Detecting High and Low Intensity Distributed Denial of Service (DDoS) Attacks

by

Abigail May Yee Koay

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Engineering.

Victoria University of Wellington
2019

Abstract

High and low-intensity attacks are two common Distributed Denial of Service (DDoS) attacks that disrupt Internet users and their daily operations. Detecting these attacks is important to ensure that communication, business operations, and education facilities can run smoothly. Many DDoS attack detection systems have been proposed in the past but still lack performance, scalability, and information sharing ability to detect both high and low-intensity DDoS attacks accurately and early. To combat these issues, this thesis studies the use of Software-Defined Networking technology, entropy-based features, and machine learning classifiers to develop three useful components, namely a good system architecture, a useful set of features, and an accurate and generalised traffic classification scheme. The findings from the experimental analysis and evaluation results of the three components provide important insights for researchers to improve the overall performance, scalability, and information sharing ability for building an accurate and early DDoS attack detection system.

To my family for their endless support.

Acknowledgement

First and foremost, I would like to express my sincere gratitude to my supervisors, Associate Professor Ian Welch, Professor Winston Seah and Dr Aaron Chen for their guidance and patience throughout my PhD journey. I appreciate the valuable feedback and suggestions that helped strengthen my technical writing and presentation skills. Without their support, I would not be able to complete this thesis.

I am grateful for the financial support of the Victoria Doctoral Scholarship, the Faculty of Graduate Research and the School of Engineering and Computer Science.

I wish to extend my gratitude towards Professor Neil Dodgson, Associate Professor Peter Andreae, Associate Professor Will Browne, Dr Lizzie Towl, Dr Diana Siwiak, and Craig Watterson for their helpful advice and encouragement. I am also thankful to Mrs Patricia Stein, who would always make the administrative process as smooth as possible for me. I wish to thank Mrs Suzan Hall and her team for arranging fun activities and lunches at the faculty, in particular, the rock painting activity.

I am deeply thankful to my colleagues in the School of Engineering and Computer Science, especially Abhi, Jordan, Miniruwani, Tony, Victoria, Yi Yin, and Yuyu for brightening my days and nights at the lab. Not to forget, my best friend, Astrid Mataele, who believed in me and would always be there to cheer me up when things got difficult. Sincere thanks to all my friends who are not mentioned here for their care and support.

I would also like to thank my thesis examination committee for reading my thesis and providing insightful feedback and encouragement during my oral examination.

Finally and most importantly, I would like to thank my family for their endless love and encouragement throughout my PhD journey. I truly appreciate all the video calls that kept me company while I ran my experiments until late at night.

Contents

List of Figures

List of Tables

Abreviations

DDoS Distributed Denial of Service.

DFA Direct Flooding Attack.

DNS Domain Name Server.

DR Detection Rate.

E3ML Entropy based with Three (3) Machine Learning Classifier Scheme.

FPR False Positive Rate.

IoT Internet-of-Things.

IP Internet Protocol.

NFV Network Function Virtualization.

SCAFE Scalable and Fault-Tolerant DDoS attack detection Architecture for Early Detection.

SDN Software Defined Networking.

TPR True Positive Rate.

Chapter 1

Introduction

The Internet has become one of the most popularly used technologies in the past decade. In the 1960s, the Internet started as a small packet switching network called the Advanced Research Projects Agency Network (ARPANET) in the United States, but has since expanded to create a better form of communication, education, and businesses on a global scale. The Internet offers many benefits and advantages to mankind, but the ”no security” in its original design, the lack of collaboration between Internet Service Providers (ISP), the absence of international laws, the shortage of Internet security agreements, and the easily available malicious tools and software have made the Internet an excellent platform for attackers to launch attacks and perform unlawful activities [1].

Distributed Denial of Service (DDoS) is the most common attack seen on the Internet [**?**]. A DDoS attack is a coordinated cyberspace attack that aims to deny services to legitimate users through flooding of useless or malicious traffic [2]. With the emerging concept of the Internet-of-Things (IoT), various devices that are connected to the Internet are often not adequately secured. As a result, attackers often exploit and take advantage of these unsecured devices by turning them into platforms for instigating large-scale and sophisticated DDoS attacks [3, 4].

1.1 Severity of the Attack

In the last five years, the size of DDoS attacks has been increasing exponentially, as shown in Figure 1.1. DDoS attacks initially started on a relatively small-scale with estimated attack traffic of 200Mbps [5]. In the early days, such a small attack would be adequate in bringing down a victim's network. In 2007, a continuous DDoS attack with a maximum of 90 Mbps crippled Estonia's Internet for almost three weeks and created major damages to the government, banking, and media websites [6]. A year later, the Internet infrastructure of Georgia, a country located at the crossroad of Western Asia and Eastern Europe, was successfully crippled for nearly a month as a result of a 200Mbps DDoS attack [7]. Thus, these attacks have resulted in significant financial loses and disruption of work to businesses and government organisations.

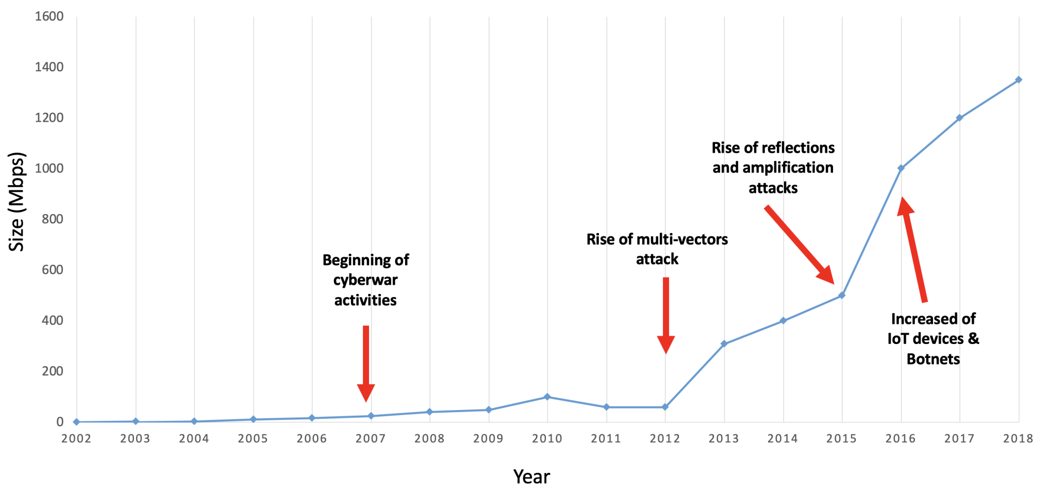


Figure 1.1: DDoS Attack Growth in Terms of Size (Mbps) from 2002-2018 [8, 9, 10]

With the expansion of the Internet, many of the attackers are increasing the scale of DDoS attacks with the aim of bringing down a broader set of victims. While the earlier form of DDoS attacks had mostly targeted a particular organisation or business, these newer DDoS attacks evolved and now possess the capability of bringing down several businesses and organisations on the Internet within a particular region.

In 2013, the whole Internet infrastructure was put to the test, when a DDoS attack with a peak of 300Gbps was launched against a not-for-profit anti-spam organisation, Spamhaus [8]. This particular attack successfully took down the Spamhaus servers and resulted in Internet congestion all over Europe, delaying users' accessibility to major websites [8]. A large DDoS attack occurred the following year against a web hosting company, OVH, with the highest recorded attack traffic of 400Gbps [9]. In 2016, another major DDoS attack, peaking at over 1Tbps using the Mirai Botnet, had resulted in the inaccessibility of many websites among which included some high profile websites like Twitter, Reddit, GitHub, and Airbnb [10].

Since the severity of DDoS attacks can disrupt many of the individuals', businesses', and government organisations' daily operations, importance should be given to DDoS attack detection so that these attacks can be mitigated more effectively. However, detecting DDoS attacks is not an easy task. Traditionally, DDoS attacks are sent by using high-intensity traffic that is directly aimed at the victim. As a case in point, high-intensity attacks such as Smurf [11] would transmit a high-rate of attack traffic in a short amount of time to their victims, and result in a sudden surge of traffic that leads to a denial-of-service. In contrast, the newer forms of DDoS attacks would indirectly send low-intensity attack traffic to their victims and focus on congesting the links shared between the victims and third-party devices such as decoy servers [12] and bots [13]. These low-intensity attacks are very effective as they are carefully coordinated by the attackers to saturate network links through the merging of the coordinated individual attack traffic at an aggregation point. The aggregated attack traffic can have legitimate source IP addresses which can be viewed anywhere other than the target link, allowing it to be indistinguishable from the actual normal traffic prior to reaching its aggregation point. Additionally, attackers may use coordination with persistent routes which are stable and not affected by load balancing mechanisms in sending attack traffic [12]. As a result, the subtly, persistence, and low-intensity of these new DDoS attacks have made it much more difficult to maintain the security of the Internet.

While there have been various approaches proposed in addressing the DDoS attacks, the existing approaches lack performance, scalability, and information sharing capability for accurate and early detection of high and low-intensity DDoS attacks [2, 14]. The ability to identify attack traffic is an essential step towards addressing the problems in DDoS attack detection. Without DDoS attack detection, it will be difficult to perform any anti-DDoS measures to handle the attack. Therefore, it is crucial to develop an understanding of DDoS attack traffic and gaps found in current detection approaches to improve DDoS attack detection.

1.2 Problems with Current Detection Solutions

An ideal DDoS detection system would aim to detect the DDoS attack traffic in real-time and to mitigate the attack as close to the source as possible so as to prevent further damages to the network [14]. This translates into two primary goals, namely high accuracy and early detection. The term *accuracy* denotes the system having a high true positive and low false positive rates in detecting DDoS attacks, and *early detection* means detecting DDoS attacks nearest to the source of the attack (i.e. at the local switch/router closest to the attack source).

Since the DDoS attack traffic can originate from multiple sources and relies on aggregated traffic volume in saturating its victim, the volume of the attack traffic is usually small when being measured at any link other than the aggregation point. A small amount of attack traffic may or may not be unusual enough for it to be detected from the vast majority of normal traffic. As such, it is important to detect the DDoS attack traffic as early as possible so that the impact of the attack on the network can be minimised. The detection system also has to be able to detect the attack traffic closer to its source with minimal false positive and miss detection rates.

As many current detection solutions primarily focus on the accuracy and early detection of DDoS attacks, the three main issues that hinder DDoS attack detection solutions to detect attacks accurately and early are:

a. Performance

Previous DDoS detection systems such as LADS [15], D-WARD [16], and NetBouncer [17] are mostly aimed at detecting high-intensity DDoS attacks. These detection systems use volumetric techniques in detecting the attacks, where they are not effective in the detection of newer DDoS attacks such as low-intensity attacks. A low-intensity attack has a lower volume and so will not trigger any alerts on the volumetric-based detectors. As a way of overcoming this problem, non-volumetric-based approaches such as entropy-based detection have been proposed to detect low-intensity DDoS attack traffic [18, 19, 20].

Although the entropy-based DDoS attack detection approach is a promising approach, its limitation lies in its lack of accuracy as a result of the type of features and thresholds used in distinguishing between the normal and attack traffic [21, 22, 23, 24, 25]. Moreover, the over-reliance of current systems on commonly used features such as the source and destination IP addresses, source and destination port numbers, and protocol identifier in detecting DDoS attacks has neglected the potential and contribution of other features such as packet content, time to live, packet arrival time, etc., that may lead to a more accurate DDoS attack traffic detection. These existing solutions do not only rely on a fixed or adaptive threshold, which can only detect a specific type of DDoS attack traffic, but also do not consider the variations of DDoS attack intensity, particularly high and low-intensity DDoS attacks.

b. Scalability

As networks become larger with increasing amounts of traffic, it has become a challenge to efficiently process the vast amount of network traffic and detect attack traffic that is located near to its respective sources. The main challenge in a scalable detection system is its ability to process traffic smoothly, without consuming too much of its processing power and incurring large communication overhead. While some of the current research solutions have increased the scalability of detection systems by using the sFlow [26] and NetFlow [27] technologies, and traffic sampling as a way to reduce processing cost, these approaches are unsuccessful in detecting stealthy attacks since not all types of traffic are subjected to analysis.

In this case, the DDoS attack detection system's scalability could be increased through the use of Software-Defined Networking (SDN) technology, which operates by separating the control plane and data plane for more efficient handling of the network traffic. Although the separation allows traffic to be analysed at a logically centralised controller without impacting the network performance on the data plane, the control plane can be congested with a sufficiently large amount of communication traffic and overhead to the controller.

c. Information sharing

The current collaborative approaches for detecting DDoS attack traffic are bounded by information shared within neighbouring nodes or nodes within the same Local Area Network (LAN) [28]. This not only limits the gathering of information that is sufficient for accurate traffic classification, but also hinders early detection since attack traffic could only be detected within each LAN or at the nodes close to the victim.

Some of the current DDoS attack detection systems use correlation, where they measure the relationship between traffic features such as arrival time, packet volume, protocol, packet size, etc. [29, 30] between two distinct traffic flows [31], between alerts from detection systems [32], and between traffic results taken at different times [33]. Although these approaches are able to improve the detection accuracy, they only focus on correlating traffic information from a single location in the network which limits their early detection capabilities.

We summarise the problems of DDoS attacks in the following problem statement:

The main problem with current DDoS attack detection approaches is the inability to detect both high and low-intensity DDoS attacks accurately and early in large-scale networks. Traditional methods such as volumetric-based traffic classification, centralised processing, and LAN-based correlation no longer effectively support accurate and early DDoS attack detection in large-scale networks due to the limitations of their performance, scalability, and information sharing capabilities. Therefore, it is important to find new methods for detecting both high and low-intensity DDoS attacks accurately and early.

1.3 Research Question

As the problems mentioned in Section 1.2 still remain as open research issues, this thesis attempts to address these problems by answering the following research question:

How can we detect high and low-intensity DDoS attacks accurately and early?

The above research question is broken down into the following sub-questions:

(a) How can we improve the early detection capability of large-scale DDoS attack detection with scalability?
This sub-question is closely tied to the *scalability* and *information sharing* issues mentioned in Section 1.2. As such, this thesis attempts to answer this question by examining the use of SDN in reducing the communication complexity and simplifying information sharing in a DDoS attack detection system as a way of improving scalability and information sharing capability, and designing an architecture that supports network-wide correlation analysis to improve the early detection capability of a DDoS attack detection system.

(b) What are the good entropy-based features and useful parameter settings for distinguishing attack traffic from normal traffic?
This sub-question is closely tied to the *performance* issue mentioned in Section 1.2. This thesis attempts to answer this question by exploring the alternative entropy measures such as Shannon [34], Tsallis [35], Rényi [36] and Xiang [19] in the computation of entropy-based features as well as in understanding the trade-off between window size and detection accuracy in entropy-based features.

(c) How can we improve the accuracy of traffic classification using entropy-based features?
This sub-question is closely tied to the *performance* issue mentioned in Section 1.2. This thesis attempts to find the best machine learning methods by using the entropy-based features for traffic classification as a way of improving the accuracy of detecting both high and low-intensity DDoS attack traffic.

1.4 Research Contributions

Figure 1.2 depicts the overview of the thesis and highlights its contributions in the last tier of the tree.

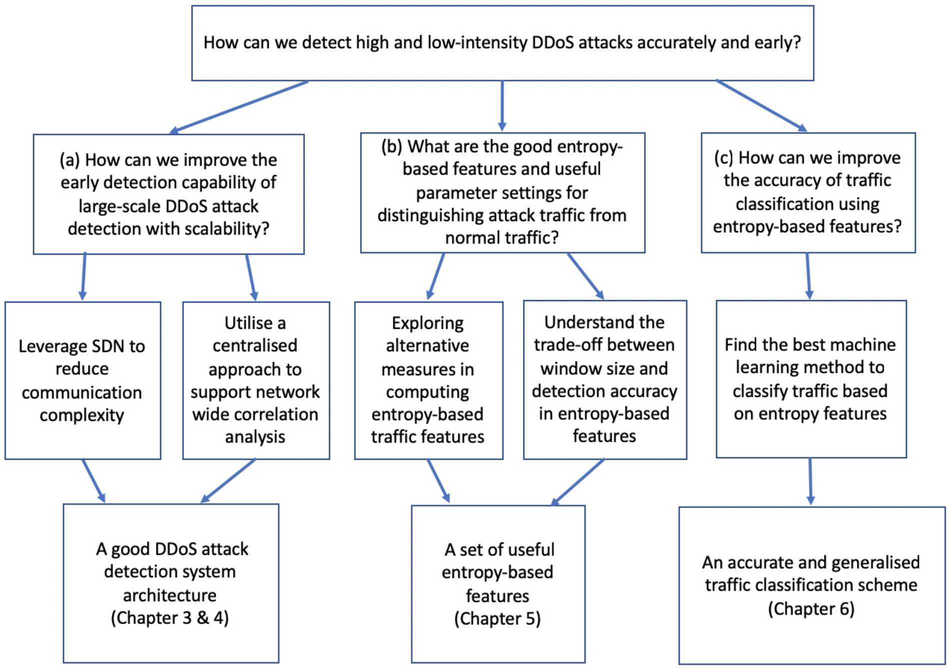

How can we detect high and low-intensity DDoS attacks accurately and early?
(a) How can we improve the early detection capability of large-scale DDoS attack detection with scalability?
(b) What are the good entropy-based features and useful parameter settings for distinguishing attack traffic from normal traffic?
(c) How can we improve the accuracy of traffic classification using entropy-based features?
Leverage SDN to reduce communication complexity
Utilise a centralised approach to support network wide correlation analysis
Exploring alternative measures in computing entropy-based traffic features
Understand the trade-off between window size and detection accuracy in entropy-based features
Find the best machine learning method to classify traffic based on entropy features
A good DDoS attack detection system architecture (Chapter 3 & 4)
A set of useful entropy-based features (Chapter 5)
An accurate and generalised traffic classification scheme (Chapter 6)

Figure 1.2: Thesis Contribution Map

The contributions are described in detail in the next few sub-sections.

(a) A good DDoS attack detection system architecture

This thesis presents a SCAlable and Fault-tolerant DDoS detection system architecture for Early detection (SCAFE) by (1) leveraging the SDN technology in reducing communication complexity and simplifying information sharing, and (2) selecting a centralised approach that supports a network-wide correlation analysis. SCAFE is based on a two-level system architecture, which improves the scalability and information sharing capability of the DDoS attack detection system. In addition, SCAFE is also designed to be fault tolerant to hardware and software faults. The first level consists of a coarse-grained detection mechanism that has the ability for identifying potential attack links, while the second level is a fine-grained detection system where traffic flows from multiple locations are collected and correlated to detect attack flows that are closest to the attack source.

The two-level detection approach in SCAFE not only improves the scalability by maintaining the processing and detection time during increases in the traffic volume or network size but also improves the fault-tolerance capability by using a set of fault-tolerant system components. Therefore, SCAFE can be regarded as a good system architecture that can be embedded in the development of an accurate and early DDoS attack detection system in large-scale networks.

(b) A set of useful entropy-based features
This thesis provides a set of useful entropy-based features that is based on two types of investigations: (1) exploring the impact of alternative measures such as Tsallis, Rényi and Zhou entropies as opposed to the commonly used Shannon entropy in detailing the DDoS attack traffic patterns in the network and (b) understanding the trade-off between window size and detection accuracy of entropy-based features in the generation of good entropy-based features.

The set of useful entropy-based features consists of entropy-based features that are constructed from raw traffic features as well as those that are constructed from the variation of two distinct entropy-based features. This list of entropy-based features is important as it provides insights on the usefulness of entropy-based features to help in the feature selection for high and low-intensity DDoS attack detection.

(c) An accurate and generalised traffic classification scheme
The multiple Entropy-based features with three (3) Machine Learning classifier (E3ML) is an accurate and generalised traffic classification scheme for DDoS attack detection, where it is a novel classification approach that uses entropy-based features for distinguishing both high and low-intensity DDoS attack traffic from normal traffic accurately. The E3ML utilises a classification model, where it is built from

exploiting the different strengths of three machine learning classifiers, namely multilayer perceptron, recurrent neural network, and alternating decision tree classifiers via a voting system.

Since the E3ML traffic classification scheme achieves a high accuracy rate in detecting both high and low DDoS attacks, the multiple classifier system is a suitable method for improving the detection accuracy of both high and low-intensity DDoS attacks.

1.5 Thesis Structure

The remaining content of this thesis is organised as follows.

- ***Chapter 2 - Background and Literature Review***
  This chapter introduces the Distributed Denial of Service attacks, discusses the related concepts such as entropy-based features, machine learning classification, correlation analysis and SDN, analyses the current detection approaches and identifies the gaps found in the current literature.

- ***Chapter 3 - Design of a Scalable and Fault-Tolerant System Architecture (SCAFE) to Handle Large Networks***
  This chapter presents the system architecture design of a scalable and fault-tolerant DDoS detection system (SCAFE) that is based on SDN and network-wide correlation analysis for large networks. The chapter also presents the design assumptions, goals and principles, the overview of the system architecture, system component roles and relationships as well as the communication paths and message interfaces of SCAFE.

- ***Chapter 4 - Evaluation of SCAFE DDoS Detection System***
  This chapter highlights the evaluation on the scalability, fault-tolerant and performance analysis of the SCAFE system architecture that was

designed in Chapter 3 in terms of its effectiveness in handling large network traffic.

- ***Chapter 5 - Identifying Good Entropy-based Features***
  This chapter describes the three-step construction of the entropy-based features. The chapter also identifies a set of useful entropy-based features by measuring its accuracy level in distinguishing between a DDoS attack and normal traffic. The influence of entropy measures and the effect of window size towards the accuracy of entropy-based features are also deliberated in this chapter.

- ***Chapter 6 - Traffic Classification using Entropy-based Features and Machine Learning Classifiers in DDoS Attack Detection***
  This chapter presents the overview and evaluation of the E3ML DDoS detection scheme for detecting the high and low-intensity DDoS attack traffic as well as the evaluation of the E3ML performance on different DDoS attacks and compares it to other state-of-the-art detection schemes.

- ***Chapter 7 - Contributions and Future Work***
  This chapter summarises the results and highlights the contributions derived from the research done in this thesis. The potential future directions in the field of high and low-intensity DDoS attack detection are also presented in this chapter.

Chapter 2

Background and Literature Review

This chapter presents the background of the DDoS attacks and some of the concepts such as entropy-based features, machine learning classifiers, correlation analysis, and SDN that form the basis of this thesis. In addition, this chapter also reviews the literature on the DDoS attack detection approaches from these concepts as well as identifying the datasets and test beds used in the evaluations of DDoS attack detection. The gaps and limitations found in DDoS attack detection are highlighted as part of the conclusion of this chapter.

2.1 Distributed Denial of Service Attack

Distributed Denial of Service (DDoS) is a type of Denial of Service (DoS) attack that aims to deny network services by crashing the targeted servers or consume their resources so that these servers can no longer provide services to legitimate users [1, 37, 38, 39]. While the attacker attacks through a single source in a DoS attack, in the case of a DDoS attack, the attacker launches attacks that originate from a variety of distributed sources. These distributed sources are often globally distributed and can amount to several hundred thousand sources [40]. Usually, these sources can be from the attacker's computers or originate from compromised devices on the Internet referred to as bots [41]. As such, DDoS attacks are larger (e.g., 600 Gbps), and more difficult to defeat than DoS attacks.

The first large-scale DDoS attack incident was reported in the year 2000, where large Internet sites such as Yahoo!, CNN, eBay, and ZDNet were left offline for several hours as a result of being flooded with high volume attack traffic [42]. Since then, the DDoS attacks have grown bigger and more complex, avoiding detection and bringing down networks and services on the Internet.

2.1.1 How to Launch an Attack?

Attackers have since come up with many strategies of launching an effective DDoS attack. An attack is considered effective when it breaches the security defences and causes major disruption or it successfully brings down an entire organisation network for several hours. This type of attack would require a large number of attack hosts (i.e. bots) launching a coordinated DoS attack against a single machine or a large organisation network. As such, a DDoS attack can be broadly categorised under two different launching methods, namely the direct and indirect flooding.

Direct Flooding

As shown in Figure 2.1, a direct flooding attack (DFA) occurs when the attacker commands the bot to send the attack traffic directly to its target through a flooding of malicious or useless traffic.

Some of the examples of DFAs include:

- **SYN Flooding**
  SYN flooding abuses the three-way handshake session in the Transmission Control Protocol (TCP) by creating many half-open connections that render a target unresponsive. This three-way handshake

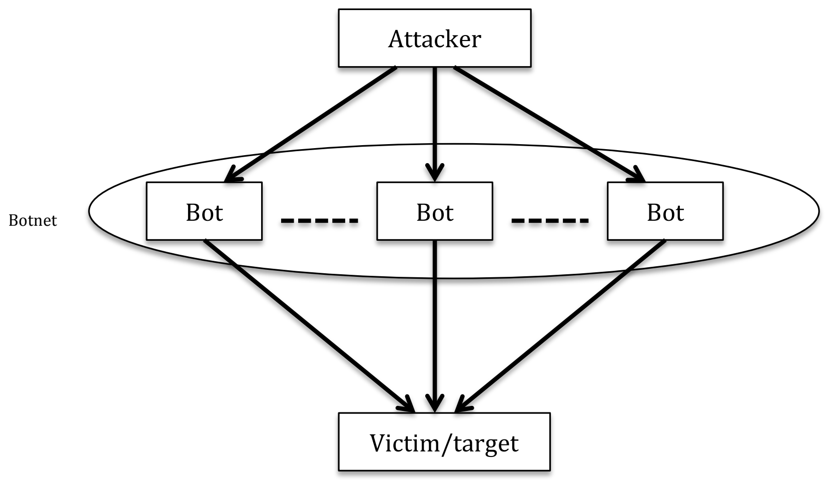


Figure 2.1: Direct Flooding Attack

session is made up of three messages that are being exchanged between the client and server prior to establishing a connection. This involves the client sending a SYN[1] message to the server with the latter reciprocating with a SYN-ACK[2] message. A connection will then be established after the client has responded with the ACK[3] message. Under the SYN flooding situation, the attacker will send as many SYN requests to its target as possible and not respond with the expected ACK message. As the server continues to anticipate the ACK reply from the client, an accumulation of half-open connections between the client and server will occur. When the number of half-open connections exceeds the server's resources, the server becomes unresponsive hence resulting in its inability to establish new connections to other legitimate users.

- **ICMP Flooding**
  ICMP Flooding or Ping Flood is a type of attack that sends a large

[1] SYN - Synchronised control flag that is used for TCP's three-way handshake.
[2] SYN-ACK - Synchronised-Acknowledgement control flag that is used for TCP's three-way handshake.
[3] ACK - Acknowledgement control flag that is used for TCP's three-way handshake.

number of ICMP echo requests (pings) that overwhelm the target so that it cannot respond to any of the new requests.

- **UDP Flooding**
  UDP Flooding abuses the User Datagram Protocol (UDP) by sending a large number of UDP datagrams to random ports of the target until the target becomes overwhelmed and unresponsive to requests from legitimate users.

Indirect Flooding

As shown in Figure 2.2, an Indirect Flooding Attack (IFA) denotes the indirect flooding of a target by sending attack traffic to third-party devices, where these devices will route the attack traffic to their targets. In some cases, these devices will amplify the attack traffic as a way of inflicting a larger impact on their targets. As such, this method aims to increase the attack's stealthiness by hiding the actual source of the attack as a way of avoiding detection.

Some of the examples of IFAs are as follows:

- **Coremelt attack [13]**
  A Coremelt attack is a fairly new IFA introduced in 2009. The Coremelt attack aims to congest the targeted links at the network core by using a collection of bots or subverted machines as attack sources as well as attack destinations [13], where the bots are paired up to send legitimate traffic to each other. The detection of a Coremelt attack has proven to be challenging because these bots send the attack traffic back and forth to each other at a low rate, and the attack traffic sent is similar to legitimate traffic. The effectiveness of this attack was demonstrated through simulation on two different network sizes with 4746 Autonomous Systems (ASes) and 720 ASes respectively, where the attack successfully cut off the connection of the top 10 ASes using only 700,000 to 1,008,000 bots [13].

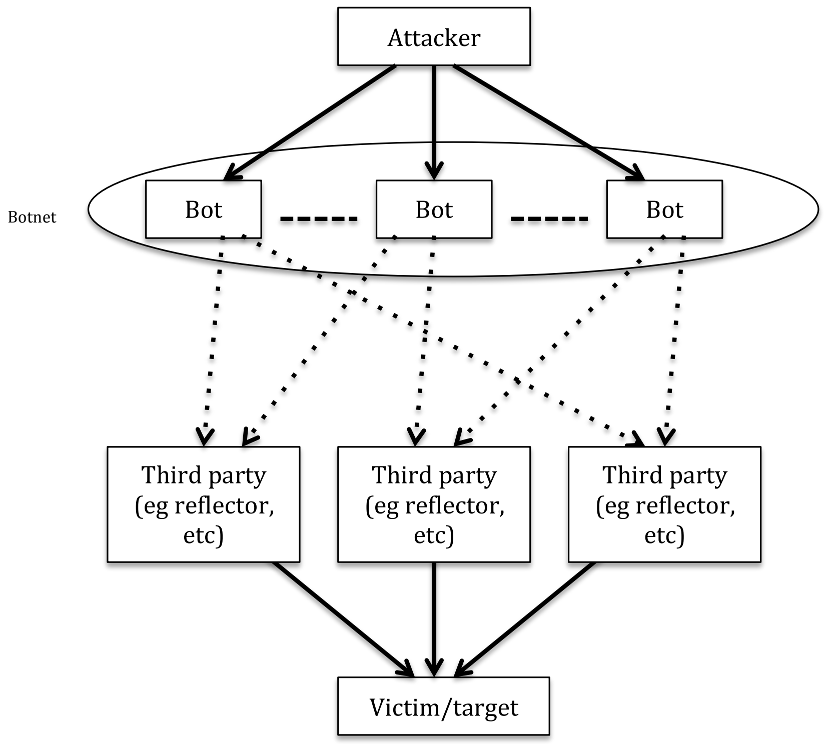

Attacker
Botnet
Bot
Bot
Bot
Third party (eg reflector, etc)
Third party (eg reflector, etc)
Third party (eg reflector, etc)
Victim/target

Figure 2.2: Indirect Flooding Attack

- **Crossfire Attack [12]**

  A Crossfire attack aims to indirectly congest the network links by sending the attack traffic to decoy servers in the network [12]. While it is similar to a Coremelt attack in terms of using bots as a means of sending out attack traffic, the main intention is not to flood the servers and cause them to be inaccessible to public users, but to act as decoys in flooding the links that are connected to the target server. The Crossfire attack also shares another similarity with the Coremelt attack, where the network links between the source and destination of the attack traffic are being targeted for congestion. For the attack to be successful, attackers would have to carefully coordinate the attack traffic by congesting all of the network links surrounding the targeted organisation's network in an attempt to block all of their access links to the Internet.

- **Reflection Attack [38]**
  A reflection attack uses reflectors on the network to direct the attack traffic to the target. In this case, the reflector is a network device that will return any packets that were sent to it, which can be web servers, DNS servers, and routers on the network. Apart from launching a reflection attack, by sending spoofed requests to the many reflectors on the Internet that will then forward the responses to their target, the attackers also make use of the combined transmission power of both the attack machines and reflectors as a way of orchestrating a powerful flooding attack on the target. Although most reflectors act as amplifiers (sending out a larger reply packet than the request packet), some of these reflectors may attenuate the packet size and can still effectively impose damages to the target by having a sufficient number of reflectors in the attack. By using reflectors, this can also create a dilution of locality in the attack streams and increase the stealthiness of the attack. In literature, these reflectors such as DNS servers, GNUTELLA servers, and TCP-based servers running on TCP implementations that suffer from predictable initial sequence numbers pose a significant threat to the Internet.

- **Amplification attack**
  An amplification attack uses a third-party device to either amplify a small request into a larger request directed to a target, or broadcasts a request using a spoofed IP address (i.e. target IP address) that directs all of the replies to the target. In a Smurf attack [11], for example, the attacker sends the ICMP echo messages to an unprotected broadcast domain and causes each host in the domain to send out an ICMP echo reply message. Similar to a reflection attack, spoofed source IP addresses are also used in the Smurf attack. As a result, all of the

hosts in the broadcast domain will send the ICMP echo reply messages to the target's IP address hence creating a massive volume of traffic that depletes their target's network bandwidth. Some of these approaches may involve the combination of a reflection attack and IP spoofing as a means of increasing the impact of the attack. This is shown by the DNS amplification attack, where the attacker takes advantage of the open recursive feature of a DNS server by sending a recursive query to the server and having it returned as large amplified response traffic to the target. In another example, although an NTP amplification attack works quite similarly to a DNS amplification attack, the only difference is that instead of exploiting the DNS resolvers, it exploits the MONLIST command on NTP servers. As the MONLIST command operates by sending the requestor a list of the last 600 hosts that had been connected to the NTP server, the attacker takes advantage of the MONLIST command and sends the MONLIST request to the server via a spoofed IP address. Since the NTP server assumes the request from the spoofed IP address to be from the target machine, it will then send the MONLIST response to the target. Once the NTP server is fully populated, the response data would be 206 times larger than the initial request data. For this reason, an amplification attack can significantly increase the traffic load hence accelerating the depletion of the target's bandwidth until it is completely crippled.

2.1.2 What are Attack Intensities?

A DDoS attack can be launched in various intensities, which can be primarily categorised as either high or low-intensity DDoS attacks.

(a) High-Intensity DDoS Attacks

In a conventional DDoS attack, the attacker uses high-intensity traffic to consume all the network and server resources [20, 43]. High-intensity attacks such as Smurf [11] would transmit packets at a high rate that result in a sudden surge of traffic flow and volume. The sudden surge can easily trigger network detection systems in the network, which enables mitigation mechanisms to halt the attack and prevent further damages in the network.

(b) Low-Intensity DDoS Attacks

A low-Intensity DDoS attack is a stealthy attack that uses a significant amount of aggregated low-intensity traffic to saturate its target. Although a low-intensity attack traffic such as Slowloris [44] does not rely on volume in causing denial-of-service, it will keep and hold as many connection ports opened as possible to prevent others from accessing the target servers. In this way, the attack can effectively cripple the target by sufficiently increasing the aggregated attack volume without setting off the detection systems. There are also other low-intensity attacks that do not seek to disrupt service entirely, but instead choose to degrade the service over a longer period to achieve economic damage [6].

2.1.3 What is the Attack Target?

DDoS attacks mostly target hosts and links on the network as a way of disabling their services.

(a) Host
The attacker targets the application, CPU, memory, service resources or even the hardware of the host which might be a component within the network (e.g. router), an edge device (e.g. switch, edge router), or a server within the network with the aim of bringing down the host.

(b) Link
The attacker attacks the network links by congesting the network

bandwidth and disabling the communication path of the targeted organisation's network. This is deemed as a more severe attack since by congesting the network link, all the hosts connected to it will also be simultaneously brought down.

2.1.4 What is used to Launch an Attack?

A DDoS attack can be launched in many ways as shown below.

Army of Bots (Botnet)

Bots are compromised machines on the Internet that the attackers use to launch attack traffic. Since bots are not only cheap, but are also easily deployable, they are sold in the dark web with many offering DDoS attack services at a very low price. The attacker only needs to upload the attack command to a Command & Control (C&C) server, where it will be pushed down to all of the connected bots. A network of bots connected to the same C&C server is called a botnet, which can have a large number of bots and these bots are often globally distributed in the launching of high impact DDoS attacks on the network. The recent Mirai botnet has shown the danger of bots on the Internet. Since 2013, bots are one of the contributing factors that leads to the exponential increase of DDoS attack size.

Some of the more popular bots used in DDoS attacks include:

(a) Code Red Worm
A worm that spreads quickly and has the ability of building up hundreds of thousands of hosts in bot armies, with each bot having a specific predefined DDoS attack target.

