Detecting Distributed Denial-of-Service Attacks and Flash Events

by

Sajal Bhatia

Bachelor of Technology (Communication & Computer Engineering)
(The LNM Institute of Information Technology, India) – 2008

Thesis submitted in accordance with the regulations for the Degree of Doctor of Philosophy

Institute for Future Environments
Science and Engineering Faculty
Queensland University of Technology

August 2013

Keywords

Denial-of-Service (DoS), Distributed Denial-of-Service (DDoS), Flash Events, Synthetic Traffic Generation

Abstract

The dependence of society on Information and Communication Technology (ICT) over the past decade has brought with it an increased vulnerability to Distributed Denial-of-Service (DDoS) attacks. These attacks harness the power of thousands, and sometimes tens or hundreds of thousands of compromised computers to attack information-providing web-services and online trading sites, resulting in significant down-time and financial losses. Consequently, the study of DDoS attacks, and the development of techniques to accurately and reliably detect and mitigate their impact is an important area of research. One particular challenge in detecting such attacks is distinguishing them from similar looking Flash Events (FEs), which occur when a server experiences an unexpected surge of requests from its legitimate clients. Distinguishing DDoS attacks from FEs is important because each requires a different set of actions to be undertaken by a network administrator. However, developing and investigating realistic techniques to distinguish between the two is complicated by an extreme lack of experimental datasets that record representative real traffic, whether attack or benign.

The work presented in this thesis addresses the above challenges and makes a number of related contributions.

The *first* contribution of the thesis is the design of a DDoS attack detection technique based on the change in the rate of previously unseen source IP addresses (IPs) of the incoming packets, and its proof-of-concept implementation. Using this simple feature of source IPs provides good scalability at higher network speeds.

The *second* contribution is the design and implementation of a model that provides the foundation for both detecting and synthesising Flash Events. The detection part is based on the subtle differences between DDoS attacks and FEs. Analysing two widely used and publicly available datasets, representing a DDoS attack and an FE, three parametric differences between DDoS attacks and FEs

are identified: change in the rate of incoming traffic, change in the rate of new source IP addresses, and the distribution of requests among source IPs. With a view to synthetic generation of FEs, the model classifies them into three broad categories: predictable, unpredictable and secondary, and describes each type via three key components: a) the volume of incoming traffic, b) the number of source IP addresses, and c) the resources being accessed.

The *third* contribution is the design and experimental implementation of an ensemble-based DDoS attack detection technique. The proposed technique combines the analysis of both network traffic features and server load characteristics to detect a wide range of network and application layer DDoS attacks and to effectively distinguish them from FEs.

The *fourth* contribution of the thesis is a traffic generation and testbed framework, cooperatively developed as part of this research, to synthetically generate different types of DDoS attacks and FEs, and to monitor their effects on the target. The testbed makes use of modest hardware through the exploitation of IP aliasing, a well-known technique available on most computing platforms, to generate synthetic benign and attack traffic originating from a wide range of valid IP addresses.

For my Parents and Sister.

Contents

Keywords i
Abstract iii
Table of Contents vii
List of Figures xi
List of Tables xv
List of Algorithms xvii
List of Equations xix
List of Acronyms xxi
Declaration xxiii
Previously Published Material xxv
Acknowledgements xxvii

Chapter 1 Introduction **1**
1.1 Background and Motivation 2
1.2 Research Aims and Questions 4
1.3 Research Contributions 6
1.4 Organisation of Thesis 8

Chapter 2 Background and Related Works **11**
2.1 Denial-of-Service (DoS) Attacks 12
2.1.1 Distributed Denial-of-Service (DDoS) Attacks 12
2.2 Taxonomy of DDoS Attacks 14
2.2.1 Types of DDoS Attack 16
2.2.2 Targets of DDoS Attack 23
2.3 DDoS Attack Detection 24
2.3.1 Network Traffic Analysis Based DDoS Detection 25
2.3.2 SNMP MIB Data Analysis Based DDoS Detection 27
2.3.3 Change Detection Techniques 29
2.3.4 Summary 32

2.4 Flash Event Detection and Modelling 33
2.4.1 Distinguishing Flash Events from DDoS Attacks 33
2.4.2 Flash Event Modelling 36
2.4.3 Summary 39
2.5 Synthetic Traffic Generation 39
2.5.1 Limitations of Existing Public Domain Datasets 39
2.5.2 Existing Traffic Generators 41
2.5.3 Testbed Design Strategies 43
2.5.4 Summary 44
2.6 Research Challenges 45
2.7 Summary 46

Chapter 3 DDoS Attack Detection Using IP Addresses 49
3.1 An IP-based DDoS Detection Strategy 50
3.2 Implementation of an IP-based DDoS Detection Strategy 54
3.2.1 IP Address Classification 54
3.2.2 DDoS Attack Detection 55
3.3 Change Detection 56
3.4 Experimental Results and Analysis 62
3.4.1 Datasets Used 63
3.4.2 Experimental Set-up 65
3.4.3 Performance Evaluation 66
3.5 DDoS Detection and Mitigation Model (D2M2) 68
3.6 Conclusion 69

Chapter 4 Identifying, Classifying, and Modelling Flash Events 71
4.1 Distinguishing Flash Events and DDoS Attacks 72
4.1.1 Change in the Rate of Incoming Traffic 73
4.1.2 Change in the Rate of New Source IP Addresses 73
4.1.3 Distribution of Requests Among Source IP Addresses . . . 74
4.1.4 Experimental Results and Analysis 75
4.2 Flash Event Classification 85
4.2.1 Predictable Flash Events 86
4.2.2 Unpredictable Flash Events 86
4.2.3 Secondary Flash Events 87
4.3 Flash Event Model 88

4.3.1 Change in the Rate of Incoming Traffic 89
4.3.2 Change in the Rate of Source IP Addresses 94
4.3.3 Change in the Pattern of Resources Accessed 95
4.4 FE Model Evaluation and Validation 96
4.4.1 Datasets 96
4.4.2 Change in the Rate of Incoming Traffic 97
4.4.3 Change in the Rate of Source IP Addresses 100
4.4.4 Change in the Pattern of Resources Accessed 101
4.4.5 Model Validation 103
4.5 Conclusion 104

Chapter 5 Synthetic Traffic Generation 107
5.1 Dataset Characteristics 108
5.2 Traffic Generation and Testbed Framework 110
5.2.1 Framework Design Requirements 110
5.2.2 Botloader: a software traffic generator 111
5.2.3 Experimental Testbed Architecture 116
5.2.4 Performance Evaluation 120
5.3 Simulating Flash Events 125
5.3.1 Datasets 125
5.3.2 Experimental Results and Analysis 127
5.4 Simulating DDoS Attacks 129
5.4.1 Datasets 130
5.4.2 Experimental Results and Analysis 130
5.5 Conclusion 132

Chapter 6 Ensemble Based DDoS Attack Detection 135
6.1 DDoS Attack Detection Model 136
6.2 Feature Set Selection 138
6.2.1 Incoming Network Traffic Features 138
6.2.2 Server-load Features 140
6.3 DDoS Attack Detection Technique 142
6.4 Experimental Set-up and Datasets 142
6.4.1 Network Architecture 143
6.4.2 Methodology 144
6.4.3 Datasets 145

6.5 Analysis of Change Detection Results for Individual Features . . . 150
6.5.1 Incoming Network Traffic Features . . . 150
6.5.2 Server-load Features . . . 159
6.6 Feature Correlation . . . 168
6.6.1 Detecting DDoS attacks and FEs . . . 169
6.7 Conclusion . . . 172

Chapter 7 Conclusion and Future Directions **175**
7.1 Summary of Contributions . . . 175
7.2 Limitations and Future Directions . . . 177
7.2.1 Improving Feature Correlation . . . 178
7.2.2 Evaluation against a Wider Range of Datasets . . . 178
7.2.3 Real-time DDoS Attack Detection and Mitigation . . . 178
7.2.4 DDoS Detection in the IPv6 Address Space . . . 179
7.2.5 Improving the Traffic Generation and Testbed Framework 179
7.3 Concluding Remarks . . . 180

Appendix A Change Detection using CUSUM **181**

Appendix B Traffic Generators **185**
B.1 Curl-loader . . . 185
B.2 IP-aliasing and Wget . . . 188

Appendix C DDoS Attack Dataset Generation **191**
C.1 HTTP GET flooding attack dataset . . . 191
C.2 SSL breakoff attack dataset . . . 192

Bibliography **195**

List of Figures

2.1 A typical set-up of a DDoS attack. 13
2.2 DDoS attack motivation (source: Arbor Networks [97]). 14
2.3 Taxonomy of DDoS Attacks. 15
2.4 TCP three-way handshake. 17
2.5 Amplification-based DDoS attack mechanism. 20
2.6 Reflection-based DDoS attack mechanism. 22

3.1 Rate of incoming traffic and new source IPs for CAIDA dataset. . 51
3.2 DDoS Detection using NSP source IPs 52
3.3 The testbed architecture for evaluating the NSP algorithm 65
3.4 UDP flooding attack with 35 new IPs per second. 65
3.5 UDP flooding attack with varying source IPs per second. 67
3.6 Deployment architecture of DDoS Detection and Mitigation Model. 68

4.1 Incoming traffic profile of the 1998 FIFA World Cup dataset. . . . 76
4.2 Incoming traffic profile around the semi-final matches of the 1998 FIFA World Cup dataset. 77
4.3 Incoming traffic profile for the CAIDA dataset 78
4.4 Incoming traffic profile for one 10 minute period (25^{th} – 35^{th} minute) within the CAIDA dataset. 79
4.5 Incoming traffic profile for a two hour period (46^{th} – 47^{th} hour) during the 2^{nd} semi-final match of the 1998 FIFA World Cup dataset. 79
4.6 New source IP addresses for the CAIDA dataset. 81
4.7 New source IP addresses for a two hour period (46^{th} – 47^{th} hour) during the 2^{nd} semi-final match of the 1998 FIFA World Cup dataset. 82
4.8 Requests per source IP distribution for the CAIDA and a subset of the 1998 FIFA World Cup dataset. 83
4.9 Packet IAT comparison for proxy and client from the the 1998 FIFA World Cup dataset. 84

4.10 Requests per source IP distribution for traffic during the first 25 minute of the CAIDA dataset. 85
4.11 Incoming traffic profile for an unpredictable FE dataset: hourly hits on Wikipedia following the death of Steve Jobs. 87
4.12 Traffic profile of an FE model proposed by Ari et al. [11] 91
4.13 Incoming traffic during the 1^{st} semi-final match of the 1998 FIFA World Cup dataset. 98
4.14 Packets per Source IP during the 1^{st} semi-final match of the 1998 FIFA World Cup dataset. 100
4.15 Source IPs during the 1^{st} semi-final match of the 1998 FIFA World Cup dataset. 101
4.16 Resources access pattern during the 1^{st} semi-final match of the 1998 FIFA World Cup dataset. 102
4.17 FE model validation for a predictable FE (1^{st} Semi-final of the 1998 FIFA World Cup). 103
4.18 FE model validation for an unpredictable FE (hourly hits on Wikipedia following the death of Steve Jobs). 104

5.1 Schematics of Botloader 113
5.2 Testbed architecture 116
5.3 Multi VLAN with IP forwarding 118
5.4 Incoming traffic comparison for the simulated and original FIFA dataset. 122
5.5 Source IP address comparison for the simulated and original FIFA dataset. 123
5.6 Resource entropy comparison for the simulated and original FIFA dataset. 124
5.7 Incoming traffic profile for the predictable FE. 127
5.8 Incoming traffic profile for the unpredictable FE. 128
5.9 Incoming traffic profile for the CAIDA dataset. 131
5.10 Incoming traffic profile for a simulated DDoS attack. 132

6.1 Schematic of the Ensemble-based DDoS Attack Detection Model. 138
6.2 Experimental set-up 143
6.3 Tcpdump influence on memory utilisation. 145
6.4 SSL protocol. 146

6.5 Incoming traffic volume variations for different DDoS attack datasets. 152
6.6 Incoming traffic volume for non-attack datasets. 153
6.7 New source IPs for different DDoS attack datasets. 154
6.8 New source IP addresses for non-attack datasets. 155
6.9 source IP addresses for different DDoS attack datasets. 156
6.10 Source IP addresses for non-attack datasets. 157
6.11 Incoming traffic distribution for different DDoS attack datasets. . 158
6.12 Incoming traffic distribution for non-attack datasets. 159
6.13 System CPU for different DDoS attack datasets. 161
6.14 System CPU for different non-attack datasets. 162
6.15 User CPU for different DDoS attack datasets. 163
6.16 User CPU for different non-attack datasets. 164
6.17 CPU Load for different DDoS attack datasets. 165
6.18 CPU Load for different non-attack datasets. 166
6.19 Memory utilisation for different DDoS attack datasets. 167
6.20 Memory utilisation for different non-attack datasets. 168

A.1 UDP flooding attack with 35 source IP addresses (CUSUM). . . . 181
A.2 UDP flooding attack with varying source IP addresses (CUSUM). 182

B.1 CPU utilisation with varying number of clients 186
B.2 Memory utilisation with varying number of clients 186
B.3 Number of requests with varying number of clients 187

List of Tables

2.1 Commonly used MIB variables. 29

3.1 Persistence of source IPs in the Auckland VIII and Darknet Trace. 63
3.2 Statistics of the Attack Traffic. 64

4.1 Macro-level Statistics of the 1998 FIFA World Cup and the CAIDA dataset. 78
4.2 Micro-level statistics of the 1998 FIFA World Cup and the CAIDA dataset during the observation period. 80
4.3 Flash Event Classification. 88
4.4 Datasets Used for Analysing Flash Event Model. 97
4.5 Flash Event Model Parameters. 99

5.1 Characteristic features of the dataset used for simulations. 120
5.2 FE model parameters for datasets used in simulations. 126
5.3 Characteristic features of the DDoS datasets used for simulations. 130

6.1 Hardware and the software components of the experimental set up 144
6.2 High-level characteristics of the datasets used for evaluation. . . . 149
6.3 Parameter values used in Change Detection Algorithm. 150
6.4 Feature Correlation Matrix. 170

List of Algorithms

3.1 *ipac* function for classifying IP addresses. 53
3.2 *ddos* function for identifying attacks. 53
3.3 Change Detection Algorithm. 60

List of Equations

3.1 Change Detection Condition. 57
3.2 Exponentially Weighted Moving Average (EWMA) 58
4.6 Flash-phase. 92
4.8 Decay-phase. 93
4.9 Pre Flash-phase and Post Decay-phase. 94

List of Acronyms

ANN Artificial Neural Networks

API Application Programming Interface

CDN Content Distribution Network

CI Critical Infrastructure

CLF Common Log Format

CPA Change Point Analysis

CUSUM Cumulative Sum

D2M2 DDoS Detection and Mitigation Model

DDoS Distributed Denial-of-Service

DoS Denial-of-Service

EWMA Exponentially Weighted Moving Average

FC Flash Crowd

FCM Feature Correlation Matrix

FE Flash Event

Gbps Gigabit per second

HMM Hidden Markov Model

HRF High-Rate Flooding

ICT Information and Communication Technology

IDS Intrusion Detection System

Mbps Megabits per second

MIB Management Information Base

NAT Network Address Translation

NIC Network Interface Card

NMS Network Management System

NSP Not Seen Previously

OID Object Identifier

OS Operating System

RMON Remote Monitoring

SNMP Simple Network Management Protocol

SOAP Simple Object Access Protocol

SSL Secure Socket Layer

TLS Transport Layer Security

Tor The Onion Route

URL Uniform Resource Locator

XML Extensible Markup Language

Declaration

The work contained in this thesis has not been previously submitted to meet requirements for an award at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

Signed: ..... **Date:** 16th Aug. 2013.

Previously Published Material

The following articles have been published, and contain material based on the content of this thesis.

- **Book Chapters**
  - George Mohay, Ejaz Ahmed, Sajal Bhatia, A. Nadarajan, B. Ravindran, Alan Tickle, and R. Vijayasarathy. Detection and Mitigation of High-Rate Flooding Attacks. In S.V. Raghavan and E. Dawson, editors, *An Investigation into the Detection and Mitigation of Denial of Service (DoS) Attacks Critical Information Infrastructure Protection* chapter 5, pages 131–181, Springer 2011.
- **Conference Papers**
  - Sajal Bhatia, Desmond Schmidt, and George Mohay. Ensemble-based DDoS Detection and Mitigation Model. In 5^{th} *International Conference on Security of Information and Networks (SIN 2012)*, pages 127–134, Jaipur, India, 25^{th} - 27^{th} October 2012.
  - Sajal Bhatia, George Mohay, Desmond Schmidt, and Alan Tickle. Modelling Web-server Flash Events. In 11^{th} *IEEE International Symposium on Network Computing and Applications (IEEE NCA12)*, pages 79–86, Cambridge, MA USA, 23^{rd} - 25^{th} August 2012.
  - Sajal Bhatia, George Mohay, Alan Tickle and Ejaz Ahmed. Parametric Differences Between a Real-world Distributed Denial-of-Service Attack and a Flash Event. In 6^{th} *International Conference on Availability, Reliability and Security (ARES 2011)*, pages 210–217, Vienna, Austria, 22^{nd} - 26^{th} August 2011.

