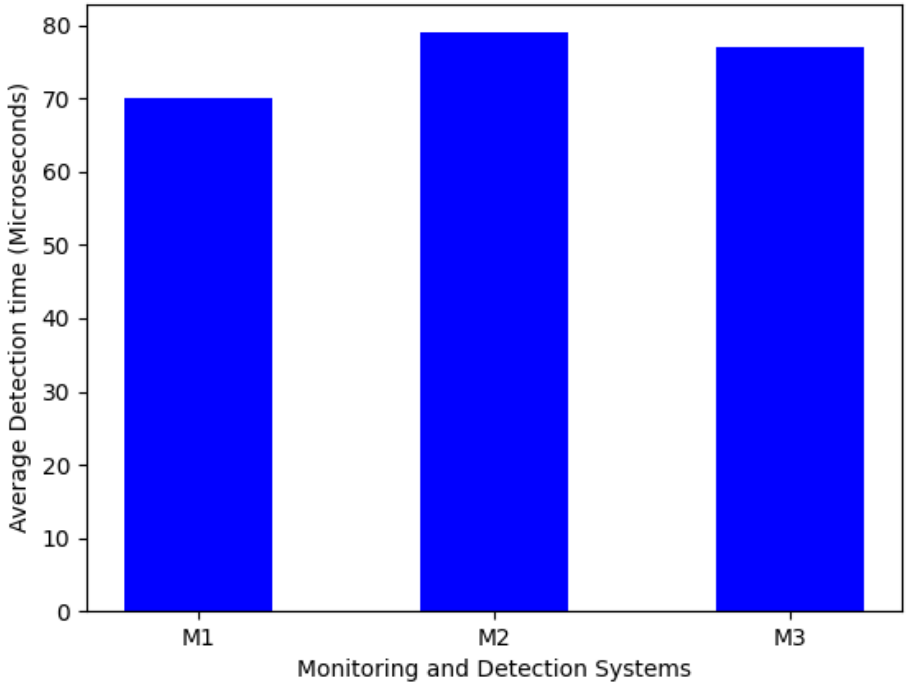

FIGURE 5.14: Average detection time across the monitoring and attack detection systems

The average detection time at M1 which was directly connected to SW1 was evaluated to be around 70μs per network flow. In addition, the average detection time at M2 and M3 were just slightly higher than that of M1 as shown in the figure.

Furthermore, the use of the predictor model was very effective as over 85% of the network flows predicted as attack were validated by the DT3 model as attack flows. This indicates that the use of the predictor model was quite effective in accurately distinguishing legitimate from attack network flows.

- **Monitoring and attacks detection at M1**

  This network monitoring and detection system for the detection of DDoS flooding attacks was deployed at SW1 of the victim's network. This was a strategic approach as all incoming network traffic are destined to pass through SW1 to get to the different destinations within the victim's network infrastructure.

  The detection system proposed in this chapter is based on a combination of the predictor model and the DT3 model implemented in Chapter 3. The predictor model was used to predict if a network flow arriving at the victim's network is legitimate, or attack based on the number of packets received for a specific flow and other flow features described earlier in this chapter. The system was tested under the network traffic conditions of SC3.

  From the tests carried out, the detection system was effective in detecting the attacks with a detection time of 70μs per network flow. In addition, the maximum packet loss ratio across all selected network flows was significantly low at about 1.5% across just three network flows of all network flows monitored. This was significantly less than almost 8% recorded without the use of a detection system.

  Furthermore, the results obtained indicated that although the use of a single monitoring and detection node can be effective in reducing the adverse effects of DDoS flooding attacks, it does not totally eliminate the threat, as the victim's network infrastructure will still experience some effects of these attacks. This shows that the use of a single monitoring and detection node is not the complete solution to completely deal with these attacks.

- **Distributed Monitoring Approach**

  Hybrid monitoring approach involves the distribution of the monitoring and detection systems across multiple strategic locations in the victim's network infrastructure.

  The distributed monitoring approach was implemented in the proposed network and the robustness of the monitoring systems was tested under the network traffic conditions in SC3. The results from the tests indicated that the distributed monitoring approach was very effective as packet loss ratio across all selected legitimate network flows was 0

  As shown in Figure 5.15, Figure 5.16 and Figure 5.17, the victim's server S1 and workstations W1 and W21 experienced a rapid increase in the volume of network traffic between t=7 and t = 8s because of the initiated DDoS attacks from various attack networks. Due to the active distributed monitoring and detection systems implemented, the attacks were effectively detected at t=8s and the packets from the attack flows were dropped. The attacks were effectively stopped, and the victim's network was restored to its normal operating conditions within 3 seconds between t = 8s and t = 11s.

  In addition, packet loss ratio was significantly reduced to 0% across the selected legitimate network flows and the network link delays were significantly reduced. The link delays achieved were on average like those experienced in SC2.

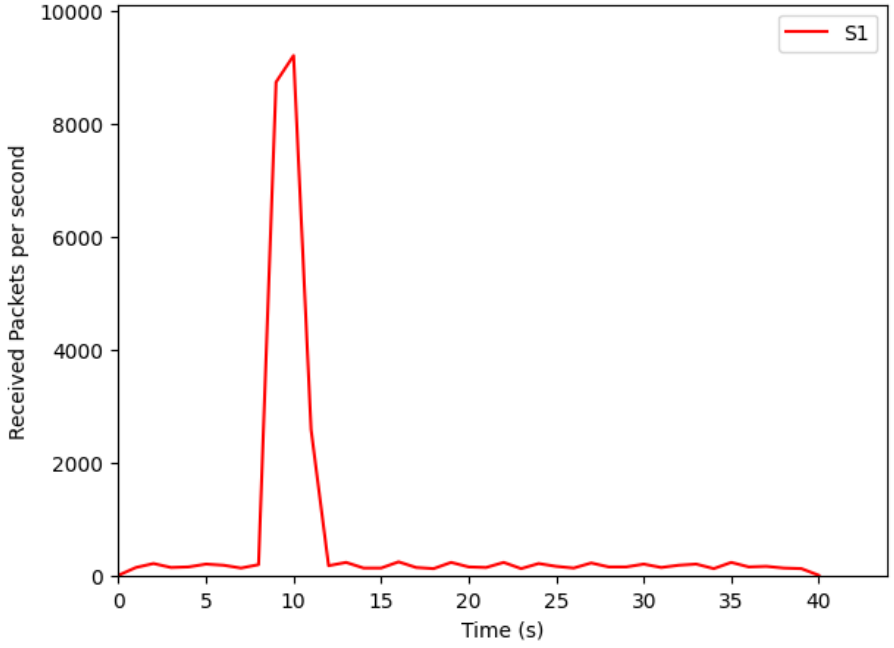

FIGURE 5.15: Network traffic condition at S1

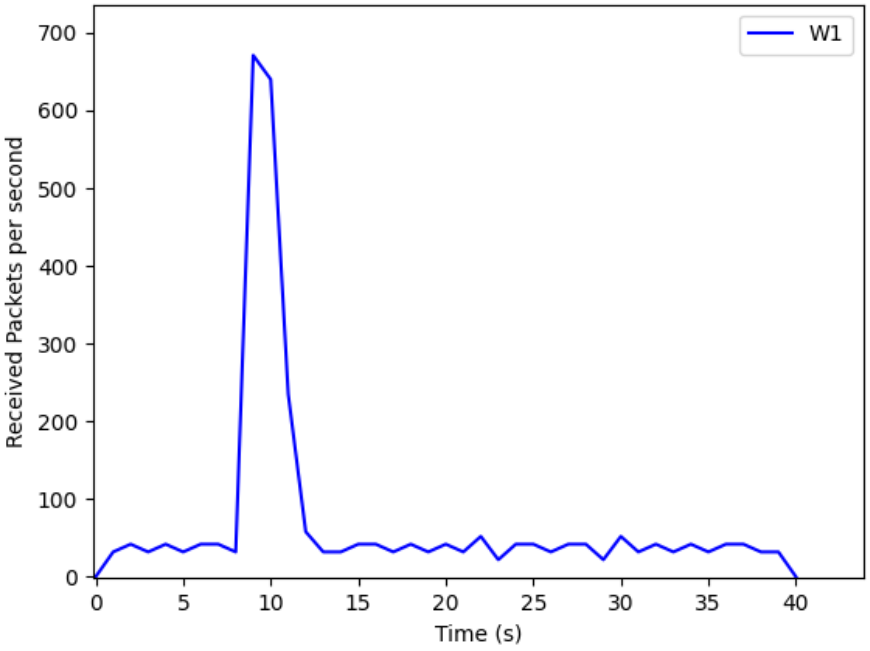

FIGURE 5.16: Network traffic condition at W1

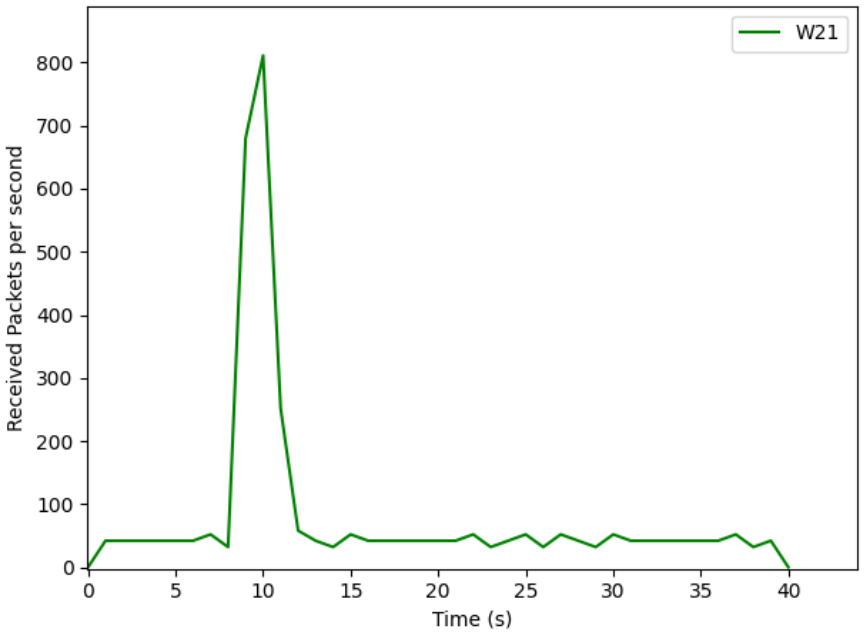

FIGURE 5.17: Network traffic condition at W21

5.5.4 Comparison between simulated data and the CAIDA 2007 dataset

The packets received at the server in the victim's network during the simulation in NS3 were captured and analysed. The network flow average packet IAT feature was analysed and compared with the results of some selected network flows from the CAIDA DDoS attack dataset to evaluate how the simulated data compares to real life DDoS attacks datasets. These results are discussed below using the same number of samples as those used across the different scenarios in Table 5.2.

- **SC1 comparison**

  Based on the results from SC1 simulation, Figure 5.18 shows the box plot of the average packet IAT for the legitimate network flows across the simulated and CAIDA datasets. Ten network flows were randomly selected from both the simulated and the CAIDA datasets and the average packet IAT of the flows were used to create the boxplots.

  As shown in Figure 5.18, there are some differences in the distribution of samples in the two datasets. The samples from the CAIDA dataset tend to have a wider distribution compared to the samples from the simulated dataset.

  Figure 5.19 shows the box plot of the average packet IAT for the attack network flows across the simulated and CAIDA datasets. Fiffty network flows were randomly selected from both the simulated and the CAIDA datasets and the average packet IAT of the flows were used to create the boxplots. As shown in Figure 5.19, there are some

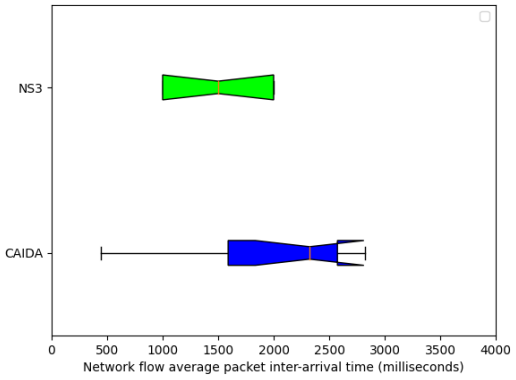

FIGURE 5.18: Average packet IAT for legitimate network flows.