(b) Internet Chat Relay (IRC)
The IRC bots are initially introduced as a way of assisting operators in managing busy chat channels, but they are used by attackers in

launching DDoS attacks on the IRC users and servers [45]. The attacker usually controls the bots via a centralised C&C server, where some of the most commonly used bots for DDoS attacks are Agobot, Nesebot, Spybot, Rxbot and Kaiten [5].

(c) Peer-to-peer (P2P)
Unlike the other bots that rely on a C&C server to communicate, the P2P bots communicate with peer bots in launching more resilient DDoS attacks [46]. This is because the attacker can use any P2P bots as the C&C server in sending the attack command to the entire bot army hence making the botnet harder to destroy. Some of the most commonly used P2P bots for DDoS attacks are Phatbot, SpamThru, Nugache and Peacomm [47].

(d) Internet-of-Things (IoT)
IoT bots are mostly based on poorly secured IoT devices connected to the Internet, where Mirai and Hajime botnets are some the more well-known IoT botnets.

Attack Software

Most attackers adopt a certain type of attack software in the launching of its DDoS attack since larger resources are required to accomodate for the increasing attack sizes. This type of attack software is an automated tool that helps attackers manage the launching of a large-scale attack. Some of the well-known attack software includes:

(a) Low-Orbit Ion Canon (LOIC)
LOIC [48] is an open source tool that can generate UDP and TCP datagrams for performing DDoS attacks. One of the disadvantages of LOIC is that it does not mask the IP address of the traffic it generates. This will then enable the attack source to be tracked easily.

(b) Tribe Flood Network (TFN)
TFN [49] allows an attacker to launch a DDoS attack such as Smurf, TCP SYN flood, UDP flood, and ICMP flood on either a specified or a random port. The TFN tool consists of a master and a set of servers, where the master sends instructions to the servers to launch DDoS attacks. DDoS attacks generated by this tool are difficult to mitigate because the communication between TFN servers and its master is mask by using ICMP Echo reply packet, where filtering ICMP echo reply packets will also cause other applications that rely on these packets to stop functioning.

(c) MStream
Mstream [50] allows an attacker to rapidly flood and deplete the bandwidth of the target using TCP ACK packets with masked source addresses.

(d) Stacheldraht
With Stacheldraht [49], an attacker can launch many types of DDoS attacks such as Smurf, TCP SYN flood, UDP flood and ICMP ECHO flood. Since Stacheldraht uses encrypted TCP and ICMP packets for communication between the attacker and attack sources, it is not easy to be traced.

(e) Trinoo
Trinoo or Trin00 [49] utilises a UDP flood to deny a particular service without spoofing its sources' IP addresses. This attack tool also allows an attacker to modify the traffic packet size as well as define the duration of an attack.

2.1.5 How do you Detect an Attack?

DDoS attack detection is a process that distinguishes attack traffic from legitimate traffic. In most networks, DDoS attack detection systems are used for detecting DDoS attacks and securing the network.

The attack detection systems are commonly classified into two categories, namely *signature detection* and *anomaly detection*. While signature detection uses a predefined set of signatures to match the incoming traffic packets of known attacks, anomaly detection uses a baseline of normal traffic patterns and classifies the traffic that deviates from these patterns as an attack. Each of these two techniques however, has its own advantages and drawbacks.

The main advantage of signature detection is that such a system can detect known attacks with a high accuracy rate by matching the traffic pattern to the patterns stored in the signature database. However, it is unable to identify any attacks that are not listed in the signature database such as zero-day attacks. In order to maintain the effectiveness of the system, the signature database would require constant updates of new attack signatures. These updates can cause scalability problems with the increasing size of the database.

The main advantage of anomaly detection is that it can detect both new and zero-day attacks more accurately. Instead of using a database to store attack signatures, this method detects the attack by measuring the deviation of traffic against the normal traffic profiles. A typical anomaly detection approach would form thresholds by using the baseline of the normal traffic to distinguish between attack and normal traffic. For example, a significant increase in the traffic statistics such as packet delays, traffic volumes or a sudden drop in the performance of the network could be an indication of an attack. However, in some cases, it might be caused by *flash crowd*[4] events or software maintenance. As such, anomaly detection is still bounded by the limitation to accurately determine the correct

[4] A flash crowd is a type of surge in the network traffic, where a high number of legitimate users are using the same Internet service or a server at the same time. Examples of flash crowd events are new product campaigns, online polling, and streaming of live events.

thresholds for distinguishing between attack and normal traffic.

Detection Strategies

Many traffic classification algorithms have been developed, which can be placed into two main categories, namely the packet-based and flow-based classifications.

(a) Packet-based Classification

The packet-based classification operates by analysing each packet in the network to determine if it is an attack or a normal packet. In a traditional DDoS attack, the attack packets have certain features that can be used for attack detection, as exemplified by the SYN flood attack packet where requests with their SYN-flag set are sent to the target server [51]. Since the recent attacks show how attackers use legitimate traffic as attack traffic in a synchronised and coordinated fashion to bring down a network link, it is futile to identify these attacks at the packet level.

(b) Flow-based Classification

The flow-based classification is used to overcome the limitations of packet-based classification. Since a flow is a stream of traffic packets that have the same attributes such as source IP address, destination IP address, source port number, destination port number and protocol identifier in a given time interval, the flow filtering mechanism can be easily deployed at the switch once an attack flow is identified. Although most of the flow-based detection methods utilise a form of distribution analysis to detect the attack flows, the flow extraction process can be time consuming and requires a lightweight application for statistics collection. The two common types of flow level detection are complete detection[5] and sampling-based detec-

[5] Complete detection takes into account all flows within a time interval for analysis.

tion[6]. While a complete detection is often not scalable and requires more resources for completing the detection process due to large incoming traffic size, flow sampling detection only requires a portion of the traffic to be analysed. However, flow sampling can be susceptible to misdetection. This is because DDoS attack traffic in the sample might be inadequate for showing signs of attack.

DDoS Attack Detection Architectures

The DDoS attack detection architecture consists of two main forms, namely centralised and distributed.

(a) Centralised Architecture

A centralised architecture operates by sending all of the network traffic to a central site for processing, which is more suited to be used in a small enterprise network. In most cases, larger networks using this architecture will cause the DDoS attack detection to be ineffective. This is because processing a large amount of traffic in a large network will result in high computational burden and a single point of failure on the central site during an attack [2].

(b) Distributed Architecture

In a distributed architecture, a DDoS attack detection system does not rely on a single device but a set of devices to reduce the computational burden of a single device and eliminate the single point of failure found in the centralised architecture. All network traffic is distributed to a set of devices for data processing to determine the occurrence of a DDoS attack, where the placement of these devices can affect the scalability and accuracy of the system [2]. The DDoS attack detection system proposed by Peng et al. [52] is able to detect

[6] Sampling based detection takes a portion of flows within a time interval for analysis

a wide range of DDoS attacks quickly using a multi-agent system for distributed attack detection as compared to the centralised method.

Attack Incidents

The size of DDoS attacks has grown significantly over the years. Motivated mostly by political, financial, and personal gains, thousands of these attacks are launched on a daily basis [43, 53]. In this section, the impact from some of the prominent DDoS attack incidents occurring in the past are described.

(a) Spamhaus Attack
The Spamhaus Project, an international organisation for tracking spam and malware on the Internet, was struck by a DDoS attack with a peak of 300Gbps in March 2013. This particular attack caused an overload of the Spamhaus servers, and created massive congestion on the Internet [54]. By using the Open Resolver DNS to send a massive amount of attack traffic to Spamhaus servers, the attacker managed to cripple the Spamhaus network for over a week.

(b) ProtonMail Attack
ProtonMail, an encrypted email service based in Europe was attacked by a DDoS attack which caused its primary data centre to go offline. This particular DDoS attack managed to keep the ProtonMail offline by congesting all of the upstream Internet Service Providers that were connected to the data centre [55]. As the attacker continuously sent the attack traffic to keep ProtonMail from being connected for more than a week, the latter was prompted to use BGP redirection as a way of mitigating the attack. During the attack, over 100 companies faced collateral damage due to the congestion of the ISPs.

(c) Dyn Attack

Dyn, a service provider company, was struck by a DDoS attack with a peak exceeding 1.2 Tbps via the Mirai botnet [56]. The series of attacks from an estimated of 100,000 bots managed to take down hundreds of popular websites such as Netflix, Github, and Twitter.

2.2 Entropy-based Features

In the recent five years, a lot of attention has been given to the inclusion of entropy in statistical-based methods for DDoS attack detection.

2.2.1 What is Entropy?

Entropy measures the uncertainty of the information provided. In network traffic, entropy captures the unusual distributional changes of traffic features in a single value [57], where sufficient observation of the changes in value can distinctly reveal the anomalies in the network. In addition, entropy can be used to generate useful features for classifying attack and normal traffic.

Generally, entropy-based features are formed by applying the entropy measures to raw traffic features such as the source and destination IP addresses, source and destination port numbers, as well as the protocol identifier. For example, with regards to the entropy of source IP address, a high entropy value would indicate a high variation, while a low entropy value denotes a low variation in the traffic packets' origins. This is useful for attack detection since a typical DDoS attack with a large number of attack sources and a single target usually has a high variation of source IP addresses and low variation of destination IP addresses as compared to the normal traffic. The entropy-based features that are formed by using the variations of two entropy-based features are known as *entropy variation features* [21].

2.2.2 Entropy Measures

Some of the most common entropy measures used in the detection of DDoS attacks are the Shannon [34], Tsallis [35] and Rényi [36] entropies.

- **Shannon entropy** was introduced by Claude E. Shannon back in 1948 for the purpose of quantifying the information gain and reducing uncertainty in communication. It was widely adopted in the classification of network traffic and can be defined in the context of a probabilistic model as shown in equation 2.1.

  $$H(X) = -\sum_{i=1}^{N} P(x_i) log_2 P(x_i) \tag{2.1}$$

  In equation 2.1, $H(X)$ represents the entropy of X. $X = x_1, ..., x_N$ is a finite set where each element has a probability of $p(x_i)$.

- **Rényi entropy** was introduced by Alfred Rényi as a type of entropy that is generalised from Shannon entropy as shown in equation 2.2.

  $$H_\alpha(X) = -\frac{1}{1-\alpha} log_2 \sum_{i=1}^{N} p_i^\alpha \tag{2.2}$$

  In equation 2.2, $H_\alpha(X)$ represents the entropy of X. $X = x_1, ..., x_N$ is a finite set where each element, x_i and $i = 1, ...N$ has a probability of $p(x_i)$, and $\alpha \geq 0, \alpha \neq 1$.

- **Tsallis entropy** was introduced by Constantino Tsallis in 1988 as a one-parameter generalization of Shannon entropy as shown in equation 2.3.

  $$H_q(X) = -\frac{1}{q-1}(1 - \sum_{i=1}^{N} p_i^q \tag{2.3}$$

  In equation 2.3, $H_q(X)$ represents the entropy of X where $X = x_1, ..., x_N$ is a finite set where each element, x_i and $i = 1, ...N$ has a probability of $p(x_i)$. q is the entropic index, where the limit $q \to 1$ will recover Shannon entropy.

- **Zhou entropy** was proposed by Xiang et al. to detect the low rates of DDoS attacks [19]. Much like the Tsallis and Renyi entropies, Zhou entropy is also a generalisation of the Shannon entropy as shown in equation 2.4.

$$H_\alpha(x) = \frac{1}{1-\alpha} log_2 \sum_{i=1}^{n} p_i^\alpha \tag{2.4}$$

where $\alpha \geq 0, \alpha \neq 1$.

2.2.3 Entropy-based Detection Approaches

A recent detection analysis has shown that entropy-based detection provides a better result than the other approaches [58]. The innovation of the entropy-based features comes from the effectiveness of entropy features in distinguishing an attack traffic from a legitimate traffic [58]. Compared to the traditional volumetric-based approaches, this method appeals more to researchers as well as to security professionals because of its simpler calculation, higher sensitivity as well as not being affected by network utilisation.

Gu et al. [23] have proposed the use of maximum entropy estimation and relative entropy in the detection of anomalies in the network traffic. The authors have classified the network packets under a two-dimensional packet class. In the first dimension, packets are divided according to their protocol related information such as TCP SYN, TCP RST, UDP SYN and UDP RST, whereas in the second dimension, packets are divided according to their destination port number. Once the packets are classified, feature selection and parameter estimation are used to generate a baseline distribution of the benign traffic and then followed by the calculation of their relative entropy, which is used to find the differences between the packet class distribution and baseline distribution of the network traffic.

Despite showing a high accuracy rate with very low false positive and false negative, this method requires a continuous supply of memory and computation time that is proportional to the traffic, and cannot be done in real time.

Xinlei Ma et al. [21], on the other hand, have presented a DDoS detection method by using both the Tsallis entropy and a variation of the Lyapunov Exponent. The authors have used a variation of the Lyapunov Exponent to quantify the exponent separation of the source and destination IP address entropies, where unless there has been no occurrence of IP spoofing, the source and destination IP addresses would exhibit similar entropy values during the same time interval. Although the experimental results showed a high true positive rate (TPR) and low False Positive Rate (FPR) in detecting DDoS attacks, there were no comparisons made with the other datasets.

Zhang et al. [24] have proposed an advanced entropy-based method in detecting Low-rate DoS (LDoS) attacks. The method uses three levels of threshold, with each threshold value being regularly adjusted and adapted to the network condition. Although this method can effectively distinguish legitimate traffic and flash crowds from LDoS attacks effectively, it consumes substantially more resources than those of the existing DDoS attack detection systems.

Bhuyan et al. [22] have used an extended entropy metric, which calculates the entropy difference of two traffic samples in the detection of DDoS attacks. This detection scheme uses a sampling method and three types of extended entropy features such as the source IP address, to effectively detect DDoS attacks. Although this approach showed high accuracy in detecting four classes of DDoS attack traffic variation, namely constant, pulsing, increasing, and dynamic, this method is not tested against DDoS attacks with low-intensity attack traffic such as those of Crossfire attacks.

Mousavi et al. [25] have introduced an early DDoS attack detection method, where it measures the destination IP address entropy of incoming packets within a specific time window to detect attack traffic. Once the destination IP address entropy drops below the threshold for five consecutive times, a DDoS attack is considered present in the network. The experimental results have shown the effectiveness of this method, where it is able to detect the presence of DDoS attacks as early as the first 250 incoming attack packets. However, the authors have only considered direct DDoS attack traffic targeting a single host. Other attacks such as indirect DDoS attacks or multi-target DDoS attacks are not considered, where these attacks could cause the destination IP address entropy to be higher than the threshold.

A summary of the recent approaches in entropy-based detection is shown in Table 2.1.

Paper	**Entropy Measures**	**Type**	**Launch Method**	**Intensity**	**Accuracy (%)**	**Classification Technique**
Two-dimensional Entropy Features [23]	Shannon	Host	Direct	High	95	Maximum Entropy Estimation
Lyapunov Exponent Separation [21]	Tsallis	Host	Direct	High	98.56	Threshold
Advanced Entropy-based Scheme (AEB) [24]	Shannon	Host	Direct	High Low	> 92	Threshold
Lightweight Extended Entropy [22]	Extended	Host	Direct	High Low	99.77	Threshold
Entropy in SDN [25]	Shannon	Host	Direct	High	96	Threshold

Table 2.1: Summary of Entropy-Based Detection System

2.3 Machine Learning Classification

Machine learning (ML) is a technique where the machines learn and acquire new knowledge based on the existing knowledge that improves their performance over time [59]. For that reason, machine learning approaches have been designed to counter large and sophisticated attacks, which traditional statistical approaches such as the single threshold or moving threshold were unable to handle. As there is a continuous evolution of network attack complexity in traffic classification, ML techniques were hence utilised in 1994 for traffic analysis in the detection of traffic anomalies [60]. As such, these ML classifiers learnt the traffic patterns from both the normal traffic and attack traffic through training, but without the necessity of setting and finding the best features or threshold values for detection purpose.

2.3.1 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 [61]. According to a widely cited survey paper[61], 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.

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 [61]. They found that the classifiers with the overall best performance across different datasets are neural networks, support vector machines and random forest (ensemble classifiers built from decision trees). These classifiers are described as follows:

- **Neural Networks**
  Artificial Neural Networks (ANNs) use highly interconnected net-

work models for classification and are used widely and extensively in attack detection methods. This is because ANNs have better robustness and fault tolerance in identifying both the known attack patterns and also unknown attack patterns [62]. Having the ability for providing satisfactory results in distinguishing attack from normal traffic, ANN is a viable candidate for DDoS detection [63]. Some of the typical examples of ANN classifiers are the Multilayer Perceptron (MLP) [64] and Recurrent Neural Network (RNN) [65].

- **Support Vector Machine**

  SVM is a form of supervised learning method that can be used to perform traffic analysis and pattern recognition by mapping the input feature vectors into a higher dimensional feature space. A standard type of SVM classifier is a non-probabilistic binary linear classifier, which uses labelled training data in the construction of a classification model.

- **Decision Trees**
  Decision trees use a tree-like model with nodes and leaves for classification. A node in a decision tree represents a traffic feature such as source address entropy where each node may connect to two other nodes or with a node and a leaf, while the leaf contains a decision value to determine the presence of an attack. Some examples of decision tree classifiers are Alternating Decision Tree (ADT) [66] and C4.5 [67], which are considered as one of the most scalable packet classification techniques [68, 69].

These ML-based approaches are designed to counter the sophisticated and evolving adversaries as a result of their abilities in coping with a significant amount of complex evolving data [70]. As opposed to traditional detection systems that use fixed threshold values to distinguish attack traffic from normal traffic, the ML approaches use classification models built from learned network behaviour to distinguish attack traffic from normal traffic.

2.3.2 Machine Learning-based Detection

Some of the machine learning-based detection approaches found in recent literature that are related to DDoS attack detection are reviewed in the next six paragraphs.

Gu et al. [63] have proposed the use of a multi-ANN classifier with an enhanced genetic algorithm (GA) for structural parameter optimisation and principal component analysis (PCA) in the improvement of its feature extraction for DDoS attack detection. By using old and up-to-date datasets (patterns) in the training process, this approach is able to detect both known and unknown DDoS attacks that have similar patterns to the training set.

Saied et al. [71], on the other hand, have used ANN to flag the known and unknown TCP, UDP, and ICMP attacks from normal traffic. The authors have compared the detection results when training with both old and up-to-date datasets, where improper training of old patterns showed poor detection results. This approach showed better accuracy, sensitivity, specificity, and precision levels than other traditional detection systems such as Snort, Probabilistic Neural Network (PNN) and Back-Propagation (BP).

In another study, Li et al. [62] have proposed the Learning Vector Quantisation (LVQ) neural network, which is a supervised version of quantisation used for pattern recognition, multi-class classification, and data compression tasks for detecting DDoS attacks. The dataset used in the experiments has been converted into numerical form and given as input to the neural network.

Horng et al. [72] have proposed using an SVM-based detection system that pre-processes the dataset through a hierarchical clustering algorithm prior to training the dataset with an SVM classifier. The hierarchical clustering algorithm is able to reduce the size of the training dataset, while sustaining the quality of the original dataset. Therefore, this approach greatly reduced the training time and achieved better detection accuracy results.

Ramamoorthi et al. [73] have proposed a real-time detection system by using an enhanced SVM (eSVM) string kernel as a way of classifying the incoming traffic flow as an attack or normal flow. In this approach, a normal profile is constructed from the user's access behaviour and it is then used by eSVM to build a model file for classifying the attack from the normal flows. Although the experimental results showed 99% accuracy in the detection of application and network DDoS attacks, eSVM is not able to support other DDoS attacks such as port scanning and DNS spoofing.

Wu et al. [74], on the other hand, have adopted a decision-tree classifier (C4.5) in their DDoS detection mechanism, where the C4.5 algorithm is used in the selection of attributes for splitting the data into further smaller subsets. The splitting procedure is repeated until all of the data in the subset belongs to the same class, or the same gain ratios of all the attributes are attained.

A summary of the mentioned machine learning-based detection systems is shown in Table 2.2, where these works and with the exception of ANN, had mostly focused on a single attack intensity, for instance in either high or low-intensity attacks [71].

2.4 Correlation Analysis

Correlation analysis is a method for finding a relationship or connection between two or more random variables. In network traffic analysis, researchers normally use correlation analysis for determining traffic feature similarities of two or more distinct traffic flows. These traffic features can be arrival time, packet volume, protocols, packet size, or source and desti-

Paper	**Attack Type**	**Launch Method**	**Attack Intensity**	**Accuracy (%)**
Multi ANN with GA [63]	Host	Direct	High	97.68
ANN [71]	Host	Direct	High Low	98
Learning Vector Quantisation (LVQ) Neural Network [62]	Host	Direct	high	99.72
Support Vector Machine (SVM) [72]	Host	Direct	High	99.5
Extended SVM [73]	Host	Direct	High	99.32
C4.5 Decision Tree [74]	Host	Direct	High	>90*

** Note that this is an estimation of its True Positive Rate since this paper only provides detection results in terms of False Positive and False Negative.*

Table 2.2: Summary of Machine Learning-based Detection System

nation addresses. DDoS attack traffic is assumed to be similar to the normal traffic because of the underlying similarity shown between the synchronisation and coordination within the DDoS attack traffic.

Correlation analysis has been widely used in collaborative detection systems as information at a single point or from a single traffic feature would not be sufficient to indicate the presence of attack traffic in the network, particularly in cases where legitimate traffic from many sources is used for launching a DDoS attack.

One of the advantages of using this method is that it can identify traffic flows that belong to a particular botnet by using the similarities existing between the flow traffic patterns and behaviour. Traffic flows that originate from the same botnet are similar because of the coordination and synchronisation within the bots. For example, Yu et al. [75] who used the flow correlation coefficient to distinguish attack flows from flash crowd traffic, found that the aggregated attack flows sent by the bots from the same botnet have similar standard deviations with those from the original attack flow. In another example, Wei et al. [76] used rank correlation-based detection for detecting attack traffic in the network, observed that the response flows as having an inherent relationship.

2.4.1 Correlation Algorithms

A popular way for measuring correlation is by using a correlation coefficient, where the value of the correlation coefficient, r, is always between -1.0 to +1.0. A positive value (larger than zero) would indicate a positive relationship between the two random variables, while a negative value (smaller than zero) denotes a negative relationship between the two random variables. A zero value implies the absence of a relationship between the two variables. The following is a list on some of the correlation algorithms for DDoS detection:

- **Pearson Product-Moment Coefficient**

  Karl Pearson introduced the Pearson's product-moment coefficient in 1895, where it is commonly used for measuring linear associations and relationships between two variables of X and Y. A positive correlation would mean a rise of X with increasing Y, while a negative correlation would denote otherwise. A zero correlation, on the other hand, implies no changes of Y when the value of X increases. This equation is shown in Equation 2.5.

$$r = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{[\sum(X_i - \bar{X})^2 \sum(Y_i - \bar{Y})^2]^{1/2}} \tag{2.5}$$

- **Spearman's Rank Correlation Coefficient**

  Charles Spearman introduced the Spearman's rank correlation coefficient in 1904 as a method for measuring the relationship between non-parametric variables, where it converts the variable's value into a ranked value before using the Pearson's product-moment coefficient in Equation 2.5 as part of its correlation value calculation.

- **Flow Correlation Coefficient**

  The Flow correlation coefficient is employed in some of the DDoS attack detection systems and was used to correlate the following information:

  - two or more features such as arrival time, packet volume, protocol, and packet size [29, 30].
  - two or more distinct traffic flows [31]
  - two or more alerts from detection systems [32]
  - detection results taken at a different time [33]

2.4.2 Correlation-based Detection

Although the success of correlation analysis techniques has led to the development of several correlation-based DDoS detection systems [77, 30, 78, 28], these systems focus on correlating information from either the neighbouring links [77, 28] or at a single link or host [30]. As they lack the ability of sharing link information such as the origin points of the attack flows between detectors at various locations, this reduces the amount of correlated information and consequently, lessens the risk of misdetection. This is especially critical for DDoS attacks that use stealthy (low-rate, indirect, persistent) attack traffic, where they are difficult to distinguish given the current detection systems. As such, this would require a wider network of information, since the information between two links that are far apart can be correlated for detection improvement (i.e. with a wider view of information content). Moreover, correlating link information nearer to the attack source can help in the early identification of the attack flows, which is seen as a crucial way for reducing the attack impact on the network.

The detection results of the network traffic will be different across different viewpoints in the network due to the different detection nodes' deployment locations [79, 80]. The attributes in each network packet at different locations and network levels will differ significantly as well since the detection methods that might be effective in some viewpoints may fare poorly in others. For this reason, a correlation of data from multiple viewpoints of the network may increase the accuracy level of the detection system. Since each of the different viewpoints consists of a detection node that is connected to a communication mechanism, the correlation of these results can help to overcome the limitation of a single viewpoint detection and consequently lead to a more accurate detection system. This method can also help to overcome entropy spoofing issues since it would be difficult for the attacker to spoof the traffic entropy at multiple locations.

Some of the correlation-based detection approaches that have been proposed in the literature are discussed in the next few paragraphs.

Li et al. [81] have proposed a correlation approach for detecting the attack at the backbone of the network through an alarm trigger even without locating the source of the attack. In this research, the authors extracted anomalies from the network-wide traffic for correlation analysis to detect changes indicative of an attack.

Ning et al. [82] on the other hand, have presented a technique by integrating two complementary types of alert correlation methods that not only enhances intrusion alert correlation efficiency, but also reduces the false positive rate. The first method is an alert correlation that was based on similarities between alert attributes, while the second method is an alert correlation that was based on the prerequisites and consequences of attacks.

In another study, Chen et al. [83] have proposed a collaborative system that was based on correlation, where each of the detection program on a router shares alert information on the abrupt changes in packet volumes to other routers of multiple domains. Since DDoS attack traffic will cause routers across multiple domains to raise a significant number of alerts, these shared alerts will then be used in a tree construction, where an attack will be detected when the tree is sufficiently large and exceeds a predefined threshold. Although the tree represents the attack traffic path, where a DDoS attack can be detected at the earliest time possible, their evaluation was only based on traditional DDoS attacks such as TCP SYN flooding, ICMP flooding, UDP flooding, and Smurf attacks.

François et al. [28] have proposed to use multiple Intrusion Protection Systems that form the overlay networks of protection rings around its customer called Firecol, where it correlates the aggregated traffic by measuring its deviation from the stored traffic profiles. This detection mechanism is not only free from false positives, but was also able to detect attacks near to the attack source at the ISP level. However, since Firecol has only focused on monitoring for the traditional direct DDoS attack traffic in the protection rings surrounding its customer, this detection mechanism may not be useful in the detection of an indirect DDoS attack.

A summary of these correlation-based detection systems is shown in Table 2.3.

Paper	**Type**	**Launch Method**	**Intensity**	**Early Detection Capability**	**Scalability**
LinkScope [77]	Link	Direct	High Low	No	Yes
Multivariate Correlation Analysis [30]	Host	Direct	High	No	Yes
Flow Correlation Analysis [78]	Various	Various	Various	No	No
Traffic Profile Deviation [28]	Host	Link	Direct	Yes	Yes
Global Correlation Coefficient [81]	Host	Direct	High	Yes	Yes
Correlation Graphs [82]	Host	Direct	Link	No	No
Distributed Change-Point Detection [83]	Host	Direct	High	Yes	Yes

Table 2.3: Summary of Correlation-based Detection System

2.5 Software-Defined Networking

Software-Defined Networking (SDN) is an emerging technology that offers the opportunity for creating a flexible and programmable network [84, 85]. This technology not only creates flexibility by decoupling the control plane[7] and the data plane[8] in network devices to allow the addition of new abstractions in the network, it also creates programmability, where network devices such as a switch can be programmed as either a controller or a forwarding switch in the network instead of the default functionality set by network vendors.

Some DDoS attack detection systems have adopted SDN technology to provide a global view of the network and reduce traffic overhead [86, 87, 88]. This global view of the network is achieved by connecting all switches in the network directly to a centralised controller, where any unknown traffic that is not listed in the switches' flow tables will be sent to the controller to determine its path [88]. On the other hand, traffic overhead in the network is reduced due to the separation of control and data planes, which allows decision making and data processing to be done on the control plane and traffic forwarding to be done on the data plane. With a global view of the network and reduction in traffic overhead, SDN technology can improve the scalability of a DDoS attack detection system.

2.5.1 SDN-based Detection

Some of the recent SDN-based DDoS attack detection approaches are discussed in this section.

Chin et al. [86] have proposed the use of a collaborative anomaly and

[7] The control plane is mostly used as the management plane for managing the connections between switches in the network and determining the paths of each new packet to its destination [84].

[8] The data plane is mostly used for user traffic to travel to its destination using the forwarding table managed by the SDN controller [84].

signature-based detection method through SDN as a way of addressing the limitations in traditional approaches to process large network traffic in real-time. This approach constantly monitors the network for anomalies using distributed monitors installed on SDN switches. Once an anomaly is detected, an alert is sent to the correlator to perform deep packet inspection and then triggers the SDN controller to mitigate the attack. Although this method can process a large amount of network traffic in a short period of time, it has only been evaluated on the TCP SYN attack and has not been considered for other types of attacks.

Lin et al. [87] have proposed an extended SDN architecture of Network Function Virtualisation (NFV) modules that can reduce traffic overhead to the SDN controller to improve scalability. As traditional network devices are inflexible and not programmable, the separation of control plane and data plane has enabled packet processing for DDoS attack detection to be done on the controller to improve efficiency and scalability. This approach further improves the scalability of the SDN-based detection system by performing most of the packet detection processes on OpenFlow virtual switches and NFV modules and only sending the outcome to the controller. As compared to conventional SDN architectures, the use of both OpenFlow virtual switches and NFV to perform traffic classification and deep packet inspection has reduced the workload and traffic overhead to the SDN controller hence increasing the overall scalability. However, one of the limitations of this method is its lack of early detection capability and information sharing.

Zheng et al. [89] have proposed to use the Commercial Off-The-Shelf (COTS) SDN switches in the detection and throttling of DDoS attacks in real time without incurring additional deployment cost. This method has been designed to detect the indirect link flooding of DDoS attacks such as the Crossfire attack through the use of adaptive correlation analysis. The use of COTS SDN switches in this case enables a selective collection of flow information without the need of new appliances, as opposed to non-SDN based approaches. Although this architecture is somewhat similar to the SCAFE DDoS detection architecture proposed in this thesis, it is important to note that this work was published in July 2018 and had run in parallel with the development of SCAFE. The limitation of COTS however, is that it is based on a single controller SDN system, where information sharing between multiple controllers is not considered. In contrast, SCAFE uses a collaborative approach, where it takes information from multiple controllers and stores it in a centralised database for information sharing.

A summary of these SDN-based DDoS attack approaches is shown in Table 2.4.

Paper	**Attack Target**	**Launch Method**	**Intensity**	**Early Detection Capability**	**Information Sharing**
Collaborative Architecture [86]	Link	Direct	High	No	Yes
Extended SDN Architecture [87]	Host	Direct	High	No	No
COTS SDN Architecture *[89]	Link Host	Indirect Direct	High Low	No	No

Table 2.4: Summary of SDN-based DDoS Attack Detection System

2.6 Summary

In this chapter, we have not only studied and reviewed the basics concepts of the DDoS attacks, but also on the four main DDoS attack detection approaches, namely entropy-based, machine learning-based, correlation-based, and SDN-based detection approaches.

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

(a) All four DDoS attack detection approaches concentrate mostly on improving the accuracy of detecting direct, high-intensity, and host-based DDoS attacks.

Very little attention is given to detecting indirect, low-intensity, or link based DDoS attacks accurately. A generalised method that can be used to detect all types of attacks, in particular, indirect, direct, link-based, host-based, high intensity and low intensity DDoS attacks is worth investigating to improve overall detection accuracy and prevent misdetection.

(b) Current DDoS attack detection approaches are lacking early detection capability for detecting various types of DDoS attack traffic accurately.

Current DDoS attack detection approaches mostly focus on detecting DDoS attack traffic locally or within the neighbouring networks. Since the impact of DDoS attack is getting larger, it is important to be able to address the attack beyond local and neighbouring networks, as close to the source of attack as possible.

(c) Current DDoS attack detection approaches lack information sharing capability among network devices in a large-scale network for detecting various types of DDoS attacks early.

Most of the entropy-based DDoS detection approaches focus on detecting DDoS attack traffic from the information obtained from a single device or location such as those from routers, switches, or other DDoS detection components. One of the limitations of using this approach is that its early detection capability would be compromised by the time needed for gathering the information necessary to detect DDoS attacks. At this point, where sufficient information has been gathered, a significant amount of damage to the network would have already occurred. The ability to obtain information from a wider network, such as from multiple locations, is crucial to improve the early detection capability of a DDoS attack detection system.

Chapter 3

Design of a Scalable and Fault-Tolerant DDoS Detection System for Large-Scale Networks

This chapter focuses on answering the first sub-question of this thesis:

> *How can we improve the early detection capability of large-scale DDoS attack detection with scalability?*

To answer the above question, this chapter presents the design and framework for developing a **SCA**lable and **F**ault-tolerant DDoS detection system architecture for **E**arly detection (**SCAFE**) in large-scale networks. This architecture is designed for detecting both the high and low-intensity DDoS attacks at an early stage as well as nearer to the source of attacks with high scalability and fault tolerance capability. An effective DDoS detection system has to be scalable[1] and fault-tolerant[2] for it to effectively

[1] Scalability is defined as the ability of the system to process traffic with increasing number of traffic flows and network links, where the processing time and message complexity increases linearly

[2] Fault tolerance is defined as the ability of the system to continue to operate despite the presence of errors or failures.

detect the attack flows[3] from a large number of traffic flows and network links early. The system's scalability can be measured through its processing time in detecting attack traffic.

SCAFE is a two-level DDoS attack detection system architecture that scales well with regards to the increasing number of links and traffic flows. This architecture operates by carrying out coarse-grained detection that requires a shorter processing time to narrow down the potential target link as well as fine-grained detection, which requires a longer processing time for detecting the attack flows. In addition, SCAFE also supports network-wide traffic monitoring and correlation analysis mechanisms that can scalably handle large networks and traffic volumes via separate communication paths between the detection components and network data links. Additionally, it leverages the SDN technology by reducing message complexity as well as simplifying information sharing by decentralising its traffic collection and processing in the identification of the attack flows in the network.

The remaining sections of this chapter present the design requirements, architecture overview, the components' roles and relationships, as well as the communication paths and message interfaces.

3.1 System Architecture Design

This section discusses the design assumptions as well as the goals and principles that have been taken into account in achieving high scalability and fault-tolerance architecture design for early DDoS attack detection.

[3] A traffic flow is a group of traffic that have the same destination address at a particular time interval.

3.1.1 Design Assumptions

A1 - A large network with the sub-networks spread across multiple locations

Several reasons have contributed to the occurrence of DDoS attacks in a large network with multiple sub-networks connected by routers. Firstly, networks in large corporations such as Microsoft and Google, have hundreds of thousands of servers spanning across hundreds of different data centres across the world that require strong protection against cyber attacks including DDoS attacks [90]. Apart from originating from many sources across multiple distant locations [12, 13], DDoS attacks are also hard to detect without considering the long-distance communications. Thirdly, the current large networks [91] have been a frequent target for DDoS attacks. Since the attack traffic can originate from sub-networks located in different locations, an early detection is required for detecting the attack traffic nearer to the attack source. For this reason, although many of the recent DDoS attack detection system architectures have been designed to function in a large network, not many of these approaches use network-wide correlation analysis to distinguish the attack from normal traffic flows.