– Ejaz Ahmed, George Mohay, Alan Tickle, and Sajal Bhatia. Use of IP addresses for high rate flooding attack detection. In 25^{th} *IFIP International Information Security Conference (SEC 2010): Security & Privacy – Silver Linings in the Cloud*, pages 124–135, Brisbane, Australia, 20^{th} - 23^{rd} September 2010.

Acknowledgements

I would like to express my sincere gratitude and appreciation to the many people I have worked with over the past few years for their constant support and inspiration, who have catalysed the journey to reach the current stage of my doctoral studies. First and foremost, my sincere thanks goes to my supervisors, Prof. George Mohay, Dr. Desmond Schmidt, and Prof. Alan Tickle, for their excellent guidance, encouragement and support. George's vision and experience, Desmond's meticulous approach, especially in technical and linguistic areas, and Alan's guidance and calming influence, greatly helped in improving the quality of research presented in this thesis. I would also like to thank Dr. Ejaz Ahmed, my associate supervisor for a brief period, for sharing his experience and constantly encouraging me to keep going. I would also like to thank Dr. Ernest Foo, Dr. Chun Ouyang, and all anonymous reviewers for reviewing my thesis/articles and providing constructive feedback and contributing in improving their quality.

It would not have been possible to complete this research without the generous support of Queensland University of Technology (QUT) for providing the tuition fee waver, the Australia-India Strategic Research Fund (AISRF) for the living stipend, and the Information Security Institute (ISI) for the top-up award and conference travel. I would also like to thank all the unit coordinators who showed confidence in me and offered me part-time teaching. This has definitely helped in lessening the nervousness of my public speaking, as well as providing financial support.

During the course of my doctoral studies, the time spent at the ISI provided me with a unique opportunity to meet a variety of interesting individuals from different corners of the world, with varied backgrounds, cultures and languages. Amongst all these friends, special thanks must go to Nishchal Kush, Dr. Farzad Salim and Dr. Mark Branagan, for their frequent help, their sense of humour and most importantly their friendship, something that I will cherish for a long time.

I would like to thank Gleb Sechenov and Edward Chang for their friendship and support, especially in setting up of the testbed infrastructure. I would also like to thank all other friends and colleagues at the ISI – Chris Djamaludin, Andrew White, Christophe Hauser, Raphael Amoah, James Mackie, Vik Tor Goh, Eesa Alsolmi, Roheena Khan, Ali Alhamdan, Chai Wen Chuah, Choudary Gorantla, Kenneth Radke, Dr. Sui Guan Teo, Kaleb Lee, Mufeed Al Mashrafi, Jothi Rangasamy, Lakshmi Kuppusamy, Janaka Alawatugoda, Prof. Emeritus Ed Dawson, Prof. Colin Boyd, Dr. Jason Smith, Prof. Andrew Clark and Dr. Juanma González Nieto for creating a cordial working environment. I would also like to thank Elizabeth Hansford and Christine Kincaid for taking care of all administrative issues and ensuring my smooth travel to and from conferences.

I would like to express my gratitude to Sachin Goyal and Bharati Gupta for their support during the final stages of my PhD. Outside QUT, I am blessed to have friends like Deborah Babulal, Nitin Singhal, Sanketa Pawar, Megha Goyal, Pragya Parihar, Dushyant Singh, and all my flatmates, particularly the Jane Street Crew, with whom I have shared the house.

Last, but by far not the least, my deep and sincere gratitude goes to my parents Suresh and Sangeeta Bhatia, and my sister Saloni Bhatia, to whom this thesis is dedicated. Without their continual and unparalleled love, help and support, I would not have been able to complete this PhD. Thanks Mum and Dad for always being beside me, and often having more faith in me than I had in myself.

Thank You

Chapter 1

Introduction

Over the past decade, advances in Information and Communication Technology (ICT) have significantly transformed the way in which information is accessed and communicated, particularly via the Web. The range of services supported by ICT has been constantly expanding and in recent years has even included the control and monitoring of key systems like power, water, gas, etc., also known as Critical Infrastructure (CI). This evolution of ICT has also entailed a significant dependence of society on the systems for storing, processing and communicating information. As a result, any malfunction in these information and communication systems directly affects, in one way or another, nearly all major aspects of contemporary society. This increasing reliance on ICT in general, and on communication networks in particular, is most keenly felt when the services delivered by these networked systems are disrupted even for relatively short periods.

These situations can be the result of a *deliberate* attempt by an attacker[1] to disrupt the services of a provider, as in the case of a Denial-of-Service (DoS) attack, or they may be the unintended consequence of a Flash Event (FE), where the sheer volume of legitimate traffic leads to a degradation of service. Both of these situations represent *anomalies* in the normal behaviour of Internet traffic. FEs and Distributed Denial-of-Service (DDoS) attacks – the latter being the case where DoS traffic originates from a number of attackers simultaneously – are the subjects of this thesis.

The rest of this chapter is organised as follows. Section 1.1 provides a brief

[1]The terms adversary and attacker are used interchangeably throughout the thesis.

overview of the background of DoS attacks and FEs and explains the motivation for the research presented in the thesis. Section 1.2 then identifies a set of research objectives and related research questions. Section 1.3 summarises the contributions of the thesis. Finally, Section 1.4 describes the overall structure of the thesis.

1.1 Background and Motivation

The Internet community has been facing the DoS problem for over two decades. One of the earliest known DoS attacks occurred in 1974 at the Computer-based Education Research Laboratory (CERL), at the University of Illinois Urbana-Champaign [39]. A novice programmer forced 31 computers in the laboratory to simultaneously power-off by exploiting the default configuration of PLATO terminals running the TUTOR programming language i.e. the default is to accept remote 'external or ext' commands. An 'ext' command issued to a PLATO terminal, not connected to an external peripheral device, would force it into a locked state, requiring it to be manually restarted. This incident, although on a small scale, introduced the concept of a DoS attack. The DoS problem was first brought onto the global stage in November 1998 with the introduction of the Morris Worm [111], a computer program written by a graduate student at Cornell University, which paralysed the Internet in its nascent stages. Even though the spread of the Morris worm is considered to be the first DoS attack to infect geographically distributed machines, it is believed that the program was originally designed without any malicious intent.

The first deliberate DoS attack on the public Internet occurred at the University of Minnesota in August 1999, more than a decade after the spread of the Morris Worm [45, 133]. Very soon after, in February 2000, a group of popular commercial websites: Yahoo, eBay, and CNN were attacked and flooded with a large number of requests, forcing them off-line, and causing significant financial losses [48]. Historically, this incident has been identified as the first deliberate large-scale DoS attack aimed at creating financial havoc, and thus can be regarded as the start of a new kind of threat to network security.

These attacks harnessed the computing power of hundreds or thousands of attacker-controlled machines (also known as 'bots') and used them collectively to send huge volumes of traffic against a target host to cause a DoS. This type of

attack is commonly known as a Distributed Denial-of-Service (DDoS) or High-Rate Flooding (HRF) attack[2]. Now decades later, and notwithstanding their conceptual simplicity, DDoS attacks using the same basic *modus operandi* still exist; the only difference being their magnitude, complexity and frequency, all of which have increased dramatically. Over the years, the DDoS problem has become so widespread that every networking or computing device connected to the Internet is potentially susceptible to such attacks. Thus the study of DDoS attacks, and the development of techniques to accurately and reliably detect and mitigate their impact, continues to be an active area of research and development.

Flash Events

The delivery of an online service can also be degraded as a result of legitimate user activity, without any malicious intent. Such situations arise when a large number of users concurrently access a web-server, either following some newsworthy event (e.g., the Olympics, the 9/11 attacks), or as a result of redirection from widely-followed websites such as Slashdot or other social media like Facebook or Twitter. These situations are called Flash Events (FEs). Both DDoS attacks and FEs represent anomalies in the normal Internet traffic, and share a number of similar characteristics, such as a substantial increase in the incoming network traffic, the overloading of the servers providing the services, and a degradation in the delivery of service.

Like DDoS attacks, FEs can also have significant financial implications. A recent example of an FE causing a substantial financial loss occurred on the 20^{th} of November, 2012, when Click Frenzy, a national online-shopping initiative in Australia similar to the US based Cyber Monday[3] event, was launched after heavy media and online publicity. The website (clickfrenzy.com.au) experienced many times its anticipated traffic volume, leading to a dramatic increase in page-load times, and the failure of the website within minutes of its launch [41, 79]. Organisers of the website had pre-arranged sales partnerships with leading Australian and International retailers and brands, many of whom had paid large amounts for advertisements, which they asked to be refunded [42].

DDoS attacks and FEs share a number of similar characteristics, and can

[2]In the context of this thesis, DDoS refers to High-Rate Flooding (HRF) attacks. Both the terms – DDoS and HRF DDoS attacks – are used interchangeably throughout the thesis.

[3]Cyber Monday is a marketing term defined by retail companies in US to persuade buyers to shop online on the Monday following the Thanksgiving.

thus be mistaken for one another [64]. A recent example of the misinterpretation of an FE as a DDoS attack occurred on the 8^{th} June 2012, when the Star Trek hero George Takei posted a link on his Facebook page to a website selling 'Takei T-shirts'. The Facebook post soon attracted the interest of his 2 million fans, and redirected them to a relatively small web-server. The click-through rate was high enough to make the web-server's ISP believe that the incoming traffic was a DDoS attack, orchestrated by nefarious bots (compromised machines) rather than by legitimate clients, forcing the site to be shut-down for several hours.

Owing to such similarities, DDoS attacks and FEs may be mistaken for one another, yet it is important to differentiate between them, since each anomaly requires a different set of actions to be undertaken by a network administrator. While the identification of DDoS attacks requires the initiation of attack mitigation strategies and the filtering out of malicious traffic, the identification of an FE ideally requires the use of load-balancing and other content distribution techniques to cater for the increased demand.

1.2 Research Aims and Questions

The overarching aim of this thesis is *to investigate and develop techniques to detect DDoS attacks accurately and reliably, and to investigate and develop techniques to differentiate DDoS attacks from FEs* with which they share similar characteristics. A necessary and related aim has been *to investigate and develop techniques to synthesise realistic emulations of DDoS attack traffic and FE traffic* that can be used to enable the primary research aim. As a result of these complementary aims, the following research questions have been identified:

Research Question 1 The literature reviewed and presented in Chapter 2 indicates that the source IP address of network traffic is widely used in determining the onset of a DDoS attack. Research question 1 seeks to simplify the application of this property to the DDoS detection problem: *Can the onset of a DDoS attack be detected based on simple network traffic characteristics such as source IP address?* Using this one simple feature of source IP address directly addresses the issue of scalability of the proposed DDoS attack detection method when applied to higher network speeds.

Research Question 2 The detection techniques available in the public domain focus on detecting DDoS attacks without considering their differentiation from FEs, and yet a reliable method for detecting DDoS attacks must also be able to distinguish these two network anomalies. This leads to the second research question: *Is it possible to differentiate DDoS attacks from FEs?*

Research Question 3 An important resource for developing and evaluating techniques to identify FEs are traffic traces of actual FE events. Unfortunately, only a very limited number of FE datasets are available, mainly due to the legal and privacy concerns associated with sharing them publicly. Given this scarcity, an obvious remedy to overcome this issue is to develop a theoretical model of FEs, and use it to synthetically generate realistic FE traffic. Although a number of FE models have been proposed, they fail to capture the essential characteristics of different types of FEs (see Section 2.4.2 for details). This gives rise to Research Question 3: *What are the essential characteristics of different types of FE and is it possible to use those characteristics to model and subsequently generate different types of synthetic FE traffic?*

Research Question 4 The extreme paucity of *recent* and *realistic* datasets reflecting real-world DDoS attacks and FEs in the public domain gives rise to Research Question 4: *Is it possible to design and implement a simple experimental traffic generation and testbed framework, using only modest hardware for the testbed and a software-only traffic generator, to synthetically generate realistic network traffic representing DDoS attacks and FEs?*

Research Question 5 The final research question follows from all the above: given real or realistic synthetic traffic representing FEs and DDoS attacks, how can these two network anomalies best be detected and differentiated? The DDoS attack detection literature surveyed for this thesis indicates a heavy use of traffic based parameters obtained either from the incoming network traffic or collected by polling the Management Information Base (MIB) of the target host using the Simple Network Management Protocol (SNMP). To the best of our knowledge, there is no DDoS detection technique in the public domain which makes use of the target machine's load-based features such as CPU and memory utilisation. Thus, one final research question is: *Can a more sophisticated ensemble-based technique based on both network traffic characteristics and server-load data (memory and*

CPU utilisation) improve the accuracy of DDoS attack detection and provide a means to identify and differentiate DDoS attacks, FEs and normal traffic?

1.3 Research Contributions

This thesis has resulted in a number of research contributions covering various facets of DDoS attacks and FEs, mainly relating to their identification and differentiation from one another and from normal traffic, and the synthetic generation of all three. Specifically this research has resulted in the following contributions and research outcomes.

Contribution 1: In Chapter 3, the design and a proof-of-concept implementation of a a DDoS attack detection technique based on the change in rate of new source IP addresses of the incoming packets is presented, thereby addressing Research Question 1. The proposed technique extends the previous research conducted in this direction by demonstrating how a simple network traffic characteristic such as source IP address of the incoming packets, coupled with a change-point analysis of the rate of arrival of new source IP addresses may be sufficient to detect the onset of a DDoS attack. The proposed technique minimises the number of features to be examined, thereby directly addressing the issue of scalability of the detection process at higher network speeds. This research outcome led to the following publication:

> Ejaz Ahmed, George Mohay, Alan Tickle, and Sajal Bhatia. Use of IP addresses for High Rate Flooding Attack Detection. In 25^{th} *IFIP International Information Security Conference (SEC 2010): Security & Privacy – Silver Linings in the Cloud*, pages 124–135, Brisbane, Australia, 20^{th} - 23^{rd} September 2010.

The chapter also outlines the conceptual design of a lightweight mitigation strategy, including a DDoS Detection and Mitigation Model (D2M2), which uses ModSecurity as the application firewall.

Contribution 2: In Chapter 4, an FE model is proposed which provides the foundations for identifying and synthesising FEs. The proposed model addresses Research Question 2 by first exposing some subtle differences between FEs and

DDoS attacks. It proposes a set of parameters for differentiating DDoS attacks from FEs, and extends previous attempts that have been made to distinguish between these two network anomalies. The proposed parametric differences are based on the analysis of two widely used and publicly available datasets representing a real-world FE and a DDoS attack, thereby providing for validation and reproducibility of the obtained results – a characteristic lacking in the majority of proposed approaches, which use proprietary datasets. This contribution led to the following publication:

> Sajal Bhatia, George Mohay, Alan Tickle and Ejaz Ahmed. Parametric Differences Between a Real-world Distributed Denial-of-Service Attack and a Flash Event. In 6^{th} *International Conference on Availability, Reliability and Security (ARES 2011)*, pages 210–217, Vienna, Austria, 22^{nd} - 26^{th} August 2011.

Chapter 4 also presents a server-side FE model with a view to synthetically generating different types of FE. The novelty of this work is its use of a simple exponential model with a small set of configurable parameters that captures various characteristic properties of different types of FEs, and can thus be used to synthesize realistic emulations of different FE traffic. This research contribution addresses Research Question 3 identified above, and led to the following publication:

> Sajal Bhatia, George Mohay, Desmond Schmidt, and Alan Tickle. Modelling web-server Flash Events. In 11^{th} *IEEE International Symposium on Network Computing and Applications (IEEE NCA12)*, pages 79–86, Cambridge, MA USA, 23^{rd} - 25^{th} August 2012.

Contribution 3: Chapter 5 presents a network traffic generation and testbed framework, co-operatively developed as a part of this research, to synthetically generate and deploy different types of DDoS attacks, FEs and normal traffic, which closely resemble real-world scenarios. The framework makes use of modest hardware and exploits IP-aliasing – a well-known technique available on most computing platforms – for assigning multiple distinct IP addresses to a single hardware address, and uses these aliases to synthetically generate normal and anomalous traffic originating from a wide range of source IPs. The testbed design, being simpler than existing approaches such as the DETER [17] and Emulab [124]

testbeds, provides a means for constructing and managing an isolated DDoS test facility at minimal cost, and thus addresses Research Question 4.