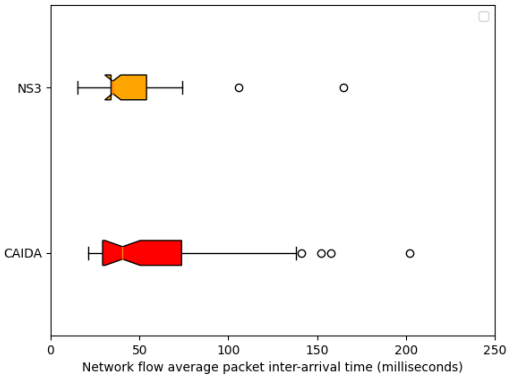

FIGURE 5.19: Average packet IAT for attack network flows.

similarities in the distribution of samples in the two datasets especially samples with lower average packet IAT. However, there exists some differences in the distribution especially regarding the maximum values as the maximum value of the CAIDA dataset is significantly larger than that of the simulated dataset.

Comparison between the two box plots show that there are some differences. These differences are due to the large sample size of the CAIDA DDoS dataset as a pool of selected samples will be more generalized compared to the dataset generated through the simulation in NS3.

- **SC2 comparison**

  Based on the results from SC2 simulation, Figure 5.20 and Figure 5.21 show the box plots of the average packet IAT for both the legitimate and attack network flows (consisting of one hundred attack and twenty legitimate network flows) across the simulated and CAIDA datasets. Twenty legitimate network flows were randomly selected from both the simulated and the CAIDA datasets and the average packet IAT of the flows were used to create the boxplots.

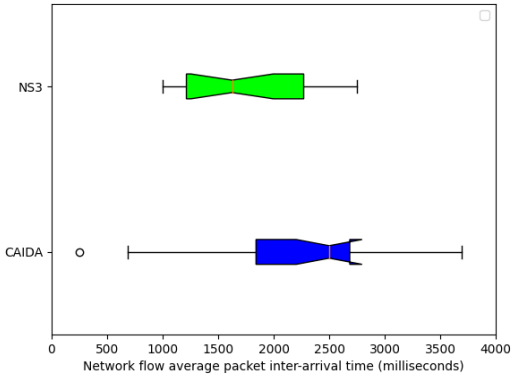

FIGURE 5.20: Average packet IAT for legitimate network flows.

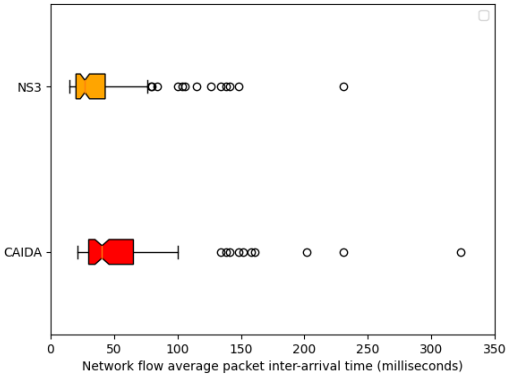

FIGURE 5.21: Average packet IAT for attack network flows.

As shown in Figure 5.20, there are some differences in the distribution of samples in the two datasets. The samples form the CAIDA dataset tend to have a wider distribution compared to the samples from the simulated dataset especially regarding the minmum and maximum values as the maximum value of the CAIDA dataset is significantly larger than that of the simulated dataset but the minimum value is lower than that of the simulated dataset.

Figure 5.21 shows the box plot of the average packet IAT for the attack network flows across the simulated and CAIDA datasets. As shown in Figure 5.21, there are some very close similarities in the distribution of samples in the two datasets especially the minimum values. The box plots of the simulated data in SC2 look closer to the CAIDA dataset than the results obtained in SC1.

- **SC3 comparison**

  Based on the results from SC3 simulation, Figure 5.22 and Figure 5.23 show the box plots of the average packet IAT for both the legitimate and attack network flows (consisting of five hundred attack and one hundred legitimate network flows) across the simulated and CAIDA datasets. One hundred legitimate network flows were randomly selected from both the simulated and the CAIDA datasets and the average packet IAT of the flows were used to create the boxplots.

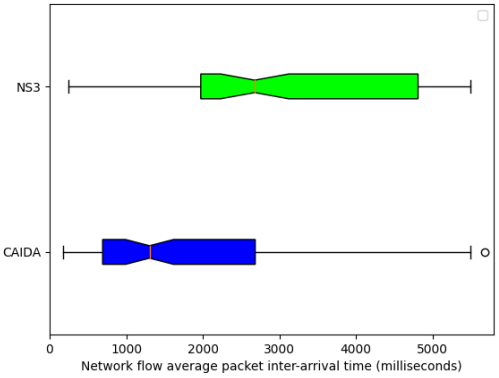

FIGURE 5.22: Average packet IAT for legitimate network flows.

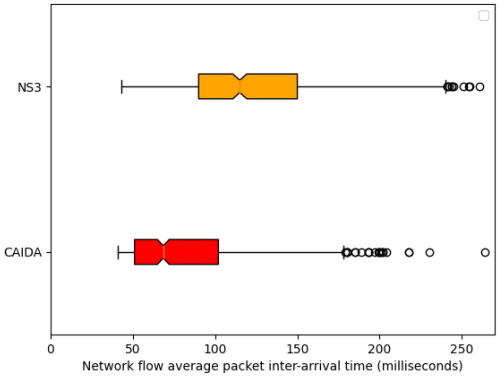

FIGURE 5.23: Average packet IAT for legitimate network flows.

As shown in Figure 5.22, there are some major similarities in the distribution of samples in the two datasets. The minimum and maximum values are very similar in both datasets as shown in Figure 5.22.

Figure 5.23 shows the box plot of the average packet IAT for the attack network flows across the simulated and CAIDA datasets. As shown in Figure 5.23, there are some very close similarities in the distribution of samples in the two datasets especially regarding the minimum values of both datasets although there are some differences in the median and interquartile range values.

Comparison between the box plots in Figure 5.22 and Figure 5.23 show that there are very close similarities compared to the results from SC1 and SC2. This shows that the higher the number of samples used, the closer the results.

5.6 Summary

The effective detection and prevention of DDoS flooding attacks is paramount in ensuring that services across multiple sectors ranging from small online business to large government and private organizations are available to legitimate users whenever it is required. However, in order for these attacks to be effectively detected and prevented, multiple monitoring techniques should be adopted due to the increasing size and complexity of the attacks.

In this chapter, the use of a single and distributed monitoring and detection systems were explored under different network traffic conditions. The monitoring and detection systems combined the use of a flowmonitor, predictor model for predicting the start of attack flows and the DT3 model for detecting the attacks which was discussed in Chapter 3.

The results obtained from the simulation of the proposed network design in NS3 without the use of monitoring and detection systems indicated the adverse effects of DDoS attacks especially with the introduction of many attack nodes in SC3. These effects included significant network link delays and packet loss in legitimate network flows due to the congestion and exhaustion of network link bandwidth. In addition, the deployment of a single network monitoring and detection system at the core switch of the victim's network significantly reduced the effects of the DDoS attacks as packet loss ratio was reduced from almost 8% for some of the selected flows without monitoring systems to around 1.5% with significant reduction in link delays and detection time of around 70μs per network flow.

However, the use of distributed network monitoring and detection systems across different network nodes in the proposed design significantly reduced the effects of the attacks through reduction of packet loss ratio to 0% and reduction in network link delays to a minimum. These results show that the use of a distributed monitoring approach provides many benefits as opposed the use of a single central monitoring approach, which were explored in this chapter. In addition, the incorporation of the predictor model shows that attacks can be detected and prevented in a timely manner which is very crucial to preventing the adverse effects of the attacks to the victim's network infrastructure.

In the next chapter, the conclusions to this thesis are provided. In addition, the various findings and contributions are presented. Furthermore, future works to greatly improve service availability through the implementation of different network monitoring and DDoS attacks detection approaches are suggested.

Chapter 6

Conclusion and Future Work

DDoS attacks are regarded as one of the most prominent threats to network security and these attacks are becoming more complex and larger. The main objectives of these attacks are to bring down network services and render them unavailable to legitimate users. In addition, the victims of these attacks range from individuals to large corporations, which can affect both lives and livelihood as covered in Chapter 2. Therefore, early and accurate detection and prevention of these attacks is paramount to maintaining stability in network services.

This thesis has attempted to answer the main research question posed in Chapter 1, which was "How can DDoS attack detection systems effectively and efficiently detect attacks early with high accuracy and low computational cost?" by proposing some key components that are useful in developing and implementing a DDoS attacks detection architecture that detects attacks early and accurate with low computational cost.

Section 6.1 highlights the major findings and contributions to the field of detection and prevention of DDoS attacks. Possible future work in the detection and prevention of DDoS attacks are outline in Section 6.2.

6.1 Findings and Contributions

A major contribution is that the architecture proposed is lightweight in nature with very little computational cost required. In addition, the architecture can be flexibly deployed at different network nodes to meet the needs of a network as part of a distributed monitoring ecosystem. This allows for early detection and prevention of attacks leading to reduction in the impact of these attacks on a network service. In summary, the findings from the different experiments conducted indicate that the detection architecture is effective in accurately detecting a wide range of DDoS attacks with high detection accuracy using minimal number of network flow features.

Details of the major conclusions from the thesis are summarised in the following subsections.

6.1.1 Introduction, Background and Related Work

The research commenced by first identifying the different characteristics of DDoS attacks and impacts on several network infrastructures. Through the analysis of some prominent real-life successful DDoS attacks, it was clear that there is high im-portance in detecting these attacks early and accurately in order to secure and keep network infrastructure running without disruption.

The study of the existing detection solutions show that there is a gap in research with respect to computational cost and flexibility in deployment. Evaluations of these so-lutions show that many detection systems are designed with little attention to pro-cessing and memory cost. In addition, the ever-increasing sizes and complexities of these attacks have rendered some of the detection solutions incapable of accurately detecting new DDoS attacks.

Furthermore, traditional detection solutions are often centralized where only a single monitoring node is implemented in the network being monitored. In this case, it is difficult to meet the different needs of the network and often requires large memory, which can be expensive. Finally, and most importantly, most of these solutions struggle with scalability and programmability, which are required in modern net-works to meet the different monitoring needs.

6.1.2 A Lightweight Decision-Tree Algorithm for detecting DDoS flooding attacks

The limitations of DDoS detection solutions, which are complex with high computa-tional cost and rigid deployment characteristics, led to the search for new approaches that result in low computational cost with flexibility in deployment to meet the dif-ferent needs of networks. The detection of a wide range of attack types also lead to the development and implementation of new detection techniques for detecting a wide range of existing DDoS attacks.

The proposed method employs the use of a robust feature selection technique to select only relevant features required for effectively and accurately distinguishing le-gitimate from DDoS attack network flows. In addition, it makes use of machine learning algorithms which using the selected features to improve detection. Further-more, it provides programmability as machine learning models can be retrained and deployed in no time.

A range of selected features based on an evaluation of the feature selection tech-nique across multiple up-to-date DDoS attack datasets were used in training a range of machine learning models.

Results from the evaluation conducted across the different models show that the DT3 model performs well across the different groups of selected features. The architec-ture presented is optimised for deployment in low-cost environments for efficient, rapid detection and prevention of DDoS attacks. To achieve a computationally efficiency architecture, the system was trained with a minimal number of features using a robust features selection approach and validated against the CIC 2017 and 2019 datasets.

With just 3 selected features which is the least in this field the DT3 model achieved a detection accuracy of over 99.9% [147]. Analysis of the design shows that the new architecture uses just 7% processing power of a core in an offline mode and provides no additional overhead to the monitored network.

6.1.3 Hardware Accelerated Application for DDoS flooding attacks detection

It is evident from recent real-life DDoS attack scenarios that there is an increasing intensity of modern attacks as discussed in Chapter 1. The largest DDoS to date is around 2.5Tbps of size, which is significantly large when compared to earlier attacks, which were in the range of Mbps. Thus, software-based solutions may not be adequate to handle high-intensity attacks. In addition, software-based solutions can easily be bypassed and targeted by the attacker.

On the other hand, hardware specific solutions are less susceptible to attacks. In addition, hardware solutions have proven to provide significantly faster classification rate leading to higher detection throughput when compared to software-based solutions. This led to the study of Chapter 4. In order to facilitate early detections, fast classification systems are required to cope with the massive network traffic of modern DDoS attacks. The proposed hybrid network monitoring and DDoS detection architecture implemented in a System-on-Chip (SoC) hardware device provides fast classification of network traffic flows. The detection solution is based on the machine-learning model proposed in Chapter 3.

The programmability of the FPGA offers great flexibility for easily retraining and deploying the detection model on the fly. The propose solution offers advantages such as fast classification (over 30 times faster than the software-based counterpart), low power consumption and flexibility in deployment. Furthermore, it is less expen-sive compared to other systems that exist in the state of the art with a detection accuracy of over 98%.