A2 - Synchronised and coordinated DDoS attack traffic using automated tools

This thesis assumes that the attackers use a botnet (a network of bots) to launch DDoS attacks. Recent top large-scale botnet-based attack traffic [92] is often generated by using automated tools such as LOIC [93], TFN [49], and MStream [50] to synchronise and coordinate the attack. Without synchronisation and coordination enabled by these tools, DDoS attack traffic cannot be aggregated properly to become an effective DDoS attack. Such highly synchronised and coordinated behaviours in network traffic are the key indicator of DDoS attacks

[30, 78, 75].

A3 - Indirect & direct DDoS attacks

This thesis assumes that DDoS attack traffic can be sent directly or indirectly to the victim. In the case of an indirect attack such as the Crossfire attack, attack traffic is forwarded to decoy servers to flood the network links attached to the victims[12]. This attack is also persistent because attack traffic is carefully coordinated to use stable routes where the route change mechanism will not be triggered. On the other hand, direct attack traffic is sent directly to the victim's server or network. A good system architecture should be able to detect both direct and indirect attack detection capability.

3.1.2 Design Goals

With the design assumptions mentioned earlier, this thesis aims to develop a DDoS attack detection architecture that meets the following goals:

G1 - Fully distributed monitoring mechanism to identify potential target links

A large network consists of many sub-networks with different locations. These sub-networks are connected through routers and links that can be the target of a DDoS attack. Target links usually have a higher concentration of attack flows than other links in the network due to the aggregated attack flows going to the target links. To identify a target link, a good detection system needs to continuously monitor every single link in the network for anomalies efficiently. However, monitoring a large number of links incurs high processing overhead to the monitor. Thus, a fully distributed monitoring mechanism, where the monitoring load is spread out over several monitors in the network, can help to minimise the processing overhead to a single monitor [94]. Moreover, a fully distributed design prevents a single point of failure when a fault occurs in the network. This design is desired in a DDoS attack detection system, because it avoids the reliance on a central unit to handle all the monitoring tasks and has the ability to scale better than a centralised design.

G2 - Centralised correlation analysis mechanism to identify attack traffic

DDoS attack traffic must be highly coordinated and synchronised for the attack to be successful. During a DDoS attack, a network link may carry both attack flows and legitimate flows. A highly distributed attack that generates low-intensity attack traffic is similar to normal traffic and hard to identify. Correlation is a good measure to identify identical traffic patterns and behaviour coming from multiple sources [95]. In addition, correlation can distinguish between attack traffic and flash crowd traffic because flash crowd traffic is found to be less correlated with other flows in the network [75].

Since attack traffic comes from multiple locations, it is essential to have a logically centralised correlation analysis mechanism to correlate information from all locations in the network. Note that this limits the scalability. For a fully scalable system, the correlation analysis mechanism can be distributed to several components during implementation to reduce communication and processing overhead in the correlator.

G3 - Separate control network for the detection system to ensure scalability with the increasing number of traffic flows or network links

A typical DDoS detection system requires the collection of traffic data in the network. Collecting traffic data from each device in the network will usually incur communication overhead. Moreover, the communication overhead typically increases the network size or traffic volume in the network as more data is being collected. This means that the communication overhead may impact network performance and unintentionally constitute an attack on the network when the communication overhead gets too large. Performing traffic data collection using a separate network or communication link to the main communication links is one way to address this issue.

3.1.3 Design Principles

This section explains some of the key principles that influence the design of SCAFE towards achieving the design goals mentioned in the previous section.

P1 - Scalability in handling traffic of various sizes

SCAFE needs to be scalable to handle networks with a large number of traffic flows and network links. In this thesis, SCAFE is designed in a way that its processing time will increase at most linearly with the number of traffic flows or network size. The scalability of SCAFE is measured in terms of processing time and message complexity.

P2 - Modularity in system architecture components

SCAFE components need to be modular in the sense that each component in the detection system is an individual building block of the detection system with specific functionality. This means that each of the components is designed with a well-defined interface and can be replaced or upgraded when necessary. Each component is created separately and has separate functionality. For example, a collector is used for collecting traffic statistics, a monitor is used for traffic monitoring, a correlator is used for correlation analysis and a database is used for storing statistical information obtained from the collectors.

P3 - Fault tolerance and availability

SCAFE must be fault tolerant with high availability so as to safeguard the continuous functionality of the detection system and to avoid having a single point of failure. This means that SCAFE needs to be designed in a way that it can handle both software and hardware faults such as a server crash, disk failure, or network failure when handling large traffic flows and network links. Each of the components in SCAFE will need to be actively replicated to ensure that SCAFE is able to function smoothly and in a fault-tolerant manner.

3.2 Architecture Overview

As shown in Figure 3.1, SCAFE is designed to include four major components: collector (packet and flow statistics collectors), monitor (local monitors), database (local and global databases) and a correlator. These components are built on SDN controllers and also Network Function Virtualisation (NFV) servers in a large network which consists of several sub-networks connected through the routers.

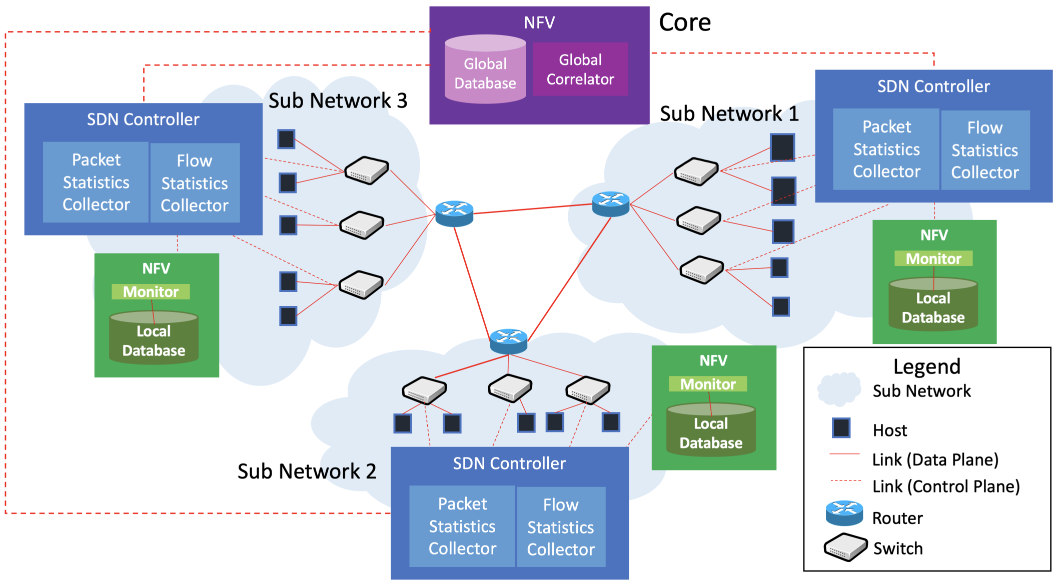


Figure 3.1: The SCAFE Architecture

There are two-levels of granularity in SCAFE, namely, coarse-grained detection and fine-grained detection. The coarse-grained detection is the first level of the detection system, where SCAFE identifies potential target links by collecting port statistics from switches. On the other hand, the fine-grained detection is the second level of the detection system, where identification of attack flows is performed by correlating the flow statistics found in target links with the flows in other edge links in the network.

3.2.1 Level 1 - Target Link Identification

The first level (Level 1) is called *target link identification*. The logical representation of this level is depicted in Figure 3.2.

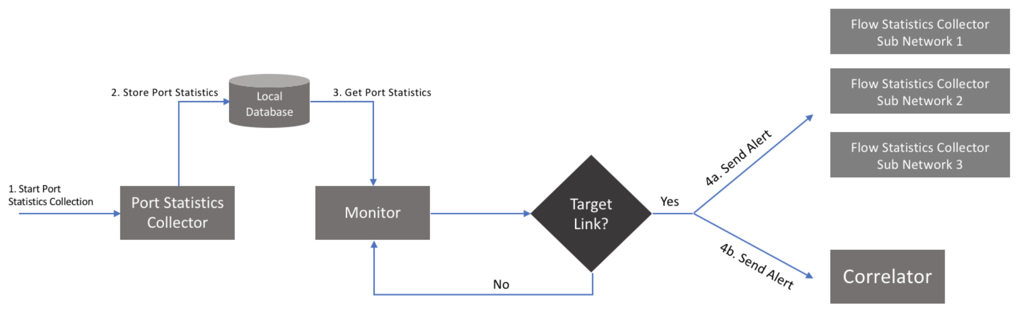

Flow Statistics Collector Sub Network 1
Flow Statistics Collector Sub Network 2
Flow Statistics Collector Sub Network 3
2. Store Port Statistics
Local Database
3. Get Port Statistics
1. Start Port Statistics Collection
Port Statistics Collector
Monitor
Target Link?
Yes
4a. Send Alert
4b. Send Alert
No
Correlator

Figure 3.2: Logical Flow of Level 1

This level performs a continuous local monitoring of network links within each of the sub-networks in the network and detects signs of congestions with regards to traffic volume[4]. These signs of congestion can be obtained by inspecting the switch/router port statistics collected as shown in Table 3.1. Once any of these port statistics exceed a predefined threshold of either a traffic volume or rate of change, the network link is considered as a potential target link.

[4] Traffic volume is the total byte count or total packet count of incoming traffic on a port which does not include the dropped byte or dropped packet count

No	Port Statistics	Definition
1	bytes_in	Number of incoming bytes into the switch/router port for each second
2	bytes_out	Number of outgoing bytes from the switch/router port for each second
3	packets_in	Number of incoming packets into the switch/ router port for each second
4	packets_out	Number of incoming packets into the switch/ router port for each second
5	dropped_in	Number of dropped packets at the incoming port of the switch/router port for each second
6	dropped_out	Number of dropped packets at the outgoing port of the switch/router for each second
7	error_in	Number of erroneous packets at the incoming port of the switch/router for each second

Table 3.1: List of Port Statistics Collected and their Definitions

Level 1 supports the coarse-grained detection by narrowing down the potential target link that contains a high amount of aggregated attack traffic. One of the limitations of the coarse-grained mechanism is that it would not be able to immediately confirm if any of the traffic flows in the detected links is an attack flow. This is because the signs of congestion or rate of change in traffic volume can also be caused by normal traffic, such as during flash crowd events. Moreover, it is also not wise to conclude that all flows in the link are attack flows, since network links in the early stages of an attack would likely contain both attack and normal flows. Although there is a likelihood of *false positive* results arising from the presence of normal traffic in the target link, this level is still regarded as the first step for narrowing down the search of DDoS attacks flows in the network. When a potential target link is detected, SCAFE will then trigger an alert to initiate the second level of the detection system, where it analyses traffic flows in the target link to detect attack flows.

3.2.2 Level 2 Attack Flow Detection

The second level (Level 2) of SCAFE is called *attack flow detection*. The logical representation is depicted in Figure 3.3.

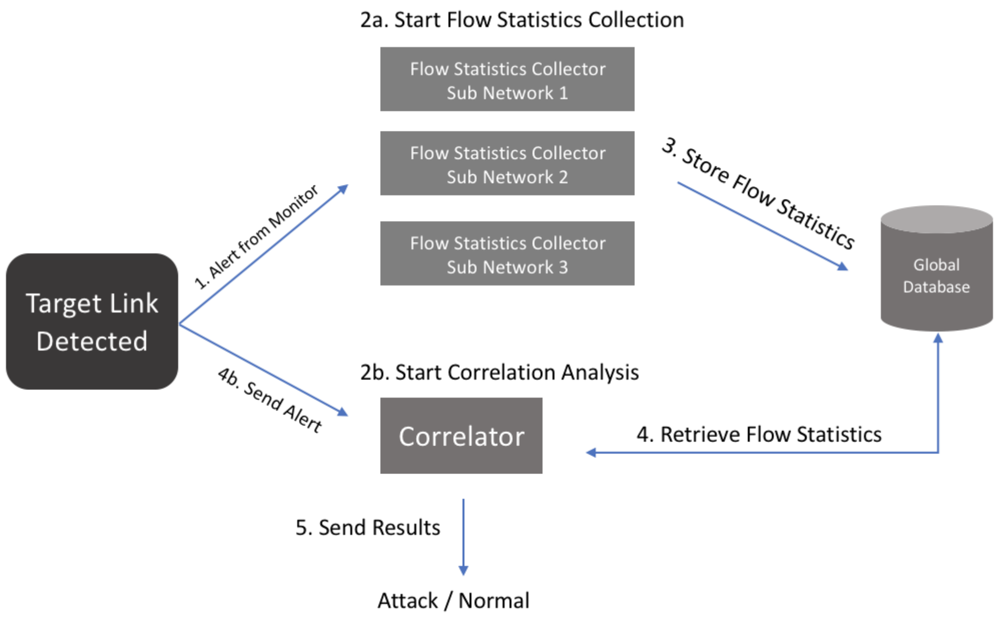

2a. Start Flow Statistics Collection
Flow Statistics Collector Sub Network 1
Flow Statistics Collector Sub Network 2
Flow Statistics Collector Sub Network 3
1. Alert from Monitor
3. Store Flow Statistics
Target Link Detected
Global Database
4b. Send Alert
2b. Start Correlation Analysis
Correlator
4. Retrieve Flow Statistics
5. Send Results
Attack / Normal

Figure 3.3: Logical Flow of Level 2

This level is used to detect DDoS attack flows found in the identified potential target link in Level 1. Once SCAFE receives an alert from the target link identification mechanism, it will then trigger the collection of traffic flow statistics for correlation analysis. The flow statistics collected at this level will consist of information obtained from the flow tables of each SDN switch in the network as described in Table 3.2. At this stage, correlation analysis of traffic flows in the potential target link and traffic flows from other network links located in the perimeter at the edge of the network is performed to measure their statistical similarities. Apart from identifying if the traffic flow is an attack or normal flow, these statistics can also be used to identify the edge link that is closest to source of the attack flow.

No	Flow Statistics	Definition
1	bytes_count	Total bytes count of a single traffic flow per second
2	duration_sec	Duration of the traffic flow
3	length	Length of traffic flow
4	packets_count	Number of packets in a single traffic flow per second

Table 3.2: List of Flow Statistics Collected and their Definitions

Level 2 is designed to perform fine-grained detection because of its ability to distinguish attack from normal flows. Unlike existing approaches that correlate the upstream and downstream network links, SCAFE extends the correlation method by correlating network links that are located further away in other sub-networks such as network links at the edge of the network.

The network-wide correlation analysis method enables aggregated attack traffic to be detected early at the edge before it reaches its victim. Traditional detection approaches only identify attack traffic near to the target link but not at the network links that are further away from the targeted links. Note that this approach is designed primarily to detect external DDoS attacks but may not be as useful in detecting internal DDoS attacks originating within the network. This is because internal DDoS attacks are less common and the attack sources are often highly distributed.

3.3 Component Roles and Relationships

The four major components in SCAFE are: collector, monitor, correlator, and database. These components interact with each other via a separate link in the network than the link used by the user traffic. Through the separation of control plane and data plane in SDN, these components are able to communicate on the control plane where user traffic travelling on the data plane will not be impacted by the components' communication overhead. In addition, these components are replicated to allow SCAFE to tolerate faults occurring in the network. This means that each component has redundant pairs and replication will be triggered when a fault occurs. The cost required to enable these components to communicate via the control plane and to perform replication would be the cost of implementing SDN-enabled devices, network cabling to connect SDN-enabled devices to the SDN controller, and redundant pairs of each SCAFE's components on the network.

3.3.1 Collector for Traffic Information Collection

A collector only communicates with local switches and controllers (in an SDN network) for the purpose of obtaining both the port and flow statistics. These statistics are then stored in databases for retrieval by local monitors and the global correlator. In this case, the collectors periodically communicate with switches, controllers, and databases through separate management links instead of data links that carry user traffic. A separate link helps to avoid extra communication overhead to the normal network communication. SDN enables the separation of data plane and control plane on the switch, where the latter is used in the polling of statistics from switches in the network. This will ensure that the current network performance (packet delay and congestion rate) will not be affected by SCAFE's operation.

3.3.2 Monitor for Target Link Identification

In SCAFE, the monitor is not only used for monitoring port statistics collected by the collector, but also for sending alerts to the correlator once a potential target link is detected. Each of these sub-networks has a local monitor that watches the links in the sub-network and continuously retrieves port statistics data from the local database and compares them with the predefined thresholds. Whenever a network link's port statistics exceed the thresholds, the network link is identified as a target link that has the potential to contain DDoS attack traffic flows.

By communicating with the database, the monitor retrieves the port statistics for every n time interval in the sub-network, where n denotes the time in seconds. For example, when n=60, this means that the monitor will send a request for port statistics retrieval at an interval of every 60 seconds. The time interval is selected or determined depending on the criticality of the network. Compared to a longer time interval, a shorter interval may result in a more accurate detection because of the higher collection of traffic information. However, a short interval can lead to frequent communication with the database, which increases the communication overhead. At each retrieval time, all of the port statistics will be collected between the current time, t, and the previous retrieval time, $t - n$. After the port statistics are retrieved, the monitor will then examine if these statistics have exceeded any of the predefined thresholds. Those statistics that exceeded the predefined threshold will then be identified as a potential DDoS attack target link. In this thesis, the time interval used is n=1, which is the default time interval in SCAFE. The time interval setting is predefined and changes in the network criticality would require manual alteration to the time interval. An adaptive time interval could be used to improve the efficiency of SCAFE, however, it is not studied in this thesis.

Algorithm 1 shows the pseudocode for target link identification. The same algorithm will be installed in every switch and router in the network. The algorithm starts by retrieving port statistics from the local database based on $switchID$ and $portID$. $switchID$ is the MAC address of a switch whereas $portID$ is the port number of a port on a switch. Next, the statistics of each $portID$ is compared to a predefined threshold, $threshold$. If any of the port statistics ($packetcount$, $bytecount$, $length$, or $duration$) is larger than or equal to $threshold$, the link connecting to the $portID$ is identified as a target link, $Link\_attack = 1$. Otherwise, the link is considered as a normal link, $Link\_attack = 0$.

When a target link is detected, the monitor will then alert the correlator in the network with the $switchID$ and the $portID$ of the potential DDoS attack target link to start Level 2 of SCAFE.

Algorithm 1 Target Link Identification in Monitor

```
Input:
  – port statistics: retrieved from the local database
Output:
  – $Link\_attack = 1$: potential attack target links
  – $Link\_attack = 0$: otherwise
  Level 1: Target link Identification in every switch/router
1: while $true$ do
2:     $getPortStatistics(switchID, portID, packetcount, bytescount, length, duration)$
3:     for $every\ switchID$ do
4:         for $every\ portID$ do
5:             if ($packetcount \,||\, bytescount \,||\, length \,||\, duration$) $\geq threshold$ then
6:                 return $Link\_attack = 1$
7:             else
8:                 return $Link\_attack = 0$
9:             end if
10:        end for
11:    end for
12: end while
```

3.3.3 Correlator for Attack Flow Detection at Edge Routers

In SCAFE, a correlator is used to correlate the flow statistics obtained from the potential target link identified by the local monitors and flow statistics from the edge links. The correlation analysis process can find similarities by correlating statistical traffic patterns in both target and edge links' traffic flows to identify attack flows. SCAFE correlates flow information between the target link and edge links instead of all other links as a way of reducing the number of links needed for correlation. Edge links are sufficient for detecting attack flows because they are the first link an attack flow passes through from an external network to reach its target. Therefore, links between edge links and target link will possess similar traffic patterns hence correlation with these links is redundant.

The correlator requires an alert from any of the local monitors to begin the correlation analysis process. This is because correlation analysis requires a large amount of processing power to compute the correlation coefficient of each pair of flows (a flow from the target link and a flow from the edge link). Since the correlator only starts correlation analysis whenever an alert is received, unnecessary correlation analysis can be avoided during a non-attack period. Avoiding these unnecessary correlations not only can improve efficiency of the detection system, but also can improve the scalability of the DDoS attack detection system. SCAFE can still be scalable with an increasing number of network links in the network because the additional network links might not be edge links hence not increasing the processing overhead of correlation analysis.

Algorithm 2 shows the pseudocode of the flow correlation analysis in the correlator. The correlator starts flow correlation analysis when the correlator receives an alert from any of the monitors in the network, indicating that there is an attack. The correlation begins by retrieving flow statistics from the global database of the target link ($targetID$) and also all the edge links ($edgeIDs$) in the network. For each pair of flows[5], where a flow from a $targetID$ ($flowID_i$) and a flow from $edgeIDs$ ($flowID_j$), its flow strength (fcc_{ij}) is calculated. If the calculated flow strength is higher than 0.8, then both flows are considered as attack flows. Otherwise, both flows are considered as normal flows.

A correlator only communicates with the global database for information retrieval and with monitors to receive alerts. A separate management interface is used for the communication between databases and monitors. This means that the communication will not be interrupted by the failure

[5] Only the pair of flows from the target link and edge links are considered.

Algorithm 2 Flow Correlation Analysis in Correlator

```
Input:
  – flow statistics: retrieved from global database
Output:
  – $Link\_attack = 1$: attack flows incoming port
  – $Link\_attack = 0$: otherwise
  Level 2: Flow Correlation Analysis
1: while $linkUnderAttack = true$ do
2:     $getFlowStatistics$
3:     for $f$ in $flowID$ do
4:         for $t = 0, t \leq 60, t++$ do
5:             $Fingerprint(flowID)[x] = getFlowStrength$
6:         end for
7:     end for
8:     for $flowID_i, flowID_j$ do
9:         $fcc_{i,j} = \frac{fcc_i}{fcc_j}$
10:        if $fcc_{i,j} > 0.8$ then
11:            attack flows
12:        else if $fcc_{i,j} \leq 0.8$ then
13:            normal flows
14:        end if
15:    end for
16: end while
```

of the production network. For example, if the production network link is under DDoS attack and experiencing congestion, this will not impact the processing time in the correlator or data retrieval rate from the global database to the correlator. Meanwhile, the production network will also not be affected by the failure of the correlator in SCAFE.

3.3.4 Database for Data Storage and Retrieval

Each sub-network has a local database and is connected to a global database. The global database resides in the core network where it connects to all sub-networks in the network. A local database has the role of storing local port statistics to be used for target link detection. A global database has the role of storing flow statistics of target links and edge links (links nearest to the hosts) in the network. SCAFE contains two different types of databases to improve its efficiency. A local database is used in local traffic monitoring, and a global database can be used in the network-wide correlation analysis. If we use only a single database to store all the information, it will increase the storage space needed and communication and processing overhead to the database. Therefore, having two databases, each dedicated to each detection level will improve the efficiency of SCAFE.

A local database is used for communication between collectors and monitors. A collector will continuously record information into the database where a monitor will retrieve the information from it. A global database facilitates indirect information exchange in between both collector and correlator. The collector will start to record information into the global database when triggered by monitors where the correlator will retrieve the information from it.

These databases will have a retention period to store data, where traffic data will be deleted after a certain period of time. A secondary database may be installed for redundancy and fault-tolerance.

3.4 Communication Path and Message Interfaces

We describe the communication path and message interfaces design between SCAFE components in both level 1 and level 2.

Variables that will be used in this section and their definitions are listed in Table 3.3.

3.4.1 Level 1 - Target Link Identification

In the first level, the communication paths and message interfaces can be divided into two separate processes, i.e. port statistics collection and traffic monitoring. Both processes perform concurrently to detect target links in the local AS.

In the port statistics collection communication path shown in Figure 3.4, the monitor sends a polling request to the switch where the switch

Table 3.3: List of Variables and their Definitions

No	Variable	Definition
1	Port_stats	Statistics information on a port.
2	bytes_in	The number of received bytes on a port.
3	bytes_out	The number of transmitted bytes on a port.
4	packets_in	The number of received packets on a port.
5	packets_out	The number of transmitted packets on a port.
6	dropped_in	The number of received packets dropped by a port.
7	dropped_out	The number of transmitted packets dropped by a port.
8	errors_in	The number of receive errors on a port.
9	Flow_collection	A command to start flow collection at the collector.
10	Flow_tables	Tables containing flow information and forwarding rules.
11	bytes_count	The number of bytes in a flow.
12	flow_name	The flow name based on destination IP.
13	duration_sec	The time a flow has been alive in seconds.
14	duration_nsec	The time a flow has been alive in nanoseconds beyond duration_sec.
15	packet_count	Number of packets in a flow
16	length	Length of a flow.
17	msg	A command to the correlator to start the correlation process.
18	dp_name	The name of a link.

will post a reply with port statistics information back to the collector. The collector will then update the local database with the new port statistics information. This process will be repeated with a predefined time interval. The time interval used is dependent on the trade-off between monitoring frequency and processing overhead. The higher the monitoring frequency, the higher the processing overhead which will impact the overall efficiency of the detection system.

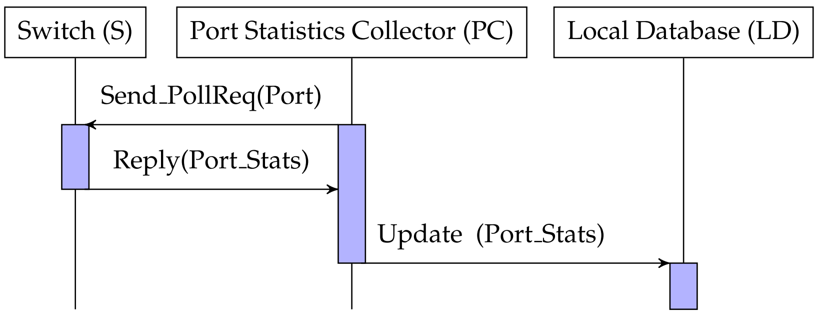


Figure 3.4: Sequence Diagram of Port Statistics Collection Function Communication Path

Descriptions for each function in the Port Statistics Collection process are as follows:

Send_PollReq(Port) Collector sends a request to the switch where it requests to query information about port statistics.

Reply(Port_stats) The switch responds to the port statistics request by sending an update of the port statistics information (Port_stats) to the collector. The port statistics information includes *bytes_in, bytes_out, packets_in, packets_out, dropped_out, dropped_in, errors_in*.

Update(Port_Stats) The collector updates the port statistics information in a time-series table of the Local Database (LD). Each column of the table represents a statistic, and its rows contain the value of the statistic based on time.

In the link monitoring communication path shown in Figure 3.5, the monitor starts sending port statistics queries of each link in the local AS to the LD. The port statistics information obtained from the LD is used for checking anomalies in the statistics. The anomalies are determined based on a set of predefined threshold values. If an anomaly is detected, an alert containing link name (dp_name) will be sent to the correlator for the second level of detection.

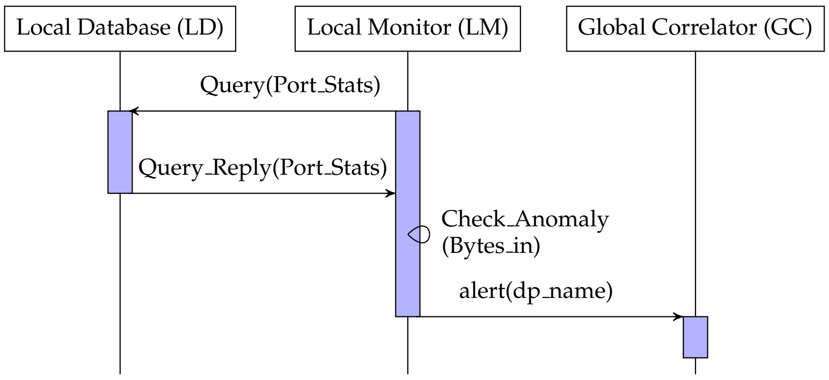


Figure 3.5: Sequence Diagram of Link Monitoring Communication Path

The description of each function in the Link Monitoring process is as follows:

Query(Port_stats) The monitor will periodically query port statistics from the local database to be used for the local monitoring process.

Query_Reply(Port_stats) The database will reply to the query from the monitor with port statistics information.

Check_anomaly(Bytes_in) The monitor will check for suspicious statistics values of every single link in the sub-network. A threshold is used to measure the suspiciousness such as incoming/outgoing bytes count and if the value exceeds the threshold value, it will indicate a link is under attack.

alert(dp_name, msg) When the monitor detects a network link, the monitor will raise an alert to the correlator to start the second stage of detection. The alert consists of a command to start correlation analysis on the link.

3.4.2 Level 2 - Attack Flow Detection

The second level of detection is triggered by the alert generated by the monitor through the link detection mechanism on the first level. This level consists of two processes, i.e. flow monitoring process, and correlation analysis process. The alert triggers both of these processes in the first level of detection and starts simultaneously.

In the flow statistics collection communication path as shown in Figure 3.6, when the correlator receives an alert, the correlator will send a request to the collector to start the flow statistics process. The collector will post a polling request to the switch where the switch will send a reply to its flow statistics back to the collector. Unlike port statistics collection, collectors in each sub-network will dump flow statistics (i.e. flow table information) to a centralised Global Database. Collectors will record flow statistics for a certain amount of time depending on the length of flow signatures required for correlation analysis.

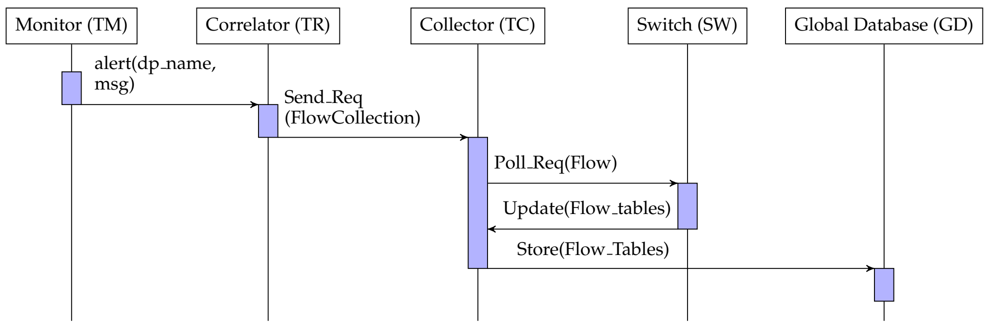


Figure 3.6: Sequence Diagram of Flow Statistics Collection Communication Path

alert(dp_name) Correlator received an alert from the Monitor that a network link is potentially under DDoS attack. The alert contains *dp_name* which contains the name of the network link that is under attack and *msg* which is a message to start the correlation process.

SendReq(Flow_collection) Collector sends a request to the switch to obtain flow statistics from the switch's flow tables.

Update(Flow_Tables) The switch responds to the flow statistics request by sending an update of the flow tables information (Flow_Tables) to the collector. The flow tables information include *bytes_count, duration_nsec, duration_sec, packets_count, length*.

In the flow correlation analysis communication path as shown in Figure 3.7, the correlator sends queries to the global database for flow statistics of the target link identified in level 1 and the global database sends a reply with a series of flow statistics queried. The correlator then uses the flow statistics for correlation analysis by calculating the flow correlation coefficient for each pair of flows. A pair of flows consists of flows from the target link and a flow from one of the edge links. If the flow correlation coefficient is higher than a predefined threshold, then two correlated flows are both considered as attack flows.

Read(Flow_Tables) The monitor will periodically read flow statistics information from the local database to be used for the network-wide correlation analysis process.

Flow_Tables(Bytes_count, flow_name, dp_name) The flow statistics needed for the network-wide correlation analysis process are *bytes_count, flow_name* and *dp_name*. This information is sent to the monitor.

Correlation(dp_name, bytes_count, flow_name) The correlator will calculate the correlation_coefficient [75] of a pair of flows made up from flows in the target link and flows in other network links. If the

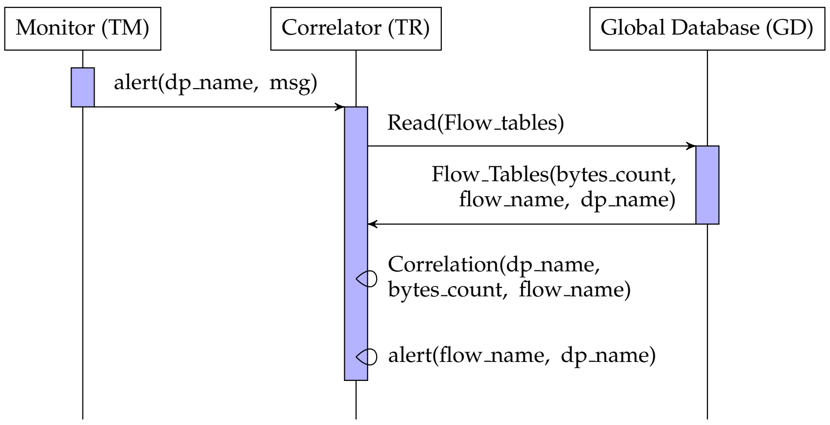

Monitor (TM)
Correlator (TR)
Global Database (GD)
alert(dp_name, msg)
Read(Flow_tables)
Flow_Tables(bytes_count, flow_name, dp_name)
Correlation(dp_name, bytes_count, flow_name)
alert(flow_name, dp_name)

Figure 3.7: Sequence Diagram of Flow Correlation Analysis Communication Path

correlation coefficient value of *bytes_count* exceeds a certain threshold, it is considered as an attack flow. A report is created at the end of the analysis to determine the incoming link at the edge switch, where incoming attack source can be recognised.

alert(flow_name, dp_name) The correlator will raise an alert to the administrator or mitigation system for further action to stop the attack which is not within the scope of this work.

3.5 Summary

This chapter presents the design of SCAFE, a scalable and fault-tolerant DDoS detection system architecture based on SDN and network-wide traffic correlation analysis. The two-level approach in SCAFE aims to improve scalability by narrowing the amount of traffic required for analysis to distinguish between attack and normal traffic flows. The two-level approach consists of a coarse-grained detection where SCAFE detects a network link that is potentially under DDoS attack and a fine-grained detection where it distinguishes attack flows from normal flows in the network.

The first design goal, where a fully distributed monitoring mechanism is desired to identify potential target link is achieved by allocating local monitor and database in each sub-network for local monitoring based on SDN technology. A fully distributed monitoring mechanism in SDN allows the use of a separate communication link (control link) for monitoring without adding additional communication overhead to the production network.

The second design goal, where a centralised correlation mechanism is desired to identify attack traffic is achieved by using a global database to store flow information from every link located in the perimeter of the edge network and a centralised correlator. The global database enables the correlator to retrieve and correlate flow information efficiently.

The third design goal, where a separate control network for SCAFE is desired to ensure scalability with the increasing number of traffic flows or network links is achieved by creating a separate control network for communication such as data storage and data retrieval between SCAFE components in various locations. The creation of a separate control network avoids additional overhead on the data network where the overhead might unintentionally constitute an attack on the network.

Overall, SCAFE fulfilled its three design goals for a scalable and fault-tolerant DDoS attack detection system architecture in large-scale network. The next chapter presents the implementation of SCAFE in a testbed environment and the evaluation of its effectiveness in achieving early detection, high accuracy, linear scalability and fault tolerance in a large-scale network.

Chapter 4

SCAFE Implementation and Evaluation

This chapter presents the implementation and evaluation of the SCAFE architecture described in Chapter 3 to validate its scalability and fault-tolerant capability in the early detection of both high and low-intensity DDoS attack traffic. The evaluation consists of both quantitative and qualitative analysis. The quantitative analysis consists of traffic distribution analysis in large networks, processing time scalability and end-to-end delay in the network. On the other hand, qualitative analysis consists of message complexity analysis and software and hardware fault-tree analysis.

4.1 SCAFE Implementation

This section describes the implementation of the SCAFE architecture for quantitative analysis. SCAFE is implemented on a large scale network using The Global Environment for Network Innovations (GENI). GENI is a virtual laboratory that provides resources to carry out large-scale experiments using real devices. A number of recent approaches in detecting DDoS attacks [86] have evaluated their techniques using GENI because of its ability to emulate a real network environment. In this section, the system implementation, such as network topology, normal traffic generation, and attack traffic generation are described in detail.