Contribution 4: In Chapter 6, a novel ensemble-based DDoS attack detection technique is proposed. The experimental implementation of the proposed technique correlates network traffic features with server-load characteristics to detect a wide variety of DDoS attacks and to effectively distinguish them from FEs. To the best of our knowledge, no other technique has been found which correlates these two rather orthogonal DDoS attack detection strategies – network traffic analysis and server-load analysis. The proposed DDoS attack detection model addresses Research Question 5, and led to the following publication:

> Sajal Bhatia, Desmond Schmidt, and George Mohay. Ensemble-based DDoS Detection and Mitigation Model. In 5^{th} *International Conference on Security of Information and Networks (SIN 2012)*, pages 127–134, Jaipur, India, 25^{th} - 27^{th} October 2012.

1.4 Organisation of Thesis

The remaining six chapters of the thesis, after the current one, are organised as follows:

Chapter 2 begins by introducing the concepts of DoS and DDoS attacks and subsequently gives an overview of DDoS attack classification, focusing on attack-types and attack-targets. It summarises some of the most pertinent research conducted in detecting DDoS attacks, and covers the two major attack detection strategies, viz., those based on network traffic analysis and SNMP MIB data analysis. The review of existing literature aids in identifying the limitations of existing attack detection approaches and presents the research challenges involved in addressing the problem. This chapter also provides a summary of research work done towards distinguishing FEs from DDoS attacks, including FE modelling, and reviews current practices in synthetic traffic generation, for both DDoS attacks and FEs. Some sections of Chapter 2 have been previously published as the following book chapter.

> George Mohay, Ejaz Ahmed, Sajal Bhatia, A. Nadarajan, B. Ravindran, Alan Tickle, and R. Vijayasarathy. Detection and Mitigation of High-Rate

Flooding Attacks. In S.V. Raghavan and E. Dawson, editors, *An Investigation into the Detection and Mitigation of Denial of Service (DoS) Attacks Critical Information Infrastructure Protection* chapter 5, pages 131–181, Springer 2011.

Chapter 3 presents a DDoS attack detection technique based on source IP addresses of the incoming network traffic. The chapter demonstrates how a simple network traffic characteristic like source IP address, coupled with a change detection technique, can be sufficient to detect the onset of DDoS attacks. The chapter provides a proof-of-concept implementation, using bit vectors, of the proposed attack detection technique. The implementation also illustrates how information about potentially anomalous source IP addresses could be used to dynamically alter source IP address based filtering rules within existing network security devices like firewalls, and thereby mitigate the effect of a DDoS attack.

Chapter 4 proposes an FE model which can be used to identify FEs, to differentiate them from DDoS attacks and to synthetically generate realistic FE network traffic. This chapter provides a comprehensive study of FEs and classifies them into three broad categories: predictable, unpredictable and secondary. It further analyses two publicly available datasets and proposes a set of parameters that can be used to distinguish DDoS attacks from FEs. Finally, the chapter presents a characterisation of FEs according to three key components and uses that characterisation to develop a server-side mathematical model for FEs. The FE model uses a small set of configurable parameters and captures the various characteristic features of different types of FEs. The proposed model is used in the following chapter to synthetically generate network traffic representing different types of FEs.

Chapter 5 outlines the characteristics of the datasets required for this research and summarises the available options for obtaining them. The chapter highlights the shortcomings of some of the highly referenced public domain datasets and off-the-shelf traffic generators used in DDoS-related research. The chapter also describes a traffic generation and testbed framework, including both the hardware set-up and the software traffic generator, cooperatively developed as a part of this research. Finally, this chapter uses the developed testbed facility to realistically emulate different types of DDoS attack and FE traffic, which are used

in the following chapter.

Chapter 6 proposes an ensemble-based DDoS attack detection model. The model correlates two orthogonal DDoS attack detection strategies – network traffic analysis based DDoS detection and server-load analysis based DDoS detection. The post-hoc implementation of the proposed attack detection model not only increases the spectrum of DDoS attacks that can be efficiently detected, but also differentiates DDoS attacks from similar looking FEs.

Chapter 7 concludes the thesis by summarising the research contributions and proposing possible directions for future research.

Chapter 2

Background and Related Works

The primary research aim of this thesis, as described in Chapter 1, is to investigate and develop techniques that detect DDoS attacks accurately and reliably, and which can differentiate them from FEs, with which they share some similar characteristics. A necessary and related aim has been to investigate and develop techniques to synthesise realistic emulations of DDoS attack and FE traffic that can be used to enable the primary research aim. In this Chapter, some of the most pertinent research conducted in detecting DDoS attacks, including those approaches seeking to distinguish them from FEs, along with current practices in synthetic traffic generation (both for DDoS attacks and FEs) will be reviewed and analysed.

This chapter is organised as follows. Section 2.1 introduces the concept of a DoS attack and its distributed form called a DDoS attack, also commonly known as a High-Rate Flooding (HRF) attack, which forms the research focus of this thesis. Section 2.2 provides an overview of the classification of DDoS attacks, focusing on the type of attacks and their targets. A detailed analysis of the research conducted in detecting DDoS attacks, using both network traffic analysis and SNMP MIB data analysis, is presented in Section 2.3. Section 2.4 presents related work in detecting and modelling FEs, and in differentiating them from DDoS attacks. Section 2.5 summarises previous work done in the synthetic generation of network traffic, both attack and benign. Section 2.6 then presents the remaining research challenges associated with the efficient detection of DDoS attacks and their separation from FEs. Finally, the chapter is summarised in

Section 2.7.

2.1 Denial-of-Service (DoS) Attacks

On Thursday, 6^{th} August, 2009 one of the most widely followed social networking and micro-blogging online services, Twitter, was brought down for several hours, silencing its millions of Tweeters [27]. This was the first major, and possibly deliberate, outage of service that the social networking website had suffered. The first official word that came in was a rather terse statement: 'Site is down – We are determining the cause and will provide an update shortly'. This message was soon updated by its co-founder Biz Stone: 'On this otherwise happy Thursday morning, Twitter is the target of a denial of service attack'. This statement, however, gave no indication to its clients as to how they were trying to defend against the attack or how long would it take to restore services.

A DoS attack is a malicious attempt by an attacker to disrupt the online services of a service provider (server) and to make it unavailable to its legitimate users. The National Information Assurance (IA) Glossary provided by the Committee on National Security Systems (CNSS) gives a more general definition and identifies DoS as [120]:

> Any action or series of actions that prevents any part of an [information system] from functioning.

A DoS attack against an online service provider can target a computing resource such as CPU, or a network resource such as the bandwidth of the victim's network link or a combination of both. The effect of a DoS attack can range from a minor increase in the service response time to complete inaccessibility, and at times having financial implications on organisations heavily reliant on the availability of their service. A recent report by Amazon suggests that even a 100 ms delay in response time causes an approximately 1% drop in their overall sales [67].

2.1.1 Distributed Denial-of-Service (DDoS) Attacks

A Distributed Denial-of-Service (DDoS) attack is a distributed variant of the more generic DoS attack, in which an array of geographically spread compromised machines (aka bots, zombies, slaves, agents) controlled by an attacker (or bot-master) are used against some target(s) to cause denial of service. A

set of these compromised machines or bots is called a *botnet*. In this form of DoS attack, the individual attacking capacity of each compromised machine is aggregated and subsequently used against a common target, thereby magnifying the effect. Figure 2.1 shows the working model of a typical DDoS attack. The bot-master takes control of an army of machines without the knowledge of their owners, commonly by infecting them with a Trojan or a backdoor program. These compromised machines are controlled by the bot-master, often via C&C (Command and Control) channels, and simultaneously used to attack a target server using the public Internet infrastructure.

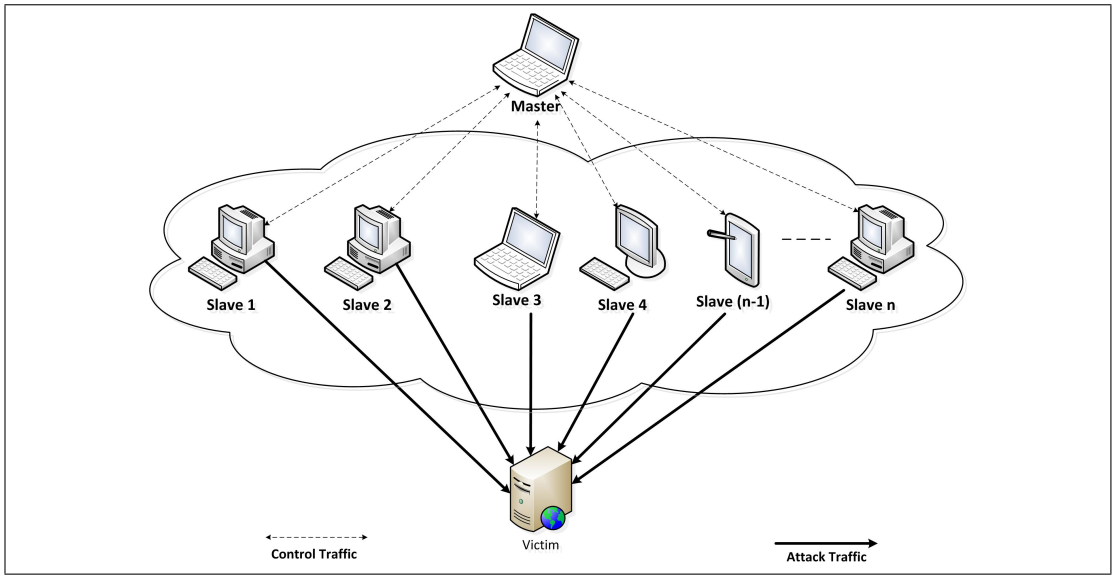


Figure 2.1: A typical set-up of a DDoS attack.

The motives behind DDoS attacks are various, ranging from fun to financial gain to political, as in the case of the attacks on Estonia and Georgia [108]. According to a recent report on Worldwide Infrastructure Security by Arbor Networks, ideologically-motivated 'hacktivism' and 'vandalism' appear to be the most readily-identified motivations behind DDoS attacks [97]. Figure 2.2 highlights the different attack motivations based on the survey conducted by Arbor Networks [97].

The *worldwide* prevalence of DDoS activity was exposed by Moore et al. using their proposed traffic monitoring technique called 'backscatter analysis' [77]. Their technique was based on the observation that during an attack, attackers often forge or 'spoof' the source IP address of packets before sending them to the target host. Source IP address spoofing may be done to conceal identity or

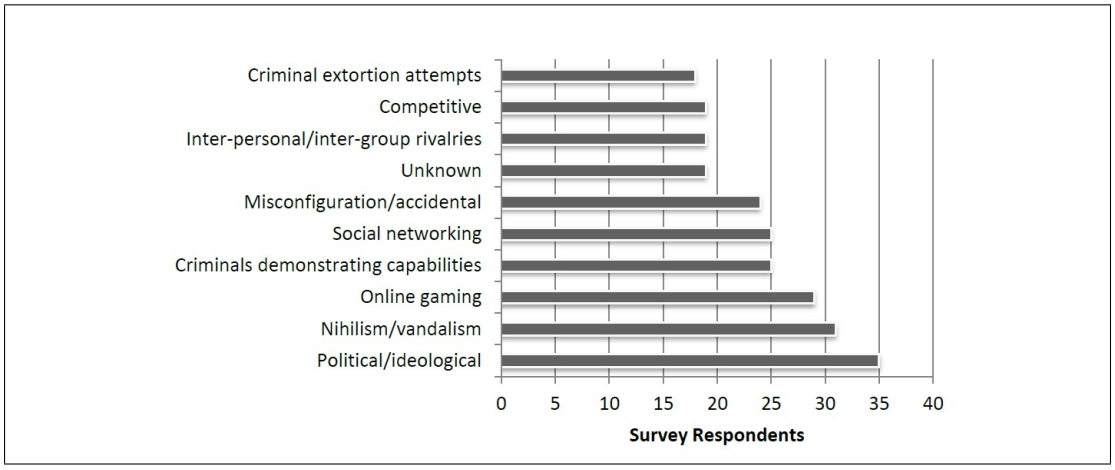


Figure 2.2: DDoS attack motivation (source: Arbor Networks [97]).

location. Subsequently, when these incoming packets from spoofed IPs arrive at the victim's machine, a response is sent back to what is believed to be a genuine host (the faked IP address). As the IP address is randomly selected, the entire Internet address space is equally likely to receive a response, a phenomenon they refer to as backscatter. Using backscatter analysis, based on the number of reply packets sent to non-existent or unallocated IPs, over a period of three weeks, 12,805 attacks directed against an approximately 5,000 distinct Internet hosts belonging to more than 2,000 different organisations – ranging from small ISPs to well-known e-commerce companies like Amazon – were observed.

A recent report by Prolexic conforms with the widespread nature of the DDoS problem, with attacks originating from a wide gamut of countries, lead by China (55.44%) and followed by Germany (9.07%) and India (8.77%), during the fourth quarter of year 2012 [3]. The published report speculates that the reason for China being at the top of this hierarchy is due to the large number of vulnerable machines (servers and normal workstations) residing within the country.

2.2 Taxonomy of DDoS Attacks

Research into DDoS attack detection has, not surprisingly, been based largely on an overall two-dimensional view of the DDoS problem – the problem generally being viewed in terms of the 'type of attack' and the 'target of attack'[1]. The classification of DDoS attacks in terms of the 'target of attack' not only foreshad-

[1]A detailed classification of DDoS attack mechanisms can be obtained from the paper by Mirkovic and Reiher [71].

ows the possibility of a DDoS attack on virtually any networked system, but also highlights the fact that the magnitude of its impact depends on the resources available to the attackers.

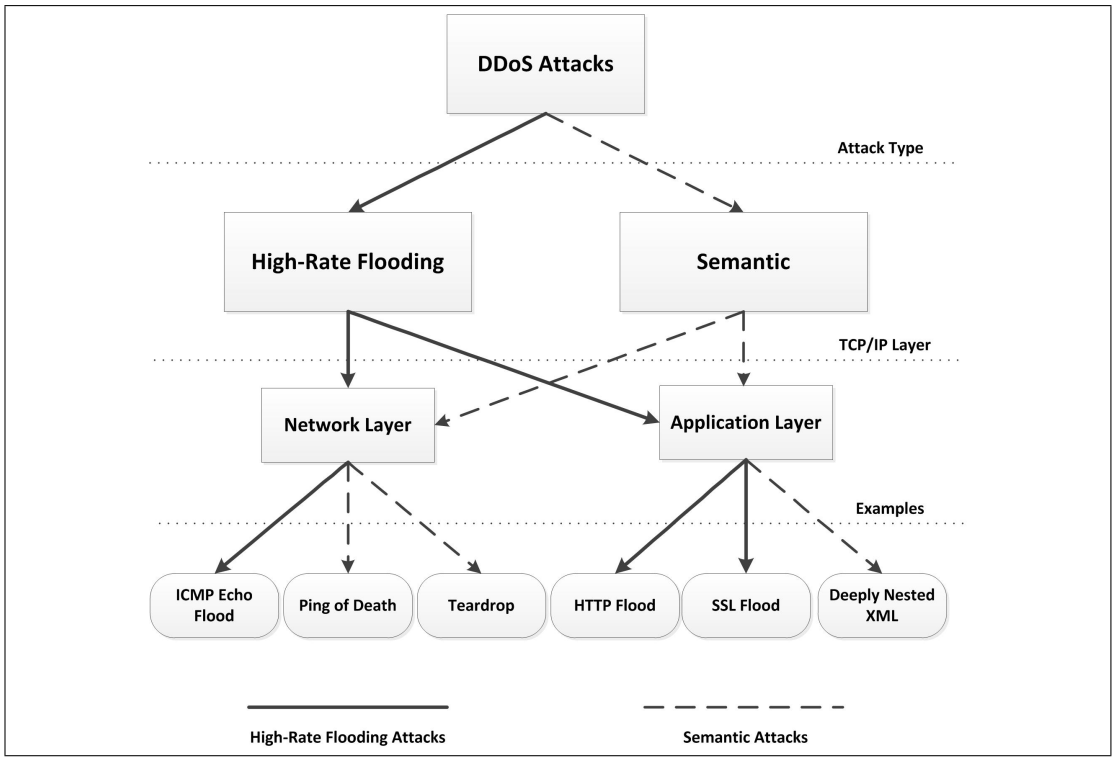


Figure 2.3: Taxonomy of DDoS Attacks.