6.1.4 Distributed Monitoring for DDoS Flooding attacks detection

Traditional network monitoring and DDoS attacks detection solutions are often centralised meaning a single node is used in monitoring and detecting attacks. Additionally, the detection system is typically deployed at a strategic location in a network. However, early detections are required in effective mitigating and preventing mod-ern attacks. To facilitate early detection of DDoS attacks, the use of multiple detection systems strategically deployed in a distributed approach across different network nodes is shown to be a much more preferable approach. This enables a distributed network load, which will be easier to manage compared to processing all network traffic at a single node.

Experiments from Chapter 5 evaluate the use of different detection approaches and their impact on the efficacy of the DDoS attacks. From the results obtained, the deployment of a centralised monitoring node at the core switch significantly reduced the effects of DDoS attacks for example, packet loss ratio was reduced from 8% without detection system to around 1.5% and reduced network link delays with detection time of around 70μs across the monitoring and detection systems deployed.

However, the use of detection systems distributed across different network nodes in the proposed network design resulted in 0% packet loss ratio and significantly low network link delays.

6.2 Future Work

The possible directions for future work related to the research field are highlighted in this section. The findings from this thesis clearly indicate that there is further work to be done to cope with the ever-increasing number of DDoS attacks, their sizes and complexities.

6.2.1 Distributed hardware detection systems in a real network environment

The proposed hybrid DDoS detection system presented in this thesis can be extended in a distributed monitoring scenario in a real network environment to evaluate the effectives of distributed over a centralised monitoring approach. In addition, the design can be modified through the use of efficient software and hardware partitioning approaches that will enable different variations of Decision-Tree algorithms to be ported to faster high computing platforms such as NetFPGA and NetFPGA Sume devices, which are capable of monitoring at faster network line rates. This could enable deployment at edge or core nodes of a network.

Furthermore, rapid design and deployment could be explored through the use of the Xilinx Aiengine (which comprises of a host of pre-modelled ML algorithms that can be tailored to solving specific problems which aids rapid design and implementation process) and Partial Reconfiguration (PR) for on the fly deployment of newly trained models without the need for restarting hardware network monitoring devices. This could aid in the dynamic deployment of IDS across various nodes in a network during run-time.

6.2.2 Extending the proposed detection solution to IoT Environments

There is evidence of successful DDoS attacks that have been initiated with IoT devices. In addition, the continuous increase in the number of IoT devices poses a security threat especially in the field of DDoS attacks, as these devices are not adequately secure leading to possible exploitations. The datasets used in conducting the experiments in this thesis did not consist of attack traffic from IoT devices. Therefore, possible future research will look into extending the proposed detection solution in an IoT environment.

6.2.3 Extension of the flow based predictive model in real network environment

Since early prediction of DDoS attacks provides an advantage for effective mitigation and prevention, potential research could be on the implementation of the proposed flow based predictive model as part of a detection system in a real network environment. This will

support detection systems in detecting DDoS attacks early, as this is paramount in the modern DDoS attacks landscape.

6.2.4 Attacks originating from wireless/mobile networks and their devices

Although extensive research has been carried out in the field of DDoS aatacks detection and prevention against wireless network infrastructures, there is still much research to be carried out regarding attacks originating from wireless/mobile networks and their devices. Potential research could look into the use of wireless and mobile devices as botnets for initiating DDoS attacks and how these attacks could be detected as early as possible to mitigate the potential damages that could be caused by this method of attacks. This is because there has been a rapid increase in the use of mobile and wireless devices over the years and these devices have become targets for executing cyber attacks.

6.3 Summary

In summary, this thesis has provided and overview into DDoS attacks detection systems, growth of attacks over the years and has contributed to the field of DDoS attack detection by proposing some useful approaches: (1) a lightweight detection architecture (DT3), (2) a hybrid accelerated hardware application for fast detection, and (3) a flow based predictor model to aid early detection. These proposed architectures can provide researchers a better insight in the use of robust feature selection techniques couple with machine learning models to develop accurate detection systems. In addition, it provides an insight in hardware acceleration of machine learning models for fast detection of attacks. Finally, it has evaluated the difference in performance between the use of a centralised monitoring approach and a distributed approach for early detection and mitigation of attacks.

References

[1] S. Yu, *Distributed denial of service attack and defense*. Springer, 2014.

[2] S. Rao and S. Rao, "Denial of service attacks and mitigation techniques: Real time implementation with detailed analysis," *This paper is from the SANS Institute Reading Room site*, 2011.

[3] M. De Donno, N. Dragoni, A. Giaretta, and A. Spognardi, "Analysis of ddos-capable iot malwares," in *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, IEEE, 2017, pp. 807–816.

[4] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in internet-of-things," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1250–1258, 2017.

[5] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*, IEEE, 2019, pp. 1–8.

[6] J. Nazario, "Ddos attack evolution," *Network Security*, vol. 2008, no. 7, pp. 7–10, 2008.

[7] A. Shield, *Threat landscape report–q1 2020*, 2020.

[8] M. Lesk, "The new front line: Estonia under cyberassault," *IEEE Security & Privacy*, vol. 5, no. 4, pp. 76–79, 2007.

[9] S. W. Korns and J. E. Kastenberg, "Georgia's cyber left hook," Army War College Carlisle Barracks Pa Strategic Studies Institute, Tech. Rep., 2009.

[10] A. Bhandari, A. Sangal, and K. Kumar, "Destination address entropy based detection and traceback approach against distributed denial of service attacks," *International Journal of Computer Network and Information Security*, vol. 7, no. 8, pp. 9–20, 2015.

[11] Cloudflare. (2020). Famous ddos attacks — the largest ddos attacks of all time, [Online]. Available: `https://www.cloudflare.com/en-gb/learning/ddos/famous-ddos-attacks/`. Accessed: 11-07-2020.

[12] L. Constantin, *Ddos attack against spamhaus was reportedly the largest in history*, 2013.

[13] W. Dean. (2014). New ddos attack breaks spamhaus records, [Online]. Available: https://www.techradar.com/news/internet/web/new-ddos-attack-breaks-spamhaus-records-1223956. Accessed: 11-07-2020.

[14] A. Studer and A. Perrig, "The coremelt attack," in *European Symposium on Research in Computer Security*, Springer, 2009, pp. 37–52.

[15] M. S. Kang, S. B. Lee, and V. D. Gligor, "The crossfire attack," in *2013 IEEE symposium on security and privacy*, IEEE, 2013, pp. 127–141.

[16] 3. Research Reports, "Global and united states ddos protection and mitigation market size, status and forecast 2021-2027," 2021. [Online]. Available: https://www.360researchreports.com/global-and-united-states-ddos-protection-and-mitigation-market-18627551, Accessed: 10-10-2021.

[17] Q. Knowledge Solutions, *Arbor networks is recognized as the 2017 market and technology leader in the global ddos mitigation market*, 2017.

[18] A. Rai and R. K. Challa, "Survey on recent ddos mitigation techniques and comparative analysis," in *2016 Second International Conference on Computational Intelligence & Communication Technology (CICT)*, IEEE, 2016, pp. 96–101.

[19] R. Kumar, P. Arun, and S. Selvakumar, "Distributed denial-of-service (ddos) threat in collaborative environment-a survey on ddos attack tools and traceback mechanisms," in *2009 IEEE International Advance Computing Conference*, IEEE, 2009, pp. 1275–1280.

[20] R. Thomas, B. Mark, T. Johnson, and J. Croall, "Netbouncer: Client-legitimacy-based high-performance ddos filtering," in *Proceedings DARPA Information Survivability Conference and Exposition*, IEEE, vol. 1, 2003, pp. 14–25.

[21] J. Mirkovic and P. Reiher, "D-ward: A source-end defense against flooding denial-of-service attacks," *IEEE transactions on Dependable and Secure Computing*, vol. 2, no. 3, pp. 216–232, 2005.

[22] V. Sekar, N. G. Duffield, O. Spatscheck, J. E. van der Merwe, and H. Zhang, "Large-scale automated ddos detection system.," in *USENIX Annual Technical Conference, General Track*, 2006, pp. 171–184.

[23] Y. Gu, A. McCallum, and D. Towsley, "Detecting anomalies in network traffic using maximum entropy estimation," in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, 2005, pp. 32–32.

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

[25] X. Ma and Y. Chen, "Ddos detection method based on chaos analysis of network traffic entropy," *IEEE Communications Letters*, vol. 18, no. 1, pp. 114–117, 2013.

[26] S. M. Mousavi and M. St-Hilaire, "Early detection of ddos attacks against sdn controllers," in *2015 International Conference on Computing, Networking and Communications (ICNC)*, IEEE, 2015, pp. 77–81.

[27] M. H. Bhuyan, D. Bhattacharyya, and J. K. Kalita, "E-ldat: A lightweight system for ddos flooding attack detection and ip traceback using extended entropy metric," *Security and Communication Networks*, vol. 9, no. 16, pp. 3251–3270, 2016.

[28] Fortinet. (2017). Ddos attack mitigation technologies demystified, [Online]. Available: `https://www.fortinet.com/content/dam/fortinet/assets/white-papers/DDoS-Attack-Mitigation-Demystified.pdf/`. Accessed: 10-10-2021.

[29] Microsoft. (2021). Azure ddos protection pricing, [Online]. Available: `https://azure.microsoft.com/en-gb/pricing/details/ddos-protection/`. Accessed: 10-10-2021.

[30] HSo. (2021). Fighting back against ddos attacks, [Online]. Available: `https://www.hso.co.uk/blog/it-challenges/improving-availability-and-uptime/ddos-protection/fighting-back-against-ddos-attacks`. Accessed: 10-10-2021.

[31] A. Koay, "Detecting high and low intensity distributed denial of service (ddos) attacks," PhD thesis, Victoria University of Wellington, 2019, pp. 1–36.

[32] P. Phaal, S. Panchen, and N. McKee, *Rfc3176: Inmon corporation's sflow: A method for monitoring traffic in switched and routed networks*, 2001.

[33] B. Claise, G. Sadasivan, V. Valluri, and M. Djernaes, "Cisco systems netflow services export version 9," 2004.

[34] A. Hussain, J. Heidemann, and C. Papadopoulos, "A framework for classifying denial of service attacks," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003, pp. 99–110.

[35] S. M. Specht and R. Lee, "Distributed denial of service: Taxonomies of attacks," *Tools, and Countermeasures*, pp. 543–550, 2004.

[36] Y. Shui, "An overview of ddos attacks," *Distributed Denial of Service Attack and Defense*, pp. 1–14, 2014.

[37] M. Fabian and M. A. Terzis, "My botnet is bigger than yours (maybe, better than yours): Why size estimates remain challenging," in *Proceedings of the 1st USENIX Workshop on Hot Topics in Understanding Botnets, Cambridge, USA*, vol. 18, 2007.

[38] L. Garber, "Denial-of-service attacks rip the internet," *Computer*, vol. 33, no. 4, pp. 12–17, 2000.

[39] HWH. (2018). Ddos attack check, [Online]. Available: `https://hyderabadwebhosting.in/blog/ddos-attack-check/`. Accessed: 10-05-2021.

[40] B. B. Gupta, R. C. Joshi, and M. Misra, "Defending against distributed denial of service attacks: Issues and challenges," *Information Security Journal: A Global Perspective*, vol. 18, no. 5, pp. 224–247, 2009.

[41] A. Srivastava, B. Gupta, A. Tyagi, A. Sharma, and A. Mishra, "A recent survey on ddos attacks and defense mechanisms," in *International Conference on Parallel Distributed Computing Technologies and Applications*, Springer, 2011, pp. 570–580.

[42] R. Sujan. (2021). Tcp 3-way handshake process, [Online]. Available: `https://www.geeksforgeeks.org/tcp-3-way-handshake-process/`. Accessed: 27-10-2021.

[43] V. Paxson, "An analysis of using reflectors for distributed denial-of-service attacks," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 3, pp. 38–47, 2001.

[44] S. Kumar, "Smurf-based distributed denial of service (ddos) attack amplification in internet," in *Second International Conference on Internet Monitoring and Protection (ICIMP 2007)*, IEEE, 2007, pp. 25–25.

[45] N. Muraleedharan and B. Janet, "Flow-based machine learning approach for slow http distributed denial of service attack classification," *International Journal of Computational Science and Engineering*, vol. 24, no. 2, pp. 147–161, 2021.

[46] C. Zhang, Z. Cai, W. Chen, X. Luo, and J. Yin, "Flow level detection and filtering of low-rate ddos," *Computer Networks*, vol. 56, no. 15, pp. 3417–3431, 2012.

[47] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna, "Your botnet is my botnet: Analysis of a botnet takeover," in *Proceedings of the 16th ACM conference on Computer and communications security*, 2009, pp. 635–647.