Network Topology

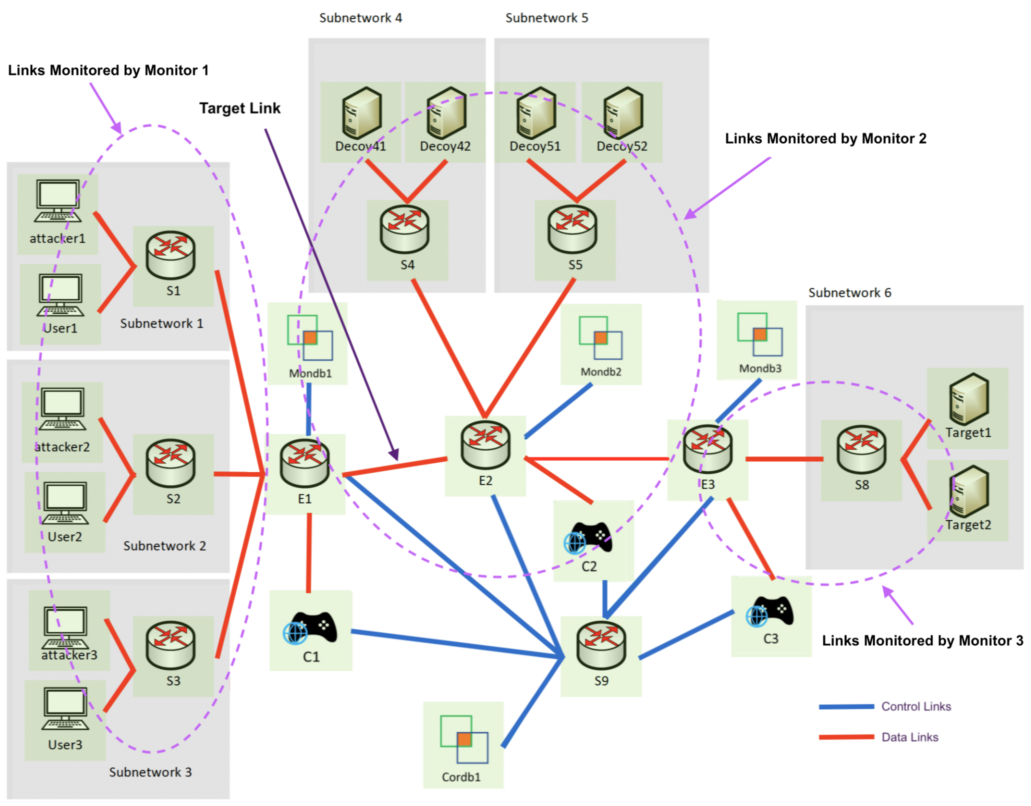


Figure 4.1: GENI Testbed Emulation Set Up

A simple network topology is used in the evaluation and is shown in Figure 4.1. The network consists of six sub-networks which represents the different locations of the network. These sub-networks are interconnected by a network of core switches ($E1$, $E2$ and $E3$) with routing capabilities. Switches in the network are connected to an SDN controller for traffic collection and monitoring. Each controller ($C1$, $C2$, and $C3$) is connected to an NFV server ($MonDB1, MonDB2,$ and $MonDB3$) where the local database and local monitoring mechanism is located. Since SCAFE uses SDN technology, there are two types of communication paths: (1) user traffic (data links) and (2) SCAFE's communication traffic (control links).

Each node in the network is a virtual machine with Ubuntu 16.04.1 LTS installed. The role of each component in the network topology is described in Table 4.1.

No	Roles	Nodes
1	Local SDN Switch	S1, S2, S3, S4, S5, S8, S9
2	Core SDN Switch	E1, E2, and E3
3	SDN Controller	C1, C2, and C3
4	Legitimate Host	User1, User2, and User3
5	Malicious Host	Attacker1, Attacker2 and Attacker3
6	Decoy Server	Decoy1, Decoy2, and Decoy3
7	Monitor and Local Database	MonDb1, MonDb2, and MonDb3
8	Correlator and Global Database	CorDb1
9	Target Host	Target1, Target 2

Table 4.1: Roles Assignment

The configurations and functionalities of the nodes in the network are as follows :

Local and Core SDN Switches - These switches are configured as virtual switches (Open vSwitch) following the OpenFlow 1.3 protocol and are used to forward traffic to its respective destination using flow tables. In each switch, the port that is attached to the production link in the network is configured as an OpenFlow vSwitch interface. Meanwhile, the port that is connected to the DDoS detection system communication link is configured as a normal port with static routes to the DDoS detection system components. All of the OpenFlow vSwitch ports are configured to connect to the controller in their local sub-network. For simplicity, switches $S1, S2$ and $S3$ are connected to controller $C1$, $S4$ and $S5$ are connected to $C2$, and $S8$ is connected to $C3$.

SDN controller - A controller is used to manage traffic flows in the network. SDN switches forward unknown packets to the controller, and the controller will decide where to forward the packet and update the switch flow tables. The controller is also used for collecting traffic statistics. For each controller node, there are two types of SDN controller: POX and FAUCET. POX is used to manage traffic flows and update flow tables in switches, while FAUCET is used for collecting traffic statistics using its Gauge Monitoring API. Each switch will connect to both controllers in the controller node.

Legitimate Host - The legitimate host is used to send normal traffic (TCP and HTTP traffic) to the target nodes in the network. Harpoon [96] is used to generate normal traffic in the node.

Malicious Host - The malicious host acts as the attacker in the network. The attacker sends DDoS attack traffic (TCP and HTTP) to its target in the network. The target can be either Decoy servers (indirect DDoS attacks) or target servers (direct DDoS attacks). D-ITG is installed to generate DDoS attack traffic.

Decoy Server - The decoy server acts as a decoy for the indirect attacks. The decoy is set as an HTTP Apache server to receive HTTP requests in the network.

Monitor and Local Database - The monitor and local database are two components in the DDoS detection system. Both of these components are installed in the same node for simplicity. The node acts as an NFV server where a monitoring script is installed for monitoring and an InfluxDB database is installed for traffic statistics storage. From our design, each sub-network will have a monitor and local database node. To overcome resource limitations, multiple sub-networks are connected to the same monitor and local database node. However, each sub-network will have a monitor and local database component in the node.

Correlator and Global Database - Like monitor and local database, the correlator and global database are installed in the same node in the NFV server. However, the correlator and global database are a centralised component where all sub-networks are connected to them. The node contains a correlation script for network-wide correlation analysis, and an InfluxDB database is installed for storing flow statistics.

4.2 Quantitative Analysis

This section presents the quantitative analysis as follows.

4.2.1 Traffic Distribution Evaluation

The traffic distribution evaluation consists of normal traffic generation and attack traffic generation.

Normal Traffic Generation

Harpoon [96] is a flow level traffic generator tool that generates normal traffic in the network. This tool is capable of generating traffic that is representative of normal TCP and UDP traffic using distributional parameters such as file size, inter-connection time, source and destination IP ranges, number of active sessions, constant bit-rate, periodic, and exponential ping-pong. The Pareto distribution is used to generate normal traffic, where it is the closest distribution to normal traffic in reality [97]. In this experiment, Harpoon generates normal traffic from three normal hosts to two target servers in the network for approximately 30 minutes. The process is repeated for a total of two hundred (200) times using five different seed numbers. In other words, normal traffic generation is iterated forty (40) times for every seed number. The average bytes and packets transmitted using Harpoon is shown in Figure 4.2.

Using Harpoon, the normal TCP traffic is generated as the background traffic by adhering to the following parameters:

File sizes: Empirical distribution of file sizes transferred from legitimate users to target servers. The normal traffic is generated using file sizes that are randomly generated following the seed settings.

Inter-connection time: Empirical distribution of time between consecutive TCP connections initiated by an IP source-destination pair. Similar to file sizes, inter-connection time is randomly generated using different seed settings.

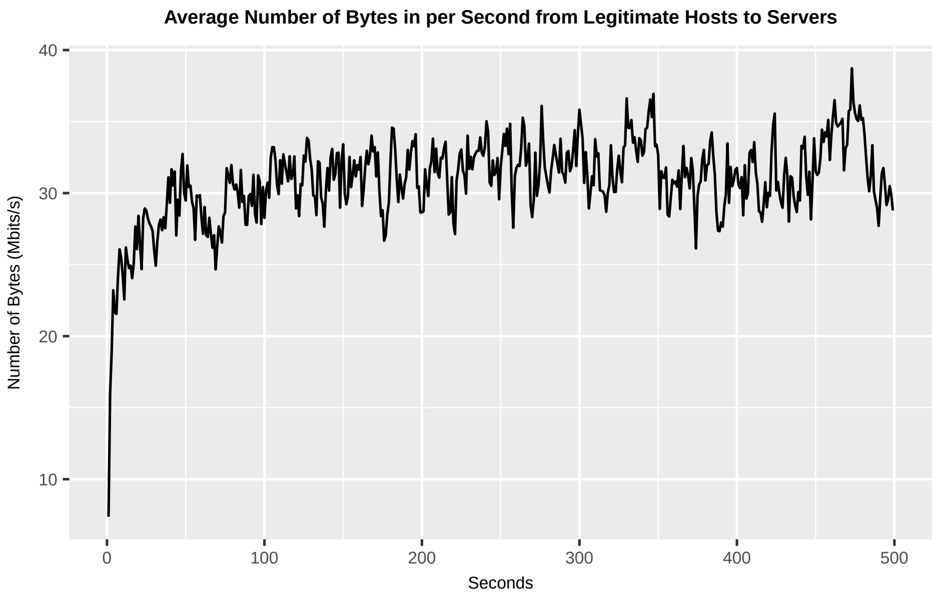
IP ranges: IP ranges are set to match the IP addresses of the legitimate host and target servers in our network. Our DDoS attack system is not sensitive to the variation of source IP addresses. Therefore, a large range of IP addresses is not needed to measure its generality.

Active sessions: The distribution of average active session during consecutive intervals. 6 active sessions are used.

Seed number: Specify the seed for random number generation. Normal traffic is generated using five different seed numbers (i.e. 1,2,3,4,5). The traffic generated using each seed provides variation in traffic patterns.

Attack Traffic Generation

The two different types of DDoS attacks, namely direct link flooding DDoS attack and indirect link flooding DDoS attack are launched through the use of three malicious hosts (attackers).



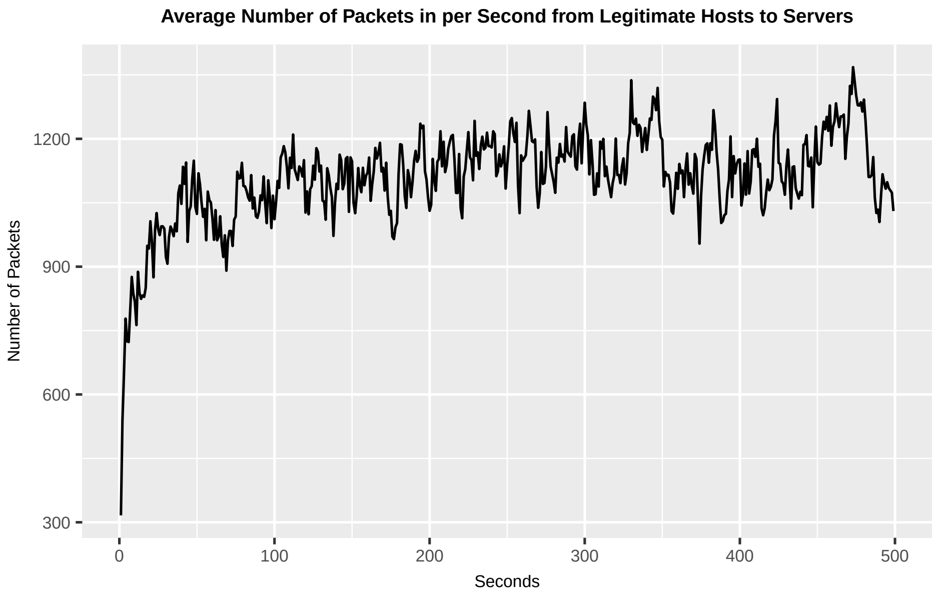


Figure 4.2: Average Bytes and Packets Transmitted using Harpoon

Figure 4.3 shows the traffic distribution of direct traffic generated using the D-ITG tool: (1) direct attack traffic generated in the network with regards to number of bytes per second, (2) direct attack traffic generated in the network with regards to number of packets per second

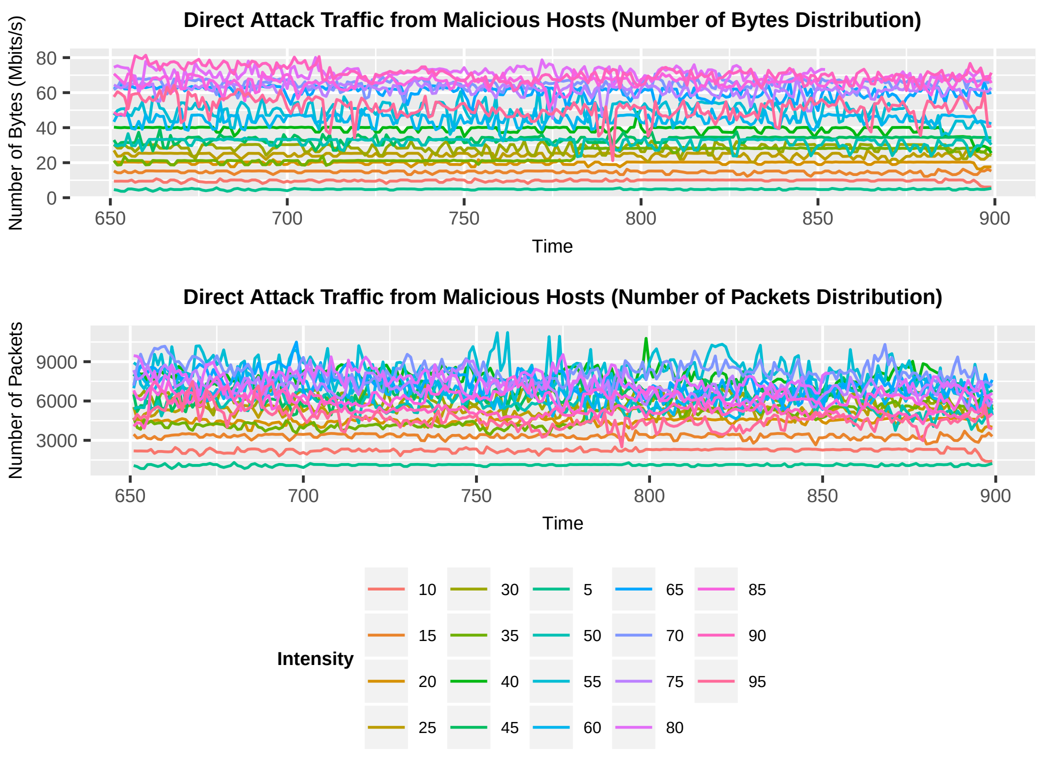


Figure 4.3: Direct Attack Traffic using D-ITG

Figure 4.4 shows the traffic distribution of indirect traffic generated using the D-ITG tool: (1) indirect attack traffic generated in the network with regards to number of bytes per second, and (2) indirect attack traffic generated in the network with regards to number of packets per second.

Different attack intensities will produce different attack versus normal traffic ratios in the network. As normal traffic intensity remains constant at 30-35 Mbps, 10Mbps of attack traffic will give the attack versus normal traffic ratio of 3.5:1. The lower the ratio of attack traffic in the network,

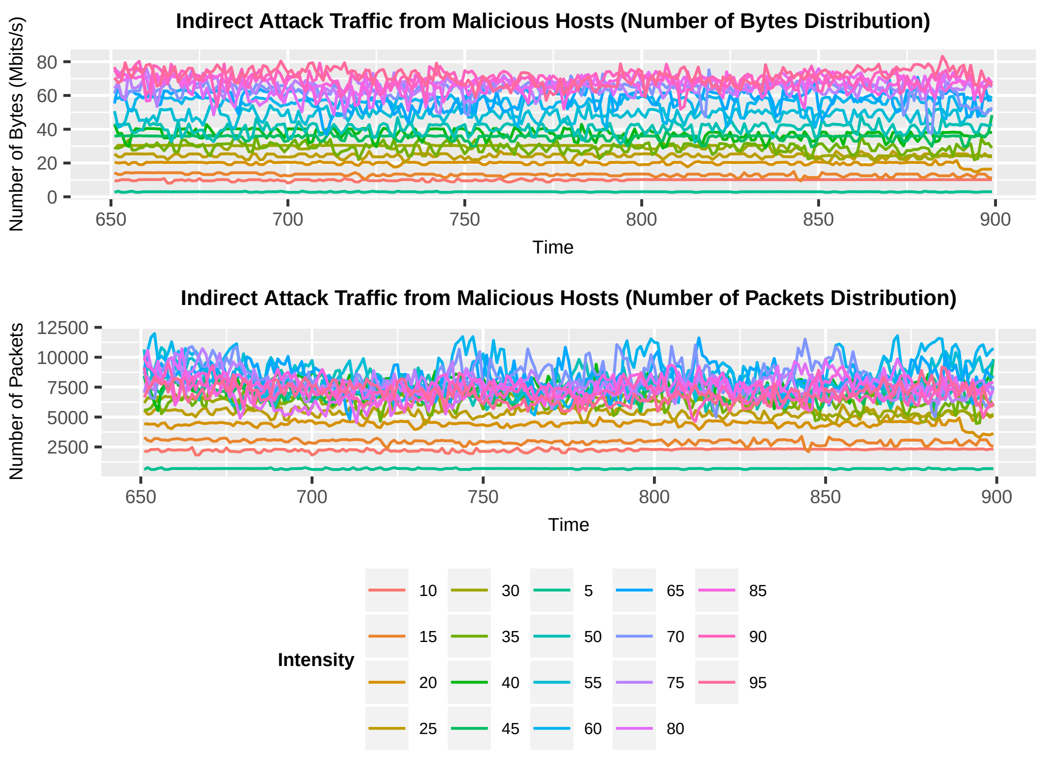


Figure 4.4: Indirect Attack Traffic using D-ITG

the more similar the traffic distribution will be to that of normal traffic, whereas a high ratio of attack traffic will show a significant difference in its traffic distribution as compared to normal traffic. Different ratios of attack traffic can also represent the location of attack traffic in the network. This is because attack traffic usually has a lower ratio nearer to the attack source and the ratio of attack traffic increases as the attack traffic gets nearer to the victim.

From Figure 4.3 and Figure 4.4, the attack traffic distribution for both direct and indirect attacks show similar traffic patterns to that of normal traffic (see Figure 4.2). Heuristically, it will be difficult to identify attack traffic from normal traffic just by using volumetric measures such as the number of packets or number of bytes in a link.

The set up of each type of attack is described as follows:

(a) Direct Link Flooding DDoS Attack

In a direct link flooding attack, the attackers will send attack traffic directly to their target hosts.

(b) Indirect Link Flooding DDoS Attack

An indirect link flooding attack consists of attackers sending attack traffic indirectly to their target by using decoys. Attack traffic is carefully coordinated to be forwarded to decoy servers to congest the targeted network links.

4.2.2 Detection Accuracy of Attack Traffic in the Network

A pair of flows are considered attack flows when their correlation coefficient exceeds the threshold value. The detection accuracy test aims to investigate the impact of certain parameter settings on the accuracy of detecting attack flows in SCAFE. The detection accuracy is measured using: detection accuracy rate (DR), and false positive rate (FPR).

The parameter settings are defined in the following:

(a) Threshold value - A fixed number in terms of megabits per second to determine the presence of DDoS attack traffic in a link. If the total number of bytes per second exceeds the pre-defined threshold, the link is considered to have DDoS attack traffic. Otherwise, the link is considered as a normal link.

(b) Observation period - An observation period is a specific time period, where network link's information is collected and analysed. Information about network links is collected every second but the decision on the network link status (i.e. attack or normal) will be made at the end of each observation period. A shorter observation time allows DDoS attacks to be detected quicker.

Impact of Threshold Values in Detecting Potential Attack Links

A link is considered as a potential attack link once its total number of bytes per second continuously exceeds the predefined threshold value during the observation period. Five different threshold values (i.e. 10, 30, 50, 70 and 90 Mbps) are considered for exploration purpose.

Figure 4.5 and 4.6 show the direct and indirect high-intensity DDoS attack detection rates of three monitors using different threshold values to differentiate attack links from normal links in the network. For each monitor located in different network locations, the detection rate does not differ significantly across different threshold values except for when the threshold value is at 10Mbps. At this value, the detection mechanism performs poorly where the average detection rate of all attack intensities is about 60% for Monitor 1 and Monitor 3, and 80% for Monitor 2. This is because of the false classification of normal traffic as an attack. The rate of normal traffic in this experiment is about 30Mbps. For other threshold values (i.e. 30, 50, 70 and 90 Mbps), the detection rate is well above 65% for all attack intensities in Monitor 1, 80% for Monitor 2 and 60% for Monitor 3. Monitor 2 reports the highest accuracy rate among the three monitors because that is where the attack traffic is aggregated. Monitor 2 aggregates more attack traffic than the other two monitors because the target link is located in the sub-network where Monitor 2 is monitoring. A higher attack traffic concentration improves detection accuracy.

Figure 4.7 and Figure 4.8 depict the FPR values obtained from three monitors in the network for direct DDoS Link attack detection and indirect DDoS attack detection. Monitor 1 has the highest FPR values, while Monitor 2 achieves the lowest FPR values. This is because the attack traffic has not reached its aggregation point in sub-network 1, where attack traffic information might not be sufficient, causing Monitor 1 to have a higher FPR than Monitor 2 and Monitor 3. In all three monitors, the FPR values decrease with increasing threshold values. This is because smaller threshold values will cause a higher number of alerts generated in the Level 1

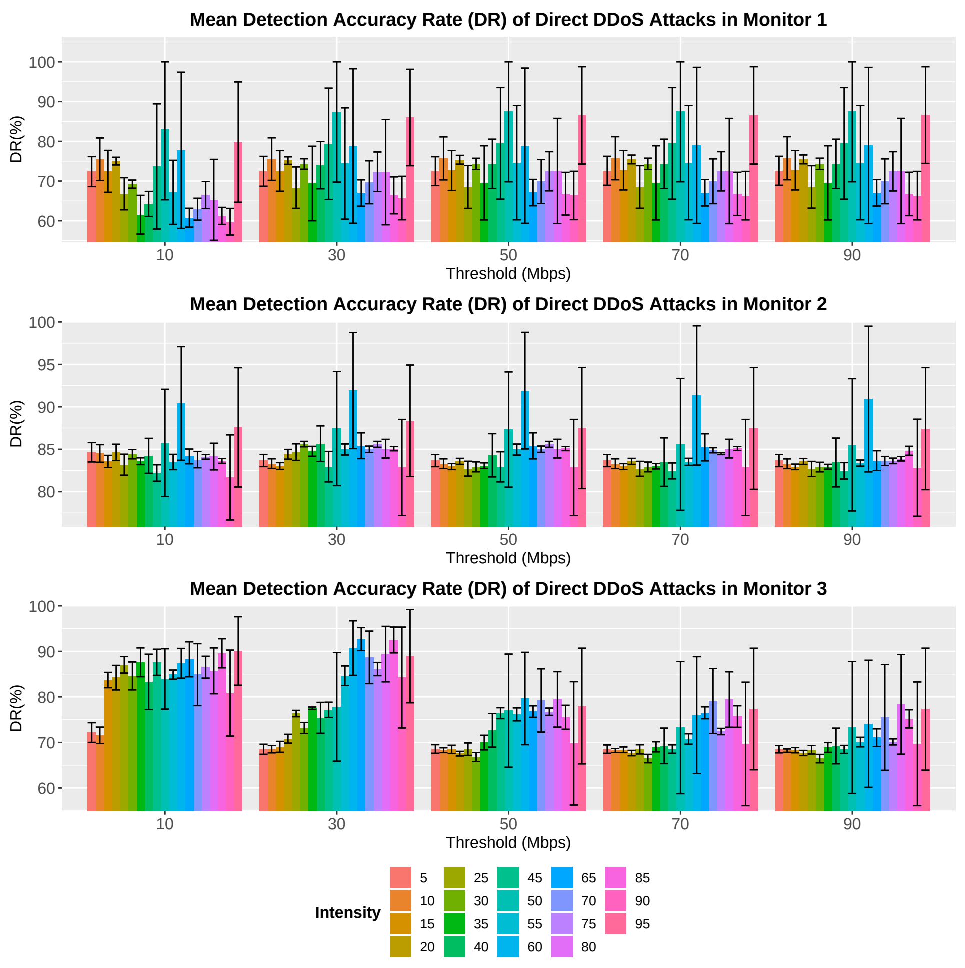


Figure 4.5: Mean Detection Accuracy Rate of Direct DDoS Attacks on All Monitors with Different Threshold Settings

detection of SCAFE, as normal links are identified as target links.

The results indicate that the threshold values selected do not significantly impact either direct or indirect attack detection accuracy. However, the FPR can be significantly reduced by using higher threshold values on all variants of attack intensities.

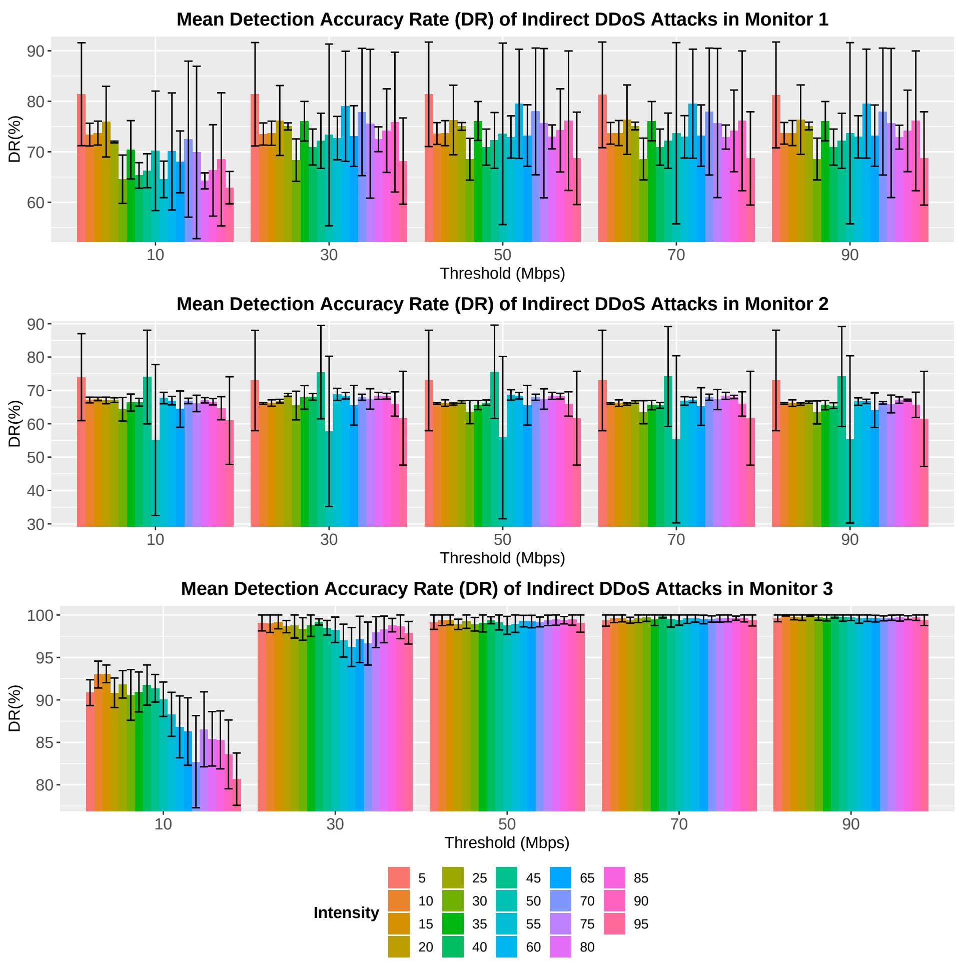


Figure 4.6: Mean Detection Accuracy Rate of Indirect DDoS Attacks on All Monitors with Different Threshold Settings

Impact of Observation Period in Detecting Target Links

This section explores the impact of observation periods on the detection accuracy. Shorter observation period allows attack traffic to be identified much earlier and allows time for mitigation processes to take action. However, data collected during the short observation period might not be sufficient to detect both high and low-intensity DDoS attack traffic accurately. A set of experiments are conducted to find out the impact of observation

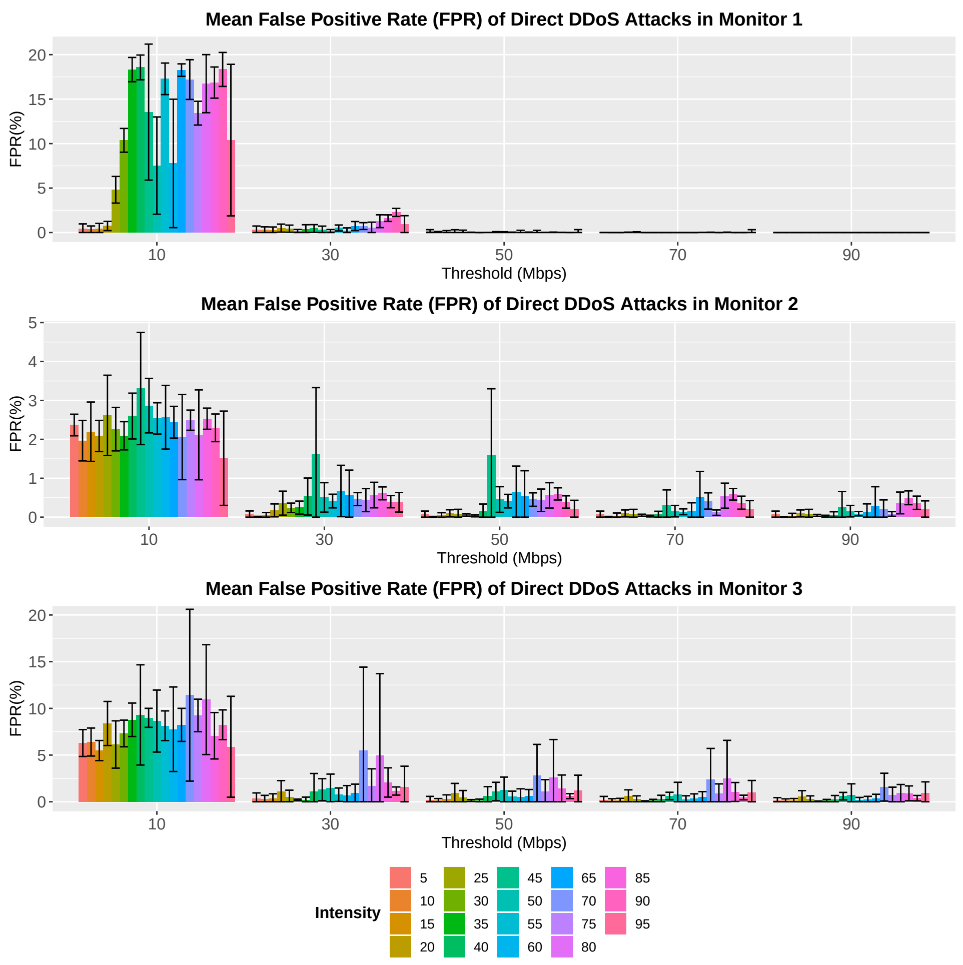


Figure 4.7: Mean False Positive Rate of Direct DDoS Attacks on All Monitors with Different Threshold Settings

period in detecting target links. 5 different observation periods are used and the results are shown in Figure 4.9 and Figure 4.10.

Figure 4.9 shows the detection accuracy rate of Monitor 1, Monitor 2 and Monitor 3 for the detection of direct DDoS attack traffic. Detection rate at Monitor 2 is shown to be the highest among the three monitors because the target link and links closest to the target links are being monitored by Monitor 2. All three monitors show a slight improvement or no

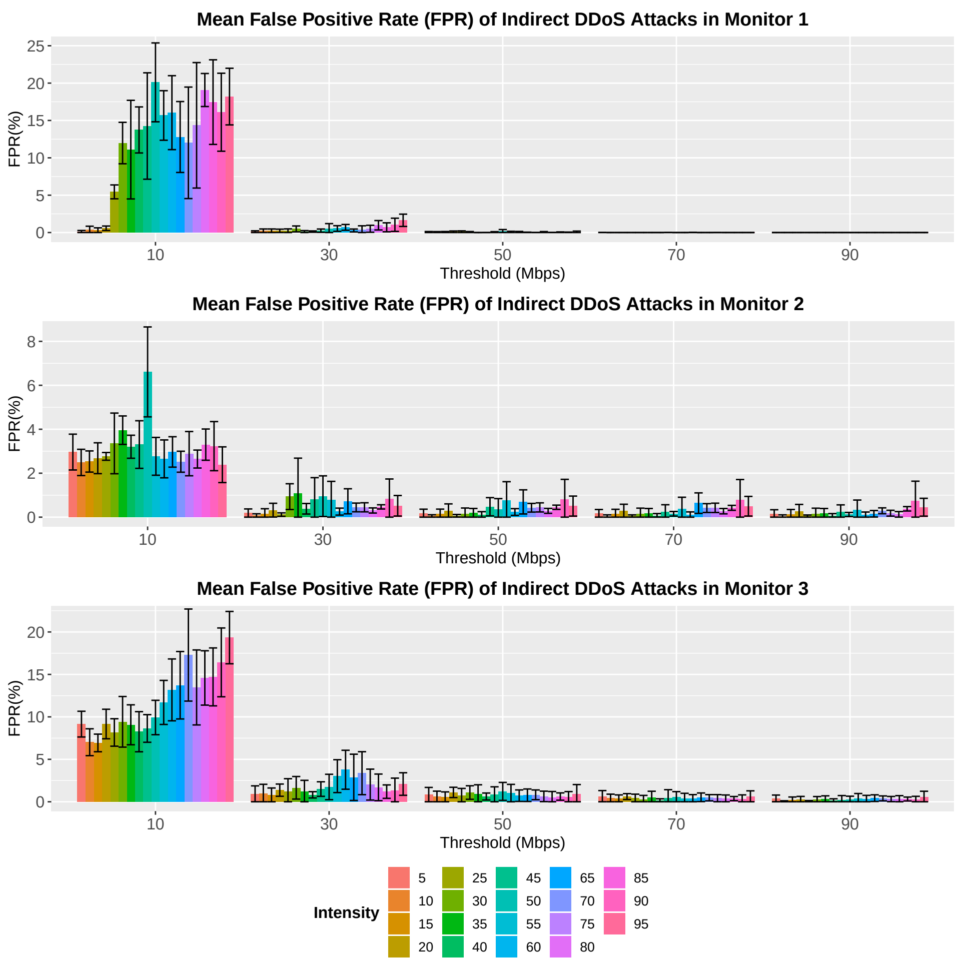


Figure 4.8: Mean False Positive Rate of Indirect DDoS Attacks on All Monitors with Different Threshold Settings

improvement at all with increasing observation periods. This means that longer observation period does not impact the detection accuracy rate significantly.