Mirkovic et al.[71] classified DDoS attacks by: degree of automation, exploited vulnerability, source address validation, possibility of characterisation, attack rate dynamics, impact on victim, victim type, and persistence of agent set. The research presented in this thesis focuses on High-Rate Flooding (HRF) DDoS attacks against the networking and computing resources of a target host, i.e., flooding attacks targeting the network and the application layer of the TCP/IP network model. Some examples of such attacks are given in Figure 2.3. It is to be noted that both HRF and semantic attacks can occur either at the network or application layer of the TCP/IP stack. An ICMP flooding attack is an example of a network layer attack targeting the available network bandwidth of the victim, whereas a HTTP flood attack targeting an application running on the target host is an example of an application layer flooding attack. Similarly, Teardrop and Deeply Nested Extensible Markup Language (XML) attacks are examples of network and application layer semantic attacks respectively.

2.2.1 Types of DDoS Attack

The increasingly critical dependence of society on networked systems presents adversaries with a multitude of possible targets. In order to exhaust their available resources and cause a DoS for legitimate clients, an adversary can mount an attack by either flooding the victim with a large number of spurious requests, as in a High-Rate Flooding (HRF) or brute force attack, or by exploiting the design or implementation flaws of a communication protocol or service application as in a *semantic attack*, as shown in Figure 2.3. The following two sub-sections discuss these two attack types, with a particular emphasis on HRF attacks, their coordination and their mechanisms.

2.2.1.1 High-Rate Flooding Attacks

High-rate flooding or brute force attacks aim at consuming critical resources on the victim, thus compromising its ability to deliver services to its legitimate clients. The *modus operandi* of these attacks is flooding the victim with a large number of packets and thus exhausting its available network or computing resources.

High-Rate flooding attacks often require the adversary to amass sufficient resources, both network bandwidth and computing power, in order to overwhelm the victim. Accumulating these resources might have been difficult in the past, but today, with the advent of advanced software techniques, higher available bandwidths, and the proliferation of compromised machines (bots) that can be 'hired' for as low as $6 an hour, it is not particularly difficult [57]. Once these desired resources have been acquired, launching high-rate flooding attacks is simple.

2.2.1.1.1 Common High-Rate Flooding Attacks

The three common HRF attacks – TCP SYN flooding, UDP flooding and HTTP flooding are now described.

TCP SYN flooding attack A classic example of a high-rate flooding attack is the 'TCP SYN flooding attack' [37]. In this attack, the adversary takes advantage of design flaws in the three-way handshake of the TCP protocol, shown in Figure 2.4. In a normal execution of a TCP connection, a client first sends a SYN packet to the server. The server, upon receiving the connection request,

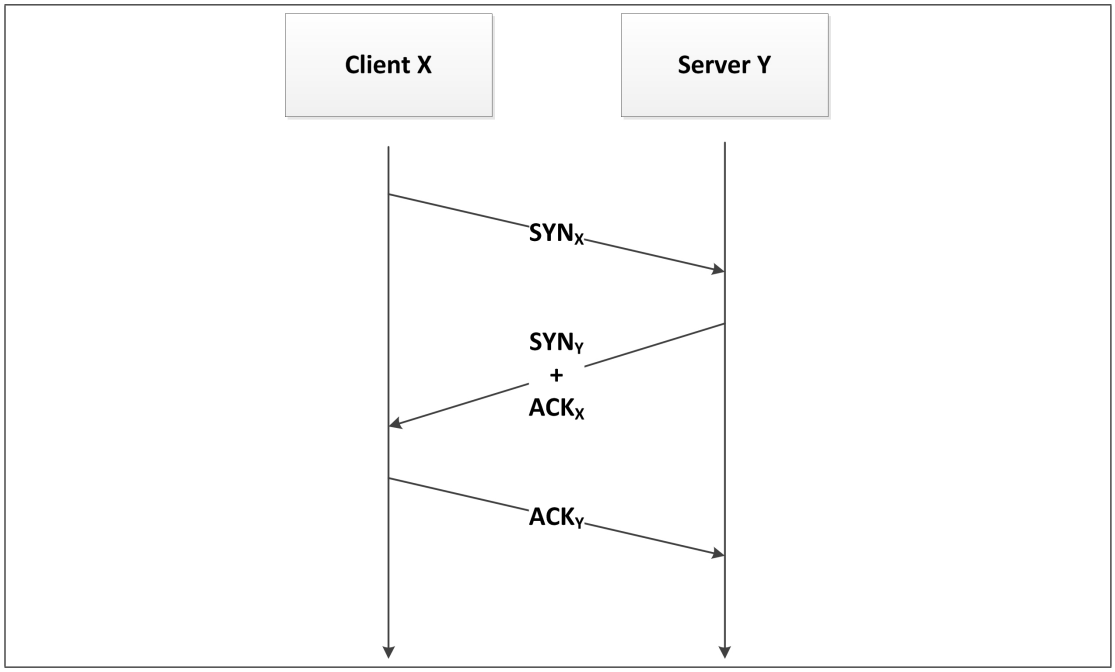


Figure 2.4: TCP three-way handshake.

opens a new session, allocates resources for this connection and responds to the client with a SYN/ACK packet. The client then responds with a ACK packet to complete the three-way handshake.

If no ACK packet is received by the server within a specified duration, a connection timeout state is reached, closing the TCP session and releasing the allocated resources. Using a lower timeout is a plausible but is an insufficient defence during a high rate TCP SYN flooding attack, in which the adversary sends a large number of SYN packets without sending the final ACK to complete the three-way handshake. This forces the server to maintain and allocate resources, albeit briefly, for each of the 'half-open' connections and eventually exhausts its available resources.

One common security measure often employed to guard against TCP SYN flooding attacks is the use of 'SYN cookies[2]' on the victim's machine [20]. SYN cookies are defined as 'particular choices of initial TCP sequence numbers by TCP servers' [20], the use of which allows the server not to drop connections when the SYN queue fills up. The TCP sequence number is encoded in the SYN-ACK response sent by the server, instead of being stored locally in the

[2]On most Linux based systems, SYN cookies can be enabled by editing `/proc/sys/net/ipv4/tcp_syncookies` file, and adding it permanently to the start-up scripts.

SYN queue. When the ACK packet comes back from the client the entry is reconstructed from the TCP sequence number. Thus, using SYN cookies, the server avoids keeping half-open states for incoming TCP connections, and so prevents memory exhaustion.

UDP flooding attack An example of a transport layer flooding attack is the 'UDP flooding attack'. In this form of attack, the adversary sends a large number of UDP packets to random ports on the target machine, usually from spoofed IP addresses [33]. As a result, the target host checks for applications running on the ports specified in the incoming packets. If no application is listening on those ports, it replies with an ICMP Destination Unreachable packet. Thus, for a large number of incoming UDP packets on random ports, the target machine can be forced to send a large number of ICMP packets, provided no application is listening on those ports, and so use up its connection bandwidth and eventually become unreachable by its clients.

HTTP flooding attack A 'HTTP flooding attack' is an example of a high-rate flooding attack operating at the application layer of the TCP/IP stack [28, 68]. In this form of attack, the adversary sends a large number of 'seemingly legitimate' HTTP requests, commonly GET and POST, to the target server requesting web-pages, usually `index.html`, and sometimes the server incurs a substantial resource (CPU or memory) cost in their processing and transfer. In this form of attack, the access requests are sent via a normal TCP connection containing a 'valid' HTTP GET or POST request, so forcing the target server to treat them as 'legitimate or normal' requests, and thus aggravating the existing problem of efficient DDoS attack detection.

Such high-rate flooding attacks appear to be conceptually simple and fairly easy to execute. However, in order to cause any substantial damage to a commercial server equipped with much higher resources than a normal desktop computer, the attacker needs to make use of an army of compromised machines, a botnet, with aggregate resources at least comparable to those of the victim. Details of such HRF attacks executed using a botnet are provided later in this section.

2.2.1.1.2 Attack Coordination

HRF attacks can be further categorised based on how the attack is coordinated:

either manually through human coordination, or automated via the use of botnets.

Manually Coordinated DDoS Attacks This type of DDoS attack is usually 'ideologically-motivated' and requires significant human coordination in order to have the desired effect on the victim. In such attacks, the attacker orchestrates a large army of volunteers with a common purpose, who use their own machines running a shared attack tool, for example connecting via the The Onion Route (Tor)[3] network to simultaneously send service requests to the target host. Depending upon the number of volunteers participating in the attack, the computing capacity of the individual machines, and the sophistication of attack tool being used, the aggregated volume of traffic sent to the victim can consume its available resources and eventually cripple its services.

The F5 attack is a classic example of such attack. This attack can be launched by coordinating a large group of volunteers and instructing them to continuously press and hold down the F5 (page reload) key while accessing a web-page through a browser. By continuously holding down the F5 key, the browser sends repeated HTTP GET requests to the web-server and so downloads a large volume of data. A teenager from Lake High School was charged with a felony for launching an F5 attack against his school's computer system [119].

A more recent example of a DDoS attack executed with human coordination was *Operation Payback*, a name given to a series of attacks conducted by a group called *Anonymous* against anti-piracy organisations and commercial websites (PayPal, Mastercard, Amazon EC2), who withdrew their ties with whistleblower website WikiLeaks [5, 26, 62]. The attacks were conducted using the LOIC (Low Orbit Ion Cannon) tool, an open source network stress testing utility, modified for launching DDoS attacks. A new feature called 'Hivemind' was added and used to connect the participants' LOIC tool to 'AnonOps' (an international communication platform used by Anonymous in Operation Payback) to receive instructions for the attack [104]. Later the tool had to be installed on the participant's machine in order to take part. The Anonymous group even created a web-page, which could effectively turn a web-browser into an attack

[3]Tor is system/network providing online anonymity to its clients. The Tor client software routes the Internet traffic via a widespread network of volunteer servers, implementing onion routing, in order to conceal user's identity such as location or usage pattern, from people/organisations conducting network surveillance or traffic analysis.

tool. It required volunteers only to visit that web-page and click on the attack button [104].

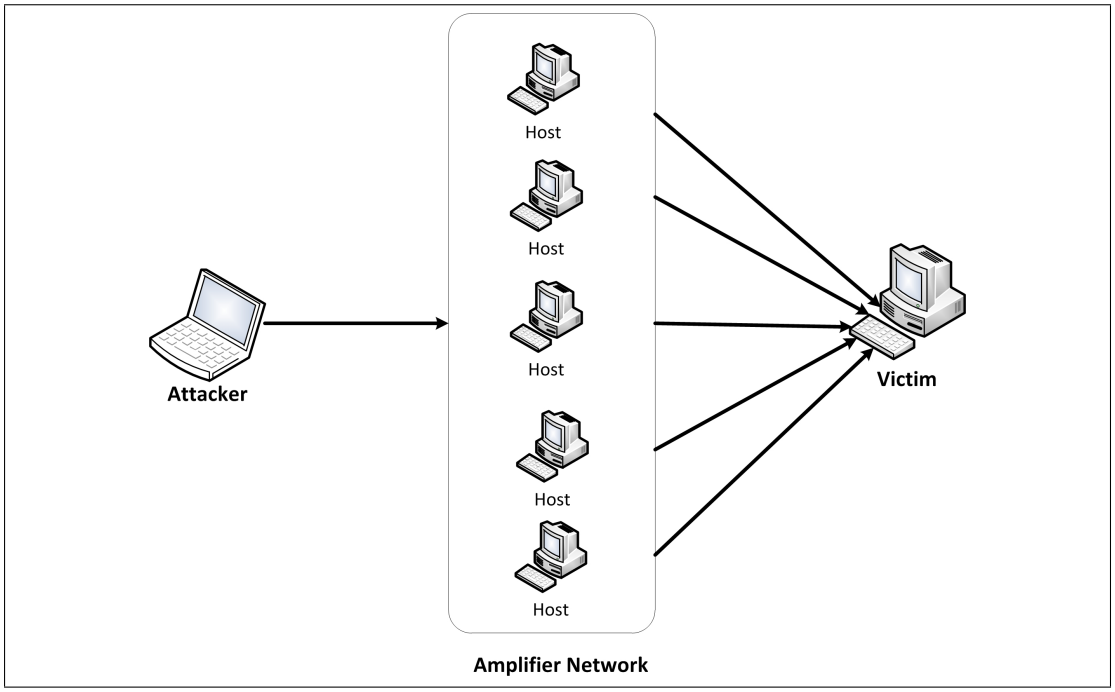


Figure 2.5: Amplification-based DDoS attack mechanism.

Automated or Semi-automated DDoS Attacks These attacks rely on exploiting aspects of network protocols and misusing them in order to amplify and/or obfuscate traffic they direct towards a victim. Such attacks can be automated or semi-automated in nature and are launched via botnets.

2.2.1.1.3 Attack Mechanisms

The most commonly used mechanisms for launching attacks – amplification and reflection are now described.

Amplification Attacks An 'Amplification-based DDoS attack' consists of an attacker, an amplification network and a victim. An amplification network is essentially a network of host machines which permits broadcast messages. Any such network, when used for communicating via a reply-based protocol like ICMP, is potentially prone to amplification-based DDoS attacks. The amplification is caused by the broadcast packet of the network, usually sent from a spoofed source IP address, which triggers a response from every host in the

network. Depending on the number of hosts present, an equivalent number of response packets are generated and directed towards the intended victim. Figure 2.5 illustrates a typical set-up for an amplification-based DDoS attack.

The Smurf Attack (or ICMP flooding attack) is a common example. In this attack, the adversary spoofs the IP address of the intended victim and sends a large number of ICMP Echo Request packets to the broadcast address of the (amplifier) network, which then sends these messages to all the hosts within the broadcast domain of the network. Every host, upon receiving the ICMP Echo Request packets, sends ICMP Echo Reply packets to the victim, thereby consuming its available bandwidth.

Reflection-based DDoS Attacks Similar to an amplification-based attack, an adversary can also exploit a reply-based protocol to potentially launch a 'reflection-based DDoS attack'. This consists of an attacker, a reflective network (or a set of reflectors) and a victim. A reflector can be any host that responds to an incoming packet by sending a reply to the source IP address of that packet [90]. Web-servers, mail servers and DNS servers are examples of reflectors, because they send reply packets such as SYN+ACK or ICMP Echo Reply in response to SYN and ICMP Echo Requests respectively. These attacks differ from amplification-based attacks, since the attacker in this case is required to make use of a set of hosts (reflectors) rather than a single host (broadcast address) in order to initiate a response and cause the desired effect. Figure 2.6 illustrates a reflection-based attack mechanism.

A DNS Reflector Attack is one example of a reflection-based attack. In this attack, the adversary first harvests a large number of DNS server IP addresses and then issues DNS queries with spoofed source IPs (instead of the sender's source IP, the source IP is spoofed to be that of the intended client), thereby resulting in a large volume of traffic (DNS replies) to the intended victims. The amplification effect of this attack is based on the fact the a small DNS query (60 bytes) can generate much larger response (512 bytes) from the name server, thus amplifying the outgoing traffic by a factor of 8.5. New RFC specifications require an even larger response from the name servers, thereby further increasing the amplification factor. While in the Smurf Attack the amplification was caused by packet(s) being sent to the broadcast address of a network, in a DNS Reflector Attack the amplification mainly occurs because of a mismatch between the query and response size, the response being significantly larger than the query.

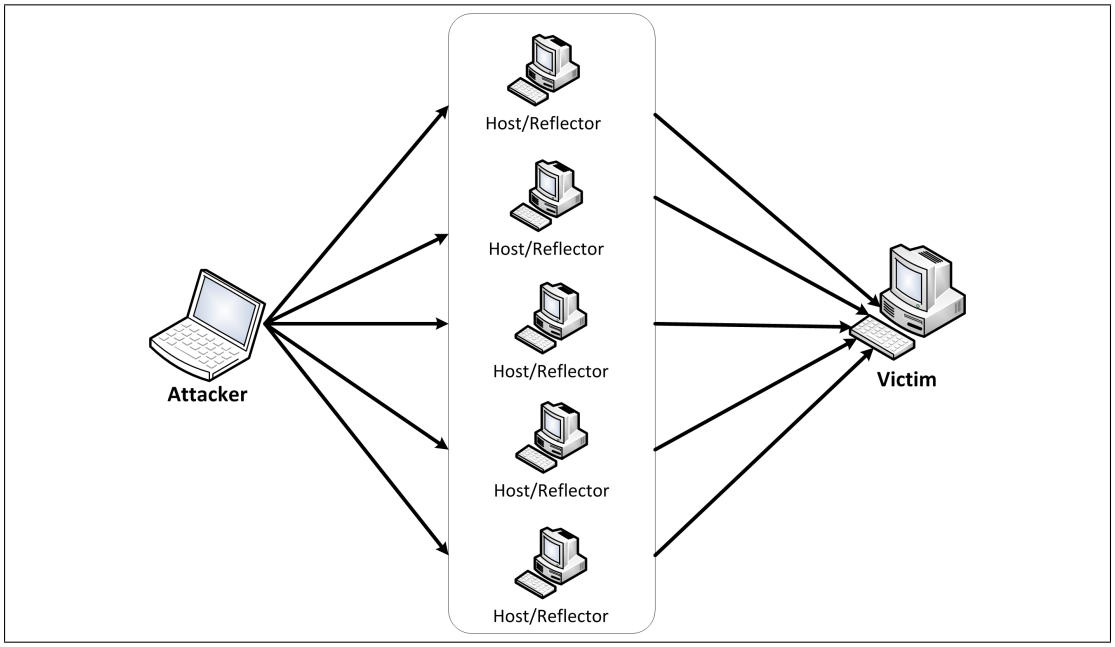


Figure 2.6: Reflection-based DDoS attack mechanism.