[48] J. Oikarinen and D. Reed, "Internet relay chat protocol," 1993.

[49] E. Cooke, F. Jahanian, and D. McPherson, "The zombie roundup: Understanding, detecting, and disrupting botnets.," *SRUTI*, vol. 5, pp. 6–6, 2005.

[50] C. Li, W. Jiang, and X. Zou, "Botnet: Survey and case study," in *2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC)*, IEEE, 2009, pp. 1184–1187.

[51] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon, "Peer-to-peer botnets: Overview and case study.," *HotBots*, vol. 7, no. 2007, 2007.

[52] P. Criscuolo, "Distributed denial of service tools, trin00, tribe flood network, tribe flood network 2000 and stacheldraht.," Lawrence Livermore National Lab., CA (US), Tech. Rep., 2000.

[53] J. Roman, B. Radek, V. Radek, and S. Libor, "Launching distributed denial of service attacks by network protocol exploitation," in *Proceedings of the 2nd international conference on Applied informatics and computing theory*, 2011, pp. 210–216.

[54] D. Dittrich, G. Weaver, S. Dietrich, and N. Long, *The mstream distributed denial of service attack tool*, 2000.

[55] A. Schulter, J. A. Reis, F. Koch, and C. B. Westphall, "A grid-based intrusion detection system," in *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06)*, IEEE, 2006, pp. 187–187.

[56] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A survey of intrusion detection techniques in cloud," *Journal of network and computer applications*, vol. 36, no. 1, pp. 42–57, 2013.

[57] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[58] A. Alazab, M. Hobbs, J. Abawajy, and M. Alazab, "Using feature selection for intrusion detection system," in *2012 international symposium on communications and information technologies (ISCIT)*, IEEE, 2012, pp. 296–301.

[59] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *computers & security*, vol. 28, no. 1-2, pp. 18–28, 2009.

[60] E. Fenil and P. Mohan Kumar, "Survey on ddos defense mechanisms," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 4, e5114, 2020.

[61] W. Eddy *et al.*, "Tcp syn flooding attacks and common mitigations," RFC 4987, August, Tech. Rep., 2007.

[62] C. V. Zhou, C. Leckie, and S. Karunasekera, "A survey of coordinated attacks and collaborative intrusion detection," *Computers & Security*, vol. 29, no. 1, pp. 124–140, 2010.

[63] T. Peng, C. Leckie, and K. Ramamohanarao, "Detecting distributed denial of service attacks by sharing distributed beliefs," in *Australasian Conference on Information Security and Privacy*, Springer, 2003, pp. 214–225.

[64] S. Mansfield-Devine, "The growth and evolution of ddos," *Network Security*, vol. 2015, no. 10, pp. 13–20, 2015.

[65] M. Prince, *The ddos that almost broke the internet*, 2013.

[66] R. Ltd. (2016). Protonmail ddos case study: Ddos prevention techniques, [Online]. Available: `https://www.radware.com/dfe06851-f81b-477b-b78b-5d1ca0f2cafc`. Accessed: 10-05-2021.

[67] S. Khandelwal, *602 gbps! this may have been the largest ddos attack in history. the hacker news, jan 8*, 2016.

[68] J. Scott Sr and W. Summit. (2016). Rise of the machines: The dyn attack was just a practice run december 2016, [Online]. Available: `https://icitech.%20org/icit-publication-the-rise-of-the-machines-the-dynattack-was-just-a-practice-run`. Accessed: 10-05-2021.

[69] G. Slocombe, "World's largest publicly revealed distributed denial of service attack," *Asia-Pacific Defence Reporter (2002)*, vol. 44, no. 3, pp. 30–31, 2018.

[70] C. Cimpanu. (2020). Aws said it mitigated a 2.3 tbps ddos attack, the largest ever, [Online]. Available: `https://www.zdnet.com/article/aws-said-it-mitigated-a-2-3-tbps-ddos-attack-the-largest-ever/`. Accessed: 10-05-2018.

[71] Y. Orzach, *Network Analysis Using Wireshark Cookbook*. Packt Publishing Ltd, 2013.

[72] A. Singh, *Instant Wireshark Starter*. Packt Publishing Ltd, 2013.

[73] B. Merino, *Instant traffic analysis with Tshark how-to*. Packt Publishing Ltd, 2013.

[74] S. Bansal and N. Bansal, "Scapy-a python tool for security testing," *Journal of Computer Science & Systems Biology*, vol. 8, no. 3, p. 140, 2015.

[75] S. Ayesha, M. K. Hanif, and R. Talib, "Overview and comparative study of dimensionality reduction techniques for high dimensional data," *Information Fusion*, vol. 59, pp. 44–58, 2020.

[76] S. Pulkit. (2018). The ultimate guide to 12 dimensionality reduction techniques (with python codes), [Online]. Available: `https://www.analyticsvidhya.com/blog/2018/08/dimensionality-reduction-techniques-python/`.

[77] S. Rosaria, A. Iris, H. Aaron, and M. Berthold, *Seven Techniques for Dimensionality Reduction*. KNIME, 2014.

[78] G. Bonaccorso, *Machine learning algorithms*. Packt Publishing Ltd, 2017.

[79] T. M. Mitchell *et al.*, *Machine learning*. McGraw-hill New York, 1997.

[80] J. Frank, "Artificial intelligence and intrusion detection: Current and future directions," in *Proceedings of the 17th national computer security conference*, Baltimore, MD, vol. 10, 1994, pp. 1–12.

[81] Y. Gu, Y. Shi, and J. Wang, "Efficient intrusion detection based on multiple neural network classifiers with improved genetic algorithm.," *JSW*, vol. 7, no. 7, pp. 1641–1648, 2012.

[82] M. Alkasassbeh, G. Al-Naymat, A. Hassanat, and M. Almseidin, "Detecting distributed denial of service attacks using data mining techniques," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 1, pp. 436–445, 2016.

[83] J. Yu, H. Lee, M.-S. Kim, and D. Park, "Traffic flooding attack detection with snmp mib using svm," *Computer Communications*, vol. 31, no. 17, pp. 4212–4219, 2008.

[84] K. Kato and V. Klyuev, "An intelligent ddos attack detection system using packet analysis and support vector machine," *IJICR*, vol. 14, no. 5, p. 3, 2014.

[85] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for internet of things," *Future Generation Computer Systems*, vol. 82, pp. 761–768, 2018.

[86] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, IEEE, 2009, pp. 1–6.

[87] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita, "Ffsc: A novel measure for low-rate and high-rate ddos attack detection using multivariate data analysis," *Security and Communication Networks*, vol. 9, no. 13, pp. 2032–2041, 2016.

[88] M. Barati, A. Abdullah, N. I. Udzir, R. Mahmod, and N. Mustapha, "Distributed denial of service detection using hybrid machine learning technique," in *2014 International Symposium on Biometrics and Security Technologies (ISBAST)*, IEEE, 2014, pp. 268–273.

[89] Netmate. (2019). Netmate homepage, [Online]. Available: `https://f00l.de/netmate/`. Accessed: 03-09-2019.

[90] A. Saied, R. E. Overill, and T. Radzik, "Detection of known and unknown ddos attacks using artificial neural networks," *Neurocomputing*, vol. 172, pp. 385–393, 2016.

[91] J. Li, Y. Liu, and L. Gu, "Ddos attack detection based on neural network," in *2010 2nd international symposium on aware computing*, IEEE, 2010, pp. 196–199.

[92] S.-J. Horng, M.-Y. Su, Y.-H. Chen, T.-W. Kao, R.-J. Chen, J.-L. Lai, and C. D. Perkasa, "A novel intrusion detection system based on hierarchical clustering and support vector machines," *Expert systems with Applications*, vol. 38, no. 1, pp. 306–313, 2011.

[93] A. Ramamoorthi, T. Subbulakshmi, and S. M. Shalinie, "Real time detection and classification of ddos attacks using enhanced svm with string kernels," in *2011 International conference on recent trends in information technology (ICRTIT)*, IEEE, 2011, pp. 91–96.

[94] Y.-C. Wu, H.-R. Tseng, W. Yang, and R.-H. Jan, "Ddos detection and traceback with decision tree and grey relational analysis," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 7, no. 2, pp. 121–136, 2011.

[95] H. Ian. (2015). Introduction to the internet of things and embedded systems, [Online]. Available: `https://www.coursera.org/learn/iot`. Accessed: 10-05-2021.

[96] J. Ganssle, *Embedded systems dictionary*. CRC Press, 2003.

[97] S. Kiourkoulis, *Ddos datasets: Use of machine learning to analyse intrusion detection performance*, 2020.

[98] C. I. for Cybersecurity. (2017). Ddos evaluation dataset (cicids2017), [Online]. Available: `https://www.unb.ca/cic/datasets/ids-2017.html`. Accessed: 27-10-2021.

[99] U. of New Brunswick. (2019). Ddos evaluation dataset (cicddos2019), [Online]. Available: `https://www.unb.ca/cic/datasets/ddos-2019.html`. Accessed: 27-10-2021.

[100] CAIDA. (2007). The caida ucsd ddos attack 2007 dataset, [Online]. Available: `http://www.caida.org/data/passive/ddos-20070804%5C_dataset.xml`. Accessed: 15-11-2017.

[101] A. Lashkari, Y. Zang, G. Owhuo, M. Mamun, and G. Gil. (2013). Cicflowmeter (formerly iscxflowmeter)—a network traffic flow analyzer, [Online]. Available: `http://www.netflowmeter.ca/netflowmeter.html`. Accessed: 10-05-2018.

[102] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," in *ICISSP*, 2018, pp. 108–116.

[103] R. Doriguzzi-Corinα, S. Millarβ, S. Scott-Haywardβ, J. Martınez-del-Rincónβ, and D. Siracusaα, "Lucid: A practical, lightweight deep learning solution for ddos attack detection," 2019.

[104] M. J. Zaki and W. Meira, *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press, 2014.

[105] S. Velliangiri, S. Alagumuthukrishnan, *et al.*, "A review of dimensionality reduction techniques for efficient computation," *Procedia Computer Science*, vol. 165, pp. 104–111, 2019.

[106] D. Mladenić, "Feature selection for dimensionality reduction," in *International Statistical and Optimization Perspectives Workshop" Subspace, Latent Structure and Feature Selection"*, Springer, 2005, pp. 84–102.

[107] N. Sharma and K. Saroha, "Study of dimension reduction methodologies in data mining," in *International Conference on Computing, Communication & Automation*, IEEE, 2015, pp. 133–137.

[108] E. Balkanli, A. N. Zincir-Heywood, and M. I. Heywood, "Feature selection for robust backscatter ddos detection," in *2015 IEEE 40th Local Computer Networks Conference Workshops (LCN Workshops)*, IEEE, 2015, pp. 611–618.

[109] J. L. Berral, N. Poggi, J. Alonso, R. Gavalda, J. Torres, and M. Parashar, "Adaptive distributed mechanism against flooding network attacks based on machine learning," in *Proceedings of the 1st ACM workshop on Workshop on AISec*, 2008, pp. 43–50.

[110] B. K. Devi, G. Preetha, G. Selvaram, and S. M. Shalinie, "An impact analysis: Real time ddos attack detection and mitigation using machine learning," in *2014 International Conference on Recent Trends in Information Technology*, IEEE, 2014, pp. 1–7.

[111] S. Hosseini and M. Azizi, "The hybrid technique for ddos detection with supervised learning algorithms," *Computer Networks*, vol. 158, pp. 35–45, 2019.

[112] C.-J. Hsieh and T.-Y. Chan, "Detection ddos attacks based on neural-network using apache spark," in *2016 international conference on applied system innovation (ICASI)*, IEEE, 2016, pp. 1–4.

[113] O. Osanaiye, H. Cai, K.-K. R. Choo, A. Dehghantanha, Z. Xu, and M. Dlodlo, "Ensemble-based multi-filter feature selection method for ddos detection in cloud computing," *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, no. 1, pp. 1–10, 2016.

[114] G. Ramadhan, Y. Kurniawan, and C.-S. Kim, "Design of tcp syn flood ddos attack detection using artificial immune systems," in *2016 6th International Conference on System Engineering and Technology (ICSET)*, IEEE, 2016, pp. 72–76.

[115] M. Suresh and R. Anitha, "Evaluating machine learning algorithms for detecting ddos attacks," in *International Conference on Network Security and Applications*, Springer, 2011, pp. 441–452.

[116] C. Wang, J. Zheng, and X. Li, "Research on ddos attacks detection based on rdf-svm," in *2017 10th International Conference on Intelligent Computation Technology and Automation (ICICTA)*, IEEE, 2017, pp. 161–165.