Figure 4.9 and 4.10 show the detection rate of Monitor 1, Monitor 2, and Monitor 3 at the 1, 5, 10, 15, and 20 second observation periods. In Figure 4.9, all three monitors show a slight improvement or no improvement at all in the detection accuracy as observation periods get longer when de-

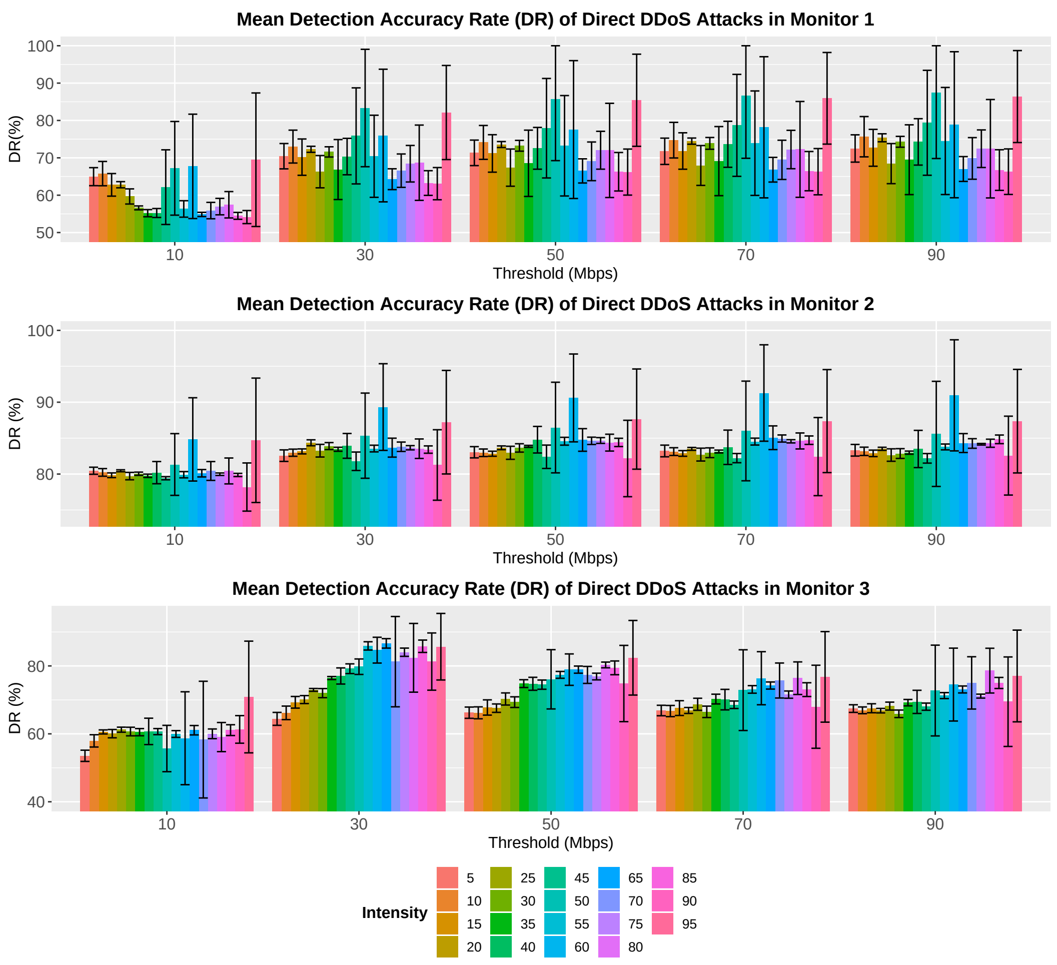


Figure 4.9: Mean Detection Accuracy Rate of Direct DDoS Attacks on All Monitors with Different Observation Periods

tecting direct DDoS attack traffic. Similarly, Figure 4.10 also shows a slight improvement or no improvement in the detection accuracy as observation periods get longer when detecting indirect DDoS attack traffic.

Figure 4.11 and 4.12 show the detection rate of Monitor 1, Monitor 2, and Monitor 3 at the 1, 5, 10, 15, and 20 second observation periods. In Figure 4.11, the false positive rates only show a slight improvement or no improvement at all, as the observation period gets longer, to detecting

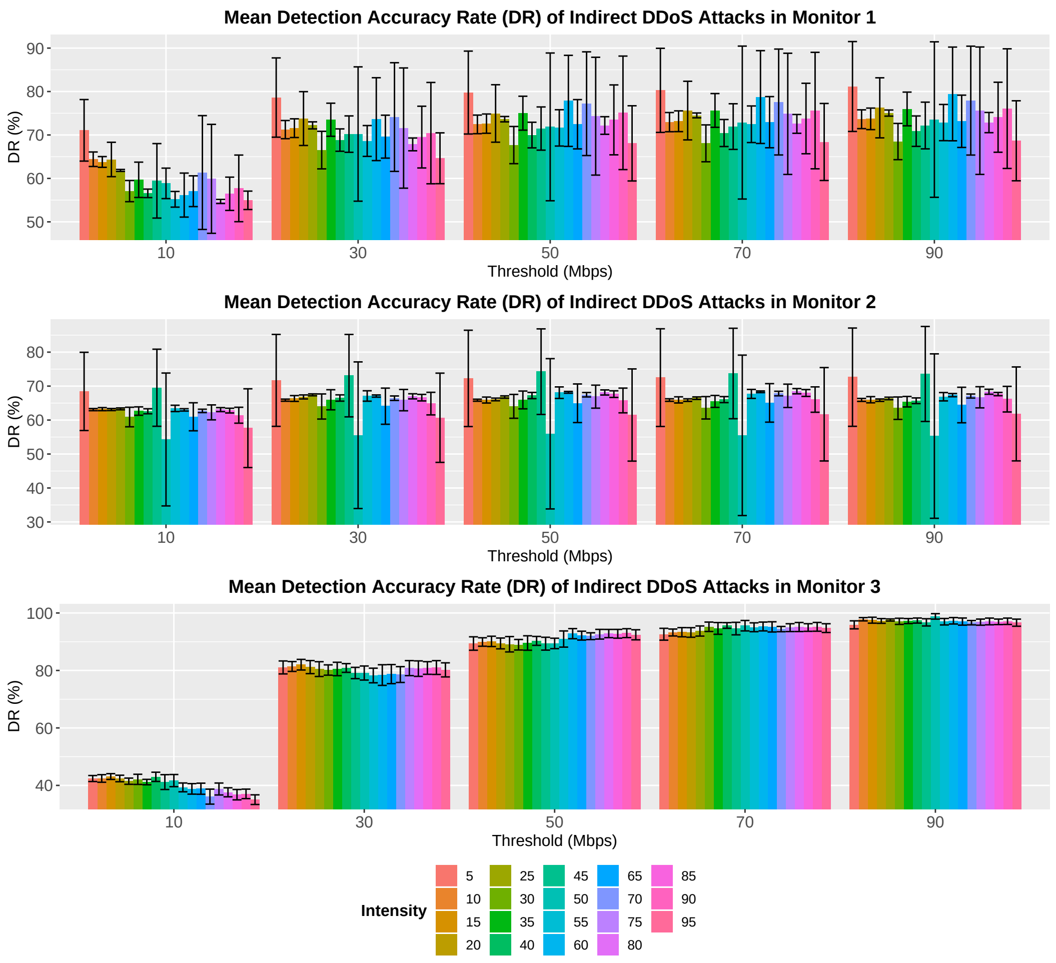


Figure 4.10: Mean Detection Accuracy Rate of Indirect DDoS Attacks on All Monitors with Different Observation Periods

direct DDoS attack links from all three monitors in the network. Similarly, Figure 4.12 shows there is only a slight improvement or no improvement at all, as the observation periods get longer, to detecting indirect DDoS attack links.

Figure 4.12 and 4.11 show FPR values when different observation periods are used. FPR values decrease linearly as the observation period gets longer on all monitors. The results show that the observation periods have

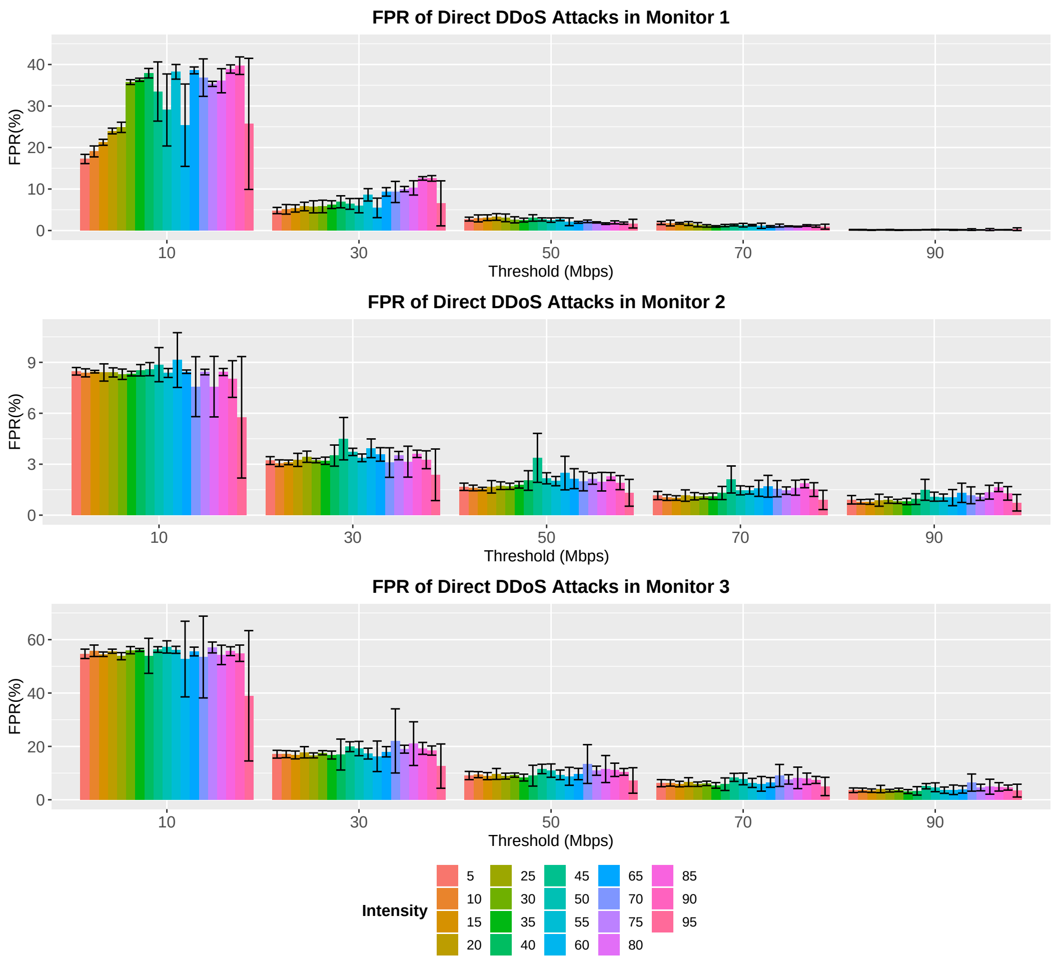


Figure 4.11: Mean False Positive Rate of Direct DDoS Attacks on All Monitors with Different Observation Periods

very little to no impact on the detection accuracy but have a linear effect on the false positive rate.

Although the results shows that no significant improvement is observed from the range of observation times evaluated, evaluation using an observation time longer than 20 seconds is not compared. This is because a longer observation period means a larger amount of data is needed for processing which will impact the efficiency of the detection system. There-

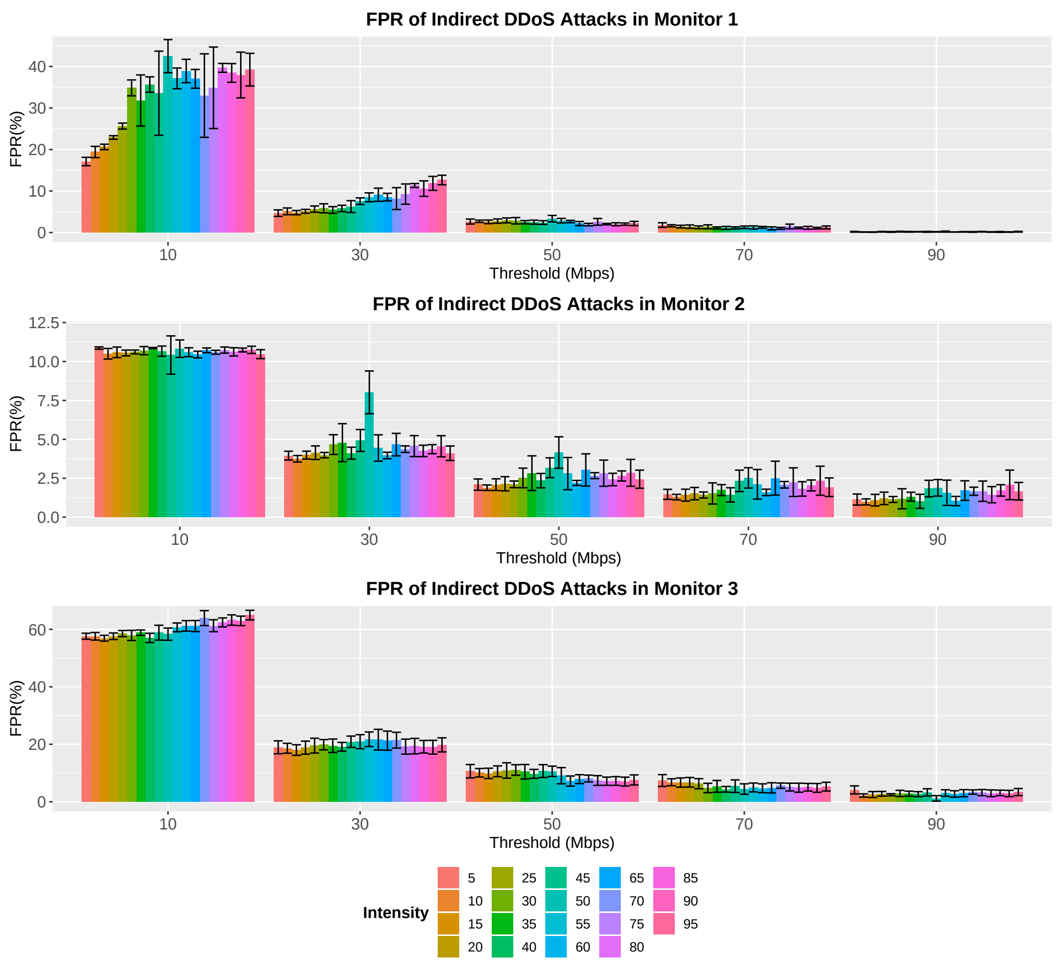


Figure 4.12: Mean False Positive Rate of Indirect DDoS Attacks on All Monitors with Different Observation Periods

fore, this thesis only compares the observation time from 1 -20 seconds.

4.2.3 Network End-to-End Delay

Figure 4.13 shows the end to end delay from users to targets in three different events; before a DDoS attack occurs, during the attack, and after the attack has stopped. This is to measure the impact of DDoS attacks on the delay time of packet transmission from users to targets.

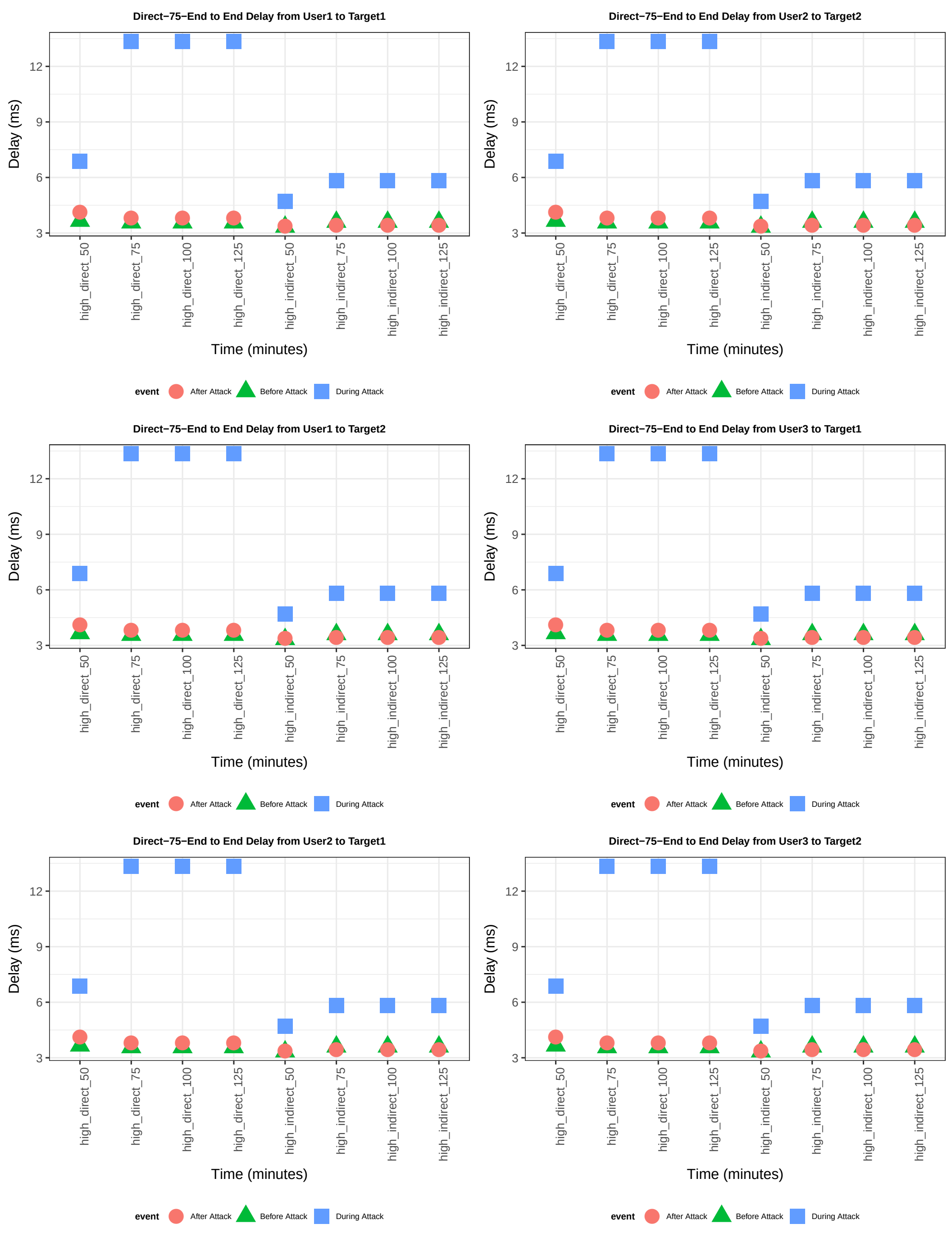


Figure 4.13: Average End-to-End Delay based on High and Low-Intensity Attacks per Time Interval

All monitors show significant increase in their end-to-end delay during an attack. This indicates that the DDoS attack is successful as user traffic is experiencing congestion and taking much longer time than usual to reach its destination. Direct DDoS attacks impact the monitor's end-to-end delay more significantly, where it is approximately 10 seconds higher than the delay time before and after an attack, as opposed to only 1-2 seconds increase in the delay time during an indirect DDoS attack.

In this evaluation, direct DDoS attacks show more significant end-to-end delay than indirect DDoS attacks because there are only two victim servers receiving attack traffic from direct DDoS attacks as opposed to four decoy servers receiving attack traffic from indirect DDoS attacks.

4.2.4 Processing Time Scalability

The processing time of SCAFE with different attack intensities, network sizes, and traffic volumes are evaluated.

Figure 4.14 depicts the average time to monitor each link in a subnetwork for each time interval. The processing time of different attack types and intensities. This figure shows the average processing time does not change significantly when the intensity of the attack increases with regards to direct and indirect DDoS attack traffic. This is because an increase of attack intensity does not increase the number of links, number of switches or number of flows in the network where the processing time could be affected based on the scalability analysis conducted in Section 4.3. The DDoS attack uses the same number of attackers (bots) in the network but with different levels of attack intensity. However, the average processing time rises during an attack and decreases after the attack because of the change in the number of flows. While an attack is happening, there is an increasing number of traffic flows from the attackers to their victims.

Figure 4.15 depicts the average time to correlate flows in a link each time local monitors in the network detect a potential target link. From

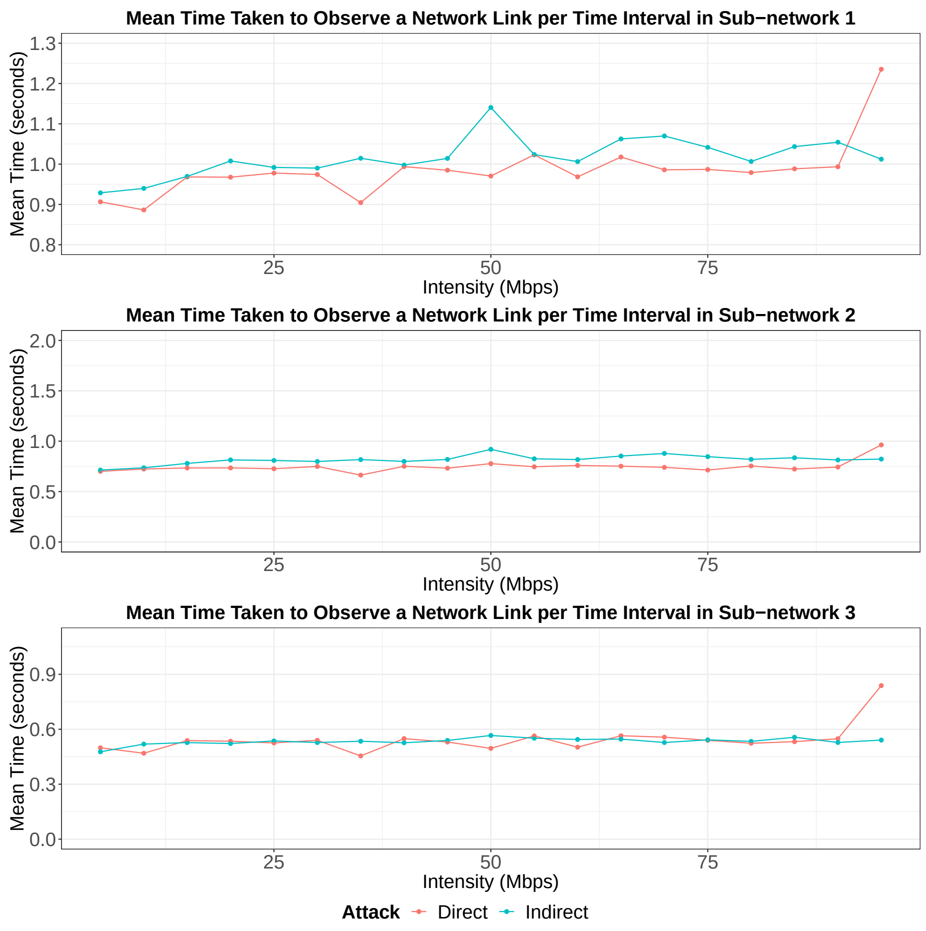


Figure 4.14: Average Time Taken for Each Monitor to Observe each Link in a Sub-Network

the figure, the average processing time increases almost linearly with the increase in attack intensity. This result validates SCAFE's design where the processing time increases linearly to the increase of DDoS attack intensity.

4.3 Qualitative Analysis

This section presents the qualitative analysis of message complexity and fault tree analysis.

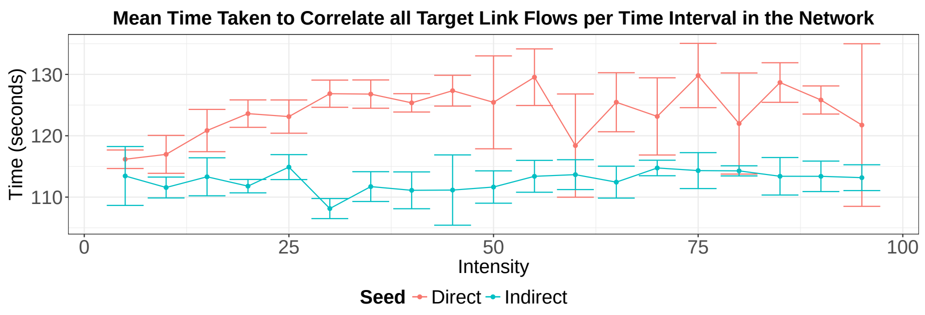


Figure 4.15: Average Time Taken for Each Controller to Correlate Each Target Link in a Sub-network based on High and Low-Intensity Attacks in a Time Interval

4.3.1 Message Complexity

Message complexity is used to evaluate SCAFE's scalability design. The message complexity is defined as the number of messages required to communicate between components in the system in order to perform DDoS detection tasks in a single unit of time per polling period as the network scales in terms of network size and traffic volume.

The notations used to calculate the message complexity are shown in Table 4.2.

A scalable system would be able to process traffic smoothly by keeping the number of messages low as the network size or traffic volume increases in magnitude. This translates to a scalability target, where the number of messages should increase sub-linearly or linearly with respect to the network size and traffic volume. If the system requires a superlinear amount of messages with the increase of the network or traffic size, the system is said to be non-scalable. To evaluate SCAFE's scalability, the complexity of each detection component is analysed. Note that the scalability can also be affected by the polling period, where longer polling period will reduce communication overheads. However, its impact is not as significant as other factors. For simplicity, it is assumed that all components use the same polling period (i.e. 1 second).

Item	Notation	Description
Switch(S)	S_n	The total number of switches in a sub-network
	S_e	The total number of switches with edge links in a network
Link (L)	L_n	The total number of links in a sub-network
	L_e	The total number of edge links in a network
Flow(F)	$\bar{F}$	The average number of flows per link
Sub-network(N)	N	The number of sub-networks in a network
Message(M)	M	The number of messages required for communication between components to execute a task in a single time interval.

Table 4.2: Notations Used in Complexity Analysis

Relationship among Links, Switches, Flows and Sub-networks towards the Network Size and Traffic Volume

The relationship among the number of links, switches, flows and sub-networks towards the size of the network and volume of the traffic is described in the following.

(a) Number of Links, L

A link connects two nodes[1] in the network. Each pair of nodes has only a single link connecting to each other. Therefore, an increase in the number of links in the network means that there is also an increase in the number of nodes in the network, where these nodes may or may not be switches in the network.

[1] A node is a network device is the network such as host, server, router, switch, and network controller.

(b) Number of Switches, S

A switch forwards traffic in the network following a set of rules in its forwarding tables. Each switch in the network would require at least one link connected to the network. There is a linear relationship between the number of switches and the number of links where the number of switches is indirectly proportional to the number of links in the network. However, it is noted that the number of links is not directly proportional to the number of switches in the network.

(c) Average Number of Flows, $\bar{F}$

A network link consists of traffic flows travelling to their respective destinations. The average number of flows, $\bar{F}$, represents the average number of distinct flows found in a single link. The number of flows in each link increases when the number of nodes (hosts and servers) in the network increases.

(d) Number of Sub-Network, N

The number of sub-networks in a network. The number of sub-network is based on the design of the network. Each sub-network represents a specific location in the network. The number of links and switches can vary across multiple sub-networks. Every sub-network will have a port statistics collector, flow statistics collector and local database.

Message Complexity of Each Component

The message complexity between components when detecting DDoS attacks using the DDoS detection architecture is described in Table 4.3.

(a) Port Statistics Collector

Periodically, the port statistics collector is responsible for running two tasks: (1) send port statistics poll requests to switches in a sub-network and (2) store port statistics information in a local database.

Component	Task	Complexity
Port Statistics Collector	Send port statistics poll request to switches in a sub-network, P_{poll}	$O(S_n)$
	Forwards port statistics information to a local database, P_{store}	$O(L_n)$
Flow Statistics Collector	Send flow statistics poll requests to edge switches in a sub-network, F_{poll}	$O(\frac{S_e}{N})$
	Forwards flow statistics information to a global database, F_{store}	$O(\bar{F}L_e)$
Monitor	Send port statistics queries to local database, W_{query}	$O(L_n)$
Correlator	Send flow statistics queries to global database, C_{query}	$O(\bar{F}L_e)$
Local Database	Return port statistics queries to monitor, DL_{reply}	$O(L_n)$
Global Database	Return flow statistics queries to correlator, DG_{reply}	$O(\bar{F}L_e)$
Switch	Return port statistics poll request to port statistics collector, S_{preply}	$O(\frac{L_n}{N})$
	Return flow statistics poll request to flow statistics collector, S_{freply}	$O(\bar{F}\frac{L_e}{S_e})$

Table 4.3: Message Complexity of Port Statistics Collector, Flow Statistics Collector, Monitor, Local Database, and Switch

For the first task, at a predefined polling interval, a port request will be sent to each switch in the sub-network to poll port statistics. Since a request is sent for each switch, it is assumed each switch will require one message to be sent from the port statistics collector. Therefore, the number of messages a port statistics collector would be required to send for polling port statistics from all switches in the sub-network will depend on the number of switches in the sub-network. For the second task, it is assumed that a single message is needed to send each link's port statistics information to the local database. Hence, the number of messages required to send the port statistics information of all links in the sub-network to the local database depends on the number of links available in the sub-network (refer to Table 4.3).

(b) Flow Statistics Collector

Tasks in the flow statistics collector are triggered when a potential attack link is found in a sub-network. Whenever an attack link is detected, the flow statistics collector will run two tasks : (1) send flow statistics poll requests to edge switches and the attack switch in a sub-network and (2) store flow statistics information in a global database. For each polling period, only one potential attack link is detected.

In the first task, where the flow statistics collector sends polling requests to edge switches and attack switch, only edge switches and one of the switches connected to the potential attack link are polled by the flow statistics collector. Each of the edge switches requires one message to poll flow statistics from the edge switches and there will only be a single switch connected to the potential attack link. Accordingly, the number of messages required by the flow statistics collector to request flow statistics from switches is dependent on the number of edge switches in the network.

In the second task, it is assumed that one message is needed to store the flow statistics of a single flow in the network. To detect the presence of DDoS attack flow in the network, the flow statistics of edge links and potential target link are needed for analysis. This means that the number of messages required to send all flow statistics information to the global database will depend on the total number of edge flows in the network (refer to Table 4.3).

(c) Monitor
Periodically, each local monitor communicates with its local database to query for port statistics information of every link in the sub-network per polling interval. Assuming that each link requires a single message to send port statistics information to the local database, the number of messages needed to query each link's port statistics information from its local database relies on the number of links in the sub-network (refer to Table 4.3).

(d) Local Database
The role of a local database is to store port statistics information and return queried port statistics information to the monitor. In a network, each sub-network has a local database. Given the assumption that each link's port statistics information is delivered through one message, the number of links in the sub-network will determine the number of messages needed by local database to return all port statistics queries to the monitor (refer to Table 4.3).

(e) Global Database
The global database acts as a storage for flow statistics information sent by the flow statistics collector and returns queried flow statistics information when requested by the correlator. The correlator sends a query for each edge and each potential target links in the network to obtain flow statistics information. Assuming that the number of flows in the potential attack link is constant or bounded above by a constant, the number of messages required to return all flow information will be dependent on the number of edges flows in the entire network (refer to Table 4.3).

(f) Switch
Switches in the SCAFE detection system report port statistics to the port statistics collector. If the switch is an edge switch, it will also return flow statistics information when polled by the flow statistics collector. In the former task, assuming that the switch returns the port statistics of each link in a single message to the port statistics collector, the number of messages needed to return all port statistics information will be dependent on the number of links in the sub-network. In the latter task, the number of edge links in a sub-network will determine the number of messages required to forward flow statistics to the flow statistics collector with the assumption that a single message is needed to send flow statistics information of a single link to the flow statistics collector (refer to Table 4.3).

(g) Correlator
The correlator's primary goal is to calculate the flow correlation coefficient of each pairwise combination of flows, which consist of a flow from the target link and a flow from one of the edge links. To achieve that, the correlator will need to communicate with the global database to obtain flow statistics information from all edge and potential attack links stored in the global database. Assuming that the correlator will send a single message to query flow statistics of each link and there is only one potential attack link, the total number of messages the correlator needs to send is dependent on the number of edge links in the entire network (refer to Table 4.3).

Message Complexity of Each Process

Recall that the SCAFE detection system consists of four processes: port statistics collection, traffic monitoring, flow statistics collection and flow correlation analysis (see Section 3.2). Each sub-network has local port statistics collection, traffic monitoring and flow statistics collection processes and they run in parallel with other sub-networks. Since the complexity of each task based on the analysis presented in section 4.3.1 has been discussed and all of them have linear complexity, the complexity values of each process is derived as shown in Table 4.4.

(a) Port Statistics Collection

As described in Section 3.4.1 and shown in Figure 3.4, port statistics collection consists of three sequential tasks associated with one polling interval. All the three tasks involve communication with other components in the system. First, the system starts by using the port statistics collector to send poll requests to all switches in the sub-network and second, each switch will reply by returning its port statistics information. Lastly, the port statistics collector will forward the port statistics information to the local database for storage. The complexity measures as shown in Table 4.4 show that there is a linear relationship between port statistics collection complexity and the number of switches and links in the sub-network. In other words, whenever the number of links or number of switches in the sub-network increases, the complexity will increase linearly. This is because the increase in the number of switches or links is directly proportional to the number of messages sent between monitor, local database and port statistics collector in the sub-network.

(b) Link Monitoring

Link monitoring has two tasks that require communication with other components in the system. These tasks are: (1) Each monitor communicates with the local database in the sub-network to query port statistics information, and (2) local database replies to monitor queries by sending queried port statistics information back to the monitor. The complexity of this process will also increase linearly with increasing number of links in the sub-network. This is because when the number of links increases in a sub-network, the monitor and local database will need to send a greater number of messages to per-

Process	**Task**	**Message Complexity, (M)**
Port Statistics Collection ($Level1_a$)	$P_{poll} \rightarrow S_{preply} \rightarrow P_{store}$	$M(Level1_a)$ $= M(P_{poll}) + M(S_{preply}) + M(P_{store})$ $= O(S_n) + O(\frac{L_n}{N}) + O(L_n)$
Link Monitoring ($Level1_b$)	$W_{query} \rightarrow S_{preply}$	$M(Level1_b)$ $= M(W_{query}) + M(DL_{preply}) + M(W_{observe})$ $= O(L_n) + O(L_n) + O(L_n)$
Flow Statistics Collection ($Level1_a$)	$F_{poll} \rightarrow S_{freply} \rightarrow F_{store}$	$M(Level2_a)$ $= M(F_{poll}) + M(S_{freply}) + M(F_{store})$ $= O(S_e) + O(\bar{F}\frac{L_e}{S_e}) + O(\bar{F}L_e)$
Flow Correlation Analysis ($Level2_b$)	$C_{query} \rightarrow DG_{freply}$	$M(Level2_b)$ $= M(C_{query}) + M(DG_{freply})$ $= O(L_e) + O(\bar{F}L_n)$

Table 4.4: Message Complexity of Port Statistics Collection, Flow Statistics Collection, Link Monitoring, and Flow Correlation Analysis Tasks in the System

form the tasks. The linearity of the process follows the equation, $O(L_n) + O(L_n) + O(L_n) \approx O(L_n)$.

(c) Flow Statistics Collection

Flow statistics collection consists of three tasks according to the diagram shown in Figure 3.6. Similar to port statistics collection, these tasks run sequentially and involve communicating with other components in the system. First, the poll requests are sent from the flow statistics collector to poll flow statistics in the edge switches on the sub-network. Second, each switch replies by returning the flow statistics information to the flow statistics collector. Lastly, the collected flow statistics will be forwarded to the global database for storage. The message complexity measures shown in Table 4.4 showed that there is a linear relationship between the number of switches and links in the sub-network. This means that whenever the number of edge links or number of edge switches in the sub-network increases, the complexity will increase linearly.

(d) Flow Correlation Analysis

The message complexity in flow correlation analysis is derived from two tasks: (1) number of messages sent from correlator to the global database for the purpose of querying flow statistics information of all edge links and target links, and (2) number of messages sent to port statistics collector by the global database to return the queried flow statistics information. The complexity of this process will increase linearly with an increase in the number of edge links and the average number of flows per link in the sub-network. As the number of edge links increases, the number of messages to query and return the queries of flow statistics information between correlator and the global database will also increase.

Overall Scalability

Based on the complexity measures of the components and processes in SCAFE, all of them scaled linearly with the increase in network size and traffic volume. A linear increase in message complexity indicates that SCAFE is scalable as the network size and traffic volume increases.