This attack was recently used against Spamhaus, a provider of anti-spam DNS-based Blocklists and Whitelists to generate allegedly the largest HRF attack yet measured [4].

An adversary, with the aid of a botnet, can employ one or more attack techniques, including amplification and reflection, to launch an attack against a target host. While the amplification technique is used to increase the magnitude of the attack, a reflection attack is generally used to conceal the attackers' (or rather the bots') identity and prevent them from being traced. These attack techniques, coupled with the distributed nature of botnets, makes such automated or semi-automated attacks extremely difficult to detect and mitigate.

2.2.1.2 Semantic Attacks

Semantic attacks exploit a design (or implementation) flaw in a communication protocol, or more often, in the application(s) installed on the victim host. This can make semantic attacks difficult to execute compared to brute force attacks, because they require an adversary to have a thorough understanding of the protocol or application being targeted. However, since semantic attacks are 'stealthy' in nature, taking advantage of *imperfections* at various levels, they can be successfully launched even with a disproportionate allocation of resources, in terms of bandwidth or processing capacity, between an attacker and a target. So a rel-

atively poorly resourced attacker can degrade the services (or potentially cause the shut down) of a much more powerful server.

The 'Ping of Death' is a classic example of a semantic attack [56]. It is executed by sending malformed ICMP (or ping) packets to the target. In this attack, the adversary sends oversized (larger than 65,535 bytes) ICMP packets instead of the normal 64 bytes. Most operating systems, released before 1997, could not efficiently handle packets of this size. Therefore, accepting such oversized packets would cause memory related (buffer overflow) issues and often end up crashing the target system.

The 'Teardrop attack' is another common example of a semantic attack [34]. This attack exploits a vulnerability in TCP/IP fragment reassembly, and is executed by sending malformed TCP/IP fragments with overlapping, oversized payloads to the target's machine. If the victim is unpatched for this exploit, the attack can easily cause the target system to crash.

A recently published application layer semantic attack exploits the Simple Object Access Protocol (SOAP) format which allows deeply nested Extensible Markup Language (XML) to be embedded into the message body [112]. When such a message is sent to a web-service provider, it forces the XML parser within the service to process the document, thereby causing memory exhaustion and leading to a DoS attack.

2.2.2 Targets of DDoS Attack

A DDoS attack may target an application, host, network or infrastructure [71]. DDoS attacks targeting different points of the communication channel may produce varied effects, but they all intend to cause a disruption of service, at some level, irrespective of their targets. For example, a DDoS attack targeting a network may differ, in the impact caused, from an attack targeting an application, but each of them intends to cause a degradation of the services being offered by the victim to its clients. In a network-based DDoS attack, the adversary may exhaust the connection bandwidth to the victim, thereby preventing legitimate clients from connecting and accessing its services. However, in an application-based DDoS attack, the adversary may send relatively few malformed packets and yet still disrupt the resource allocation strategies of the application under attack. Out of all possible targets of DDoS attacks – application, host, network and infrastructure – this thesis focuses on developing detection techniques

against HRF DDoS attacks that target networks and applications running on the victim host.

2.2.2.1 Attacks Targeting Networks

DDoS attacks targeting networks aim at consuming the incoming bandwidth of the target network by sending a large number of packets. An attacker launching DDoS attacks at the network layer can easily take advantage of source IP address spoofing to not only send attack packets in large numbers but also to prevent itself from being detected or traced. The prime focus of DDoS attacks targeting networks is on the *volume of traffic* being sent rather than the actual *content* of the packets, thereby consuming the network bandwidth. TCP SYN, UDP and ICMP flooding attacks discussed above (2.2.1.1) are some common examples of DDoS attacks targeting a network.

2.2.2.2 Attacks Targeting Applications

In recent years, DDoS attacks have not only increased dramatically in frequency and size, but have also become more sophisticated, with the focus slowly shifting from more general network-based attacks to more specific application-based attacks, targeting the security holes at layer 7 or the application layer of the TCP/IP stack [82].

Application layer DDoS attacks operate by exploiting the design or implementation level flaws of the applications running on the victim's machine, thus preventing its legitimate clients from accessing them. Application layer attacks can be low enough in volume not to trigger any 'volume based' alerts and are usually launched with 'seemingly legitimate' attack packets, thus making them difficult to detect. HTTP Request Attacks and DNS Amplification Attacks are some of the popular examples of DDoS attacks targeting applications.

2.3 DDoS Attack Detection

As discussed in Section 2.2, DDoS attacks come in various guises and constitute an extremely pernicious threat within the Internet domain [81]. Notwithstanding the amount of research done in the past decade, early and accurate detection of DDoS attacks remains a tough challenge. Consequently, the investigation of techniques and processes for efficient and timely detection of such attacks

continues to be an active area of research [92]. The primary focus of this thesis is the investigation, development and evaluation of techniques for detecting a wide variety of DDoS attacks. DDoS attack detection has predominantly followed two broad directions: network traffic analysis and SNMP MIB data analysis. A selection of the most pertinent literature related to each of these two DDoS detection strategies is now examined.

2.3.1 Network Traffic Analysis Based DDoS Detection

The very nature of HRF attacks suggests that one of the focal points for developing attack detection techniques should be the detection of *anomalies in the incoming network traffic*, with the intention of efficiently identifying traffic likely to constitute a DDoS attack. One of the key components of network traffic, apart from traffic volume, is the source IP address of the packets. Consequently, there is a distinct body of knowledge surrounding the use of source IP addresses, among other techniques, for detecting the onset of anomalous network activities such as a DoS attack [15, 36, 47, 91, 93, 113], and in particular those in which the attacker uses 'spoofed' source IP addresses [54]. The majority of these techniques focus on detecting DDoS attacks without considering FEs, although there are some approaches that take FEs into account, by tagging them as an abnormal activity for further analysis and characterising them in an attempt to facilitate their differentiation from DDoS attacks [15, 54]. A detailed discussion of these techniques is presented in Section 2.4. This section presents an overview of the existing DDoS attack detection techniques based on the analysis of the incoming network traffic.

An integral part of any attack detection process is the need for a feature or a set of features that capture the inherent characteristics of an attack, which appear more frequently in the so-called attack class, but are less prominent in the non-attack or normal traffic class [92]. In addition, in order to cater for continuously changing attack vectors, it is important to identify and use those features that an attacker is less likely to change [92].

A mechanism to detect SYN flooding attacks using Bloom filters was proposed by He et al.[49]. Their proposed mechanism maintained a list of client IP addresses using a Bloom filter. If a SYN request from a given client appeared in the network traffic being monitored, a corresponding counter was incremented. However, if a SYN+ACK packet originated from the same client, the same counter

was decremented. Thus, by checking these counters being maintained for each client, their proposed technique could detect SYN flooding attacks. A similar approach was adopted by Wang et al.[121], using the ratio of the number of SYN and FIN+RST packets (closing the connection) as the detection metric for detecting SYN flooding attacks. During such an attack, the number of SYN packets would be significantly greater than the FIN+RST packets, since the attacker's intention is to keep a large number of half-open TCP connections on the victim's machine, thereby exhausting both the networking (bandwidth) and computing (memory) resources.

Over the years other IP address related features have also been used to indicate the occurrence of DDoS attacks, including some basic features such as the traffic volume per IP address [47] or a change in the number of distinct network flows i.e. a unique combination of source IP, destination IP, source port, destination port, and the protocol [15]. Statistical based approaches on IP addresses have also been used for detection purposes. Feinstein et al.[43] used the distribution of source IP addresses as a detection criterion. Their approach assumed that the distribution of source IPs during an attack is uniform, which can be detected using entropy and Chi-square statistics. The experimental results they presented show a dramatic increase in the Chi-square value during DDoS attacks. The detection threshold used in their proposed technique was, however, dependent on the underlying statistical result and thus required modification to detect different types of attack.

Two important aspects associated with IP address based DDoS detection strategies are: the practicality of storing the statistical information for each of the 2^{32} elements in the IPv4 address space, and the need to accommodate a sharp increase in the rate of arrival of new source IP addresses during an attack [92]. This necessitates minimising the number of IP addresses being monitored, and optimising the way in which information about IP addresses is stored. Gil and Poletto [47] used a dynamic 4-level 256-ary tree (MULTOPS) to store traffic data for each IP address. Peng et al.[91] pointed out that during a heavy traffic scenario such as a DDoS attack, such a data structure could itself succumb to a memory exhaustion attack. Hence Peng et al. and Takada et al.[91, 92, 113] stored only those IP addresses which either completed a three-way TCP handshake or crossed a pre-defined threshold for the minimum number of packets originating from an IP address.

Peng et al.[91] used history based source IP address filtering at the edge router to detect a DDoS attack. Their proposed mechanism maintains a historical database of all valid IP addresses i.e. those completing the three-way TCP handshake. This database is updated using a sliding window in order to store the most recent addresses. Whenever the edge router gets overloaded, the IP Address Database (IAD) is then used to decide whether to accept the incoming packets. During an attack, only packets originating from source IP addresses present in the database are allowed access. However, the IAD can be corrupted by those source IP addresses which first complete a three-way handshake and later on participate in the attack.

Clustering of IP addresses has also been used by researchers for DDoS detection. Le et al.[63] used IP aggregation to detect DDoS attacks and differentiate them from FEs. Their technique was based on the premise that during an FE, most of the sources are close geographically to each other, whereas in case of DDoS attacks traffic originates from widely distributed sources. An intrinsic challenge associated with IP based attack detection techniques is the scalability of the solution when applied to the IPv6 address space where the number of addresses is 2^{128} as compared to 2^{32} for IPv4.

An orthogonal attack detection strategy to network traffic analysis is monitoring the target server, usually via SNMP, for various MIB variables, and using that information to detect an anomaly. An overview of such techniques, when applied for detecting DDoS attack, is now presented.

2.3.2 SNMP MIB Data Analysis Based DDoS Detection

The second area for DDoS detection that has evolved has been statistical analysis of Management Information Base (MIB) data collected via Simple Network Management Protocol (SNMP) agents. This integrates existing detection systems, like an Intrusion Detection System (IDS) with SNMP-based Network Management Systems (NMSs), to detect the onset of a DDoS attack [29, 96].

The Simple Network Management Protocol (SNMP), as defined by the Internet Engineering Task Force (IETF), is a standard for network management activities [32], and is commonly used for monitoring networking equipment such as servers and routers. An SNMP-managed network consists of three main components: a managed device being monitored, an SNMP agent (a daemon or service running on the managed device), and an NMS (an application running on

the system/manager monitoring the managed device). The SNMP agent hosts a Management Information Base (MIB) which can be queried by the NMS. MIB is a tree structured database containing information (state and configuration) about the agent and the monitored objects, identified by their Object Identifier (OID). MIB objects with similar characteristics are aggregated to form MIB Groups.

Cabrera et al.[29] proposed a methodology for using an NMS for early detection of DDoS attacks. Their work was solely dependent on using MIB traffic variables collected from the systems (attacker and the target) participating in the attacks. The attack datasets used for analysis were synthetically created using TFN2K and Trin00 tools [38]. The proposed methodology used 91 MIB variables corresponding to five groups – `ip, icmp, tcp, udp` and `snmp`, collected at a 5 second sampling interval for 2 hours, and was able to detect ICMP and UDP flooding attacks.

Streilein et al.[110] proposed a flooding-based DoS attack detection technique using data polled from Remote Monitoring (RMON) capable devices. RMON is a special purpose MIB for SNMP which tracks low-level network usage indicators such as packet size, packet transmission rate, data rate etc. The attack detection capabilities of their technique relied on the RMON capabilities of the existing infrastructure, such as switches and routers, instead of special purpose hardware. Using four months of RMON data collected from a switch in a production network and superimposing it with the simulated attack data, their technique was able to differentiate the attack traffic from the normal data using packet count, packet size, and the ratio of packet size to packet count, as the detection parameters.

Park et al.[87] used 16 MIB objects from six groups – `system, interface, ip, tcp, udp` and `icmp`, to propose a lightweight and fast detection technique for flooding attacks. The MIB, belonging to the `system` group – system.sysUpTime – provided measures of data rate (bits per second) and traffic rate (packets per second), rather than system load. In their detection algorithm, a correlation of the SNMP MIB data was performed and compared against pre-defined constant thresholds to determine if the traffic is completely normal or possibly abnormal. Stacheldraht [40] was used to generate flooding attack traces and their technique was able to detect a TCP-SYN flooding attack 8.3 seconds after its start.

Bao [14] and Yu [129] used the MIB variables presented in Table 2.1, and proposed a fast, lightweight, hierarchical SVM-based DDoS attack detection mechanism. At the first level, a one-class SVM was used to distinguish attack traffic

Table 2.1: Commonly used MIB variables.

MIB Group	SNMP MIB Object
ip	ip.ipInReceives
	ip.ipInDelivers
	ip.ipOutRequests
	ip.ipOutDiscards
tcp	tcp.tcpAttemptFails
	tcp.tcpOutRsts
udp	udp.udpInErrors
icmp	icmp.icmpInMsgs
	icmp.icmpInErrors
	icmp.icmpInDestUnreachs
	icmp.icmpOutMsgs
	icmp.icmpOutErrors
	icmp.icmpOutDestUnreachs

from normal traffic, and at the second level the attack traffic was separated into different attack types, including TCP-SYN flood, UDP flood and ICMP flood. The proposed mechanism could efficiently detect both known and novel attacks, and the results obtained from analysing MIB data in conjunction with IDSs were comparable to results achieved by analysing the traffic data.

A key observation drawn from the reviewed literature related to SNMP MIB data analysis based DDoS attack detection is that all SNMP based work deals only with traffic features, and does not take into account system load features.

2.3.3 Change Detection Techniques

Once the attack vector has been identified, the next challenge is to decide upon a suitable technique that can use this information to detect the onset of a malicious activity such as a DDoS attack. In intrusion detection systems, two widely used strategies for detecting malicious activities are: misuse or signature-based detection and anomaly-based detection.

A misuse-based detection uses a signature repository of known malicious activities and uses it to compare with the traffic derived signature to identify the occurrence of an attack or intrusion. While signature based techniques are suitable for detecting known attacks, they have limitations in detecting new or previously unknown attacks, such as zero day attacks, whose signatures are not present in

the signature repository. Bro [89] and Snort [100] are some of the widely used signature-based intrusion detection systems.

An anomaly-based detection strategy, on the other hand, relies upon the normal profile of the system being monitored, and using this profile as the benchmark for differentiating normal from anomalous activity. The normal profile is generally learned from historical data, which is then used to detect anomalous activities, commonly indicated by a noticeable deviation from normal activity. In contrast to the signature-based detection strategies, anomaly-based detection is able to detect new attacks, however, it can also potentially suffer from a relatively high false detection rate i.e. normal behaviour being classified as abnormal behaviour. An important component of these detection approaches is change detection [60].

Any occurrence of anomalous activities, including DDoS attacks and FEs, is also accompanied by a change in the statistical properties of the observed parameters. Hence, the problem of detecting such anomalous activities can be formulated as a change detection problem where the aim is to detect a change in the observed parameters with minimal delay and false detection rate [65, 114, 115]. Detecting changes in the statistical properties of various parameters has been extensively studied and applied in different domains such as network traffic, image processing, seismology, and financial analysis. The book by Basseville et al. [16] provides an overview on the various change detection techniques and their applications. Amongst all different techniques that have been used to detect changes, Cumulative Sum (CUSUM), moving average, and spectrum analysis are some of the most commonly used for detecting anomalous network activities such as DDoS attacks.

CUSUM was first studied by Page more than 50 years ago [85], and thereafter has been applied to various problem domains including DDoS attacks. As its name indicates, the CUSUM technique involves calculating the cumulative sum of a sequence of reading of a given parameter. For detecting changes, a sum of the difference between the actual and expected parameter values of a sequence, also known as a CUSUM value, is compared against a threshold value. A CUSUM value greater than the threshold indicates a change in the statical properties of the given parameter sequence. Carl et al. [31] performed a comparative study of three widely used DoS attack detection techniques – activity profiling, change-point detection, wavelet analysis – and found change-point detection to

be computationally less expensive and lighter in terms of memory utilisation, as compared to the other two techniques.