[117] A. R. Yusof, N. I. Udzir, A. Selamat, H. Hamdan, and M. T. Abdullah, "Adaptive feature selection for denial of services (dos) attack," in *2017 IEEE Conference on Application, Information and Network Security (AINS)*, IEEE, 2017, pp. 81–84.

[118] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *The journal of machine learning research*, vol. 15, no. 1, pp. 3133–3181, 2014.

[119] R. Timofeev, "Classification and regression trees (cart) theory and applications," *Humboldt University, Berlin*, 2004.

[120] D. Steinberg and P. Colla, "Cart: Classification and regression trees," *The top ten algorithms in data mining*, vol. 9, p. 179, 2009.

[121] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[122] Y. Freund and L. Mason, "The alternating decision tree learning algorithm," in *icml*, Citeseer, vol. 99, 1999, pp. 124–133.

[123] D. E. Taylor, "Survey and taxonomy of packet classification techniques," *ACM Computing Surveys (CSUR)*, vol. 37, no. 3, pp. 238–275, 2005.

[124] A. D. Joseph, P. Laskov, F. Roli, J. D. Tygar, and B. Nelson, "Machine learning methods for computer security (dagstuhl perspectives workshop 12371)," in *Dagstuhl Manifestos*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, vol. 3, 2013.

[125] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[126] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[127] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[128] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[129] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[130] V. Selis, "Establishing trusted machine-to-machine communications in the internet of things through the use of behavioural tests," PhD thesis, University of Liverpool, 2017.

[131] M. J. Orr *et al.*, *Introduction to radial basis function networks*, 1996.

[132] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments," *Computer Networks*, vol. 62, pp. 122–136, 2014.

[133] A. Veselỳ and D. Brechlerova, "Neural networks in intrusion detection systems," *Agriculturejournals. cz*, pp. 156–165, 2009.

[134] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[135] M. Sheng, W. Hou, and J. Jiang, "Implementation of accelerating video preprocessing based on zynq platform resource management," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1544, 2020, p. 012 112.

[136] A. Alhamwi, "Co-design hardware/software of real time vision system on fpga for obstacle detection," PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier, 2016.

[137] D. B. Thomas, L. Howes, and W. Luk, "A comparison of cpus, gpus, fpgas, and massively parallel processor arrays for random number generation," in *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*, 2009, pp. 63–72.

[138] J. Fung and S. Mann, "Using multiple graphics cards as a general purpose parallel computer: Applications to computer vision," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, IEEE, vol. 1, 2004, pp. 805–808.

[139] C. Adams, "Fpga or gpu?-the evolution continues," *Military Embedded Systems*, vol. 10, no. 6, p. 16, 2014.

[140] H. B. Kommuru and H. Mahmoodi, "Asic design flow tutorial using synopsys tools," *Nano-Electronics & Computing Research Lab, School of Engineering, San Francisco State University San Francisco, CA, Spring*, 2009.

[141] ARM. (2016). Arm cortex-a9 processor, [Online]. Available: `http://www.arm.com/cortex-a9.php`. Accessed: 25-06-2021.

[142] C. Maxfield, *The design warrior's guide to FPGAs: devices, tools and flows.* Elsevier, 2004.

[143] H. H. Schmit, S. Cadambi, M. Moe, and S. C. Goldstein, "Pipeline reconfigurable fpgas," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 24, no. 2, pp. 129–146, 2000.

[144] G. W. Morris, D. B. Thomas, and W. Luk, "Fpga accelerated low-latency market data feed processing," in *2009 17th IEEE Symposium on High Performance Interconnects*, IEEE, 2009, pp. 83–89.

[145] L. Semiconductor, "High-speed serdes interfaces in high value fpgas," *White Paper, Lattice Semiconductor Corp*, 2009.

[146] K. Parnell and R. Bryner, "Comparing and contrasting fpga and microprocessor system design and development," *WP213*, vol. 1, no. 1, pp. 1–32, 2004.

[147] G. Lucky, F. Jjunju, and A. Marshall, "A lightweight decision-tree algorithm for detecting ddos flooding attacks," in *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, IEEE, 2020, pp. 382–389.

[148] M. A. Vega-Rodríguez, J. M. Sánchez-Pérez, and J. A. Gómez-Pulido, "An fpga-based implementation for median filter meeting the real-time requirements of automated visual inspection systems," in *Proc. 10th Mediterranean Conf. Control and Automation*, 2002.

[149] M. A. Altuncu, T. Guven, Y. Becerikli, and S. Sahin, "Real-time system implementation for image processing with hardware/software co-design on the xilinx zynq platform," *International Journal of Information and Electronics Engineering*, vol. 5, no. 6, p. 473, 2015.

[150] M. Kiran, K. M. War, L. M. Kuan, L. K. Meng, and L. W. Kin, "Implementing image processing algorithms using 'hardware in the loop'approach for xilinx fpga," in *2008 International Conference on Electronic Design*, IEEE, 2008, pp. 1–6.

[151] Xilinx. (2013). Zedboard, [Online]. Available: `https://www.xilinx.com/products/boards-and-kits/1-8dyf-11.html`. Accessed: 20-07-2017.

[152] V. Xilinx, *Vivado design suite user guide-high-level synthesis*, 2017.

[153] Xillybus. (2010). Xillybus: Principle of operation, [Online]. Available: `http://xillybus.com/doc/xilinx-pcie-principle-of-operation`. Accessed: 05-03-2019.

[154] N. Hoque, H. Kashyap, and D. K. Bhattacharyya, "Real-time ddos attack detection using fpga," *Computer Communications*, vol. 110, pp. 48–58, 2017.

[155] B. Nagy, P. Orosz, T. Tóthfalusi, L. Kovács, and P. Varga, "Detecting ddos attacks within milliseconds by using fpga-based hardware acceleration," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, IEEE, 2018, pp. 1–4.

[156] B. Nagy, P. Orosz, and P. Varga, "Low-reaction time fpga-based ddos detector," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, IEEE, 2018, pp. 1–2.

[157] J. W. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, and J. Luo, "Netfpga–an open platform for gigabit-rate network switching and routing," in *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)*, IEEE, 2007, pp. 160–161.

[158] N. Zilberman, Y. Audzevich, G. A. Covington, and A. W. Moore, "Netfpga sume: Toward 100 gbps as research commodity," *IEEE micro*, vol. 34, no. 5, pp. 32–41, 2014.

[159] A. Balyk, M. Karpinski, A. Naglik, G. Shangytbayeva, and I. Romanets, "Using graphic network simulator 3 for ddos attacks simulation," *International Journal of Computing*, vol. 16, no. 4, pp. 219–225, 2017.

[160] L. Campanile, M. Gribaudo, M. Iacono, F. Marulli, and M. Mastroianni, "Computer network simulation with ns-3: A systematic literature review," *Electronics*, vol. 9, no. 2, p. 272, 2020.

[161] NS3. (2021). Research, [Online]. Available: `https://www.nsnam.org/research/`. Accessed: 01-10-2021.

[162] S. Khanvilkar, F. Bashir, D. Schonfeld, and A. Khokhar, *Multimedia networks and communication*, 2004.

[163] C. Inc. (2012). Cisco; cisco network hardware news and technology, [Online]. Available: `http://ciscorouterswitch.over-blog.com/article-what-makes-lan-different-from-wan-100014883.htm`. Accessed: 10-05-2021.

[164] I. Telchemy, "Impact of delay in voice over ip services," *Telchemy Application Notes, Series VoIP Performance Management*, vol. 1, no. 7, 2006.

[165] X. Xiao, *Technical, commercial and regulatory challenges of QoS: An internet service model perspective*. Morgan Kaufmann, 2008.