From the evaluation results, SCAFE scales well with increasing traffic volume and network size, which is suitable for large-network deployments. By taking advantage of the separation of control and data planes in SDN technology, SCAFE is able to collect both port and flow statistics through the control link. This avoids adding extra overhead to the network data link that may lead to congestion. Also, each sub-network has a local controller to prevent overloading the controller in a large network environment.

4.3.2 Fault-Tree Analysis

The fault-tree analysis is used to evaluate both software and hardware fault tolerance of SCAFE. Software faults are related to the functionality of the system components where they experience crashing or being compromised. Hardware faults are related to the hardware of each component in the SCAFE detection system such as correlators, monitor, collectors, and databases. Both software and hardware faults can be further categorised into fail-stop failure where the faults resulted in the DDoS detection stopping completely or Byzantine failure where faults result in the SCAFE detection system providing inaccurate results.

This analysis identifies potential faults in the SCAFE detection system. SCAFE is designed to be in a general active replication environment, where secondary(i.e. redundant) components are actively replicating their respective primary components. Active replication means that both primary and secondary components can run in parallel while receiving the inputs and deterministically generating the same output. With this design, each component is stateless, where all states are the same in both primary and secondary components to ensure the system can function smoothly with faults.

Hardware Related Faults

SCAFE is designed to be fault-tolerant by using simple component redundancy. Each component is modular and can be actively replicated, where primary and secondary components run in parallel. Figure 4.16 shows a detailed diagram of the hardware component faults and their impacts on the functionality of SCAFE.

In figure 4.16, it shows that the detection will fail if the correlator fails. Correlator failure can be caused by three events which are (1) Target link identification failure where it did not trigger the correlator to start correlation analysis when there is DDoS attack traffic in the network, and (2) Correlator stops working due to the correlator crashing, and (3) correlator faulty due to a corrupted database, the component being compromised or human error in the configuration or updates of the component. Traffic collection can cause target link identification failure, monitor stops working, or monitor is faulty. Traffic collection failure is due to the collector crashing, or the collector is faulty due to being compromised or configuration error.

The system can tolerate these faults, and the fault-tolerance methods are explained in the following:

> **(a) What will happen when a switch is ill-functioning? How does it impact the accuracy of detection?**
>
> Switches in the network are installed with SDN capabilities where each switch is responsible for sending port and traffic statistics to the SDN controller. When a switch is ill-functioning, it means that the switch is overloaded and unable to post statistics to the controller. Also, the switch is unable to forward or receive traffic smoothly. The

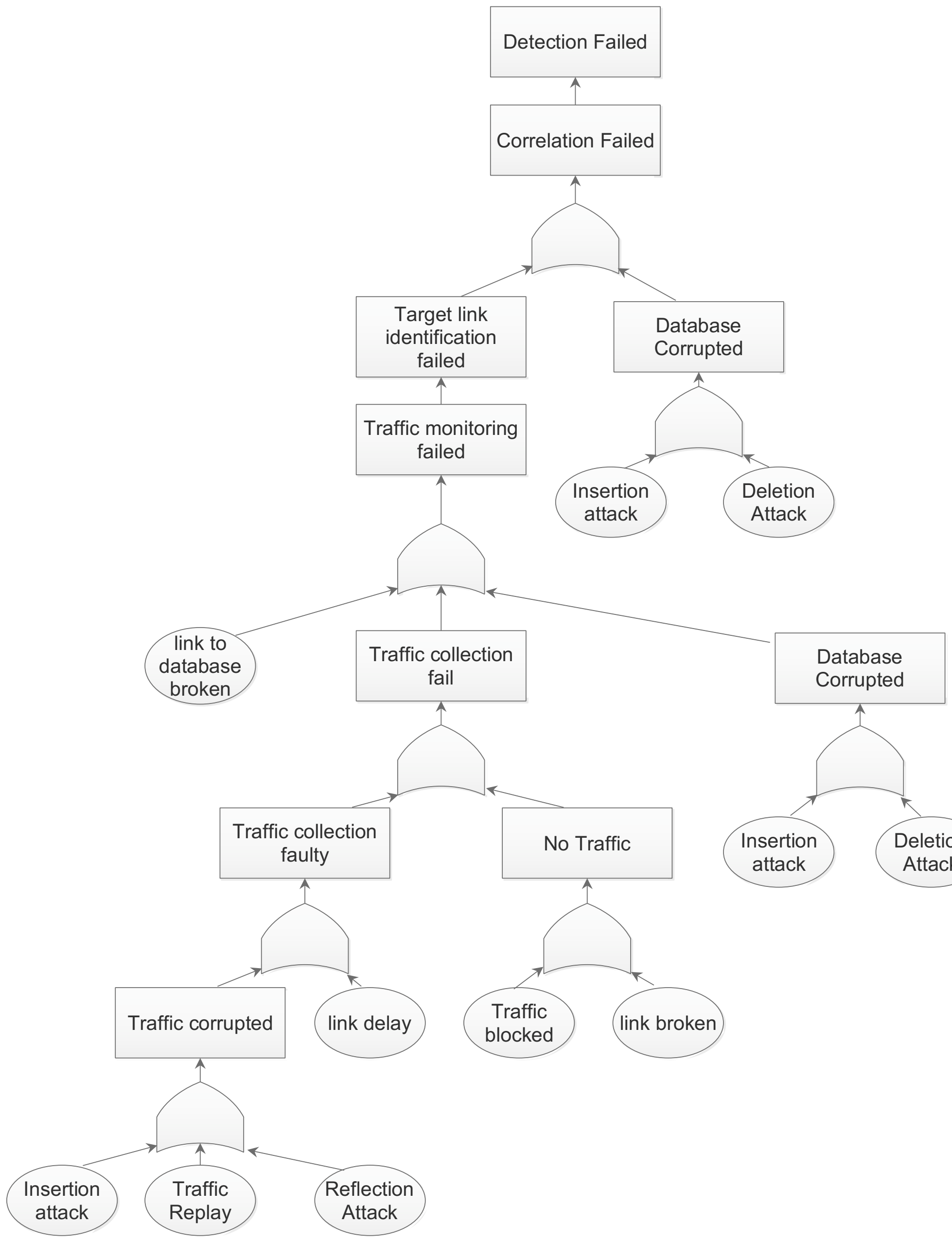


Figure 4.16: Fault Tree for Hardware Faults

system accuracy is not impacted by the missing statistics from the failed switch since there will be no traffic regardless of attack traffic or normal traffic passing through the switch. According to the design of conventional networks, traffic intended to be forwarded to a failed switch will be re-forwarded to another switch when failure happens. Therefore, the traffic statistics will be captured by another switch in the network to be analysed for attack flows.

(b) What will happen when a collector fails? How does it impact the accuracy of detection?

A collector in the SCAFE detection system is responsible for collecting port and flow statistics in its sub-network. When a port statistics collector in a particular sub-network fails, the secondary collector, which runs in parallel with the primary collector, continues port/flow statistics collection to detect congested links in the network. Therefore, the accuracy of the detection system will not be impacted. However, if the secondary collector fails before the primary collector is fixed, then the accuracy of the detection system might be reduced because the monitor in the sub-network is unable to monitor the traffic of that particular sub-network. This will result in misdetection which will impact the overall accuracy of the detection system.

(c) What will happen when a monitor fails? How does it impact the accuracy of detection?

A monitor is responsible for monitoring port statistics collected by the collector to detect congested network links in the sub-network that can potentially be a DDoS attack target link. When the monitor fails, its secondary monitor will continue the monitoring process. This is similar to the fault-tolerance ability of the collector. A failed monitor will result in DDoS target links in the particular sub-network going undetected and reduce the accuracy of the system linearly.

(d) What will happen if the correlator fails? How does it impact the accuracy of detection?

A correlator is responsible for correlating traffic flows from different parts of the network to detect DDoS attack traffic. The correlator requires flow statistics stored in the global database for correlation analysis. This is the most crucial part of the system where fine-grained detection to detect attack traffic flows is executed. Without the correlator, the flow correlation analysis is unable to perform, and the detection system will only identify DDoS attack target links. The correlator is also redundant whereby a replica correlator will resume the correlation analysis function when the correlator fails. The accuracy of SCAFE to detect traffic flows will exponentially reduce when both correlator and replica correlator fails.

(e) What will happen if the database is not working? How does it impact the accuracy of detection?

The local and global databases are responsible for storing traffic statistics for monitoring and correlation analysis. The collector from each sub-network will update the local database with port statistics periodically and with flow statistics when the attack flow detection process is triggered. Port statistics are later retrieved for link monitoring, and flow statistics are extracted for correlation analysis. If a database is not working, then it means that links cannot be monitored and attack flows cannot be detected. However, SCAFE is designed for link monitoring process to be decentralised where each sub-network has a local database. Therefore, the failure of the database will cause the detection accuracy to reduce linearly depending on the number of links unable to be monitored due to the failed database.

(f) What will happen if one of the network links in the detection system communication path fails? How does it impact the accuracy of detection?

Network links are responsible for the communication channel between components in the detection system. If a network link is not working or has failed, components in the detection system will not be able to communicate with each other. For example, if the network link connecting between the collector and a local database failed, the collector will not be able to update nor retrieve information from the database.

Software Related Faults

SCAFE is designed to be fault tolerant by duplicating the functionality of the software. Each component is modular and can its functionality can be copied or transfer to another device when the component stops working. Figure 4.17 shows a detailed diagram of the fault tolerance level of the system impacting its software functionality.

In figure 4.17, it shows that the detection will fail if the correlation analysis fails. The correlation analysis process can be caused by two events which are (1) Target link identification failure where it did not trigger the correlator to start correlation analysis when there is DDoS attack traffic in the network, and (2) the global database is corrupted, and no data can be retrieved for the correlation analysis process. Target link identification failure can be caused by the inability of traffic monitoring to detect any attack links because traffic collection on the port statistics collector failed, the local database is corrupted, or connection between the components used for traffic monitoring is broken. Port statistics collection can fail due to corrupted traffic or link delay causing the information to be dropped before reaching the collector for processing.

The system can tolerate these software faults, and the fault-tolerance methods are explained in the following:

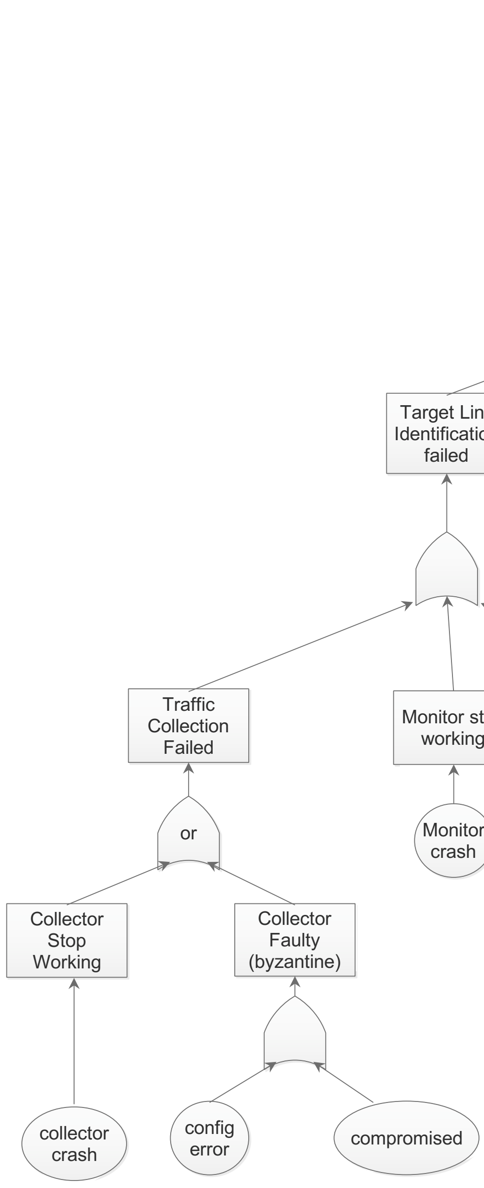


Figure 4.17: Fault Tree for Software Faults

(a) What will happen when the traffic collection mechanism fails?

The traffic collection mechanism runs periodically in the SCAFE detection system to detect DDoS attack target links. This mechanism runs on an NFV server locally on each sub-network. When the traffic collection mechanism fails, a redundant traffic collection mechanism will start and continue with the collection mechanism by sending requests to switches to poll port statistics until the failed mechanism is restored. The accuracy of the SCAFE detection system will not be impacted because a redundant traffic collection mechanism is used to ensure no port statistics are lost and missing for the use in the target link identification process.

(b) What will happen when the correlation mechanism fails?

The correlation mechanism only runs when a target DDoS attack link is detected. When the correlation mechanism is not working, the SCAFE detection system will be unable to detect attack flows. This would also impact the accuracy of the SCAFE detection system significantly. To tolerate faults impacting the correlation mechanism, the mechanism can be duplicated on another device. When the correlation mechanism fails, the correlation process can be transferred to the other device to detect attack flows.

Overall System Fault Tolerance

SCAFE is tolerant of components crashing, corruption, and broken network links because the detection components are designed to be modular and can be duplicated to allow active replications to prevent the DDoS detection system failing or its accuracy decreasing during a fault.

4.4 Summary

This chapter demonstrated the scalability and fault tolerance capability of SCAFE through qualitative and quantitative analysis of the system architecture. SCAFE is implemented in the GENI testbed for quantitative analysis.

The quantitative analysis shows that the use of threshold values only impacts the false positive rate of DDoS attack detection in both direct and indirect DDoS attacks. On the other hand, the observation period does not affect detection accuracy significantly where SCAFE is able to provide similar accuracy with a shorter observation period. This validates the early detection capability of SCAFE.

The qualitative analysis shows the scalability analysis with regards to message complexity and fault-tolerant analysis based on fault trees. The scalability analysis proves that SCAFE components and processes are able to scale linearly as the amount of traffic flow and the number of network links increases. On the other hand, the fault-tolerant analysis shows the behaviour of SCAFE during the event of software and hardware faults by using fault trees. The modularity and active replication of SCAFE components increase the fault tolerance level of the DDoS attack detection system. Moreover, the decentralised design of some components and mechanisms in SCAFE allows partial failures when both primary and secondary components or mechanisms fail.

Overall SCAFE is scalable and fault-tolerant. However, its accuracy can be improved. Normal volumetric features are used in investigating the detection accuracy in this chapter. However, alternative features can also be used to detect DDOS attacks. Next chapter explores the use of an alternative traffic features called entropy-based features to increase the accuracy in distinguishing attack traffic from normal traffic.

Chapter 5

Identifying Good Entropy-based Features

This chapter focuses in answering the second sub-question of this thesis:

> *What are the good entropy-based features and useful parameter settings for distinguishing attack traffic from normal traffic?*

Although entropy is popularly used in detecting DDoS attack traffic, it still lacks the generality in detecting various intensity DDoS attacks accurately. This thesis hypothesises that using different entropy measures, window sizes, and the entropy-based features may affect the accuracy of detecting DDoS attacks. This means that using some particular entropy measures, window sizes, and entropy-based features in DDoS attack detection may provide higher accuracy than other measures in distinguishing attack traffic amongst normal traffic. Therefore, this chapter explores various entropy measures, gains an understanding of the trade-off between window size and accuracy, and identifies a set of useful entropy-based features and parameter settings for constructing good entropy-based features that are useful in the identification of both high and low-intensity DDoS attacks.

In the next three subsections, we present the dataset used, steps to construct entropy-based features, the influence of entropy measures and the influence of window sizes in revealing attack traffic patterns. Then, we discuss the relationship of entropy-based features with the SCAFE architecture and conclude the chapter by providing a summary on the usefulness of entropy features.

5.1 Dataset

This chapter uses the UNB ISCX 2012 intrusion detection evaluation dataset (ISCXIDS2012) [98]. This dataset contains seven days (Monday through to Sunday) of network activities which include high and low-intensity attack traffic. IRC Botnet DDoS attack traffic represents the high-intensity DDoS attack and HTTP Denial of Service attack traffic represents the low-intensity DDoS attack traffic. Each day has a different combination of attacks. This dataset was chosen because it is one of the recent datasets that is popularly used by other researchers.

5.2 Constructing Entropy-based Features

Entropy-based features can be constructed using two steps: (1) extract features from the raw dataset, (2) compute entropy values based on predefined entropy measures using a specific time interval.

5.2.1 Step 1 - Extract Features from Raw Dataset

All possible traffic features that can be extracted from packet header information are used to construct entropy-based features except redundant features (i.e., Absolute time, resolved/unresolved addresses) or features that contains null values (i.e., Cisco VSAN, 802.1Q VLAN id, Expert Info

Severity). Table 5.1 shows the list of traffic features that can be extracted from packet header information.

No	Features	Definition
1	Delta Time	Time since the previous packet was captured.
2	Source IP Address	Source IP address of the packet
3	Destination IP Address	Destination IP address of the packet
4	Source Port Number	Source port number of the packet
5	Destination Port Number	Destination port number of the packet
6	Source MAC Address	Hardware address of the previous network router/ host the packet is coming from
7	Destination MAC Address	Hardware address of the next-hop network router/host the packet is going to
8	Source Network Address	Source Network address of the packet
9	Destination Network Address	Destination Network address of the packet
10	Protocol Identifier	Type of protocol identifier such as HTTP, TELNET, and DNS
11	Packet Length	Size of packets in bits
12	IP DSCP Value	Differentiated Services Code Point value of each packet
13	TCP Sequence Number	TCP sequence number relative to the first seen segment in a TCP session
14	TCP Window Length	Maximum amount of received data, in bytes, that can be buffered at one time on the receiver
15	TCP Payload	Size of TCP Payload

Table 5.1: List of Traffic Features from a Tcpdump File Extracted using Wireshark

5.2.2 Step 2 - Compute Entropy Values of Features based on a Pre-Defined Entropy Measure and Window Size

Two types of entropy-based features are computed by calculating their entropy values: *regular entropy-based features* created by calculating the entropy of single traffic features and *entropy variation features* created by calculating the variation between two distinct *regular entropy-based features*. Table 5.2 shows the list of regular entropy-based features and Table 5.3 shows the list of entropy variation features computed from features extracted in Section 5.2.1.

Regular Entropy-based Features	
Delta Time Entropy (D.Time)	Protocol Identifier Entropy (Protocol)
Source IP Address Entropy (S.IP)	Destination IP Address Entropy(D.IP)
Source Port Address Entropy (S.Port)	Destination Port Address Entropy (D.Port)
Source MAC Address Entropy (S.MAC)	Destination MAC Address Entropy (D.MAC)
Source Network Address Entropy (S.Net)	Destination Network Address Entropy (D.Net)
Packet Length Entropy (P.Length)	IP DSCP Value Entropy (DSCP)
TCP Sequence Number Entropy (Seq)	TCP Window Length Entropy (W.Length)
TCP Payload Entropy (Payload)	

Table 5.2: List of Regular Entropy-based Features Constructed

Entropy Variation Features
Separation IP Address Entropy(V.IP) [21]
Separation Port Number Entropy(V.Port) [99]
Separation MAC Address Entropy (V.MAC) [99]
Separation TCP Information Entropy(V.TCP)[99]

Table 5.3: List of Entropy Variation Features Constructed

5.2.3 Influence of Entropy Measures in Traffic Patterns

This subsection focuses on examining the influence of different entropy algorithms in the accuracy of detecting DDoS attack traffic. Four different entropy measures, namely, Shannon, Tsallis, Rényi and Zhou entropy measures were used in this evaluation.

Network Traffic Containing High-Intensity DDoS Attack

The Tuesday's network activities in the ISCXIDS2012 dataset contains the IRC Botnet based DDoS attack traffic. This dataset is used as a representation of network traffic containing high-intensity DDoS attack traffic as shown in Figure 5.1, Figure 5.2, and Figure 5.3.

Figure 5.1 shows the traffic patterns of high-intensity DDoS attack traffic and normal traffic based on the seven entropy features, namely Delta Time, Source & Destination IP Addresses, Source & Destination Port Numbers and Source & Destination MAC Addresses.

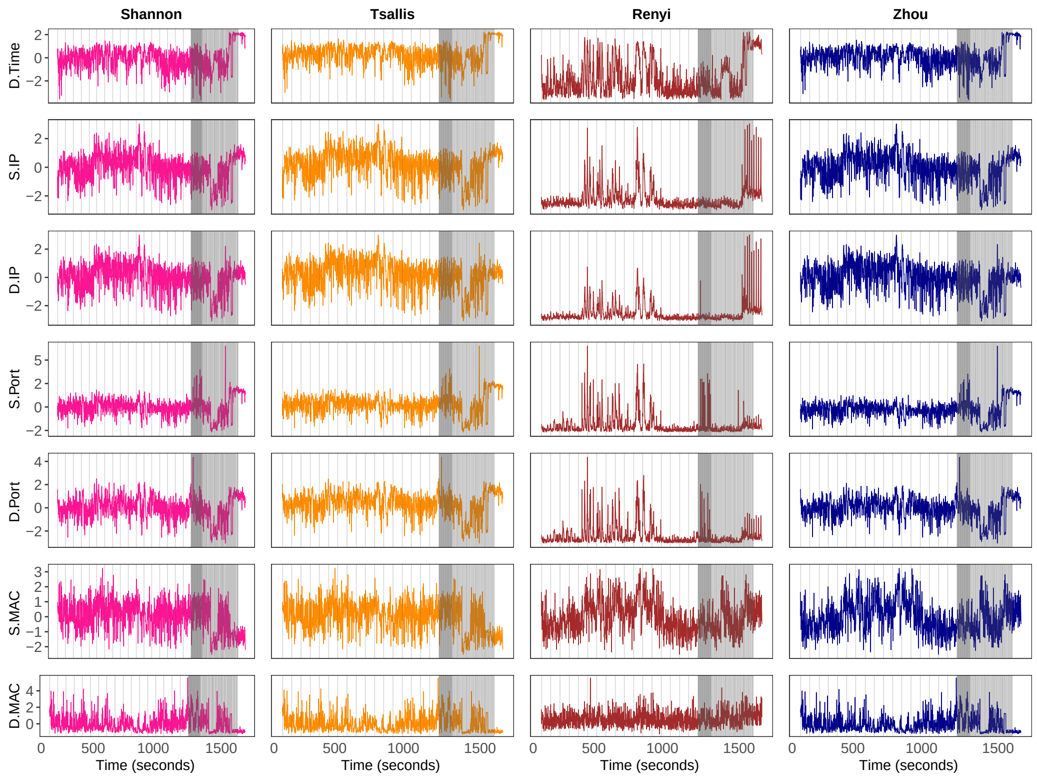


Figure 5.1: Different Entropy Measures of Entropy-based Features with High-Intensity Attack Traffic - 1

Figure 5.2 shows the traffic patterns of high-intensity DDoS attack and normal traffic of Protocol Identifier, IP DSCP Value, TCP Sequence Number, TCP Window Length, TCP Payload, Source Network Address and Destination Network Address.

Figure 5.3 depicts the traffic patterns of high-intensity DDoS attack and normal traffic of entropy variation features, namely Separation IP Address, Separation Port Number, Separation MAC Address, and Separation TCP Information.

In all three figures (Figure 5.1, Figure 5.2, Figure 5.3), the grey area indicates the presence of DDoS attack, while the non-grey area indicates normal traffic. From these figures, entropy values generated using Shannon, Tsallis and Zhou entropy measures provide a similar traffic pattern

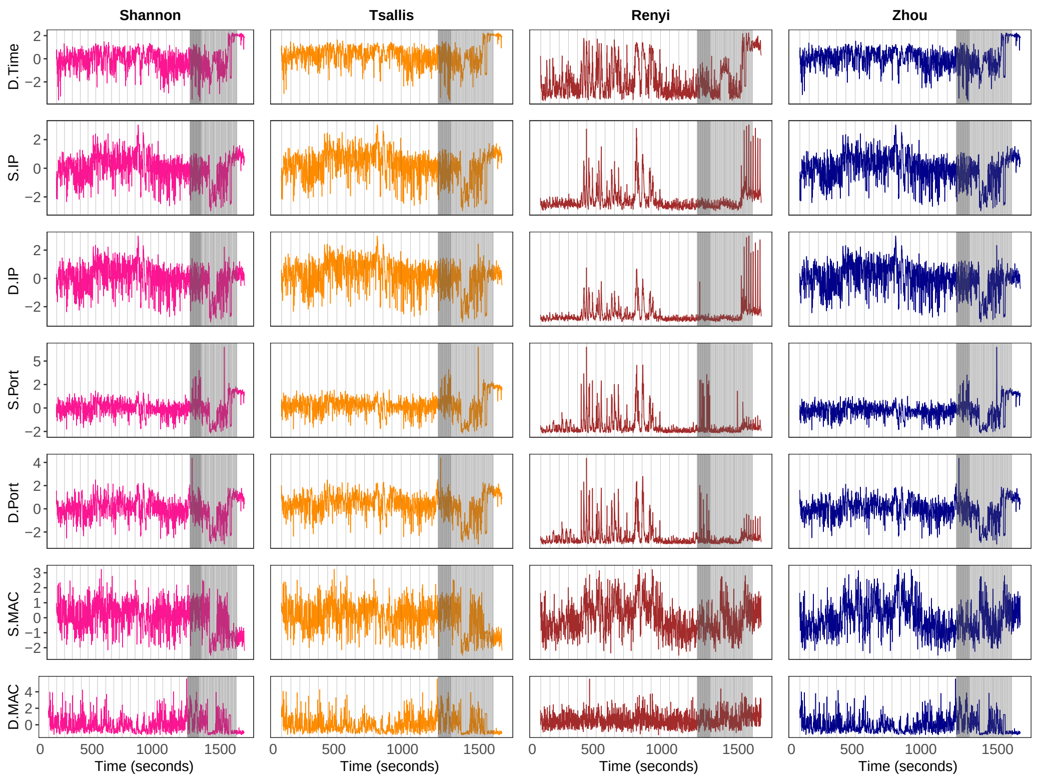


Figure 5.2: Different Entropy Measures of Entropy-based Features with High-Intensity Attack Traffic - 2

whereas Rényi entropy gives quite a different traffic pattern than the others. Since Tsallis and Zhou entropies are a generalisation of Shannon entropy, traffic patterns generated will be similar.

In this evaluation, we observe that it is possible to distinguish high-intensity DDoS attack traffic from normal traffic quite easily when entropy measures are applied to most traffic features except for entropy measures calculated using the TCP Sequence ("Seq") field as shown in Figure 5.3. This is because in most entropy-based features generated, the entropy values of attack traffic have a much smaller range than normal traffic. For example, in Figure 5.1, the attack traffic entropy values of the Source IP address feature, using the Shannon entropy algorithm, lie between 0.5 to

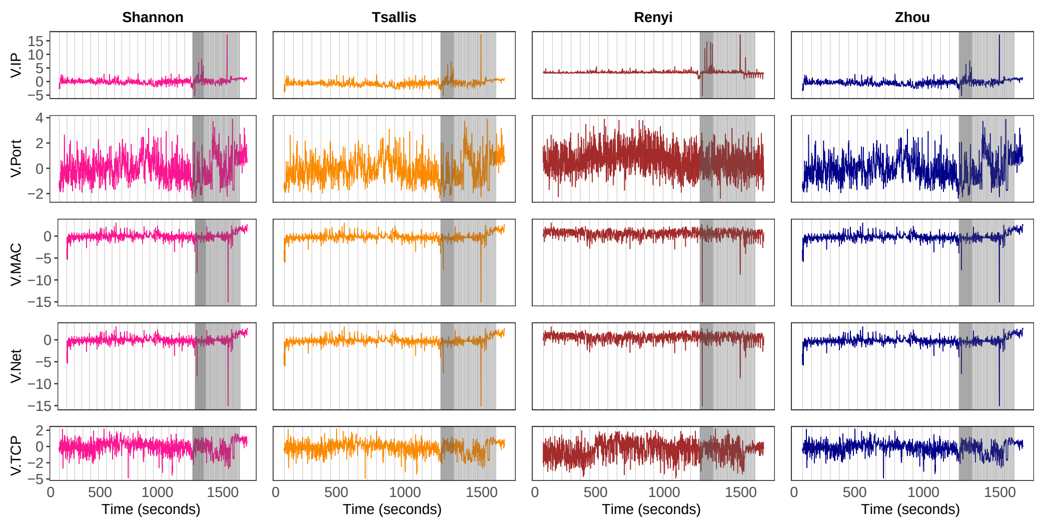


Figure 5.3: Different Entropy Measures of Entropy-based Features with High-Intensity Attack Traffic - 3

0.75 whereas the normal traffic entropy values of the same entropy lie between 0.15 to 0.9. Rényi entropy performs poorly in providing significant differences between attack traffic and normal traffic as both traffic entropy values lie in a similar range.

The differences between these entropies, as shown in the figures, are the distributional differences between attack and normal traffic. There is not much of a difference in the distributional patterns of attack and normal traffic using entropy-based features constructed using Tsallis entropy, which may not be useful in identifying stealthy DDoS attacks. This is because the differences between attack and normal traffic entropy values may be too small to be noticeable and can be easily misclassified.

Shannon and Zhou entropies provide more distinct differences in distributional patterns and entropy values between attack and normal traffic. Unlike Shannon and Zhou entropies, Rényi entropy gives almost no difference in the traffic patterns between attack traffic and normal traffic. However, Rényi entropy shows significant differences between the traffic patterns of attack traffic and normal traffic when applied to TCP Window

Length (”W.Length”) and Delta Time between packets (”D.Time”) entropy features, in which it shows more explicit differences than other entropy algorithms (Zhou, Shannon, and Tsallis) examined.

Overall, Rényi entropy does not perform well at distinguishing high-intensity DDoS attack traffic from normal traffic whereas Shannon, Tsallis and Zhou entropies perform better and can identify DDoS attack traffic relatively well.

Network Traffic Containing Low-Intensity DDoS Attack

The Monday’s network activities in ISCXIDS2012 dataset contains HTTP Denial of Service attack traffic. We use this dataset to represent network traffic containing low-intensity DDoS attack as shown in Figure 5.4, Figure 5.5, and Figure 5.6.

Figure 5.4 depicts the traffic patterns of low-intensity DDoS attack and normal traffic based on the seven entropy features, namely Delta Time, Source & Destination IP Addresses, Source & Destination Port Numbers and Source & Destination MAC Addresses.

Figure 5.5 shows the traffic patterns of high-intensity DDoS attack and normal traffic of Protocol Identifier, IP DSCP Value, TCP Sequence Number, TCP Window Length, TCP Payload, Source Network Address and Destination Network Address.

Figure 5.6 depicts the traffic patterns of low-intensity DDoS attack and normal traffic of entropy variation features, namely Separation IP Address, Separation Port Number, Separation MAC Address, and Separation TCP Information.

Similar to Figures 5.1, Figures 5.2, and Figures 5.3, the grey area shown Figures 5.4,Figures 5.5,and Figures 5.6 indicate that DDoS attack traffic is present in that time interval. Unlike high-intensity DDoS attacks, it is difficult to distinguish between low-intensity attack traffic and normal traffic. Most entropy-based features such as Delta Time (”D.Time”), source IP address (“S.IP”), destination IP address (“D.IP”), source port number

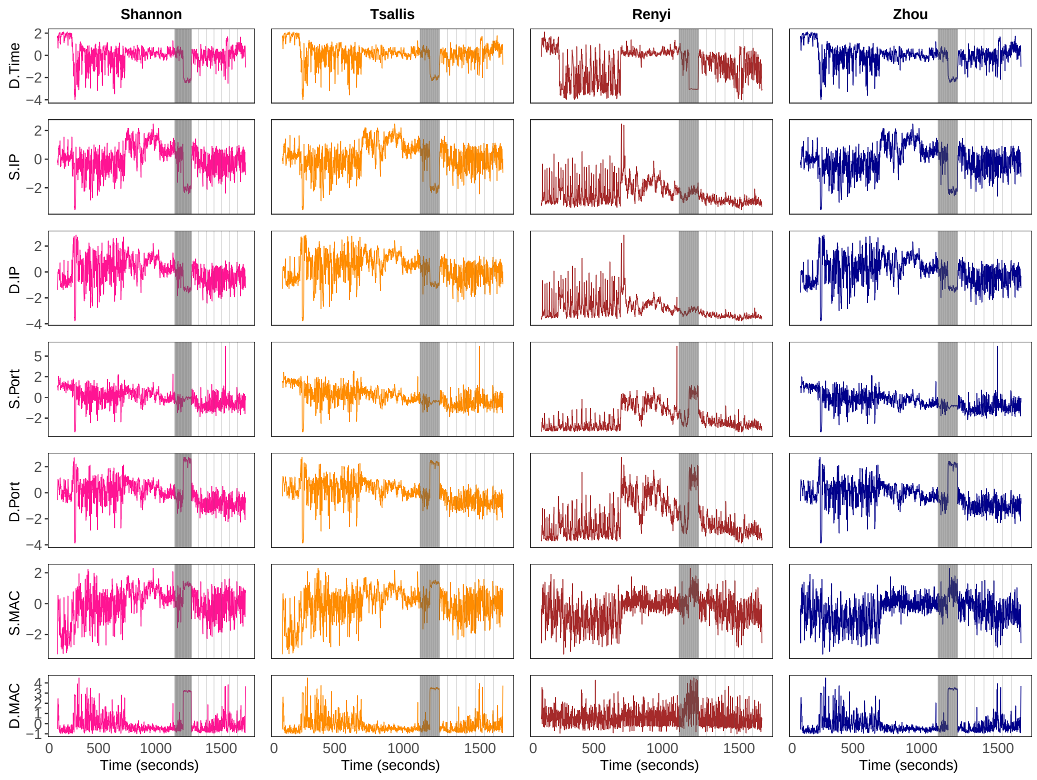


Figure 5.4: Different Entropy Measures of Entropy-based Features with Low-Intensity Attack Traffic - 1

("S.Port"), destination port number ("D.Port"), source MAC address ("S.MAC"), destination MAC address ("D.MAC"), protocol identifier ("Protocol"), and packet length ("P.Length"), show the entropy values decreases during the attack. However, this is true for only a small part of the attack, specifically in the middle of the attack (i.e. around the 1200-second area in the graphs). This phenomenon indicates that the low-intensity attack requires some time before it shows a significant change in the traffic distribution in the network.

TCP Sequence ("Seq") and Window Length ("W. Length") entropies shown in Figure 5.5 using Shannon, and Zhou entropies show a clear distinction between attack traffic and normal traffic.