Bo et al. [23] proposed an application of non-parametric CUSUM[4] for detection of worm attacks. A time series of the number of distinct hosts (IPs) a given source attempts to connect to was used to estimate the CUSUM score, which was then subjected to a threshold test in order to identify an attack. Chan et al. [35] also used non-parametric CUSUM to identify worms that use a list of potential target IPs in order to propagate through a network. Every incoming source IP was assigned a weight and a non-parametric CUSUM was applied on the total weighted source IP count. Wang et al. [121, 122] applied a non-parametric CUSUM to detect SYN flooding attacks. The difference between SYN and FIN+RST was used as the detection metric to detect SYN flooding attacks with a high accuracy and small detection delay.

A variation of the moving average technique, Exponentially Weighted Moving Average (EWMA), has also been commonly used for detecting intrusions [127, 128] and flooding attacks [105]. (A detailed description of the EWMA-based change detection technique used within this research, in the work presented in Chapters 3 and 6, is described in Section 3.2.2 of Chapter 3.) EWMA was first studied by Roberts [99], and relies on determining whether the value of the observed parameter in a given time interval has exceeded a particular threshold. This threshold is calculated in an adaptive manner and is based on the mean value of the observed parameter, which in turn is computed from the recent observations of that parameter.

Paul [88] used EWMA for detecting flooding-based DoS attacks against web-servers. Münz et al. [80] compared the capabilities of Shewhart, CUSUM, and EWMA for detecting anomalies in network traffic. Instead of using a direct time-series of network traffic, a time-series of prediction errors was used in order to cope with seasonal variations and serial correlation. The backbone network traffic of an ISP was used for evaluation. The authors concluded that CUSUM, although generally favoured, does not perform any better than relatively simpler techniques (Shewhart and EWMA) when applied to time-series of prediction errors. Siris et al. [105] compared the use of an adaptive threshold algorithm and a parametric CUSUM for detecting TCP SYN flooding attacks. Real traffic

[4]A parametric CUSUM algorithm requires an underlying model or distribution, both for the pre-change and post-change periods, of the parameter being analysed. In cases where pre-change and post-change distributions are not available, a non-parametric CUSUM is used.

traces were used for a comparative analysis based on the detection probability, the false alarm ratio, and the detection delay. The authors concluded that while both the algorithms gave good results for high-intensity attacks (HRF flooding attacks), the parametric CUSUM exhibited better performance for low-intensity attacks, without being more complex.

The reviewed literature thus indicates the use of both CUSUM and EWMA as suitable techniques for detecting abrupt changes in the parameters being observed. Although, preliminary work related to Chapter 3, published in [8, 75], used a sliding-window-based non-parametric CUSUM proposed by Ahmed et al. [6, 7], a change detection technique using EWMA is presented in Chapter 3 and then used throughout the work described in both Chapters 3 and 6. EWMA was chosen mainly because it is robust, flexible and simple to implement as compared to a sliding-window-based non-parametric CUSUM, and does not require a somewhat arbitrary selection of a window size to suit different datasets. (In the case of Chapter 3, however, the experimental results based on CUSUM are presented in Appendix A for comparison).

2.3.4 Summary

Some key observations from the above literature review of network traffic and SNMP MIB based DDoS attack detection techniques are:

- source IP address appears to be a useful parameter to indicate the onset of a DDoS attack, either when used in isolation or in combination with other parameters [15, 36, 47, 91, 93, 113].

- the majority of the SNMP MIB based detection techniques use a rather fixed set of MIB variables, based only on the incoming and outgoing network traffic [14, 87, 129], as shown in Table 2.1). The use of this one-dimensional set of MIB variables limits the attack detection capability to network layer DDoS attacks.

- to the best of our knowledge, no attack detection technique has been developed or even proposed that uses load based features such as CPU and memory utilisation (Section 2.3.2). Like network traffic based features, these load based features can be collected by probing the victim machine at regular intervals via SNMP. The use of such load based features can po-

tentially not only detect a variety of DDoS attacks, including application layer attacks, but also differentiate them from FEs.

- the majority of attack detection techniques are tested and evaluated against either the more than decade-old KDD Cup 1999 Data [50], or datasets not available in the public domain (private datasets), or datasets generated using old traffic generation tools (such as TFN2K [109] and Trin00 [38]), most of which were written long ago and are now obsolete.

The majority of detection techniques discussed in this section focus on attack detection without considering FEs, which share a number of characteristics with DDoS attacks. The following section discusses some of the available techniques which detect FEs and distinguish them from DDoS attacks. An overview of work relating to FE models is also presented in the following section.

2.4 Flash Event Detection and Modelling

An intrinsic challenge in addressing the problem of DDoS detection is to effectively distinguish DDoS attacks from Flash Events generated by legitimate clients.

The term Flash Crowd (FC) was coined by Larry Niven in a 1973 science-fiction story to describe a situation where huge crowds of people went back in time to re-visit historic events [84]. In the World Wide Web context, a Flash Event (FE) refers to the situation where a large number of legitimate users concurrently send requests to a web-server, either following some newsworthy event or as a result of redirection from popular websites like Slashdot or other social media. This can lead to a dramatic increase in incoming traffic and subsequently cause a degradation in service. Some examples of FEs include popular websites experiencing a surge in incoming traffic after major world events (e.g. earthquakes, 9/11 attacks), or during major national or international events such as sporting events or elections etc. An overview of the research conducted in detecting FEs and differentiating them from DDoS attacks is now presented.

2.4.1 Distinguishing Flash Events from DDoS Attacks

DDoS attacks and FEs represent anomalies in the Internet traffic that cause performance degradation. However, each results from a different section of the

web-community with varied intentions. DDoS attacks, executed by an attacker, are a deliberate attempt to disrupt a service intended for its legitimate clients. On the other hand, FEs are created by legitimate users of the web-community trying to access a popular web-resource with non-malicious intent. From a network administrator's point of view, a precise distinction between DDoS attacks and FEs is important because these events require an entirely different set of actions to be undertaken after they have been successfully identified. Detection of an FE requires an increase in the number of Content Distribution Networks (CDNs) and other load sharing mechanisms to accommodate more legitimate users [54]. Whereas, in the case of a DDoS attack, the network administrator needs to enable various attack mitigation mechanisms to filter out malicious traffic and continue uninterrupted access for legitimate clients.

DDoS attacks are usually executed using a set of geographically separated compromised machines (bots), controlled by a bot-master, except when the DDoS attacks have resulted from coordinated voluntary 'hacktivists' such as Anonymous. On the other hand, an FE originates from legitimate clients trying to access a web-resource. Therefore, this problem of differentiating DDoS attacks from FEs can be mapped onto the problem of differentiating bots (machines) from humans. Amongst all available techniques for achieving this, CAPTCHAs [9] have been heavily used. CAPTCHAs are graphical puzzles based on the premise that humans can solve them but machines or bots cannot. Kandula et al. [55] used these puzzles to build a system based on probabilistic authentication to protect a web-server. However, the use of such puzzles introduces additional delays for legitimate clients, and various puzzle-breaking mechanisms have now been developed which make it possible for compromised machines (bots) to break the visual CAPTCHAs [78].

Using private datasets containing HTTP traffic from two busy web-servers, Jung et al.[54] proposed an enhancement to CDNs to better assist web-servers during FE scenarios. One of the traffic traces used was from a play-along website for a popular TV show, and another from the Chilean election web-site, both being FEs. They first categorised FEs into predictable and unpredictable, based on whether the web-site is aware beforehand of the possibility of its occurrence. Their work further characterised FEs in terms of traffic patterns, client characteristics, and file references. Using the above mentioned predictable FE datasets, their work concluded that FEs (and DDoS attacks) are accompanied

by a dramatic increase in the number of incoming requests, and a decrease in the per-client request rate. These results deviate from the analysis presented in this thesis, which shows that even in the case of a predictable flash event, the intensification in the incoming network traffic to the victim can be gradual rather than abrupt. It is also reasonable to suppose that, in an unpredictable FE, this parameter might behave differently. The lack of datasets representing FEs, both predictable and unpredictable, contributes to such speculation. Hyund et al.[86] have used the FE characteristics proposed by Jung et al.[54], and have reported the use of randomness checking to differentiate FEs from DDoS attacks. However, their results, as in the case of [54], are based on datasets of predictable FEs not available in the public domain.

Yi et al.[125] suggested a novel hidden semi-Markov based anomaly detector for detecting and differentiating shrew HTTP flood attacks from Flash Crowds by monitoring the file popularity of the web-server. A shrew attack is a low-rate DoS attack which sends 'legitimate appearing' requests at a sufficiently low rate to elude detection by counter DoS techniques [61]. Their proposed technique used Principal Component Analysis (PCA) and Independent Component Analysis (ICA) to abstract the multivariate observation vector. However, one disadvantage of their model, as acknowledged by the authors, was that the proposed scheme was unable to differentiate and filter out malicious sources from normal ones, so that their model merely serves as an alert function to trigger more complex monitoring mechanisms.

Le et al. used discrete wavelet transform on source probability, source IP address distribution map, source IP address aggregation and request volume intensification to differentiate FEs from spoofed DoS attacks. Yu et al.[130] proposed a technique to distinguish between a DDoS attack and a Flash Crowd based on flow similarity. They define flow as the packets passing through the same router which have a common destination address. Their technique is based on the premise that there is a stronger similarity between the flows of a DDoS attack as compared to those in flash crowds. They used three abstract distance metrics (Abstract distance, Jeffrey distance and Sibon distance) to measure and compare the flow similarity among DDoS and flash crowd flows based on the number of packets (presumably packets per flow). Their research shows that Sibon distance is the most suitable metric to measure the flow similarity. When tested on real-datasets, the proposed algorithm produced a differentiating accuracy of around

65%.

The following section summarises the recent research conducted in modelling FEs.

2.4.2 Flash Event Modelling

In order to evaluate the efficiency of a DDoS detection and FE separation technique, the availability of representative datasets, both for DDoS attacks and FEs, is needed. However, the extreme scarcity of publicly available datasets hinders the requisite research in this area. For the FE domain, only a very limited number of datasets representing different types of FEs are publicly available. The majority of these datasets are web-server logs in Common Log Format (CLF) which makes them difficult to use for experimentation, e.g. by replaying them over a network using the *tcpreplay* utility in order to test various FE and DDoS detection mechanisms, or by measuring their effects (such as CPU, Memory and Bandwidth utilisation) on the target server. While this lack of representative datasets is limiting, there is nonetheless sufficient data available to make useful progress in developing models of FE traffic and using them to generate synthetic FE traffic, which can then be used for evaluating the proposed techniques (as is done in Chapter 4 of this thesis). In this section, some of the related work done in modelling FE traffic is presented.

A significant amount of research has been conducted in workload characterisation of web-servers [13]. However, much of this work has been directed towards understanding typical web-traffic behaviour to improve caching and content distribution capabilities [54]. Workload analysis of web-servers during extremely heavy traffic situations, as in an FE, is comparatively less researched.

Ari et al.[11] proposed a model for Flash Crowd traffic consisting of three phases: a *ramp-up phase*, a *sustained traffic phase* and a *ramp-down phase*. Their proposed model focused on the time duration of each phase rather than on the shape of the curve. Each of the three phases in the proposed model is controlled by the *shock_level*, a parameter used to define the order of magnitude increase in the average request rate, and assumes a linear increase and decrease in the incoming traffic. The proposed model defines each of the three phases as:

$$l_1 = \frac{1}{\log(1 + shock\_level)}$$

$$l_2 = \log(1 + shock\_level)$$

$$l_3 = n \times \log(1 + shock\_level)$$

where l_1, l_2, and l_3 represent the duration of the ramp-up phase, the sustained phase and the ramp-down phase respectively. Their proposed model appears overly simplistic as the behaviour is likely to vary significantly between different types of FE e.g., the traffic seen during *Slashdot triggered FEs* tends to increase quickly by a large amount and then fade off slowly [10]. Such FEs occur when short news articles are posted along with a Web link redirecting the reader to another web-server containing a full description of the associated story, and the receiving web-server can be quickly overwhelmed. In addition, a sustained traffic phase or 'plateau period' can be extremely short-lived and is usually orders of magnitude smaller than the other two phases – the ramp-up phase and the ramp-down phase. Analysis of the real-world FE datasets, available in the public domain, presented in Chapter 4 (Section 4.3) shows the absence of any such plateau period.

Wendell et al.[123] analysed FC traffic (they have used the term FC instead of FE) based on the data collected from CoralCDN, an open CDN made up of globally cooperating proxies [44]. They defined FCs as successive intervals of time for which the rate of incoming requests to a fully-qualified domain name increases exponentially. In other words, they say that a domain experiences a FC if

$$r_{t_i} > 2^i \cdot r_{t_0}, \quad \forall i \in [0, k]$$

and

$$\max_i r_{t_i} > m \quad \wedge \quad \max_i r_{t_i} > n \cdot r_{avg}$$

where r_{t_i} is the average per-minute request rate over a time period t_i. Constants m and n are defined as the minimum per-minute request rate, and the difference between the average and maximum sustained request rate respectively. The proposed definition of FC traffic and the choice of constants ($m = 500$, $n = 15$ and $k = 3$), presented in the paper, appeared to be ad hoc, lacking supporting arguments. Their work focuses on different aspects of online content service delivery, rather than on developing a detailed model of FEs, which can then be used to synthetically generate similar network traffic, as in this thesis.

Bodik et al.[24] analysed five datasets for volume and data spikes, and observed variations in steepness, magnitude, duration and spatial locality. The

datasets used for the analysis were: World Cup 1998, UC Berkeley EECS Website, Ebates.com, Wikipedia.org, and a partial source of data from Twitter. The analysis was used to create a workload model, which served as an input to a closed-loop workload generator to synthesise similar workload traffic in a test environment. The volume spike in their proposed model was parametrised into: t_0, t_1, t_2, t_3, and M, representing the start-time of the spike, the peak-time, the end-time of the period with a flat workload, the end-time of the spike, and the constant M defined as the relative increase in the workload volume. The workload volume profile was multiplied by a factor c_t to represent a new profile during a spike. The multiplication factor c_t was set to 1.0 before time t_0. It linearly increased from t_0 before reaching M at time t_1, then remained unchanged between t_1 and t_2, then started to decrease linearly between t_2 and t_3, and stayed at 1.0 after t_3. Hence, their workload volume model was very similar to the one proposed by Ari et al. [11] i.e. a linearly increasing ramp-up period, a sustained period, and a linearly decreasing ramp-down period of the traffic during an FE. In addition, their proposed workload volume model was validated against a UC Berkeley EECS Website dataset, which is not available in the public domain.

Zhang et al.[131] presented a model for the *magnitude* of FCs in BitTorrent where the service capacity increases with its peers, characteristically different from FCs seen on web-servers. They define a BitTorrent FC as a significant increase in the number of peers in a *swarm* – a group of peers sharing a torrent. Their proposed FC model consists of four components: the *arrival time* (time between the creation of swarm and the start of an FC), the *duration* (time between the start and the end of the increase in the number of peers), the *plateau period* (duration for which the number of peers remains fairly constant) and the *magnitude* (indicating the significance of an FC in terms of the increase). Their model resembles the model proposed by Ari et al. [11] in terms of having a distinct plateau period. The magnitude of the proposed FC model was defined as:

$$M = \frac{\Delta n}{\Delta t} \times \frac{\Delta n}{\max(n_b, c)}$$

where Δn and Δt represent the increase in the swarm size and the duration of the FC (in minutes) respectively. n_b is the swarm size at the start of an FC and c is the seeder capacity i.e. the number of peers a well-provisioned seeder can serve in a swarm. The proposed model, however, takes into account the fact that the service capacity of a Bit Torrent swarm increases with its peers. This

characteristic property of FCs seen in BitTorrent is different from the FCs seen on web-servers, which have finite serving capacities. Hence, the proposed model was not useful for modelling FEs defined as a part of this research.

2.4.3 Summary

The above research highlights:

- differences in the various models proposed that need to be resolved, and
- the difficulties in comparative evaluation of different FE models due to the lack of available datasets

The latter, and a similar lack of publicly available DDoS datasets, has prompted the work described in this thesis (Chapter 5) on synthetically generated traffic to provide traffic datasets that may be used for evaluation purposes.

2.5 Synthetic Traffic Generation

Legal and privacy issues associated with sharing captured data are the key underlying reasons for the relative lack of real datasets for use in evaluating DDoS detection techniques and traffic models for DDoS attacks and FEs. This section reviews current traffic generation techniques after first detailing the limitations of current publicly available traffic datasets.