Appendix A

Analysis of CAIDA 2007 DDoS Attacks Dataset

A.1 CAIDA 2007 Dataset Analysis

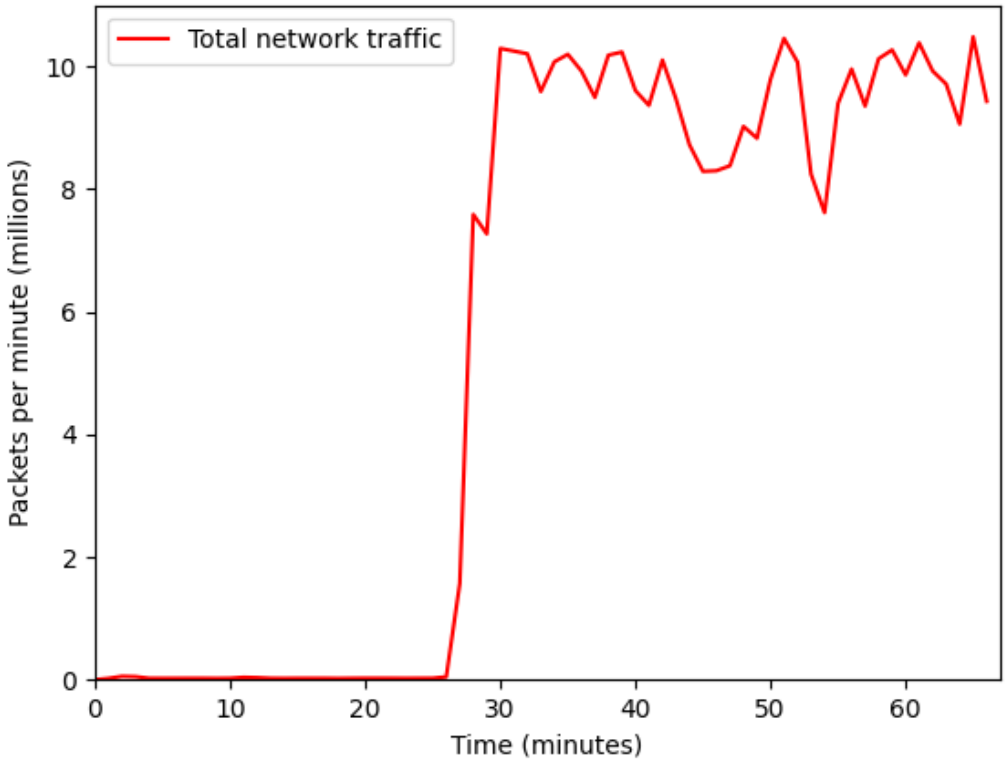

FIGURE A.1: Overall network traffic variation in the dataset

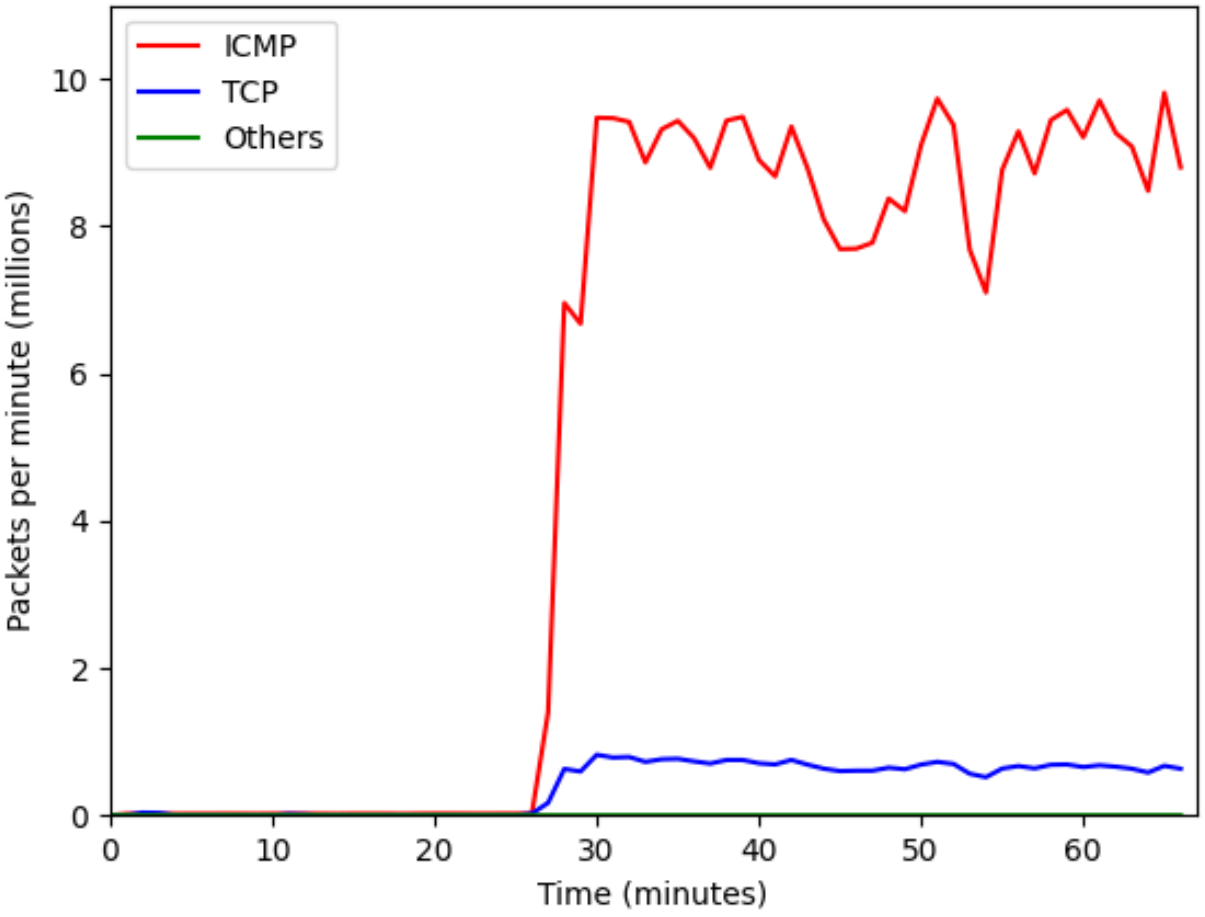

FIGURE A.2: Network traffic variation based on network protocols

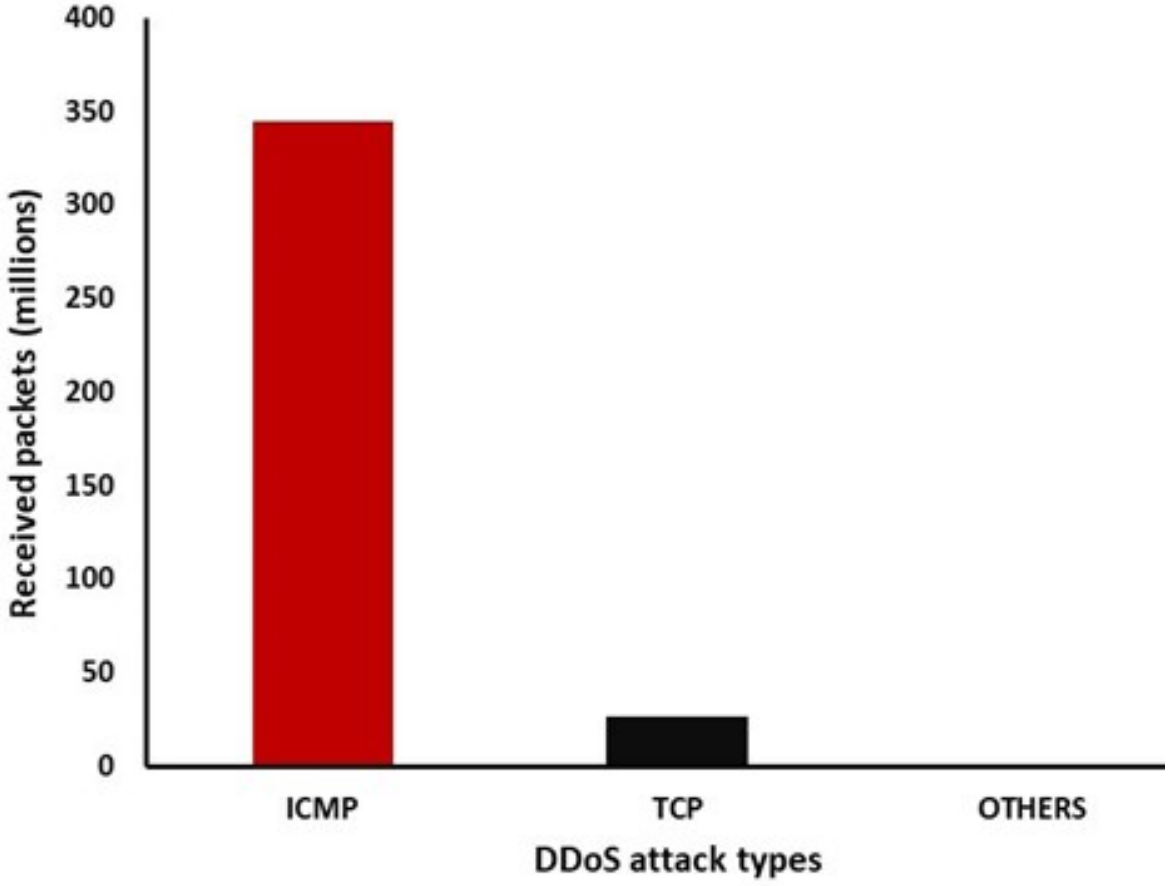

FIGURE A.3: Number of attack packets in the dataset based on network protocol

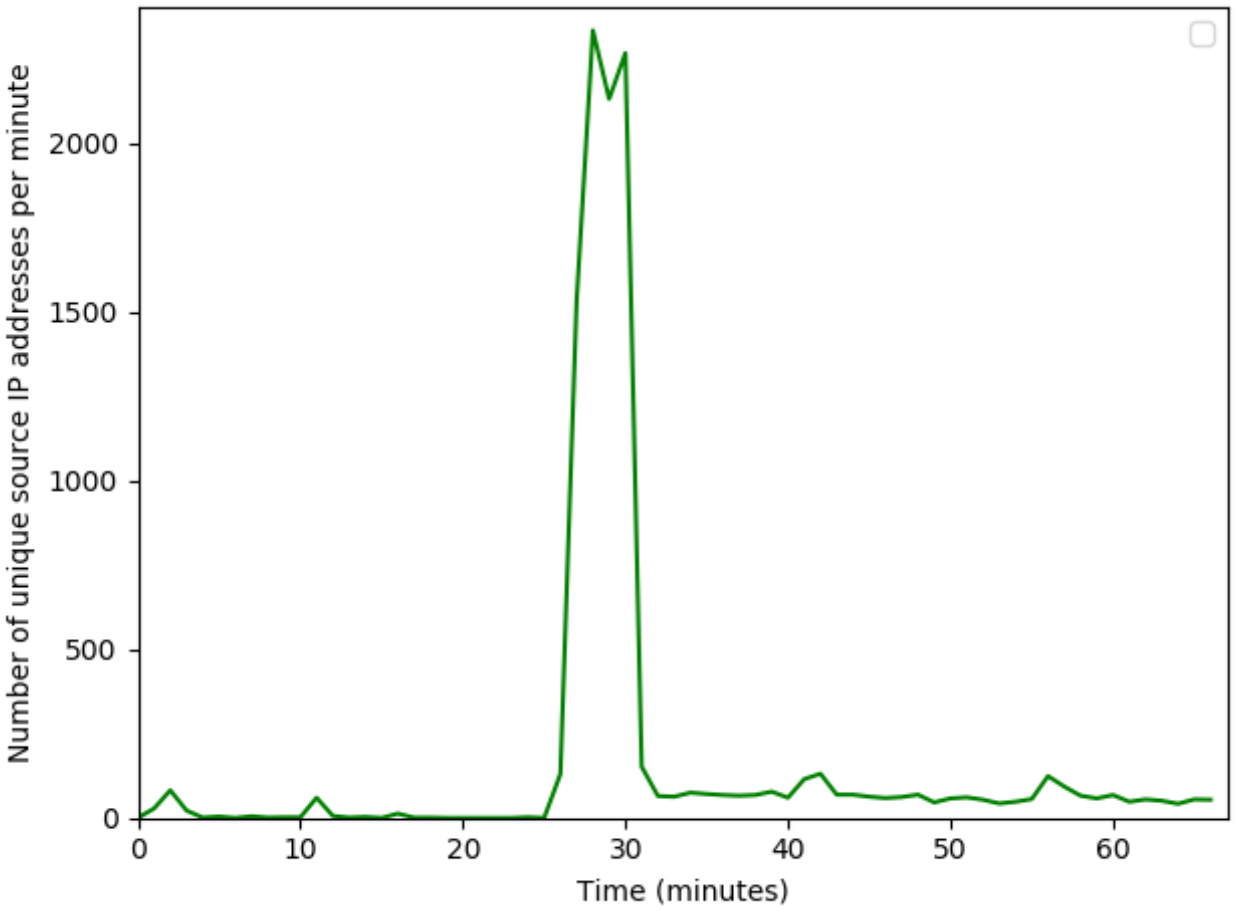

FIGURE A.4: Number of unique source IP addresses over the traffic duration

Appendix B

Design, Implementation and Performance of ML Classifiers

B.1 DT3 Construction

Input: Suppose D is the supplied sample dataset **Output:** Decision-Tree – DT model **Features:** *A* represents the selected features of the DT which is determined by applying the Low Variance Filter technique described in Chapter 3.

Procedures:

1. Initialize all the data values in the dataset D, calculate the initial entropy of D using:

$$Entropy(D) = \sum_{i=1}^{N} -p_i log_2 p_i \tag{B.1}$$

where p_i is the probability of picking a flow record in class i from D, v is the number of class labels (benign or attack flow). If all the samples in a dataset are from a single class label, v=1, then $p_i = 1$ and the entropy will be zero. However, if each sample has its own class label, $p_i = 1/v$ and the entropy will be at its largest when Entropy (D)=log_2 v. In this application, v =2 (two class labels, benign or attack) and the dataset is balanced (equal number of records for each class label)

2. The split entropy of the sample dataset D is calculated. Suppose that a feature partitions the sample dataset D into D_L, D_T and D_R. The split entropy is calculated as the sum of the entropy values for all subsets as shown below:

$$\begin{aligned} Entropy_F(D) = \frac{D_L}{D} Entropy(D_L) + \\ \frac{D_T}{D} Entropy(D_T) + \frac{D_R}{D} Entropy(D_R) \end{aligned} \tag{B.2}$$

where F is any feature in A, D_L, D_T and D_R are subsets from D with a split criteria of F. $|D|$ is the total number of samples in the sample dataset D. D_L, D_T and D_R are the total number of samples in D_L, D_T and D_R respectively

3. Calculate the information gain of feature F. The information gain is used to determine if a selected feature F can effectively reduce the overall entropy using:

$$Gain(F) = Entropy(D) - Entropy_F(D) \tag{B.3}$$

With this, the higher the information gain, the larger the reduction in entropy and this means the better the attribute for splitting the samples. ID3 algorithm uses the information gain in splitting samples in a dataset for the construction of the DT models. However, it is prone to overfitting as it does this by assigning the different classification results to each sample [125].

4. Determine the information gain ratio. To overcome the problem of overfitting faced by ID3 algorithm, C4.5 algorithm uses a split information value which is calculated using:

$$SplitInfo(F) = -\sum_{i=1}^{k} \frac{|D_i|}{|D|} log_2 \frac{|D_i|}{|D|} \tag{B.4}$$

For feature F in A, the information gain ratio can be calculated using:

$$GainRatio(F) = \frac{InformationGain(F)}{SplitInfo(F)} \tag{B.5}$$

The information gain ratio of each node (from the root to the terminal nodes) of the DT model needs to be calculated. The features that achieve the maximum information gain ratio at each stage are selected and stored at the corresponding node. This process is recursively executed for subsets split with multiple classes from the sample dataset D until a leaf node is reached where all samples belong to a single class (a 'pure' subset). The paths from one node to the other are the classification rules. The DT model uses these classification rules by comparing the sample data values to the different rules to determine the class label (either benign or attack). Once the construction of the DT model is complete, the process terminates.

The optimized DT model after pruning consists a minimum of thirty samples per leaf node with a tree depth of 4. Smaller threshold had minimum impact on the performance with increase in complexity of the model whereas larger threshold values led to reduction in the DT model accurately distinguishing normal from attack records.