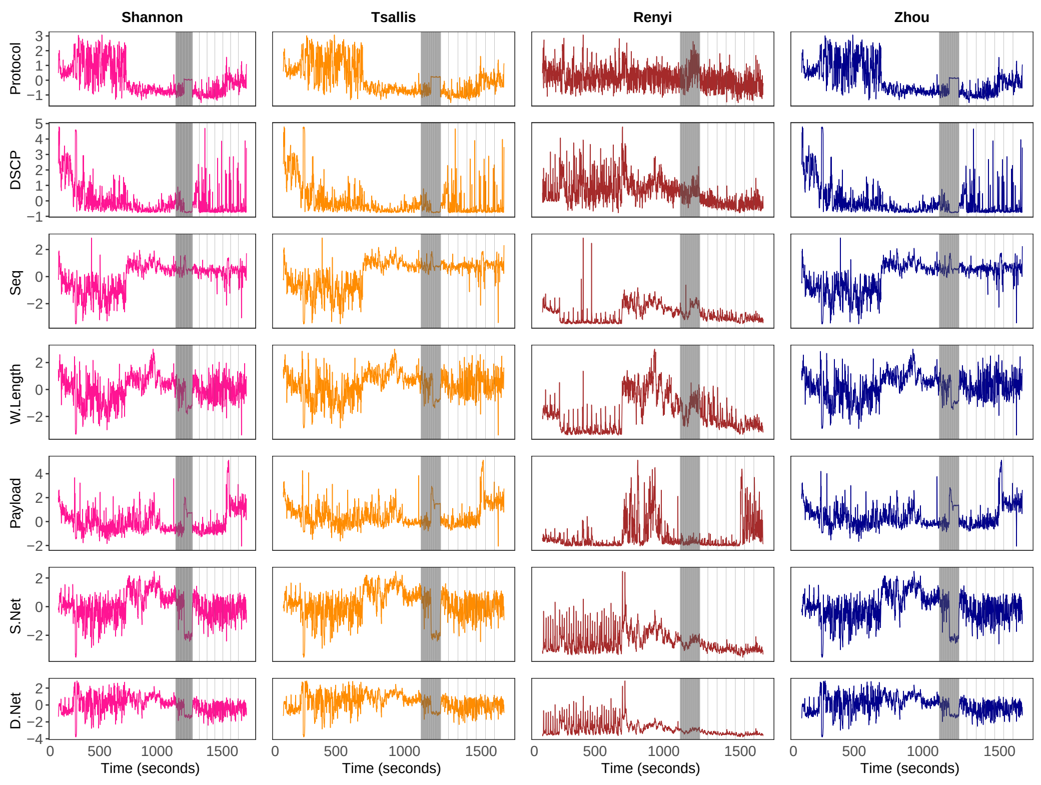


Figure 5.5: Different Entropy Measures of Entropy-based Features with Low-Intensity Attack Traffic - 2

5.2.4 Effects of Window Size in Traffic Patterns

This subsection focuses on examining the influence of window size in calculating entropy values for DDoS detection on network traffic containing both high and low-intensity DDoS traffic. As the window size determines which traffic is used to calculate an entropy value, it is important to find the right size to prevent inaccuracy in detecting attack traffic from normal traffic. For example, if the window size is set too large, DDoS attacks that lasted for a shorter period than the window size may be hidden and the entropy value computed may not show the distinct difference between attack traffic and normal traffic. However, if the window size is set too small, entropy values generated may be too sensitive to the changes in the

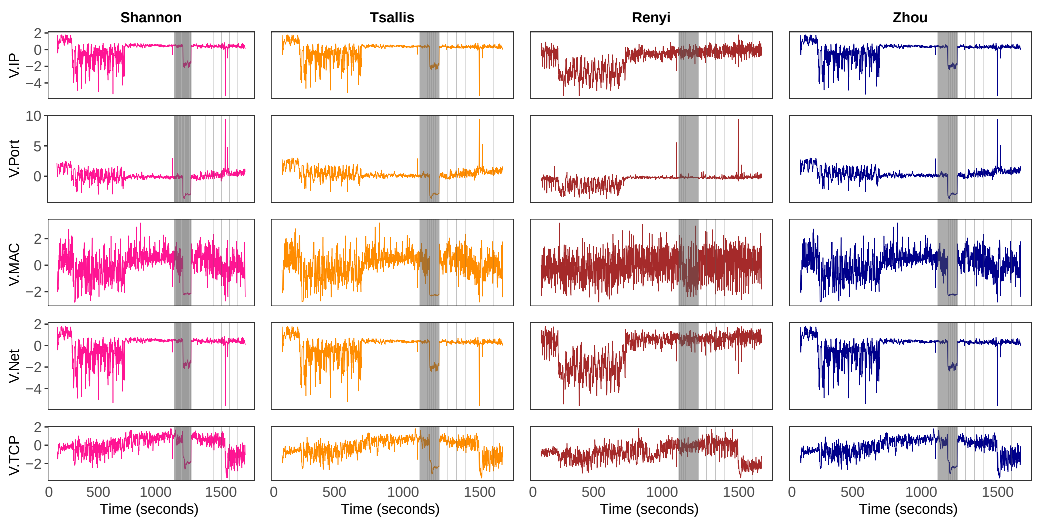


Figure 5.6: Different Entropy Measures of Entropy-based Features with High-Intensity Attack Traffic - 3

traffic. This means that a slight change in the network can be regarded as an attack even though it is not. In this case, a lot of false alarms may occur.

Network Traffic Containing High-Intensity DDoS Attack

Here, we compare six different window sizes (30, 60, 90, 120, 150, and 180 seconds) and observe the traffic patterns generated. Traffic patterns based on the entropy values of traffic features are shown in Figure 5.7 and Figure 5.8.

Inspecting the figures, we observe that all features have similar traffic patterns even though different window intervals are applied. We observe that there are almost no differences in traffic patterns between these four window intervals. Entropy is being calculated more frequently in the 30-second interval compared to the 60-second interval, but both gave similar traffic patterns. The lack of differences in attack traffic and normal traffic patterns suggest that the size of the window used for generating traffic feature entropy values only gives a slight effect on the accuracy of DDoS attack detection.

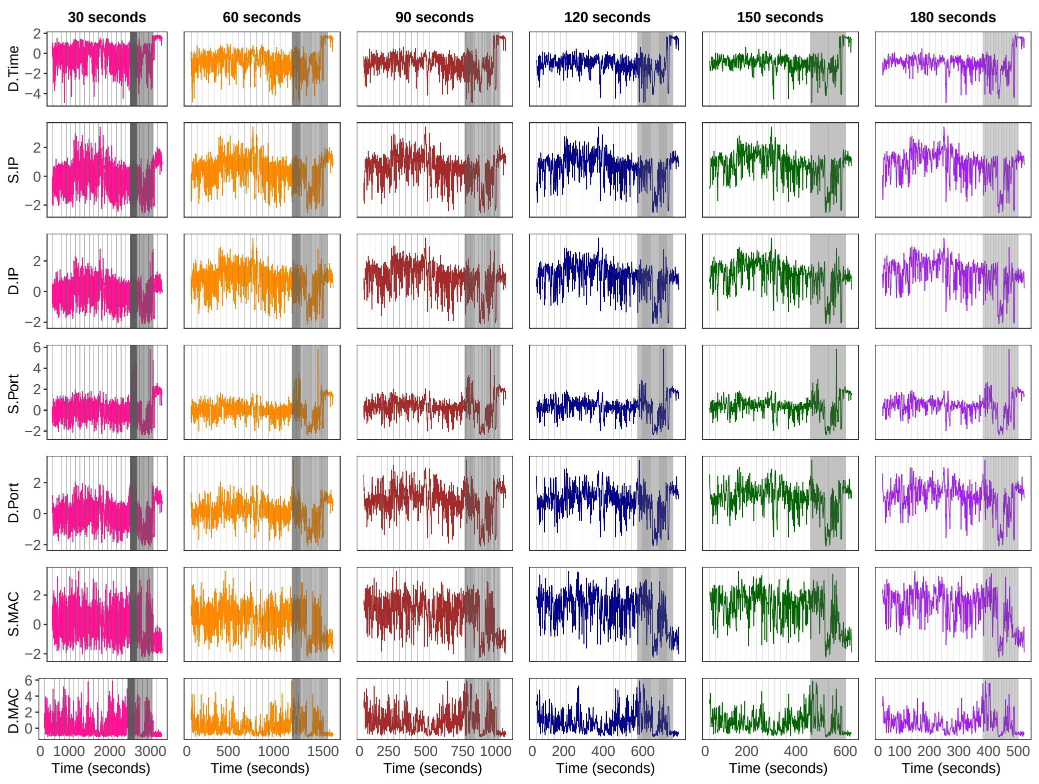


Figure 5.7: Shannon entropy-based features with high-intensity attack constructed using different window sizes - 1

5.2.5 Relationship with SCAFE Architecture

The entropy-based features constructed in this chapter can be used in both Level 1 and 2 of the SCAFE architecture discussed in Chapter 3. In the Target Link Identification level (Level 1) of the detection system, entropy-based features can be used to indicate signs of DDoS attacks by measuring their entropy values. When the value of these entropy-based features exceeds the threshold, an alert can be raised to trigger the start of SCAFE's Attack Flow Detection Level (Level 2). In level 2, the value of entropy-based features in network links can be used to measure the correlation between two links as opposed to using flow information in the current SCAFE architecture design.

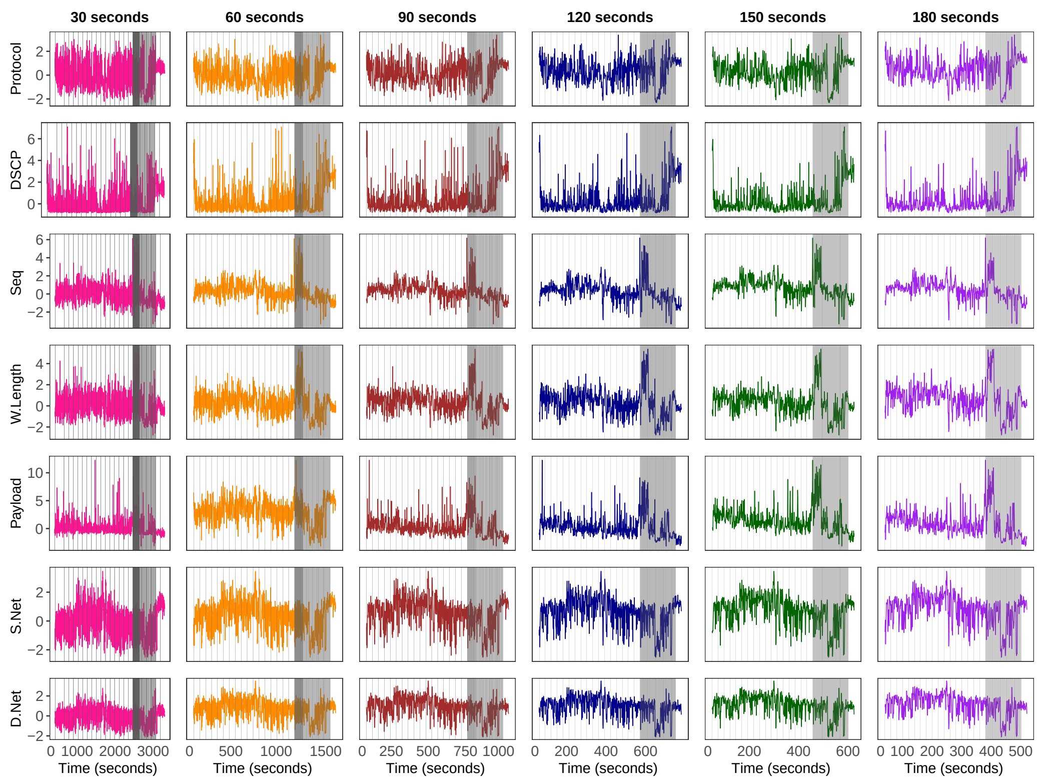


Figure 5.8: Shannon Entropy-based Features with High-Intensity Attack Constructed using Different Window Sizes - 2

The effectiveness of entropy-based features in the SCAFE architecture could not be evaluated in this research due to a constraint on resources. SCAFE was implemented on GENI which is a shared testbed for cybersecurity research. Constructing entropy-based features in GENI would require traffic data to be collected on the packet level, which would require a huge amount of storage space. Since GENI provides a limited amount of disk space (i.e. 10GB) in each node in a shared environment, it would not be possible to perform data collection. In addition, downloading a large amount of data from GENI would also take a huge amount of time. Therefore, the usage of entropy-based features in the SCAFE architecture is not studied in this thesis.

5.2.6 Summary

In this chapter, we examine the usefulness of entropy-based features in detecting DDoS attacks. We start by analysing each entropy-based feature as shown in Figure 5.1 to Figure 5.8. We found that entropy-based features such as Delta Time (Figure 5.1), Destination Port Number (Figure 5.1), Destination MAC Address (Figure 5.1), Protocol Identifier (Figure 5.2), and Packet Length (Figure 5.3) can show a more distinct difference between attack and normal traffic.

Although some entropy-based features can be effective in detecting a DDoS attack, they may not be effective for all types of DDoS attack. For example, Delta Time can be effective against DDoS attacks that send attack traffic at a constant rate but may not be effective against DDoS attacks that send attack traffic at a variable rate that is similar to the rate of normal traffic. Destination IP Address is effective against DDoS attacks that send attack traffic to the same IP Address but may not be effective against DDoS attacks that send attack traffic to multiple IP addresses such as the Crossfire attack [12]. An attacker can easily defeat the detection scheme based on single entropy features by randomising the attack traffic sending rate and IP addresses.

Also, at the earlier stage of an attack, the temporal change of a single entropy feature may be too small to be noticed by the detection scheme, especially when it is observed close to the attack source. Temporal changes are changes that could be observed over time. Entropy values before an attack and during an attack could be different based on the characteristics of attack traffic and its differences with normal traffic. These differences might not be noticeable in the early stage of an attack, before the aggregated attack traffic meets at the aggregation point, but become more noticeable after some time where attack volumes are increasing over time.

We conclude that entropy-based features are useful in DDoS attack detection, but their accuracy relies on the type of entropy features used, the type of entropy measures used and the type of attacks that are being detected. Therefore, the next chapter will investigate a generalised traffic classification based on entropy-based features that can detect various types of DDoS attack traffic with consistent accuracy.

Chapter 6

Traffic Classification using Multiple Entropy-based Features and Three (3) Machine Learning Classifiers in DDoS Attack Detection

The third sub-question for developing an accurate and early DDoS attack detection system is:

> *How can we improve the accuracy of traffic classification using entropy-based features?*

Since entropy-based features were found to be promising for distinguishing between attack traffic and normal traffic in the previous chapter, this chapter aims to answer the aforementioned sub-question by proposing a generalised traffic classification scheme: Entropy-based Features and Three (3) Machine Learning Classifiers (E3ML). The rationale behind the design of E3ML is that it utilises the strength of entropy-based features studied in Chapter 5 with machine learning classifiers to improve detection accuracy in the detection of both high and low-intensity DDoS attacks. By utilising the traffic patterns of multiple entropy-based features and a voting mechanism based on MLP, RNN and ADT classifiers, E3ML is able to improve the overall accuracy of both high and low-intensity DDoS attack detection in the network.

The remainder of this chapter presents the overview of the E3ML DDoS attack detection scheme, its attack detection model and the evaluation results.

6.1 Overview

E3ML is a traffic classification scheme that is designed to address the performance issues mentioned in Section 1.2. All entropy-based features identified in Chapter 5 are used to expose different types of DDoS attack traffic in the network. These features act as inputs to E3ML's classification model, where the classification model is formed through a voting mechanism based on three machine learning classifiers: multilayer perceptron (MLP), recurrent neural network (RNN) and alternating decision tree (ADT). An initial pilot study was conducted, where every single ML classifier available in the WEKA library was evaluated using 10 cross-fold validation. The top three classifiers were chosen as the ML classifiers used in E3ML after comparing the performance results of all the classifiers: MLP, RNN, and ADT gave the highest accuracy among all other classifiers found in WEKA.

E3ML attack detection scheme is composed of two parts: (1)Entropy-based Feature Construction and (2) Attack Detection.

6.1.1 Entropy-based Feature Construction

The first part of E3ML, an entropy-based feature construction module (Figure 6.1) for constructing two types of entropy-based features: *regular entropy-*

based features and *entropy variation features*. *Regular entropy-based features* are created from the entropy of raw traffic features, whereas the *entropy variation features* are created by calculating the variation between two distinct *regular entropy-based features*. Since each entropy-based feature studied in Chapter 5 gave different accuracy results for different DDoS attacks, this thesis hypothesises that combining these two type of features may increase the consistency in identifying different types of DDoS attacks accurately.

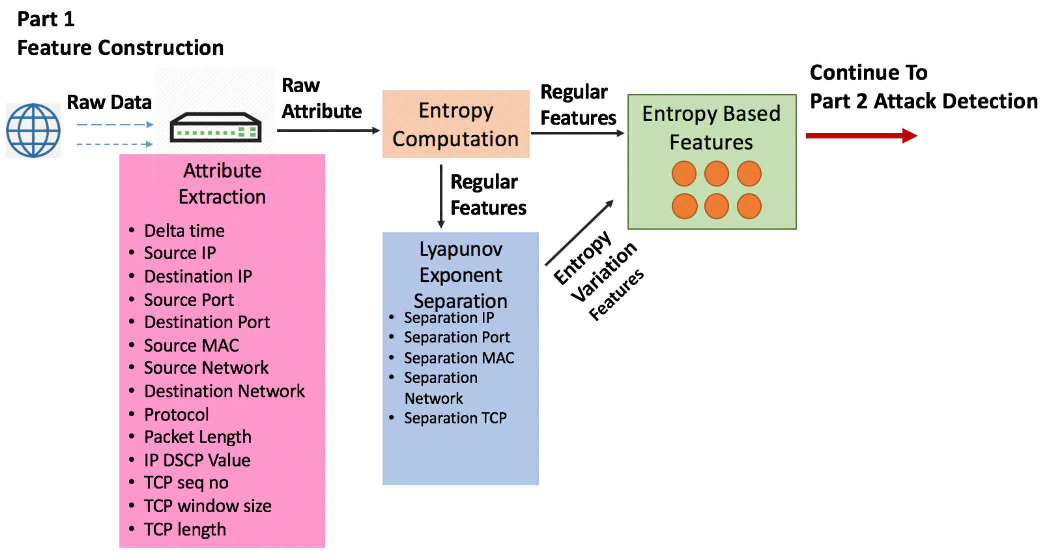


Figure 6.1: Feature Construction

The feature construction is done in three steps.

1. **Raw feature extraction**: The raw features are extracted from each packet header. The features such as source IP address, destination IP address, source port number, destination port number, and protocol identifier, are commonly used features, whereas delta time, packet length, TCP sequence number, TCP window length are the uncommon features used in DDoS attack detection [100].

2. **Entropy calculation**: E3ML computes the entropy value of each traffic feature based on the Shannon entropy using specific window size,

W where $W = 60$ seconds in all of our experiments. Since the previous study in Section 5.2 has shown that the entropy measures and window size selections have a very little impact on the accuracy of DDoS attack detection, Shannon entropy is used as the entropy measure and 60 seconds as the window size for evaluation of the E3ML detection scheme. A window size of 60 seconds is used because it was shown as a suitable classification point and a common practice in the literature [21]. The Shannon entropy measure is used because it is a commonly used entropy measure in traffic classification. Features computed in this stage are known as regular entropy-based features as described in Table 6.1.

3. **Build entropy variation features**: By using combinations of two regular entropy-based features generated from previous steps, entropy variation features are generated using the Lyapunov exponent separation method proposed by Ma et al. [21] as shown in Equation (6.1).

$$\lambda_k = \frac{1}{t_k} ln \frac{\hat{\mathrm{H}}_s(k)}{\hat{\mathrm{H}}_d(k)} \tag{6.1}$$

From Equation (6.1), λ_k is the rate of separation between two distinct features, for example, the entropy of source IP address ($\hat{\mathrm{H}}_s(k)$) at time t_k and the entropy of destination IP address ($\hat{\mathrm{H}}_d(k)$) at time t_k where k is the time sequence of the traffic. This method generates a new feature that can distinguish attack traffic effectively on high-intensity DDoS attacks [21]. Four new entropy variation features (*separation port number, separation MAC address, separation network address* and *separation TCP value*) are generated using this method to evaluate their effectiveness at detecting a wider range of DDoS attacks, in particular, both high and low-intensity DDoS attacks.

These features are described in Table 6.2.

No	Features	Definition
1	Delta time	Entropy value of Delta Time of the network traffic in a specific time interval
2	Source IP Address	Entropy value of Source IP address of the network traffic in a specific time interval
3	Destination IP Address	Entropy value Destination IP address of the network traffic in a specific time interval
4	Source Port Address	Entropy value Source port number of the network traffic in a specific time interval
5	Destination Port Address	Entropy value Destination port address of the network traffic in a specific time interval
6	Source MAC Address	Entropy value of source MAC address of the network traffic in a specific time interval
7	Destination MAC Address	Entropy value of destination MAC address of the network traffic in a specific time interval
8	Source Network Address	Entropy value of Source Network address of the network traffic in a specific time interval
9	Destination Network Address	Entropy value of Destination Network address of the network traffic in a specific time interval
10	Protocol	Entropy value of protocol of the network traffic in a specific time interval
11	Packet Length	Entropy value of packet length of the network traffic in a specific time interval
12	IP DSCP Value	Entropy value of IP DSCP Value of the network traffic in a specific time interval
13	TCP Sequence Number	Entropy value of TCP sequence number of the network traffic in a specific time interval
14	TCP Window Size	Entropy value of TCP Window Size of the network traffic in a specific time interval
15	TCP Length	Entropy value of TCP Length of the network traffic in a specific time interval

Table 6.1: List of Entropy-based Features Constructed

6.1.2 Attack detection

Figure. 6.2 shows the second part of E3ML called Attack Detection. The attack traffic is detected through a classification model that is built upon a voting system. The voting system uses a simple majority voting technique, where the classification results of both MLP and RNN needs to be the same for a consensus to be reached. If both MLP and RNN are unable to reach

Table 6.2: Entropy Variation Features generated using Lyapunov Exponent Separation

No	attributes	Definition
1	Separation IP	Exponent separation of source and destination IP Address
2	Separation Port (new)	Exponent separation of source and destination Port numbers
3	Separation MAC (new)	Exponent separation of source and destination MAC Address
4	Separation Network (new)	Exponent separation of source and destination Network Address
5	Separation TCP (new)	Exponent separation of TCP window size and destination TCP Length

a consensus or resulted in a tie, the classification result of a third classifier, ADT will be taken into account for the final decision. The voting system in E3ML is summarised in Algorithm 3.

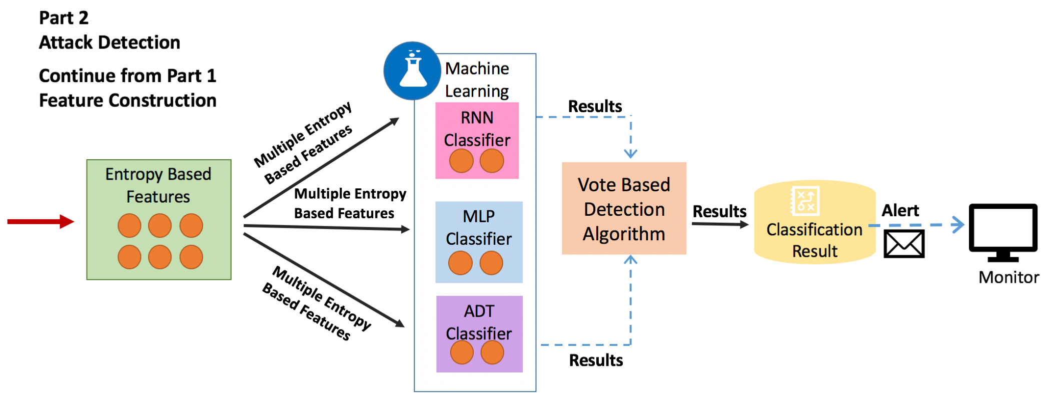


Figure 6.2: Attack Detection

The advantage of this algorithm lies in the use of multiple ML classifiers, where two ML classifiers will need to have the same results or a third ML classifier will used to break the tie. This approach not only increases the accuracy E3ML, but also avoids the reliance on a single ML classifier to detect both high-intensity and low-intensity DDoS attacks. In addition, the use of multiple ML classifiers allows the exploitation and collective use of multiple ML classifiers. Among all the ML classifiers tested (i.e. Naive Bayesian, Linear Regression, SVM, etc), MLP, ADT, and RNN were found to give the best performance. Although these ML classifiers top the

Algorithm 3 E3ML Algorithm

```
1: procedure TRAFFIC CLASSIFICATION
2:     Build training model for each classifier (MLP, RNN, and ADT)
3:     Let $X = \{$a set of entropy features of traffic with respect to a time interval, $T\}$
4:     for every $X$ do
5:         Get classification result ($C$) from MLP and RNN
6:         if $C_{\text{MLP}}$ == $C_{\text{RNN}}$ == attack then
7:             Majority vote == true
8:             Detection result == attack
9:         else if $C_{\text{MLP}}$ == $C_{\text{RNN}}$ == normal then
10:            Majority vote == true
11:            Detection result == normal
12:        else if $C_{\text{MLP}}$ != $C_{\text{RNN}}$ then
13:            Majority vote == not true
14:            Get classification result ($C$) from ADT (arbiter)
15:            if $C_{\text{ADT}}$ == $C_{\text{RNN}} ==$ attack then
16:                Detection result == attack
17:            else if $C_{\text{ADT}}$ == $C_{\text{RNN}} ==$ normal then
18:                Detection result == normal
19:            else if $C_{\text{ADT}}$ != $C_{\text{RNN}}$ then
20:                Detection result == attack
21:            end if
22:        end if
23:    end for
24: end procedure
```

other ML classifiers in distinguishing attack from normal traffic, they still lack performance consistency between different types of DDoS attacks. In particular, RNN can detect attack traffic effectively with high precision on high-intensity attacks but does not perform well with low-intensity attacks. On the other hand, MLP performs better in detecting normal traffic but at the same time has poor overall precision. ADT performs relatively well in both low-intensity attack traffic and high-intensity attack traffic, but its overall precision for both attacks is not as high as RNN.

Multiple ML classifiers can be used to overcome the shortcomings of a single ML classifier. The rationale behind the selection of multiple ML classifiers and a voting system is to take advantage of the strength of each classifier to distinguish attack traffic from normal traffic. Since RNN performed poorly in detecting normal traffic, MLP and ADT can be used to improve decision making. For example, when traffic is considered as attack traffic by RNN and as normal traffic by MLP, ADT will act as a tie breaker by choosing a side based on its classification result. In the E3ML algorithm, the traffic is said to be malicious if ADT and RNN disagrees with each other or ADT and RNN both say that it is attack traffic. Otherwise, the traffic will be considered as normal traffic.

6.2 Performance Evaluation

This section analyses the efficacy of the E3ML DDoS attack detection scheme. E3ML is implemented using JAVA and WEKA Data Mining Software [101] is integrated as the ML library. The evaluation metrics, datasets used for evaluation and also the evaluation results are detailed in the next three subsections.

6.2.1 Evaluation Metrics

The evaluation metrics used in the evaluation of E3ML are *precision*, *recall*, and *F1 score*.

- True Positive (*TP*) - The amount of attack traffic correctly detected as attack traffic.
- False Positive (*FP*) - The amount of normal traffic incorrectly detected as attack traffic.
- False Negative (*FN*) - The amount of attack traffic incorrectly detected as normal traffic.
- Precision - The ratio of correctly detected attack traffic (*TP*) against the total of detected attack traffic (*TP*+ *FP*). The equation is described as:

  $$precision = \frac{TP}{TP + FP} \tag{6.2}$$

  In other words, precision measures the number of detected attacks that were actually attacks. The higher the precision, the better the performance of the classifier.
- Recall - The ratio of correctly detected attack traffic (*TP*) against the total of actual attack traffic (*TP*+ *FN*). The equation is described as:

  $$recall = \frac{TP}{TP + FN} \tag{6.3}$$

  Recall is also known as the sensitivity measure, where it measures the number of actual attacks that are detected as attacks. The higher the recall, the higher the number of attacks the classifier correctly detects.
- F1 score - The harmonic mean of precision and recall. The equation is described as:

  $$F1\text{-}Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{6.4}$$

  The higher the F1-score, the better the performance for a classifier. If the F1-score reaches 1, both precision and recall are perfect.

6.2.2 Datasets

E3ML is evaluated and validated by using a relatively new dataset from the University of New Brunswick 2012 ISCX dataset (ISCX'12) [98] and the commonly used 1998 DARPA Intrusion Detection dataset (DARPA'98) [102]. Although the DARPA'98 is two decades old, this dataset is still being used for the comparison and evaluation of DDoS attack detection and defence approaches in the literature [103, 104].

The ISCX'12 dataset is provided by the University of New Brunswick. This dataset contains seven days of traffic data (Monday - Sunday) and focuses on more recent forms of attacks such as the high-intensity application layer attacks generated using IRC botnets and low-intensity attacks generated using the Slowloris tool. Monday's traffic contains HTTP DoS traffic which is high-intensity and normal traffic data, whereas Tuesday's traffic consists of IRC botnet-based DDoS attack traffic which is low-intensity and normal traffic data. Both Monday's (High-Intensity) and Tuesday's (Low-Intensity) traffic data are used to evaluate E3ML performance on distinguishing high and low intensity DDoS attack traffic from normal traffic.

As for the DARPA'98 dataset provided by MIT Lincoln Lab, it contains more traditional attacks such as SMURF, Neptune, and Land. Each of these attacks uses a different attack intensity in sending its attack traffic hence this dataset is considered to have a mix-intensity attack traffic. This dataset contains seven weeks (Week 1-7) of training data and two weeks (Week 8-9) of testing data, which includes 27 types of network-based attacks that can be summarised into four categories: Denial-of-Service (DoS), Remote to Local User (R2L), User to Superuser (U2S), and Probe. The training data from Week 1 and Week 2 containing mix-intensity attack traffic and normal traffic are used to evaluate E3ML performance on distinguishing mix-intensity attack traffic from normal traffic. Week 1 training data is labelled as Mix-Intensity 1 dataset and Week 2 training data is labelled as Mix-Intensity 2 dataset. Note that E3ML focuses only on detecting DoS attacks (SMURF, Neptune, Pod, Teardrop, and Land attacks) for this dataset. Other attacks are considered as noise and labelled as normal traffic.

6.2.3 Evaluation Results

These datasets are evaluated by randomly selecting 70% of attack traffic and normal traffic from each dataset as suggested by Tan et al. [105] as a way of avoiding inaccurate detection due to the hidden bias in the sequential data. A 10-fold cross-validation and default parameter settings of ADT and MLP in WEKA [101] are used in the performance evaluation of E3ML. Since WEKA does not have an official RNN classifier, the Elman RNN package [106] is adopted into the WEKA library and its default parameter settings are used.

In this evaluation, two sets of entropy-based features are used: (1) 20 common and uncommon entropy-based features (source and destination IP addresses, source and destination port addresses, protocol identifier, delta time, source and destination MAC addresses, source and destination Network addresses, packet length, IP DSCP value, TCP sequence number, TCP Payload, TCP Window length) and (2) 5 commonly used entropy-based features (source and destination IP addresses, source and destination port addresses, and protocol identifier). These features are used as inputs to the machine learning classifiers.

20 entropy-based features are selected based on the study done in Chapter 5. Since not all single entropy-based classifiers can distinguish between both high and low-intensity DDoS attack traffic and normal traffic, all constructed entropy-based features are used to leverage the strength of both common and uncommon entropy-based features for improving detection accuracy. For comparison, 5 commonly used entropy-based features are used. Other combinations of entropy-based features are not tested as the aim of this chapter is to show the effectiveness of using both uncommon

and common entropy-based features. Finding the optimal set of entropy-based features will require additional study which is beyond the scope of this this thesis.

As shown in Figure 6.3, 20 entropy-based features showed better F1-scores as compared to the 5 commonly used entropy-based features. The results were consistent for all four classifiers except for the ADT classifier in the high-intensity dataset. However, this can be ignored since there is only a mere 0.06% difference compared to the other classifiers where the difference is at least 1% in the F1-score between the 20 features and 5 features.

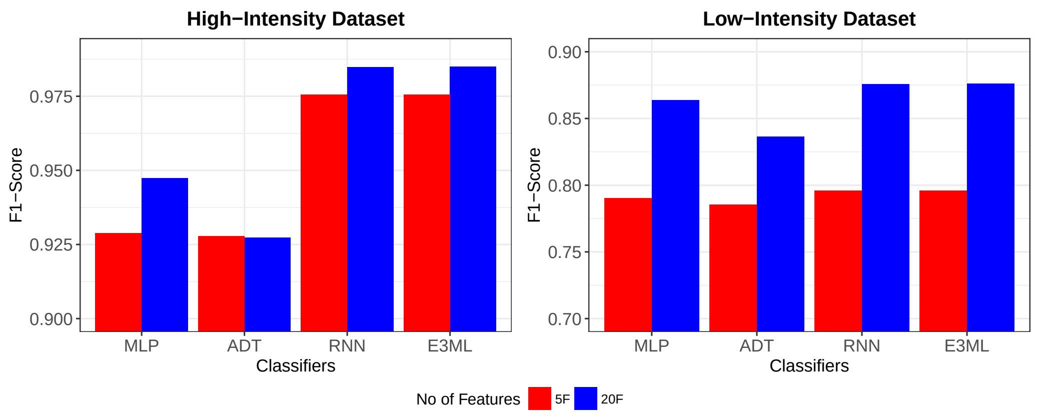


Figure 6.3: Effects of 5 Commonly used Entropy-based Features - 5F (red) vs 20 Entropy-based Features - 20F (blue)

Figure 6.4 shows the precision, recall and F1-score of E3ML and three single ML classifiers in detecting both high and low-intensity DDoS attacks. E3ML consistently achieved the highest or second highest precision and F1-score as compared to ADT, RNN, and MLP on High-Intensity, Mix-Intensity 1 and Mix-Intensity 2 datasets. E3ML consistently achieved the lowest or second lowest in terms of recall on all four datasets. E3ML achieved the highest or second highest in terms of F1 score on Low-Intensity dataset.

Table 6.3 presents the average F1-scores along with standard deviation

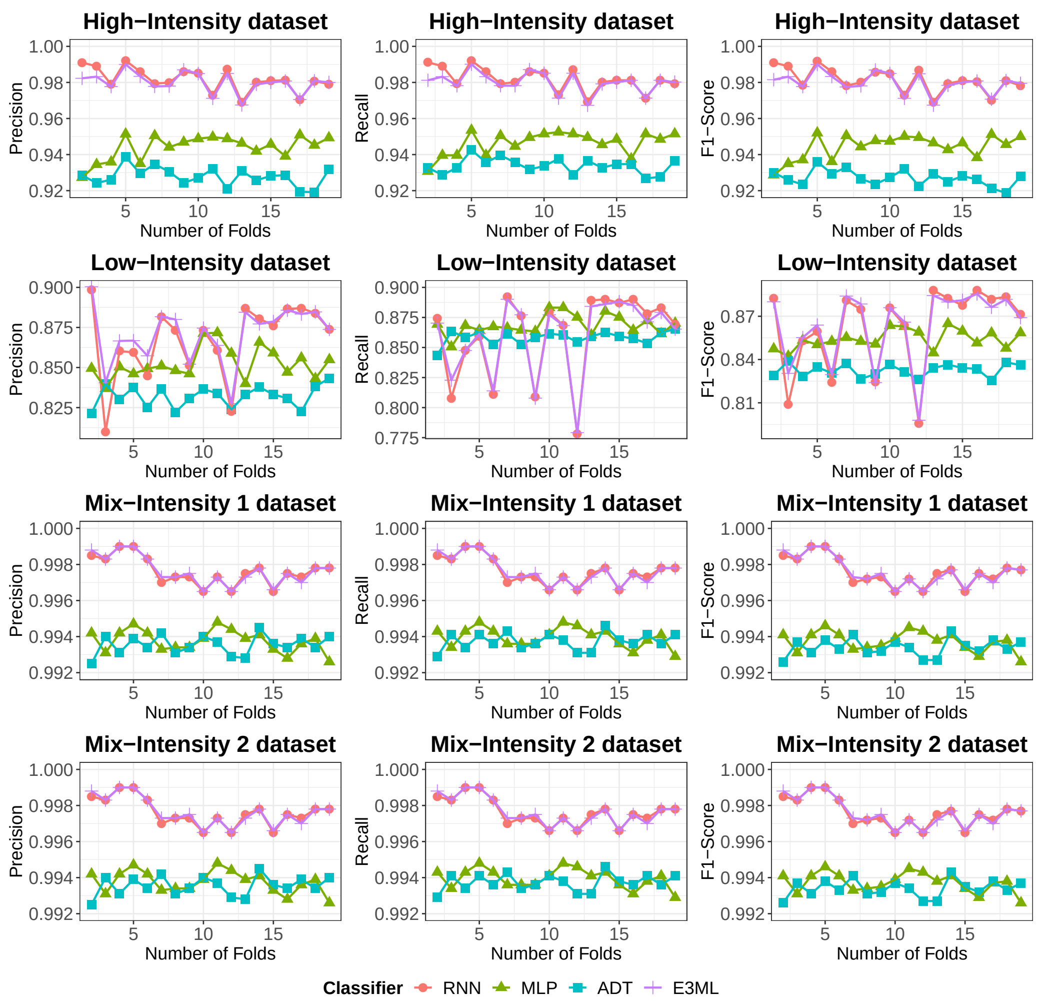


Figure 6.4: Precision, Recall and F1-scores of ADT, MLP, RNN, and E3ML in High, Low and Mix-Intensity DDoS Attack Detection

of E3ML and other single ML classifiers on various DDoS attack intensity datasets. The average F1-scores of each classifier for each dataset is calculated by averaging the F1-scores obtained through x-fold cross-validation, where $x = \{2, \cdots, 19\}$. The results clearly show that E3ML outperforms other single ML classifiers in all four datasets.