2.5.1 Limitations of Existing Public Domain Datasets

The KDD Cup 1999 Data is one of the most referenced and possibly the only reliably-labelled dataset available in the public domain [50]. This is the dataset used for the 3^{rd} International Knowledge Discovery and Data Mining Tools Competition held in conjunction with the 5^{th} International Conference on Knowledge Discovery and Data Mining (KDD99). The competition was to develop a network intrusion detector and the dataset contained a wide variety of intrusions, simulated in a military environment, that had to be audited. However, due to its age (1999) and other limitations e.g., its prime usage is for evaluating signature-based intrusion detection systems [70, 116], it is not really suitable for testing DDoS detection and mitigation mechanisms.

Other widely referenced datasets within the DDoS attack detection domain are the network traces from Waikato Internet Trace Storage Project [1]. However, these datasets have their own limitations: IP addresses are pseudonymised, packets are truncated 20 bytes after the transport header, and any retained payloads of UDP packets (except the DNS) are zeroed. A more recent addition has been the 'CAIDA DDoS Attack 2007 Dataset' containing an approximately one hour traffic trace of a DDoS attack which aimed at consuming the computing resource of the targeted server [51]. However, the available traces have had their IP addresses pseudonymised using CryptoPAn and their payloads removed, thereby limiting their usability. In the FE domain, a very limited number of datasets representing different types of FEs are publicly available. The majority of these datasets are web-server logs in Common Log Format, thereby making them difficult to use, e.g. it is not possible to replay them over a network to test FE detection algorithms.

Besides these specific limitations, these datasets also share some characteristics which greatly limit their usability:

- Excepting the CAIDA DDoS Attack 2007 Dataset, the other datasets are old and obsolete in the context of continuously evolving DDoS attacks with complex attack vectors.

- Barring KDD Cup 1999 Data, the datasets are not labelled, thus making it difficult to filter out the attack traffic from the background traffic, precluding detection strategies relying on machine learning algorithms.

- Even though the target addresses of the packets in these datasets can be altered and subsequently directed to a test machine while replaying, the real effects of the network trace still cannot be reproduced due to the non-availability of the services addressed in the original trace.

- The missing payloads completely eliminates the possibility of developing any *content-based* analysis technique.

Not all the aforementioned deficiencies affect the research presented in this thesis, but several of them do, and hence there is a need to investigate the problem of synthetic traffic generation.

2.5.2 Existing Traffic Generators

A number of synthetic traffic generation techniques have been implemented on various software and hardware platforms. The hardware traffic generators can do much the same as their software counterparts, but at higher speed and at greater cost. Hardware traffic generators like the SmartBits 600 [106] can artificially generate configurable flooding attacks, but they fail to interact with the application at the target machine. Preliminary experiments with the SmartBits generator highlighted significant differences in how the computers and the other networking devices like switches and routers responded to hardware-generated and software-generated network traffic. The number of packets being dropped by the kernel was found to be much higher for hardware-generated network traffic. On the other hand, most of the software-based traffic generators available in the public domain are based on simple client-server models and are limited by the number of different scenarios they are capable of simulating. Moreover, most of these software tools were written long ago and thus need serious modifications before they can be of any practical use.

One weakness shared by hardware traffic generators and a majority of software based traffic generators is their inability to conduct any meaningful interactions with the target. Their sole aim is to deposit packets onto the wire that conform to the given traffic profile, as expressed by parameters like traffic volume, traffic distribution, packet size, and protocol type. The following open-source software traffic generators were investigated: D-ITG [25], Harpoon [107], fudp[5], Hping [102] and curl-loader [53]. Of these, fudp is used in work described in Chapter 3 for injecting malicious packets in the normal background traffic to test a DDoS attack detection technique based on the change in rate of new source IPs. The functionality of the fudp tool was improved to incorporate the use of random source IP addresses. Apart from fudp, Curl-loader [53] was the other tool that was extensively tested and used to conduct preliminary experiments for generating datasets with the desired characteristics. The investigation of Curl-loader also led to the use of IP aliasing with Wget [83], (both available on most computing platforms) for synthetic traffic generation. IP aliasing is a well known technique for assigning multiple IP addresses to a single network interface and Wget is a command line utility for retrieving files, commonly via HTTP.

[5]fudp – http://sourceforge.net/projects/usoft/

2.5.2.1 Curl-loader

Curl-loader is a C-based open-source tool designed to simulate the behaviour of a large number of HTTP/HTTPS and FTP/FTPS clients, each with its own unique source IP address. Curl-loader has the ability to generate large numbers of 'virtual clients' (VCs), each having their own valid unique source IP and shared MAC address. In order to generate synthetic network traffic, comprising both malicious and normal data on the experimental testbed infrastructure, the scalability of this approach was tested and was seen to be governed by (a) processing capacity of the individual machines (i.e. the number and speed of CPUs and the amount of memory); and (b) the number of distinct platforms available in the experimental testbed. Details of the experiments conducted using both these options are presented in Appendix B (Section B.1).

Curl-loader placed heavy demands on the host systems' resources (CPU and memory) and proved far from an ideal tool for generating synthetic traffic originating from a large number of source IPs at the desired rate. The tool also had to be modified to incorporate randomness in the source IP addresses and to create various attack profiles. These modifications led to instabilities in performance. This, combined by the need for further modifications to incorporate different types of attacks, eventually led to the abandonment of this approach.

2.5.2.2 IP-aliasing with wget

IP aliasing is a well-known technique, available on most computing platforms, for assigning a large number of distinct IPs to a single hardware (Network Interface Card or NIC) address. On Linux and BSD, for example, thousands of unique IP addresses can be assigned to a single Ethernet card. IP-aliasing on the Linux platform can be achieved via the *ifconfig* utility. Using IP-aliasing, traffic originating from a single host can be divided into multiple *network-flows*[6], with each flow having a unique value for its source IP address, and so overall the traffic can be made to appear (to the target) to be coming from many separate machines.

GNU Wget [83] is a non-interactive command line utility for retrieving files using the HTTP, HTTPS and FTP protocols. As wget is an non-interactive tool,

[6]A TCP/IP network flow is defined as a sequence of packets from a source machine to a destination machine with unique values for source IP, source port, destination IP, destination port, and protocol.

it can be easily integrated with custom scripts, cron jobs and other command line utilities available in Linux without the support of X-windows. Therefore, when used with IP-aliasing, wget can potentially create the appearance of an array of machines ('bots' for DDoS attacks and 'legitimate clients' for FEs), each with a unique source IP address, but sharing the hardware address of the physical interface. This array of machines can then be used to synthetically generate realistic DDoS attacks and FEs.

The approach of using wget with IP-aliasing provided a method for crafting application layer DDoS attacks such as the HTTP flooding attack. However, the time it took to create aliases increased with the number of requested aliases. Also, an increase in the number of aliases resulted in a corresponding increase in the host machines' CPU and memory consumption. Experimental results of using IP-aliasing with Wget are provided in Appendix B (Section B.2). In addition to resource consumption constraints, the approach was also limited by the variety of attacks (only HTTP based) that it was capable of generating, thereby lacking the desired flexibility. The perceived drawbacks with this technique and with the use of curl-loader led to the development of a customised software traffic generator called Botloader, which was cooperatively developed as a part of this research, and is discussed in more detail in Chapter 5.

2.5.3 Testbed Design Strategies

An essential requirement for replaying synthetic or original traffic traces is having an experimental setup to play it on. To obtain experimental results also requires some means of managing an attack and measuring its effects. These requirements thus imply the existence of some kind of testbed in addition to the traffic generation software. The reviewed literature in this field suggests that there are three commonly used testbed design strategies, viz. simulation, emulation and direct physical generation.

2.5.3.1 Simulation

Network simulators like ns–2 [69] or Opnet [74] have often been used to simulate DDoS attacks and measure their effects [21, 101, 124]. However, the realism of a method in which the attackers, targets and network devices are all simulated has recently been called into question [72]. This lack of realism, combined with a slower speed of traffic replay, makes simulation an unattractive proposition

for addressing the DDoS attack detection techniques developed in this thesis, particularly for High Rate Flooding attacks.

2.5.3.2 Emulation

Emulation is a step forward in realism over simulation: real machines are used as attackers and targets. This approach gained traction with the Emulab [124], DETER [17] and Planetlab [94] testbeds. The DETER and Emulab testbeds allowed users to get access to desired machines, in a central and isolated (from the Internet) facility, whereas Planetlab, being a distributed testbed set-up, offers shared access to machines, via a VM (virtual machine) software, thus allowing user isolation.

In emulation only the network topology is recreated in software; targets and attackers are represented by physical machines, with a real Operating System (OS) and applications. The network is represented by soft-routers, and by virtual LANs. Although this is more realistic than the simulation approach, and allows the facility to be shared, the high cost of constructing and maintaining testbeds of this type seems to limit their size to around 300 machines. Real world DDoS attacks on the other hand, typically comprise at least several thousand attackers [98].

2.5.3.3 Direct physical generation

In this technique the desired network topology is built by physically arranging a network of routers, switches, and computers. The main disadvantage is lack of flexibility: each experiment has to be set up by hand, and the facility can't be shared. On the other hand, larger scale facilities can be built, although at greater cost. For example, Calvet et al. were able to create a network of 3000 virtual machines using only 98 blade servers [30]. The experimental testbed, cooperatively developed as a part of this research and explained in detail in Chapter 5 (see Section 5.2 for details), uses this approach as it allows a decoupling of hardware and software, and allows for more realistic experiments.

2.5.4 Summary

Research associated with network anomaly detection, particularly DDoS attack detection and FE separation, suffer immensely from accurate evaluation mainly

originating from the extreme scarcity of *recent* and *realistic* network data available in the public domain. Most of the publicly available datasets are either too old [50] to reflect the current network trends or are heavily anonymised (to eliminate any associated privacy and legal concerns) to give any useful information, or are available in a format like Common Log Format (CLF), which limits their usability [12]. The other obvious alternative to overcoming this scarcity of publicly available datasets, necessary for evaluating any DDoS attack detection technique, is the use of traffic generators. However, the majority of traffic generator tools, both hardware and software based, are only useful for flooding the connecting pipe with packets originating from spoofed source IPs rather than for generating stateful and responsive TCP traffic, suggesting that the synthetic generation of more realistic traffic may be one way around the problem.

The generated traffic also requires an experimental setup on which to be played. Directly building the desired network topology for experiments using physical networking equipment and computers provides greater realism over simulation or emulation approaches to testbed design, and can be set up with more modest equipment and computing resources.

2.6 Research Challenges

The focus of this thesis is to address some of the research challenges highlighted above, in particular challenges relating to the detection of DDoS attacks, the modelling of FEs, and the differentiation of DDoS attacks from FE traffic. In addition, the thesis addresses pre-existing work on traffic generation and its deployment for use in addressing these challenges. The challenges are:

1. **Extending previous work on the use of IP addresses for detecting DDoS attacks:** the literature reviewed indicates that the source IP address of incoming packets may be a useful metric, either in isolation or in combination with other traffic parameters, for identifying the onset of a DDoS attack. This leads to Research Question 1: *Can the onset of a DDoS attack be detected based on simple network traffic characteristics such as source IP address?* Chapter 3 describes the design, implementation and evaluation of an attack detection technique based on the change in rate of previously unseen or new source IP addresses of incoming packets.

2. **Distinguishing DDoS attacks from Flash Events and modelling Flash Events:** the majority of the attack detection techniques reviewed focus on DDoS attacks without considering FEs. A likely contributing factor to this is the general lack of publicly available real datasets of both DDoS and FE traffic. This leads to Research Questions 2 and 3, addressed in Chapter 4. Research Question 2: *Is it possible to differentiate DDoS attacks from FEs?* and Research Question 3: *What are the essential characteristics of different types of FE and is it possible to use those characteristics to model and subsequently generate different types of synthetic FE traffic?*

3. **Traffic generation and testbed framework:** The problems of the scarcity of publicly available datasets and the limitations of using off-the-shelf software and hardware traffic generators leads to Research Question 4: *Is it possible to design and implement a simple experimental traffic generation and testbed framework, using only modest hardware for the testbed and a software-only traffic generator, to synthetically generate realistic network traffic representing DDoS attacks and FEs?* This question is addressed in Chapter 5, which explores the design and implementation of the DDoS/FE testbed which includes a software traffic generator – Botloader, cooperatively developed as a part of this research.

4. **Ensemble-based DDoS detection strategy:** as previously mentioned, to the best of our knowledge, no DDoS attack detection technique yet proposed combines network traffic and server-load analysis. This leads to Research Question 5: *Can a more sophisticated ensemble-based technique based on both network traffic characteristics and server-load data (memory and CPU utilisation) improve the accuracy of DDoS attack detection and provide a means to identify and differentiate DDoS attacks, FEs and normal traffic?* Chapter 6 addresses this challenge by designing, implementing and evaluating an ensemble-based DDoS detection technique.

2.7 Summary

This chapter reviewed previous work relating to DDoS attacks. It examined current DDoS attack detection techniques, both from the network traffic analysis

side and SNMP MIB data analysis side, and identified their limitations. It also described the previous work done in the area of characterising FEs for detecting FEs and differentiating them from DDoS attacks, and presented an analysis of existing FE models and current practices for synthetic traffic generation.

Based on this survey of previous related work, four research challenges were identified for detecting DDoS attacks and distinguishing them from FEs. These challenges will now be explored in the following chapters.

Chapter 3

DDoS Attack Detection Using IP Addresses

It is now more than a decade since the first full-scale High-Rate Flooding (HRF) DDoS attacks were unleashed on the Internet community [45]. The vector for that attack was a set of compromised machines (bots), controlled by an attacker (bot-master) and used to direct high volumes of unwanted network traffic towards a target host. Even after a decade, and notwithstanding their conceptual simplicity, HRF DDoS attacks in various guises still continue to constitute a pernicious threat within the Internet community [81]. However, an efficient and timely detection of such attacks, combined with the formulation and implementation of an effective attack mitigation strategy, remains a challenging problem. This chapter attempts to provide a solution to the DDoS[1] attack detection problem, and outlines the design of a corresponding lightweight mitigation strategy.

The work presented in this chapter addresses Research Question 1 identified earlier in Chapter 1 (see Section 1.2 for details) i.e. *Can the onset of a DDoS attack be detected based on simple network traffic characteristics such as source IP address?* This chapter describes an approach based on the source IP address of the incoming packets, and demonstrates how that simple network traffic feature, coupled with a change-point analysis of the rate of arrival of new IP addresses, is sufficient in some circumstances to detect the onset of a DDoS attack. Importantly, the use of a single network feature directly addresses the scalability of

[1] In the context of this thesis, DDoS refers to High-Rate Flooding (HRF) attacks.

the detection process when subjected to high network speeds. The chapter concludes by presenting a scheme to download the legitimate IP addresses identified by the above detection strategy to an application firewall to provide a mitigation facility.

The chapter is organised as follows. Section 3.1 provides a high-level design of a source IP address based DDoS attack detection approach. This approach comprises two main functions: *ipac* (for classifying IP addresses of the incoming packets), and *ddos* (to identify an attack). Section 3.2 provides the implementation details of the *ipac* function, using bit vectors, and the *ddos* function used in the attack detection technique. Section 3.3 presents the change detection algorithm used in the proposed technique. Section 3.4 presents the experimental results and evaluates the technique against different attack datasets. Section 3.5 presents the conceptual design for a DDoS Detection and Mitigation Model (D2M2). Finally, Section 3.6 summarises the research presented in the chapter and provides concluding remarks.

3.1 An IP-based DDoS Detection Strategy

The onset of a HRF DDoS attack is typically accompanied by an increase in the number of unsolicited packets originating from previously unseen or new source IP addresses arriving at the target host. A real-world example of a DDoS attack with this characteristic is the CAIDA DDoS Attack 2007 dataset – a popular dataset used by DDoS researchers containing approximately one hour of pseudonymised traffic traces from a DDoS attack [51].

The characteristics of this attack in terms of the rate of incoming traffic (left y-axis) and previously unseen or new source IP addresses (right y-axis) are presented in Figure 3.1. The figure shows a dramatic increase in the number of previously unseen IP addresses at the onset of the attack. This characteristic feature of a HRF DDoS attack forms the basis of the proposed DDoS attack detection approach, namely, to analyse the incoming traffic in terms of the source IP addresses of the packets received, and to reliably determine if the current incoming network traffic represents a DDoS attack.