B.2 Performance of ML Models with Chi-squared Feature Selection Technique

B.2.1 Performance with three selected Features

Table B.1: DT performance results on distinguishing DDoS attacks and benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	0.96	0.7	0.81	0.8614
DDoS	0.81	0.98	0.89	
Average	0.88	0.86	0.86	

Table B.2: RF performance results on distinguishing DDoS attacks and benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	0.99	0.7	0.82	0.8690
DDoS	0.82	0.99	0.90	
Average	0.89	0.87	0.86	

Table B.3: MLP performance results on distinguishing DDoS attacks and benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	0.99	0.42	0.59	0.7482
DDoS	0.69	1.00	0.82	
Average	0.82	0.75	0.72	

Table B.4: SVM performance results on distinguishing DDoS attacks and benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	1.00	0.51	0.67	0.7873
DDoS	0.73	1.00	0.84	
Average	0.84	0.79	0.77	

B.2.2 Performance with five selected Features

Table B.5: DT performance results on distinguishing DDoS attacks and benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	1.00	0.86	0.92	0.9390
DDoS	0.9	1.00	0.95	
Average	0.94	0.94	0.94	

Table B.6: RF performance results on distinguishing DDoS attacks and benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	1.00	0.86	0.92	0.9392
DDoS	0.9	1.00	0.95	
Average	0.94	0.94	0.94	

Table B.7: MLP performance results on distinguishing DDoS attacks and benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	1.00	0.50	0.67	0.7855
DDoS	0.73	1.00	0.84	
Average	0.84	0.79	0.77	

Table B.8: SVM performance results on distinguishing DDoS attacks and benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	1.00	0.47	0.64	0.7699
DDoS	0.71	1.00	0.83	
Average	0.84	0.77	0.75	

B.2.3 Performance with seven selected Features

Table B.9: DT performance results on distinguishing DDoS attacks from benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	1.00	1.00	1.00	0.9984
DDoS	1.00	1.00	1.00	
Average	1.00	1.00	1.00	

Table B.10: RF performance results on distinguishing DDoS attacks from benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	1.00	1.00	1.00	0.9987
DDoS	1.00	1.00	1.00	
Average	1.00	1.00	1.00	

Table B.11: MLP performance results on distinguishing DDoS attacks from benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	1.00	0.81	0.90	0.9183
DDoS	0.87	1.00	0.93	
Average	0.93	0.92	0.92	

Table B.12: SVM performance results on distinguishing DDoS attacks and benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	1.00	0.71	0.83	0.87325
DDoS	0.82	1.00	0.90	
Average	0.90	0.87	0.87	

B.3 Performance of ML Models with BFE and FFS Feature Selection Techniques

B.3.1 Performance with three selected Features

Table B.13: DT performance results on distinguishing DDoS attacks and benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	1.00	0.87	0.93	0.94153
DDoS	0.91	1.00	0.95	
Average	0.95	0.94	0.94	

Table B.14: RF performance results on distinguishing DDoS attacks and benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	1.00	0.87	0.93	0.94153
DDoS	0.91	1.00	0.95	
Average	0.95	0.94	0.94	

Table B.15: MLP performance results on distinguishing DDoS attacks and benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	1.00	0.85	0.92	0.93543
DDoS	0.90	1.00	0.95	
Average	0.94	0.94	0.93	

Table B.16: SVM performance results on distinguishing DDoS attacks and benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	1.00	0.87	0.93	0.94153
DDoS	0.91	1.00	0.95	
Average	0.95	0.94	0.94	

B.3.2 Performance with five selected Features

Table B.17: DT performance results on distinguishing DDoS attacks and benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	1.00	0.89	0.94	0.95381
DDoS	0.93	1.00	0.96	
Average	0.96	0.95	0.95	

Table B.18: RF performance results on distinguishing DDoS attacks and benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	1.00	0.89	0.94	0.95381
DDoS	0.93	1.00	0.96	
Average	0.96	0.95	0.95	

Table B.19: MLP performance results on distinguishing DDoS attacks and benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	1.00	0.88	0.94	0.94939
DDoS	0.92	1.00	0.96	
Average	0.95	0.95	0.95	

Table B.20: SVM performance results on distinguishing DDoS attacks and benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	0.99	0.89	0.94	0.94938
DDoS	0.92	0.99	0.96	
Average	0.95	0.95	0.95	

B.3.3 Performance with seven selected Features

Table B.21: DT performance results on distinguishing DDoS attacks from benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	1.00	0.90	0.95	0.95873
DDoS	0.93	1.00	0.97	
Average	0.96	0.96	0.96	

Table B.22: RF performance results on distinguishing DDoS attacks from benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	1.00	0.90	0.95	0.95873
DDoS	0.93	1.00	0.97	
Average	0.96	0.96	0.96	

Table B.23: MLP performance results on distinguishing DDoS attacks from benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	1.00	0.90	0.95	0.95800
DDoS	0.93	1.00	0.96	
Average	0.96	0.96	0.96	

Table B.24: SVM performance results on distinguishing DDoS attacks and benign network traffic flows

	Precision	**Sensitivity**	**F-measure**	**Accuracy**
Benign	0.99	0.90	0.95	0.95485
DDoS	0.93	0.99	0.96	
Average	0.96	0.95	0.95	

B.4 Real-time Monitoring and Network Flow Pre-processing Application

```
# Packet Capture and Pre-processor Application using Scapy
import socket
from struct import *
import datetime,time
import sys,math
import subprocess,os
from scapy.all import *
import csv

main_packet_size = {}           # dictionary to store list

#of packet sizes for each flow (Here key = flowID, value = list of packet
    sizes)
flow_list = []                  # contains the flowIDs (a combination of SIP,
    DIP,
#srcPort, dstPort)
main_inter_arrival_time = {}    # dictionary to store list of IATs for each
    flow
#(Here key = flowID, value = list of IATs)
last_time={}                    # for each flow we store timestamp of the
#last packet arrival
length = 0                      # Number of packets in a flow
avg_pkt_size = 0                # average packet size of a flow
std_dev_pkt_size = 0            # standard deviation of packet sizes in a
    flow
avg_iat = 0                     # average IAT of a flow
std_dev_IAT = 0                 # standard deviation of IAT in a flow
pktsize = 0                     # packet size
total_bytes = 0                 # total bytes in a flow
flow_duration = 0               # Total flow duration
minPacket = 30
var_pkt_size = 0                # variance of packet sizes of a flow
proto = ""                      # whether a packet/flow is ICMP, UDP or TCP
start_time  = time.time()       # initial time


def sniffPackets(packet):
    if packet.haslayer(IP):
        pckt_src=packet[IP].src
        pckt_dst=packet[IP].dst
        pckt_ttl=packet[IP].ttl
        pckt_proto=packet[IP].proto
        time1  = time.time() - start_time
        pktsize =packet[IP].len

        # Filtering only the required traffic bassed on Protocol
        # and destination IP address
```

```
        if pckt_proto==1 or pckt_proto==6 or pckt_proto==17
        and pckt_dst=="192.168.32.139":

            # Three Tuple for classifying network packets to flows
            key = str(pckt_src)+"|"+ str(pckt_dst)+"|"+str(pckt_proto)
            print(key)

            if (key in flow_list):

                main_packet_size[key].append(pktsize)
                lasttime = last_time[key]
                diff = round(float(time1) - float(lasttime),8)
                main_inter_arrival_time[key].append(diff)
                last_time[key] = time1

                # Call proprocessing function once the number of packets
                # threshold is met
                if (len(main_packet_size[key]) == 10):
                    print(key,"ten packets arrived so far.......")
                    calculate(key,proto)

            else:
                flow_list.append(key)
                main_packet_size[key] = [pktsize]
                main_inter_arrival_time[key] = []
                last_time[key] = time1

# Function for network flow pre-processing and computing values
# for the selected features

def calculate(key,proto):
    feature_vector = " "
    packet_list = main_packet_size[key]  # packet_list contains the list of
    #packet sizes for the flow   in consideration
    length=len(packet_list)  # number of packets
    total_bytes = int(sum(packet_list))     # Total_bytes in the flow
    avg_pkt_size = total_bytes/length
    summ = 0
    # for j in packet_list:     # calculating standard deviation of packet
   sizes
    # summ = summ + (avg_pkt_size - j)*(avg_pkt_size - j)
    # var_pkt_size = summ/length    # variance of packet sizes
    # std_dev_pkt_size = int(math.sqrt(var_pkt_size))
    inter_arrival_time_list = main_inter_arrival_time[key]
    # a list containing IATs for the flows
    length = len(inter_arrival_time_list)
    length1 = len(inter_arrival_time_list)
    if (length > 0):
        flow_duration = sum(inter_arrival_time_list)    # flow duration
        r1 = (length1/flow_duration)
        avg_iat = 1000*1.0*flow_duration/length # averag IAT in miliSeconds
```

```
        summ=0

    for iat in inter_arrival_time_list:
        iat = 1000.0*iat  # covert each IAT to ms, and then calculate
        summ=summ + (iat-avg_iat)*(iat-avg_iat)

        # std_dev_IAT = round(math.sqrt(1.0*summ/(length)),2)
        # standard deviation of IATs
        # Selected netwrok flow features sent to DT mpdel for decision
        feature_vector = ("Average packet size: "+str(avg_pkt_size)+"
        AVG_IAT: "+str(round(float(avg_iat),2))+" Total bytes: "+str(
    total_bytes))
        print(feature_vector)

def main():
    # Real-time packet capturing and processing
    print("custom packet sniffer")
    cap = sniff(filter="ip",iface="eth0",prn=sniffPackets)

if __name__ == '__main__':
    main()
```