Based on the study by Demšar [107], the Wilcoxon signed rank test [108] is a preferred method used to compare two ML classifiers. In this

Table 6.3: The Comparison of F1-Scores between E3ML and other Machine Learning Classifiers on Various Intensity Datasets

Dataset	E3ML	ADT	RNN	MLP
High-intensity	**0.98 ± 0.02**	0.59 ± 0.03	0.90 ± 0.03	0.90 ± 0.04
Low-intensity	**0.86 ± 0.02**	0.34 ± 0.02	0.57 ± 0.07	0.57 ± 0.02
Mix-intensity 1	**1.00 ± 0.02**	0.84 ± 0.01	0.95 ± 0.02	0.95 ± 0.01
Mix-intensity 2	**1.00 ± 0.02**	0.84 ± 0.01	0.95 ± 0.02	0.95 ± 0.01

thesis, the two-tailed Wilcoxon signed rank test is performed to measure the significance of the differences in performance between E3ML and single ML classifiers on various intensity datasets. E3ML is considered to be significantly better than a single ML classifier when the significance level, p-value, is smaller than 0.05 (5% significance level). The test results are shown in Table 6.4, Table 6.5, and Table 6.6.

As shown in Table 6.4, E3ML is significantly better than ADT, MLP, and RNN for all datasets in terms of precision. The results indicate that E3ML is able to detect attack traffic more accurately than the other classifiers tested.

Table 6.4: The Significance Test (p-value $>$ 0.05) on the Precision

Dataset	E3ML vs ADT	E3ML vs RNN	E3ML vs MLP
High-intensity	**0.0002**	**0.0027**	**0.0002**
Low-intensity	**0.0002**	**0.0278**	**0.00288**
Mix-intensity 1	**0.0002**	**0.01878**	**0.0002**
Mix-intensity 2	**0.0002**	**0.01878**	**0.0002**

Table 6.4 shows the significance of the recall between E3ML and other single ML classifiers (i.e. RNN, MLP, and ADT). E3ML is evidently better than all three classifiers in terms of recall on three out of the four datasets tested. The recall of the low-intensity dataset is shown to be less significant, which indicates that E3ML performance is similar to the performance

of all three classifiers in terms of recall.

Table 6.5: The Significance Test (p-value > 0.05) on the Recall

Dataset	E3ML vs ADT	E3ML vs RNN	E3ML vs MLP
High-intensity	**0.0002**	**0.0042**	**0.0002**
Low-intensity	0.61708	0.1141	0.48392
Mix-intensity 1	**0.0002**	**0.01878**	**0.0002**
Mix-intensity 2	**0.0002**	**0.01878**	**0.0002**

Similar to the significance test shown in Table 6.5, E3ML performance in terms of F1-score is significantly better than ADT, MLP and RNN for high-intensity, mix-intensity 1 and mix-intensity 2 datasets as shown in Table 6.6. However, for the low-intensity dataset, E3ML is only significantly better than ADT but not for RNN and MLP.

Table 6.6: The Significance Test (p-value > 0.05) on the F1-Score

Dataset	E3ML vs ADT	E3ML vs RNN	E3ML vs MLP
High-intensity	**0.0002**	**0.0139**	**0.0002**
Low-intensity	**0.0012**	0.47152	0.0703
Mix-intensity 1	**0.0002**	**0.01878**	**0.0002**
Mix-intensity 2	**0.0002**	**0.01878**	**0.0002**

Based of the Wilcoxon Signed Rank test, it can be concluded that E3ML is significantly better than all three classifiers in all datasets except for low-intensity datasets where the significance level of E3ML is not prominent.

E3ML is also compared with existing entropy-based approaches and other recent approaches as shown in Table 6.7. The Exponent Separation Detection Algorithm (ESDA) proposed by Ma et al. [21] is implemented for comparison and investigates the performance of the approach with different threshold values, T_k, including the one used in their paper, $T_k = 0.1$ in the case of a positive effect on the detection accuracy. In addition, E3ML is also compared to the reported results of another recent approach by Tan et al. [105]. These approaches are evaluated using the High-Intensity dataset (IRC-Botnet attack traffic) and 10-fold cross validation.

Table 6.7: Comparison of Detection Performance on ISCX 2012 Dataset

Approaches	Non Entropy-based	Entropy-based approach				
	[105] EMD-L_i	ESDA [21]				Our Approach
		Threshold (T)				
		0.1	-0.1	-0.5	-0.01	
TPR	90.04%	4.93%	46.48%	43.66%	42.96%	**94.74%**

The evaluation results showed that the E3ML approach outperformed the entropy-based methods and other state-of-the-art approaches such as computer vision and evolutionary computation based methods. E3ML achieved 4.74% higher precision than EMD-L_i (computer vision technique based on Earth Mover's Distance). E3ML has also proven to be more accurate than the chaos analysis technique in [21] by more than double its accuracy rate on all four threshold values tested.

6.3 Summary

This chapter proposed a DDoS detection scheme based on multiple entropy-based features and machine learning classifiers to improve detection accuracy and generality. The technique, Entropy-based Three Machine Learning Classifiers (E3ML) consists of feature construction to generate two types of entropy-based features (i.e., *regular entropy-based features* and *entropy variation features*) and attack detection to classify network traffic into attack or normal using our detection algorithm. E3ML also contains a voting system to compare classification results of two different machine learning classifiers, MLP and RNN and later compare with ADT when a tie happens between MLP and RNN.

Results from performance evaluations showed that E3ML could detect DDoS attacks with different intensities effectively across datasets. Even though E3ML is only slightly more accurate than RNN, E3ML shows potential in differentiating attack traffic from normal traffic consistently across different datasets with high accuracy and low misdetection. E3ML outperformed several recent approaches (EMD-L_i and ESDA) and showed that it could produce consistently high accuracy results with datasets containing different kinds of DDoS attacks.

This chapter found that a combination of multiple entropy-based features performed better than single entropy-based features. In addition, the evaluation results showed that machine learning acts as a better threshold selector or classifier in determining whether traffic is an attack traffic or a normal traffic than a fixed threshold. The technique, developed based on multiple entropy-based features and machine learning techniques, provides a promising direction for DDoS detection.

Chapter 7

Contributions and Future Work

The main research question of this thesis as described in Chapter 1 was ***"How can we detect high and low-intensity DDoS attacks accurately and early?"***. In answering this question, this thesis proposed three useful components for developing an accurate and early high and low-intensity DDoS attack detection system.

The first section of this chapter presents the main contributions to the field of DDoS attack detection. Section 7.2 outlines the possible future work for DDoS attack detection. This thesis concludes by providing a summary of this research.

7.1 Contributions

7.1.1 Design for a Scalable and Fault-Tolerant DDoS Detection System for Early Detection (SCAFE)

SCAFE is designed to detect DDoS attacks early, where attack traffic is detected at the edge of the network, closer to the attack source. The SCAFE architecture is scalable and fault tolerant so as to ensure that DDoS attack detection could operate smoothly even with large amount of traffic or faults occurring in the large network. The findings discovered during the construction of SCAFE are described in the following:

(a) A decentralised approach allows linear scalability in the traffic information collection process This thesis finds that by decentralising the traffic collection mechanism with lightweight local monitors and databases, an increase in the amount of collected traffic will increase the message complexity linearly. SCAFE uses a decentralised traffic collection approach through the GAUGE monitoring mechanism installed in each monitor, where these monitors collect the traffic statistics information of every single link in its respective subnetwork. A decentralised approach also allows monitors and databases to be added to the network for a more scalable monitoring when the size of a network increases.

(b) The network-wide correlation analysis approach using information gathered from multiple locations improves the early detection capability of a DDoS attack detection system

This thesis finds that network-wide correlation analysis improves the early detection capability by detecting the DDoS attack traffic at the edge of the network, where the edge of the network is the closest possible point to the source of the attack that is connected to the DDoS attack detection system. SCAFE employs SDN technology in its traffic monitoring mechanism to provide a comprehensive view of the network, where traffic can be monitored at multiple locations. The early detection is achieved by correlating information collected from multiple points since information at a single point is usually not prominent enough to indicate the presence of attack traffic.

(c) Using a separate network for communication between detection system components avoids additional overhead to the data network

The separate network is made possible by leveraging SDN technology, where the technology allows the separation of control plane and data plane in the network. The SCAFE components are connected via a control network for communication (e.g. data requests and retrievals) between components. The dedicated network is separated from the main data links to avoid adding additional overhead to the network that might unintentionally constitute an attack to the network.

7.1.2 Providing a Set of Useful Entropy-based Features

Section 3.1 has identified a list of useful entropy-based features for distinguishing DDoS attack traffic from normal traffic. The entropy-based features consist of common, uncommon, and entropy variation features that were constructed from packet header information in the network within a particular time interval. The findings are summarised in the following.

(a) Not all regular entropy-based features provide clear distinction between attack and normal traffic patterns

A total of 15 regular entropy-based features, which consisted of 5 common and 10 uncommon features in DDoS detection were evaluated individually for their effectiveness in distinguishing high and low intensity attack traffic from normal traffic. Not all five common features, namely source IP address, destination IP address, source port number, destination port number, and protocol identifier were able to provide a relatively clear distinction between attack traffic and normal traffic for both high and low-intensity DDoS attacks. In particular, only source port address and destination port address showed relatively clear distinction between high-intensity attack and normal traffic patterns, whereas destination IP address, source port address and protocol identifier provided similar patterns in both attack and normal traffic for low-intensity DDoS attack traffic. Ten less common features, namely delta time, source MAC address, destination MAC address, source network address, destination network address, packet length, IP DSCP value, TCP sequence number, TCP Window size and TCP length were evaluated. Only packet length was able to provide some distinction between high-intensity attack and normal traffic, whereas window length and sequence were able to provide some distinction between low-intensity attack and normal traffic.

(b) Five useful entropy-based features from entropy variation of two distinct entropy-based features

The entropy variation features are constructed from the Lyapunov separation of two distinct features. The five features found to be useful, where the entropy variation features derived from the Lyapunov separation of two distinct features, are:

– IP address separation - The separation between source IP address and destination IP address

– Port number separation - The separation between source port number and destination port number)

– MAC address separation - The separation between source MAC address and destination MAC address

– Network address separation - The separation between source network address and destination network address)

– TCP separation - The separation between TCP window size and TCP window length

(c) Entropy Measures such as Shannon, Tsallis, and Zhou show clearer DDoS attack traffic patterns than Rényi entropy

Based on our findings, Shannon, Tsallis and Zhou entropy measures show clearer DDoS attack traffic patterns than Rényi entropy. The only difference found in the first three entropy measures is the gap difference between attack and normal traffic entropy values. Shannon entropy provides a bigger gap between the normal traffic entropy values and attack traffic entropy values, which indicates a more distinctive difference between attack and normal traffic.

(d) Window size section used in entropy construction has minimal impact on overall accuracy

The time interval window size used in constructing entropy-based features has minimal effect on the accuracy of DDoS attack detection. Hypothetically, window size can influence the traffic patterns if a high concentration of attack traffic is being captured in a single window. However, DDoS attack traffic concentration varies depending on the aggregation level and also the type of DDoS attack being launched. In the evaluation shown in Section 3.1.2, there is no clear difference between the traffic patterns of different window sizes in revealing DDoS attack traffic.

7.1.3 Multiple Entropy-based Features and Multiple Machine Learning Classifiers Increase Accuracy

The E3ML DDoS detection scheme proposed in Chapter 3.2 focuses on improving the accuracy of DDoS detection through effective traffic classification. Entropy is used to construct traffic features, where these features are then used as the inputs to the machine learning classification model. This model integrates three machine learning classification models, namely Alternating Decision Tree (ADT), Recursive Neural Network (RNN) and Multilayer Perceptron (MLP), which forms a voting system to determine the most accurate result for detecting attack traffic. E3ML is evaluated using low-intensity, high-intensity and mix-intensity datasets and is found to generate consistent results as being the best or second best classifier in four different types of datasets containing high, low and mix-intensities DDoS attack traffic.

(a) A clear potential in the use of uncommon entropy-based features in DDoS attack detection

The E3ML DDoS attack detection scheme demonstrated that uncommonly used entropy-based traffic features in DDoS attack detection (i.e. delta time, packet length, TCP Window size and TCP length) and entropy variation features (i.e. separation IP, separation port, separation MAC, separation network and separation TCP) with multiple machine learning classifiers (ADT, MLP and RNN) can reveal attack traffic patterns more distinctively even with low-intensity DDoS attack. The evaluation results shown in Section 6.2.3 have clearly indicated that using 20 entropy-based features (both common and uncommon features) have higher detection accuracy that using five commonly used entropy-based features.

(b) A single machine learning classifier is unable to accurately detect both high and low-intensity DDoS attack traffic

A single machine learning classifier is shown in Section 6.2.3 to be less precise that E3ML which uses multiple machine learning classifiers in classifying network traffic.

7.2 Future Work

This section lists the potential directions for future work related to the research in this thesis.

(a) Extending E3ML and SCAFE approaches to IOT networks The datasets used in this thesis do not contain traffic from IOT devices. Since IOT devices are becoming a popular target for attackers to exploit for DDoS attacks, it is a good direction to extended approaches to be able to distinguish between IOT attack traffic from normal traffic.

(b) Distributed correlation analysis to further improve scalability Level 2 in the SCAFE DDoS attack detection architecture, which is used for detecting attack flows in network links, is currently centralised. Distributed methods can be applied to this detection level to achieve full scalability.

(c) Feature selection methods for optimization to improve the multiple classifier method The E3ML approach uses a set of entropy-based features derived from raw traffic features and the variation of two distinct entropy-based features. Feature selection methods can be applied to optimize the use of entropy-based features and further improve detection accuracy.

(d) Exploring other traffic features for network wide correlation to further improve accuracy This thesis uses byte count, flow duration, packet count and flow length features from traffic flows for correlation. Exploring other features in the flow tables that might be useful can potentially improve the detection accuracy of a DDoS attack detection system.

7.3 Summary

In summary, this thesis has contributed to the field of DDoS attack detection by proposing three useful components: (1) a good architecture (SCAFE), (2) a set of useful features (regular entropy and entropy variation features), and (3) a generalised traffic classification technique (E3ML). These proposed components can help researchers gain better insights in the use of entropy, machine learning, and SDN techniques to further improve the detection of high and low-intensity DDoS attacks accurately and early.

Bibliography

[1] Shui Yu. An Overview of DDoS Attacks. In *Distributed Denial of Service Attack and Defense*, pages 1–14. Springer, 2014.

[2] Chenfeng Vincent Zhou, Christopher Leckie, and Shanika Karunasekera. A Survey of Coordinated Attacks and Collaborative Intrusion Detection. *Computers & Security*, 29(1):124–140, 2010.

[3] Michele De Donno, Nicola Dragoni, Alberto Giaretta, and Angelo Spognardi. Analysis of DDoS-Capable IoT Malwares. In *Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 807–816. IEEE, 2017.

[4] Yuchen Yang, Longfei Wu, Guisheng Yin, Lijie Li, and Hongbin Zhao. A Survey on Security and Privacy Issues in Internet-of-Things. *IEEE Internet of Things Journal*, 4(5):1250–1258, 2017.

[5] Jose Nazario. Ddos Attack Evolution. *Network Security*, 2008(7):7–10, 2008.

[6] Michael Lesk. The New Front Line: Estonia under Cyberassault. *IEEE Security & Privacy*, 5(4), 2007.

[7] Stephen W Korns and Joshua E Kastenberg. Georgia's Cyber Left Hook. *Parameters*, 38(4):60, 2008.

[8] Lucian Constantin. DDoS Attack against Spamhaus was Reportedly the Largest in History. http://features.techworld.com/security/3437607/ddos-attack-against-spamhaus-was-reportedly-the-largest-in-history/, 2013.

[9] Dean Wilson. New DDoS Attack Breaks Spamhaus Records. http://www.techradar.com/news/internet/web/new-ddos-attack-breaks-spamhaus-records-1223956, Feb 2014.

[10] C Williams. Today the Web was Broken by Countless Hacked DevicesYour 60-Second Summary. *The Register*, 21, 2016.

[11] Sanjeev Kumar. Smurf-based Distributed Denial of Service (DDoS) Attack Amplification in Internet. In *Proceedings of the Second International Conference on Internet Monitoring and Protection (ICIMP)*, pages 25–25. IEEE, 2007.

[12] Min Suk Kang, Soo Bum Lee, and Virgil D Gligor. The Crossfire Attack. In *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, pages 127–141. IEEE, 2013.

[13] Ahren Studer and Adrian Perrig. The Coremelt Attack. In *European Symposium on Research in Computer Security*, pages 37–52. Springer, 2009.

[14] Saman Taghavi Zargar, James Joshi, and David Tipper. A Survey of Defense Mechanisms against Distributed Denial of Service (DDoS) Flooding Attacks. *IEEE Communications Surveys & Tutorials*, 15(4):2046–2069, 2013.

[15] Vyas Sekar, Nick G Duffield, Oliver Spatscheck, Jacobus E van der Merwe, and Hui Zhang. LADS: Large-Scale Automated DDoS Detection System. In *USENIX Annual Technical Conference, General Track*, pages 171–184, 2006.

[16] Jelena Mirkovic and Peter Reiher. D-WARD: A Source-End Defense against Flooding Denial-of-Service Attacks. *IEEE Transactions on Dependable and Secure Computing*, 2(3):216–232, 2005.

[17] Roshan Thomas, Brian Mark, Tommy Johnson, and James Croall. NetBouncer: Client-Legitimacy-based High-Performance DDoS Filtering. In *Proceedings of the 2003 DARPA Information Survivability Conference and Exposition*, volume 1, pages 14–25. IEEE, 2003.

[18] Haiqin Liu and Min Sik Kim. Real-time Detection of Stealthy DDoS Attacks using Time-series Decomposition. In *IEEE International Conference onCommunications (ICC), 2010*, pages 1–6. IEEE, 2010.

[19] Yang Xiang, Ke Li, and Wanlei Zhou. Low-rate DDoS Attacks Detection and Traceback by using New Information Metrics. *IEEE Transactions on Information Forensics and Security*, 6(2):426–437, 2011.

[20] Changwang Zhang, Zhiping Cai, Weifeng Chen, Xiapu Luo, and Jianping Yin. Flow Level Detection and Filtering of Low-Rate DDoS. *Computer Networks*, 56(15):3417–3431, 2012.

[21] Xinlei Ma and Yonghong Chen. DDoS Detection Method based on Chaos Analysis of Network Traffic Entropy. *IEEE Communications Letters*, 18(1):114–117, 2014.

[22] Monowar H Bhuyan, DK Bhattacharyya, and JK Kalita. E-ldat: A Lightweight System for DDoS Flooding Attack Detection and IP Traceback using Extended Entropy Metric. *Security and Communication Networks*, 9(16):3251–3270, 2016.

[23] Yu Gu, Andrew McCallum, and Don Towsley. Detecting Anomalies in Network Traffic using Maximum Entropy Estimation. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, pages 32–32. USENIX Association, 2005.

[24] Jie Zhang, Zheng Qin, Lu Ou, Pei Jiang, JianRong Liu, and AX Liu. An Advanced Entropy-based DDoS Detection Scheme. In *Proceedings of the International Conference on Information Networking and Automation (ICINA)*, volume 2, pages V2–67. IEEE, 2010.

[25] Seyed Mohammad Mousavi and Marc St-Hilaire. Early Detection of DDoS Attacks against SDN Controllers. In *Proceedings of the International Conference on Computing, Networking and Communications (ICNC)*, pages 77–81. IEEE, 2015.

[26] Peter Phaal, Sonia Panchen, and Neil McKee. InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks. Technical report, 2001.

[27] Benoit Claise. Cisco Systems Netflow Services Export Version 9. Technical report, 2004.

[28] Jérôme François, Issam Aib, and Raouf Boutaba. FireCol: A Collaborative Protection Network for the Detection of Flooding DDoS attacks. *IEEE/ACM Transactions on Networking (TON)*, 20(6):1828–1841, 2012.

[29] George Nychis, Vyas Sekar, David G Andersen, Hyong Kim, and Hui Zhang. An Empirical Evaluation of Entropy-based Traffic Anomaly Detection. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*, pages 151–156. ACM, 2008.

[30] Zhiyuan Tan, Aruna Jamdagni, Xiangjian He, Priyadarsi Nanda, and Ren Ping Liu. A System for Denial-of-Service Attack Detection based on Multivariate Correlation Analysis. *IEEE Transactions on Parallel and Distributed Systems*, 25(2):447–456, 2014.

[31] Jun Zhang, Yang Xiang, Yu Wang, Wanlei Zhou, Yong Xiang, and Yong Guan. Network Traffic Classification using Correlation In-

formation. *IEEE Transactions on Parallel and Distributed Systems*, 24(1):104–117, 2013.

[32] Riyanat Shittu, Alex Healing, Robert Ghanea-Hercock, Robin Bloomfield, and Muttukrishnan Rajarajan. Intrusion Alert Prioritisation and Attack Detection using Post-Correlation Analysis. *Computers & Security*, 50:1–15, 2015.

[33] Guofei Gu, Junjie Zhang, and Wenke Lee. BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS)*, volume 8, pages 1–18, 2008.

[34] Claude E Shannon. Communication Theory of Secrecy Systems. *Bell Labs Technical Journal*, 28(4):656–715, 1949.

[35] Constantino Tsallis. Possible Generalization of Boltzmann-Gibbs Statistics. *Journal of Statistical Physics*, 52(1-2):479–487, 1988.

[36] Alfréd Rényi. On Measures of Entropy and Information. Technical report, HUNGARIAN ACADEMY OF SCIENCES Budapest Hungary, 1961.

[37] Alefiya Hussain, John Heidemann, and Christos Papadopoulos. A Framework for Classifying Denial of Service Attacks. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 99–110. ACM, 2003.

[38] Vern Paxson. An Analysis of using Reflectors for Distributed Denial-of-Service Attacks. *ACM SIGCOMM Computer Communication Review*, 31(3):38–47, 2001.

[39] Stephen M Specht and Ruby B Lee. Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures. In *ISCA PDCS*, pages 543–550, 2004.

[40] Moheeb Abu Rajab, Jay Zafross, Fabian Monrose, and Andreas 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*, 2007.

[41] Bill McCarty. Botnets: Big and Bigger. *IEEE Security & Privacy*, 99(4):87–90, 2003.

[42] Lee Garber. Denial-of-Service Attacks Rip the Internet. *IEEE Computer*, 33(4):12–17, 2000.

[43] Steve Mansfield-Devine. The Growth and Evolution of DDoS. *Network Security*, 2015(10):13–20, 2015.

[44] Gkberk Yaltirakli. Slowloris. https://github.com/gkbrk/slowloris, 2015.

[45] Evan Cooke, Farnam Jahanian, and Danny McPherson. The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. *SRUTI*, 5:6–6, 2005.

[46] Chao Li, Wei Jiang, and Xin Zou. Botnet: Survey and Case Study. In *Proceedings of the Fourth Innovative Computing, Information and Control (ICICIC)*, pages 1184–1187. IEEE, 2009.

[47] Julian B Grizzard, Vikram Sharma, Chris Nunnery, Brent ByungHoon Kang, and David Dagon. Peer-to-Peer Botnets: Overview and Case Study. *HotBots*, 7:1–1, 2007.

[48] Jasek Roman, Benda Radek, Vala Radek, and Sarga Libor. Launching Distributed Denial of Service Attacks by Network Protocol Exploitation. In *Proceedings of the 2nd International Conference on Applied Informatics and Computing Theory, ser. AICT*, volume 11, pages 210–216, 2011.

[49] Paul J Criscuolo. Distributed Denial of Service: Trin00, Tribe Flood Network, Tribe Flood Network 2000, and Stacheldraht ciac-2319. Technical report, DTIC Document, 2000.

[50] David Dittrich, George Weaver, Sven Dietrich, and Neil Long. The Mstream Distributed Denial of Service Attack Tool. http://staff. washington. edu/dittrich/misc/mstream. analysis. txt, 2000.

[51] Wesley Eddy. TCP SYN Flooding Attacks and Common Mitigations. Technical report, 2007.

[52] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Detecting Distributed Denial of Service Attacks by Sharing Distributed Beliefs. In *Australasian Conference on Information Security and Privacy*, pages 214–225. Springer, 2003.

[53] Google Ideas Arbor Network Inc. Digital Attack Map. http://www.digitalattackmap.com/, 2013.

[54] Matthew Prince. The DDoS that Almost Broke the Internet. *CloudFlare blog, March*, 27:2013, 2013.

[55] Radware Ltd. ProtonMail DDoS Case Study: DDoS Prevention Techniques. https://security.radware.com/ddos-experts-insider/ert-case-studies/protonmail-overcomers-sophisticated-ddos-ransom-attack/, January 2016.

[56] James Scott Sr and Winter Summit. Rise of the Machines: The Dyn Attack Was Just a Practice Run December 2016. 2016.

[57] Anukool Lakhina, Mark Crovella, and Christophe Diot. *Mining Anomalies using Traffic Feature Distributions*, volume 35. 2005.

[58] İlker Özçelik and Richard R Brooks. Deceiving Entropy based DoS Detection. *Computers & Security*, 48:234–245, 2015.

[59] Tom M Mitchell. Machine Learning. *Burr Ridge, IL: McGraw Hill*, 45:995, 1997.

[60] Jeremy Frank. Artificial iIntelligence and Intrusion Detection: Current and Future Directions. In *Proceedings of the 17th National Computer Security Conference*, volume 10, pages 1–12. Baltimore, USA, 1994.

[61] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do We Need Hundreds of Classifiers to Solve Real World Classification Problems? *The Journal of Machine Learning Research*, 15(1):3133–3181, 2014.

[62] Jin Li, Yong Liu, and Lin Gu. DDoS Attack Detection based on Neural Network. In *Aware Computing (ISAC), 2010 2nd International Symposium on*, pages 196–199. IEEE, 2010.

[63] Yuesheng Gu, Yongchang Shi, and Jianping Wang. Efficient Intrusion Detection based on Multiple Neural Network Classifiers with Improved Genetic Algorithm. *Journal of Software*, 7(7):1641–1648, 2012.

[64] K Giotis, Christos Argyropoulos, Georgios Androulidakis, Dimitrios Kalogeras, and Vasilis Maglaris. Combining OpenFlow and sFlow for an Effective and Scalable Anomaly Detection and Mitigation Mechanism on SDN Environments. *Computer Networks*, 62:122–136, 2014.

[65] Mark JL Orr. Introduction to Radial Basis Function Networks. 1996.

[66] Yoav Freund and Llew Mason. The Alternating Decision Tree Learning Algorithm. In *In Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*, volume 99, pages 124–133, 1999.

[67] J Ross Quinlan. *C4. 5: Programs for Machine Learning*. Elsevier, 2014.

[68] David E Taylor. Survey and Taxonomy of Packet Classification Techniques. *ACM Computing Surveys (CSUR)*, 37(3):238–275, 2005.

[69] Weirong Jiang and Viktor K Prasanna. Scalable Packet Classification on FPGA. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(9):1668–1680, 2012.

[70] Anthony D Joseph, Pavel Laskov, Fabio Roli, J Doug Tygar, and Blaine Nelson. Machine Learning Methods for Computer Security (Dagstuhl Perspectives Workshop 12371). *Dagstuhl Manifestos*, 3(1), 2013.

[71] Alan Saied, Richard E Overill, and Tomasz Radzik. Artificial Neural Networks in the Detection of Known and Unknown DDoS Attacks: Proof-of-Concept. In *Highlights of Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection*, pages 309–320. Springer, 2014.

[72] Shi-Jinn Horng, Ming-Yang Su, Yuan-Hsin Chen, Tzong-Wann Kao, Rong-Jian Chen, Jui-Lin Lai, and Citra Dwi Perkasa. A Novel Intrusion Detection System based on Hierarchical Clustering and Support Vector Machines. *Expert systems with Applications*, 38(1):306–313, 2011.

[73] A Ramamoorthi, T Subbulakshmi, and S Mercy Shalinie. Real Time Detection and Classification of DDoS Attacks using Enhanced SVM with String Kernels. In *Proceedings of the International Conference on Recent Trends in Information Technology (ICRTIT)*, pages 91–96. IEEE, 2011.

[74] Yi-Chi Wu, Huei-Ru Tseng, Wuu Yang, and Rong-Hong Jan. DDoS Detection and Traceback with Decision Tree and Grey Relational Analysis. *International Journal of Ad Hoc and Ubiquitous Computing*, 7(2):121–136, 2011.

[75] Shui Yu, Wanlei Zhou, Weijia Jia, Song Guo, Yong Xiang, and Feilong Tang. Discriminating DDoS attacks from Flash Crowds using Flow Correlation Coefficient. *IEEE Transactions on Parallel and Distributed Systems*, 23(6):1073–1080, 2012.

[76] Wei Wei, Feng Chen, Yingjie Xia, and Guang Jin. A Rank Correlation based Detection against Distributed Reflection DoS Attacks. *IEEE Communications Letters*, 17(1):173–175, 2013.

[77] Lei Xue, Xiapu Luo, Edmond WW Chan, and Xian Zhan. Towards Detecting Target Link Flooding Attack. In *LISA*, pages 81–96, 2014.

[78] Peng Xiao, Wenyu Qu, Heng Qi, and Zhiyang Li. Detecting DDoS Attacks against Data Center with Correlation Analysis. *Computer Communications*, 67:66–74, 2015.

[79] Christos Douligeris and Aikaterini Mitrokotsa. DDoS Attacks and Defense Mechanisms: Classification and State-of-the-Art. *Computer Networks*, 44(5):643–666, 2004.

[80] Jelena Mirkovic and Peter Reiher. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53, 2004.

[81] Zong-Lin Li, Guang-Min Hu, and Dan Yang. Global Abnormal Correlation Analysis for DDoS Attack Detection. In *IEEE Symposium on Computers and Communications (ISCC)*, pages 310–315. IEEE, 2008.

[82] Peng Ning, Dingbang Xu, Christopher G Healey, and Robert St Amant. Building Attack Scenarios through Integration of Complementary Alert Correlation Method. In *Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS)*, volume 4, pages 97–111, 2004.

[83] Yu Chen, Kai Hwang, and Wei-Shinn Ku. Collaborative Detection of DDoS Attacks over Multiple Network Domains. *IEEE Transactions on Parallel and Distributed Systems*, 18(12):1649–1662, 2007.

[84] Keith Kirkpatrick. Software-Defined Networking. *Communications of the ACM*, 56(9):16–19, 2013.

[85] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[86] Tommy Chin, Xenia Mountrouidou, Xiangyang Li, and Kaiqi Xiong. An SDN-Supported Collaborative Approach for DDoS Flooding Detection and Containment. In *Military Communications Conference, MILCOM*, pages 659–664. IEEE, 2015.

[87] Ying-Dar Lin, Po-Ching Lin, Chih-Hung Yeh, Yao-Chun Wang, and Yuan-Cheng Lai. An Extended SDN Architecture for Network Function Virtualization with a Case Study on Intrusion Prevention. *IEEE Network*, 29(3):48–53, 2015.

[88] Bing Wang, Yao Zheng, Wenjing Lou, and Y Thomas Hou. DDoS Attack Protection in the Era of Cloud Computing and Software-Defined Networking. *Computer Networks*, 81:308–319, 2015.

[89] Jing Zheng, Qi Li, Guofei Gu, Jiahao Cao, David KY Yau, and Jianping Wu. Realtime DDoS Defense using COTS SDN Switches via Adaptive Correlation Analysis. *IEEE Transactions on Information Forensics and Security*, 13(7):1838–1853, 2018.

[90] Ali Ghiasi, Rich Baca, Ghiasi Quantum, and LLC Commscope. Overview of Largest Data Centers. In *Proceedings of the 802.3 bs Task Force Interim meeting*, 2014.

[91] Simon Knight, Hung X Nguyen, Nick Falkner, Rhys Bowden, and Matthew Roughan. The Internet Topology Zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.

[92] Martin McKeay et al. The Q4 2016 State of the Internet/Security Report, 2017.

[93] AM Batishchev. LOIC (Low Orbit Ion Cannon). http://sourceforge.net/projects/loic, 2004.

[94] Ahmed Patel, Mona Taghavi, Kaveh Bakhtiyari, and Joaquim Celestino JúNior. An Intrusion Detection and Prevention System in Cloud Computing: A Systematic Review. *Journal of Network and Computer Applications*, 36(1):25–41, 2013.

[95] Joseph Lee Rodgers and W Alan Nicewander. Thirteen Ways to Look at the Correlation Coefficient. *The American Statistician*, 42(1):59–66, 1988.

[96] Joel Sommers, Hyungsuk Kim, and Paul Barford. Harpoon: A Flow-Level Traffic Generator for Router and Network Tests. In *ACM SIGMETRICS Performance Evaluation Review*, volume 32, pages 392–392. ACM, 2004.

[97] Mark E Crovella and Azer Bestavros. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. *ACM SIGMETRICS Performance Evaluation Review*, 24(1):160–169, 1996.

[98] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A Ghorbani. Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection. *Computers & Security*, 31(3):357–374, 2012.

[99] Abigail Koay, Aaron Chen, Ian Welch, and Winston KG Seah. A New Multi Classifier System using Entropy-based Features in DDoS

Attack Detection. In *Proceedings of the 2018 International Conference of Information Networking (ICOIN)*, pages 162–167. IEEE, 2018.

[100] Thuy TT Nguyen and Grenville Armitage. A Survey of Techniques for Internet Traffic Classification using Machine Learning. *IEEE Communications Surveys & Tutorials*, 10(4):56–76, 2008.

[101] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA Data Mining Software: an Update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[102] MIT Lincoln Lab. 2000 DARPA Intrusion Detection Scenario Specific Datasets. https://www.ll.mit.edu/r-d/datasets/2000-darpa-intrusion-detection-scenario-specific-datasets.

[103] Jakub Breier and Jana Branišová. Anomaly Detection from Log Files using Data Mining Techniques. In *Information Science and Applications*, pages 449–457. Springer, 2015.

[104] Sahil Garg and Shalini Batra. A Novel Ensembled Technique for Anomaly Detection. *International Journal of Communication Systems*, 30(11), 2017.

[105] Zhiyuan Tan, Aruna Jamdagni, Xiangjian He, Priyadarsi Nanda, Ren Ping Liu, and Jiankun Hu. Detection of Denial-of-Service Attacks based on Computer Vision Techniques. *IEEE Transactions on Computers*, 64(9):2519–2533, 2015.

[106] John Salatas. Implementation of Elman Recurrent Neural Network in WEKA. https://jsalatas.ictpro.gr/implementation-of-elman-recurrent-neural-network-in-weka/, 2010.

[107] Janez Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7(Jan):1–30, 2006.

[108] Frank Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83, 1945.