Appendix C

Hardware Design

C.1 Overall Hardware Design in Vivado Design Suite

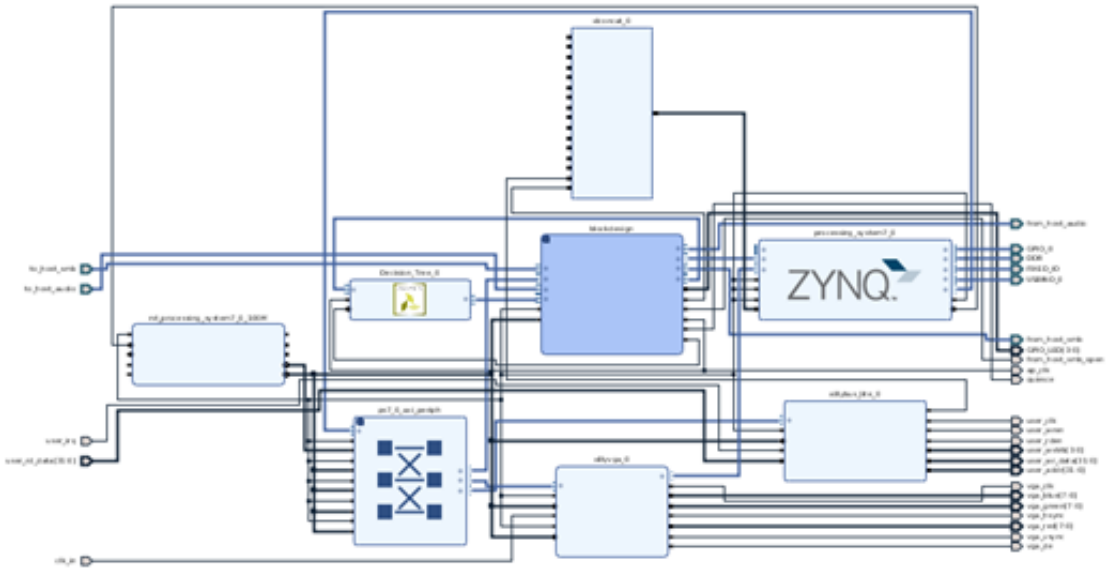

FIGURE C.1: Overall hardware design

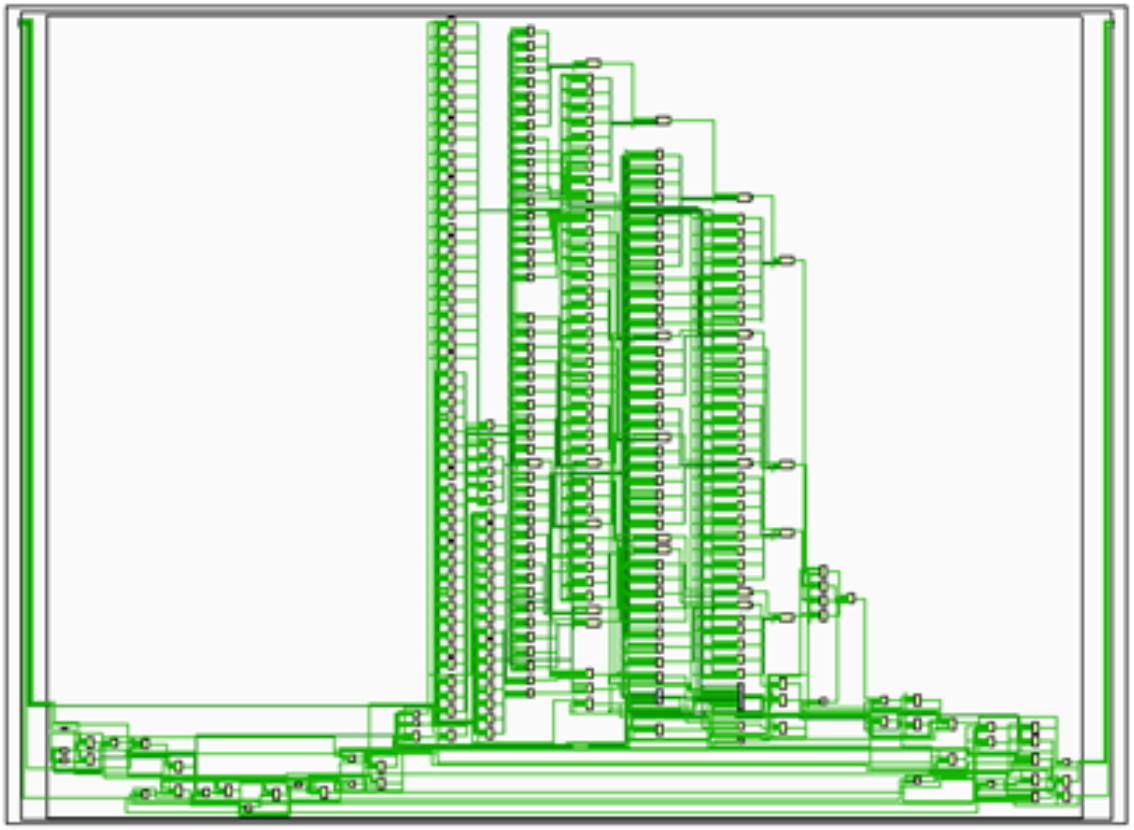

FIGURE C.2: Internal view of the DT3 hardware IP block

Appendix D

NS3 Network Testbed

D.1 Overall network design

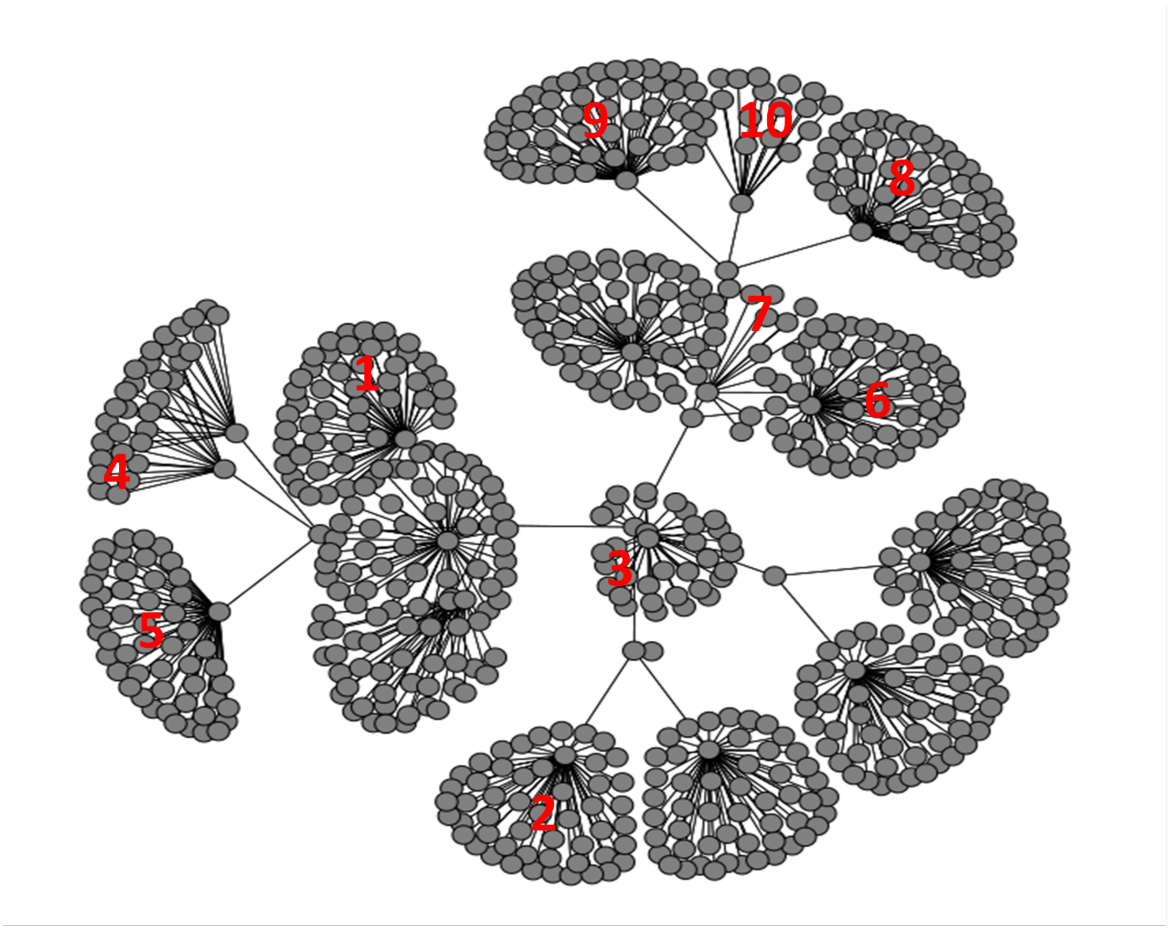

FIGURE D.1: Final Network simulation model in NS3

D.2 Network Flow Monitoring in NS3

```
// #
    ##############################################################################

// Main routine
// #
    ##############################################################################

NS_LOG_COMPONENT_DEFINE ("Multiple-LANs-implementation")
#define vssearch(loc,vec) std::find ((vec).begin(),(vec).end(),(loc))!= (vec)
    .end()

void Monitor (FlowMonitorHelper *flowmon, Ptr<FlowMonitor> monitor)
{
  float total_bytes, Iat;
  float avg_pkt_s;
  int r=0, v=0;
  double seconds = 0.0;
  monitor->SerializeToXmlFile ("scenario1.xml" , true, true );
  monitor->CheckForLostPackets ();
  std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();
  Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier>(
    flowmon->
  GetClassifier ());

  for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator iter = stats.
    begin ();
  iter != stats.end (); ++iter)
  {
    int x[9] = {60,7,5,47,42,48,59,406,112};
    int y[5] = {64,269,1621,6033,2811};
    int z[11] = {1004,2047,6222,38,1584,4438,636,250,168,125,199};

    Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (iter->first);

    if (t.destinationAddress == "192.168.1.2")  # Monitored destination
    address
    {

      Iat = ((iter->second.timeLastRxPacket.GetSeconds()-
      iter->second.timeFirstTxPacket.GetSeconds())/(iter->second.rxPackets)
   *1000);
      avg_pkt_s = (iter->second.rxBytes)/(iter->second.rxPackets);
      total_bytes = iter->second.rxBytes;
      //int A = avg_pkt_s;

      // Classification using the DT3 model
      r = decision(x,y,z,avg_pkt_s,&total_bytes,&Iat,v);

      seconds = (double) Simulator::Now ().GetSeconds ();
```

```
      NS_LOG_INFO (" ");
      std::cout << "Elapsed seconds\t\t: " << seconds << " s" << std::endl;
      NS_LOG_UNCOND("Flow ID: " << iter->first << " Src Addr "<< t.
    sourceAddress <<
      " Dst Addr " << t.destinationAddress);
      NS_LOG_UNCOND("Tx Packets = " << iter->second.txPackets);
      NS_LOG_UNCOND("Rx Packets = " << iter->second.rxPackets);
      NS_LOG_UNCOND("Flow outcome = " << r);
      NS_LOG_UNCOND("Throughput: " << iter->second.rxBytes * 8.0 /
      (iter->second.timeLastRxPacket.GetSeconds()-iter->
      second.timeFirstTxPacket.GetSeconds()) /1024 << " Kbps");
    }
  }
  Simulator::Schedule (Seconds(2), &Monitor, flowmon, monitor);
}
```

Appendix E

Extracted Network Flow Features

E.1 Network Flow Features

Table E.1: Network flow features and their meanings

Feature Name	**Description**
Fl_Dur	Flow duration
Tot_Fwd_Pkt	Total packets in the forward direction
Tot_Bwd_Pkt	Total packets in the backward direction
Tot_l_Fwd_Pkt	Total size of packet in forward direction
Fwd_Pkt_l_Max	Maximum size of packet in forward direction
Fwd_Pkt_l_Min	Minimum size of packet in forward direction
Fwd_Pkt_l_Mean	Average size of packet in forward direction
Fwd_Pkt_l_Std	Standard deviation size of packet in forward direction
Bwd_Pkt_l_Max	Maximum size of packet in backward direction
Bwd_Pkt_l_Min	Minimum size of packet in backward direction
Bwd_Pkt_l_Mean	Mean size of packet in backward direction
Bwd_Pkt_l_Std	Standard deviation size of packet in backward direction
Fl_Byte_S	flow byte rate that is number of packets transferred per second
Fl_Pkt_S	flow packets rate that is number of packets transferred per second
Fl_IAT_Mean	Average time between two flows
Fl_IAT_Std	Standard deviation time two flows
Fl_IAT_Max	Maximum time between two flows
Fl_IAT_Min	Minimum time between two flows
Fwd_IAT_Tot	Total time between two packets sent in the forward direction
Fwd_IAT_Mean	Mean time between two packets sent in the forward direction
Fwd_IAT_Std	Standard deviation time between two packets sent in the forward direction
Fwd_IAT_Max	Maximum time between two packets sent in the forward direction
Fwd_IAT_Min	Minimum time between two packets sent in the forward direction
Bwd_IAT_Tot	Total time between two packets sent in the backward direction
Bwd_IAT_Mean	Mean time between two packets sent in the backward direction
Bwd_IAT_Std	Standard deviation time between two packets sent in the backward direction

TABLE E.2: Network flow features and their meanings cont.

Feature Name	**Description**
Bwd_IAT_Max	Maximum time between two packets sent in the backward direction
Bwd_IAT_Min	Minimum time between two packets sent in the backward direction
Fwd_PSH_Flag	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
Bwd_PSH_Flag	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
Fwd_URG_Flag	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
Bwd_URG_Flag	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
Fwd_Hdr_Len	Total bytes used for headers in the forward direction
Bwd_Hdr_Len	Total bytes used for headers in the forward direction
Fwd_Pkt_S	Number of forward packets per second
Bwd_Pkt_S	Number of backward packets per second
Pkt_Len_Min	Minimum length of a flow
Pkt_Len_Max	Maximum length of a flow
Pkt_Len_Mean	Mean length of a flow
Pkt_Len_Std	Standard deviation length of a flow
Pkt_Len_Va	Minimum inter-arrival time of packet
FIN_Cnt	Number of packets with FIN
SYN_Cnt	Number of packets with SYN
RST_Cnt	Number of packets with RST
PSH_Cnt	Number of packets with PUSH
ACK_Cnt	Number of packets with ACK
URG_Cnt	Number of packets with URG
CWE_Cnt	Number of packets with CWE
ECE_Cnt	Number of packets with ECE
Down_Up_Ratio	Download and upload ratio
Pkt_Size_Mean	Average size of packet
Fwd_Seg_Mean	Average size observed in the forward direction
Bwd_Seg_Mean	Average size observed in the backward direction
Fwd_Byte_Blk_Mean	Average number of bytes bulk rate in the forward direction
Fwd_Pkt_Blk_Mean	Average number of packets bulk rate in the forward direction
Fwd_Blk_rate_Mean	Average number of bulk rate in the forward direction
Bwd_Byte_Blk_Mean	Average number of bytes bulk rate in the backward direction
Bwd_Pkt_Blk_Mean	Average number of packets bulk rate in the backward direction
Bwd_Blk_rate_Mean	Average number of bulk rate in the backward direction
SubFl_Fwd_Pkt	The average number of packets in a sub flow in the forward direction
SubFl_Fwd_Byte	The average number of bytes in a sub flow in the forward direction
SubFl_Bwd_Pkt	The average number of packets in a sub flow in the backward direction

TABLE E.3: Network flow features and their meanings cont.

Feature Name	**Description**
SubFl_Bwd_Byte	The average number of bytes in a sub flow in the backward direction
Fwd_Win_Byte	Number of bytes sent in initial window in the forward direction
Bwd_Win_Byte	Number of bytes sent in initial window in the backward direction
Fwd_Act_Pkt	Number of packets with at least 1 byte of TCP data payload in the forward direction
Fwd_Seg_Min	Minimum segment size observed in the forward direction
ATV_Mean	Mean time a flow was active before becoming idle
ATV_Std	Standard deviation time a flow was active before becoming idle
ATV_Max	Maximum time a flow was active before becoming idle
ATV_Min	Minimum time a flow was active before becoming idle
IDL_Mean	Mean time a flow was idle before becoming active
IDL_Std	Standard deviation time a flow was idle before becoming active
IDL_Max	Maximum time a flow was idle before becoming active
IDL_Min	Minimum time a flow was idle before becoming active