Figure 3.2 illustrates the proposed DDoS attack detection approach, cooperatively developed as a part of this research[2]. It consists of two high-level functions

[2]While the other work presented in the remainder of the thesis is primarily the candidate's

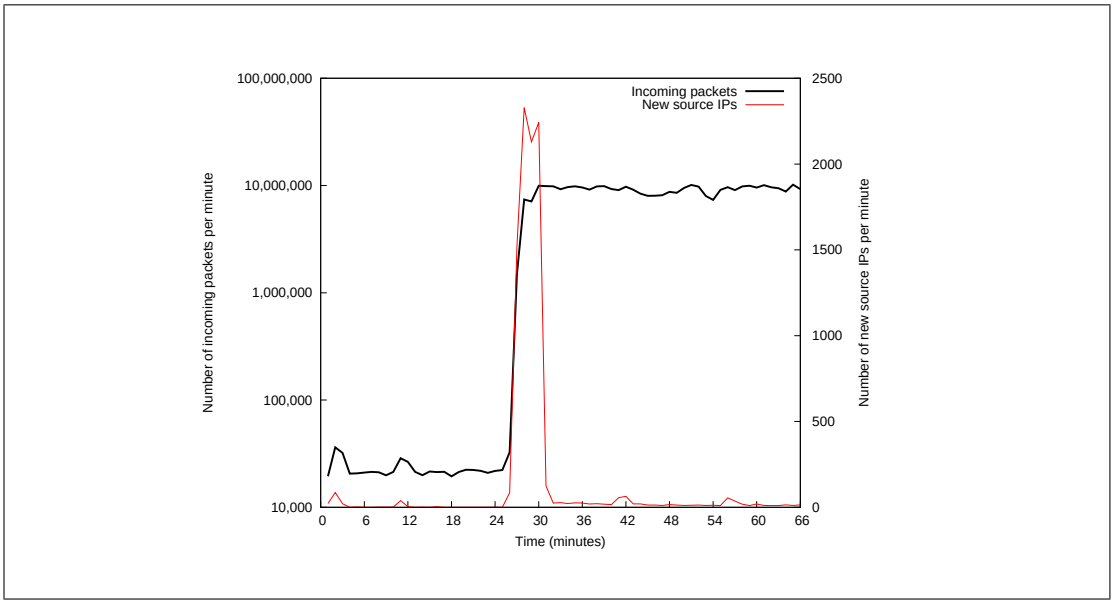


Figure 3.1: Rate of incoming traffic and new source IPs for CAIDA dataset.

– *ipac* and *ddos*:

- *ipac* function for classifying the source IP addresses of the packets received
- *ddos* function for identifying an attack

The *ipac* or IP address classification function first extracts the IP addresses of the incoming network traffic. It then determines whether the source IP address has been seen previously or is new (Not Seen Previously – NSP). The resulting time series of the rate of previously unseen or new IP addresses is then is analysed by the *ddos* function to identify whether the system is under attack. The system being protected against DDoS attacks is represented in terms of two independent states: **NA** (not under attack), which is the system state when receiving non-attack or normal traffic, and **A** (under attack), the system state when receiving DDoS attack traffic.

In order to learn the network behaviour under normal (non-attack) conditions, the *ipac* function is first applied to normal network traffic without the activation of the *ddos* function. After the training period, the *ddos* function is invoked at regular intervals (1 to 10 seconds), and based on the rate of arrival of packets from previously unseen IP addresses (as calculated by the *ipac* function), the *ddos* function then determines if the system is under attack.

contribution, the work presented in Sections 3.1 and 3.2 of this chapter was in particular a cooperative effort.

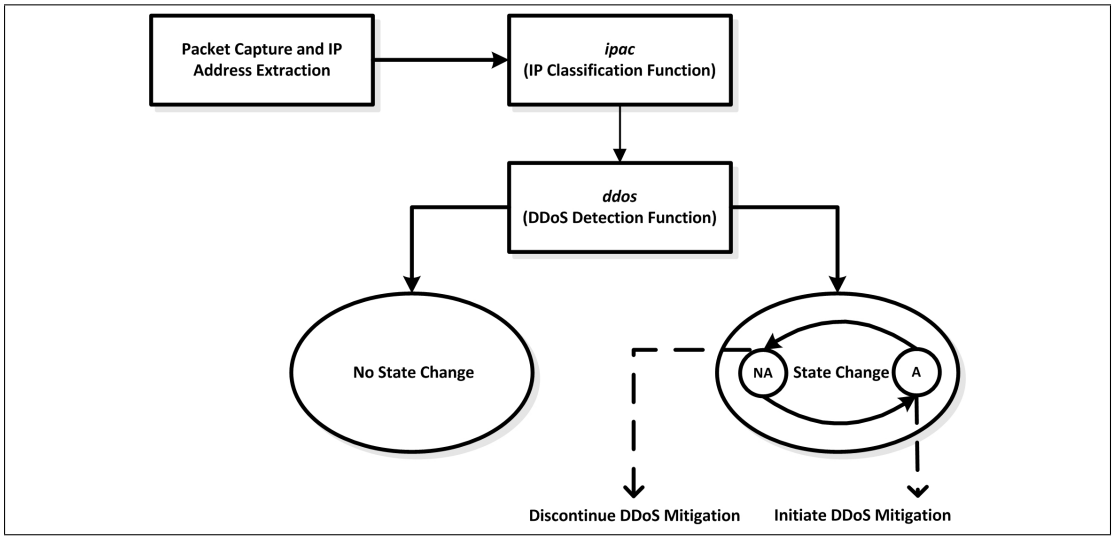


Figure 3.2: DDoS Detection using NSP source IPs

A transition from state **NA** to **A** signals that an attack has been identified, and the *ddos* function then generates a 'white list', which can then be used for mitigation, for example, by only allowing traffic from IP addresses within the white list. On the other hand, a transition from state **A** to **NA** is indicative of the end of an attack, and can be the prompt for the relaxation of the mitigation strategy, for example, by ceasing to use the previously generated white list for traffic filtering. The focus in this section is however on detection, mitigation is discussed further in Section 3.5.

In order to perform the source IP address classification of the incoming packets, *ipac* maintains two data structures representing the IP addresses of the packets received: $\boldsymbol{W}$ (White-list) and $\boldsymbol{R}$ (Recent). The data structure $\boldsymbol{W}$ is initialised with the source IPs of known attack-free or normal traffic i.e. legitimate clients. The data structure $\boldsymbol{R}$ is initialised to be empty at system start-up, and is used to temporarily hold the source IPs of the packets arriving from new (NSP) IP addresses. Only once the *ddos* function has recently determined that the system is not under attack can those addresses be copied to the white-list – the data structure $\boldsymbol{W}$. This avoids polluting the white-listed (legitimate) source IPs stored in $\boldsymbol{W}$. If, on the other hand, the *ddos* function determines that the system was recently under attack, then the data structure $\boldsymbol{R}$ is discarded.

The *ipac* function (described in Algorithm 3.1) analyses the source IP address of an incoming packet, and if the *ip* address is new or has not been seen previously, it updates the data structure $\boldsymbol{R}$.

Algorithm 3.1: *ipac* function for classifying IP addresses.

Input: Source IP address (*ip*) of the incoming packets.
Output: Previously unseen or new source IP addresses (*newIP*).

```
if NOT ((ip in W) OR (ip in R)) then
    INC(newIP)          /* increment the counter for new IPs */
    add ip to R                    /* update data structure R */
end
```

The *ddos* function (described in Algorithm 3.2) is periodically invoked, at 1 to 10 second intervals, to analyse recent changes in the arrival of packets from NSP IP addresses, as calculated by the *ipac* function. The *ddos* function calls the *StateChange* function, which in turn uses a change detection technique (see Section 3.3 for details) to detect abrupt changes in the rate of arrival of packets from NSP IPs, specifically in the change from the state **NA** to **A**.

Algorithm 3.2: *ddos* function for identifying attacks.

Input: Time series of the rate of new source IP addresses.
Output: System State – **A** or **NA**

```
if (in state NA) then
    if NOT (StateChange(NA)) then          /* no state change */
        add R to W          /* add IP addresses to white-list */
        else                                /* state change to A */
            state = A
            send white-list to the application firewall
            R = empty
    end
end
else if (in state A) then
    if StateChange(A) then                 /* change state to NA */
        state = NA
        inform the application firewall to stop using the white-list
    end
end
```

There are a few limitations to this approach. The first is that whenever the system is deemed under attack, new IP addresses are treated as malicious. This can give rise to false positives. In Chapter 6, the proposed ensemble-based DDoS detection technique aims to limit these false positives by extending the

rejection criterion of the incoming packets beyond the simple property of NSP IP addresses. It does this by making use of additional network-based parameters such as traffic volume, traffic volume per IP address, along with server-load based parameters such as CPU and memory utilisation, in tandem with NSP IP addresses to improve the identification of malicious traffic.

A second limitation concerns the malicious IP addresses observed during normal network operations. For example, if an attacker first performed reconnaissance of the target network by sending a small number of packets (such as ICMP echo requests) then these IP addresses would be added to the white-list, giving rise to false negatives, before actually flooding the target with a large number of packets. This can be addressed not only by using the additional network and server-load based features described above, but also by using smaller historical time periods, for example, by using only IP addresses observed during the last 24 hours as trusted IP addresses.

An additional limitation concerns the time taken by the execution of the *ipac* and *ddos* functions. The intention is to keep these two functions small and efficient. However, if they are slow, incoming packets may not be properly processed. Section 3.2 discusses the implementation of the NSP algorithm using bit vectors: a data structure that compactly stores the IP addresses as a bit-array.

3.2 Implementation of an IP-based DDoS Detection Strategy

This section describes the implementation details of the main components of the IP-based DDoS attack detection technique – IP addresses classification, and DDoS attack detection.

3.2.1 IP Address Classification

Using the change in rate of new source IP addresses as the primary feature to detect the onset of a DDoS attack requires keeping track of source IP addresses already seen in the network under normal, non-attack network conditions. At these times, the rate of arrival of new source IP addresses is relatively low, i.e. the target server experiences a small number of previously unseen (new) source IPs. On the other hand, the onset of a DDoS attack is generally marked by

a substantial increase in the rate of arrival of new source IP addresses [92]. Therefore, the implementation of the IP address classification function (*ipac*) requires the use of a compact and efficient data structure.

Bit Vector Implementation

In order to accommodate all the IP addresses in the IPv4 address space, a bit vector can be used as the data structure to represent $\boldsymbol{W}$ (White-list) and $\boldsymbol{R}$ (Recent) in the implemented NSP algorithm. In the IPv4 address space, in order to accommodate all of 2^{32} IP addresses using a bit vector, 0.5 GB of space is required. In contrast, a bit vector in IPv6 address space needs to represent 2^{128} IP addresses. This increase in the address size from 32 bits in IPv4 to 128 bit in IPv6 significantly increases storage requirements. One possible solution to the scaling problem in the *ipac* function would be to accommodate the larger IPv6 address space by using a more efficient data structure such as a bloom filter [22].

In the IP address classification function (*ipac*), whenever an incoming packet is received, a bit at a specified position in the bit vector is set to indicate the presence of a given IP address. A separate counter is then used to calculate the number of new source IP addresses observed during an interval, resulting in a time series of the rate of previously unseen or new IP addresses. This implementation via bit vectors allows for a number of optimisations to the proposed NSP algorithm. In the *ddos* function, the step 'add R to W' can be optimised by modifying the *ipac* function so that at the time of setting the appropriate bit in the data structure R, a temporary copy of the IP address value, or better still, a copy of the address of the bit being set can be kept. This optimisation allows the 'add R to W' step to be implemented as a series of bit operations rather than as an OR operation between the entire bit vectors $\boldsymbol{R}$ and $\boldsymbol{W}$. Another similar optimisation relates to the step 'R = empty' in the *ddos* function. This step is optimised by unsetting the relevant bits rather then zeroing the entire bit vector $\boldsymbol{R}$.

3.2.2 DDoS Attack Detection

As previously discussed in Section 3.1, the onset of a HRF DDoS attack is typically accompanied by an increase in the number of unsolicited packets arriving at the target host from previously unseen or new IP addresses. In such attacks, an

increase in the number of unsolicited packets usually results from a large number of compromised hosts (bots), controlled by a bot master, which send a high volume of traffic to the target host. As a result, the target host not only experiences an increase in incoming traffic volume but also an increase in the number of sources sending such traffic. Thus, the problem of identifying the onset of a DDoS attack can be formulated as a change detection problem. This involves detecting changes in the statistical properties of the incoming traffic parameters of interest, such as the change in rate of new source IP addresses.

The IP address classification function (*ipac*) identifies the number of new source IP addresses within a given measurement interval. It then calls the *ddos* function (change detection algorithm) to identify the onset of a DDoS attack. For detecting abrupt changes in the parameter under investigation i.e. the rate of new source IP addresses, a change detection technique called Exponentially Weighted Moving Average (EWMA) has been used, and is discussed in the following section.

3.3 Change Detection

At the onset of a network anomaly such as a DDoS attack, it is expected that the statistical properties of the parameters indicating anomalous activity may change abruptly. In order to detect such changes in the parameters being observed, various detection techniques have been proposed by researchers. As previously discussed in Chapter 2 (see Section 2.3.3 for details), Cumulative Sum (CUSUM), moving average, and spectrum analysis are some of the most commonly used techniques for detecting anomalous network activities. In the case of High-Rate Flooding (HRF) attacks, the reviewed literature also indicates the use of either CUSUM or Exponentially Weighted Moving Average (EWMA) as suitable techniques for detecting abrupt changes.

The EWMA change detection technique was chosen because of its simplicity, flexibility, robustness and effectiveness in detecting high intensity attacks [105]. It also has a lower false positive rate as compared to CUSUM for large dataset samples, or whenever the parameters are estimated [58]. Chapters 3 and 6 use EWMA to detect changes in the 'parameters of interest', indicative of the onset of a DDoS attack. In this chapter the parameter of interest is the number of new or Not Seen Previously (NSP) IP addresses. Although preliminary work,

published in [8, 75], used a sliding-window-based non-parametric CUSUM algorithm proposed by Ahmed et al. [6, 7] for change detection, EWMA has been used instead in this thesis. The two sets of results (EWMA, and CUSUM) are, however, closely comparable. The experimental results of attack detection of the same attacks used in this chapter, but based on CUSUM are presented in Appendix A.

EWMA Algorithm

EWMA relies on examining whether the value of the parameter being observed in a given time interval exceeds a particular threshold value. In order to cater for constantly changing traffic scenarios, as in the case of DDoS attacks, this technique uses a threshold that is adaptively computed instead of using a predefined threshold value. This threshold is based on the mean value of a given parameter, which in turn is calculated from the recent observations made for that particular parameter. The threshold computed at each sampling interval is then used in the decision making process. This algorithm used is now described in detail.

Let x_t represent the value of a given parameter at a time t, and μ_{t-1} represent an average value of that parameter computed from the measurements prior to time t. Time t is a cardinal number and indicates the sampling time instance of the time series, rather than an actual time stamp. A significant change in the parameter x_t is indicated if,

$$\boxed{x_t \geqslant (1+\eta)\mu_{t-1}} \tag{3.1}$$

Change Detection Condition.

where $\eta > 0$ indicates the fractional change from the mean (μ) that constitutes a 'change' in the behaviour of the parameter (x_t) being observed. With the onset of a DDoS attack, these parameters show a substantial increase in their observed values. Hence, the change condition (Equation 3.1) is used only to detect positive changes in the parameter values. After successfully identifying a change, the end of an anomalous activity is identified with the violation of the change condition (Equation 3.1).

The mean values of the parameter μ_t can be estimated, either by using a historical time period window, or by using an exponentially weighted moving av-

erage (EWMA) of its previous occurrences until that time interval. The EWMA technique gives a maximum weight to the most recent observation, and an exponentially decreasing weight (from most recent to the first) to older observations [99]. The current implementation of the change detection technique makes use of EWMA to compute the mean value of a parameter at each sampling time interval.

Exponentially Weighted Moving Average (EWMA) X_t of a parameter Y_t at a given time t, as first studied by Roberts [99] is defined as:

$$\boxed{X_t = \begin{cases} Y_1 & \text{if } t = 1 \\ \lambda Y_t + (1-\lambda) X_{t-1} & \text{if } t > 1 \end{cases}} \tag{3.2}$$

Exponentially Weighted Moving Average (EWMA)

where

- Y_t is the value of the parameter being observed at a time period t. The value of Y_t can either be the individually observed values, or an average computed using a given sampling technique.
- X_t is the EWMA value of the parameter Y_t at a time period t. X_1 can be initialised i.e. at time $t = 1$, either by using the parameter value at that time i.e. Y_1, or an average of first few time intervals.
- λ is the EWMA coefficient (also known as a smoothing or weighting factor) with a value between 0 and 1. A lower value of λ (close to 0) gives more weight to the mean calculated up until that time period and ignores the current value, and a higher value of λ (close to 1) gives more weight to the current value and ignores the mean calculated up until that time period.

The change detection technique used in this chapter makes use of EWMA technique to calculate the mean EWMA (μ_t) for the parameter of interest (x_t) at each sampling time interval (t). Thus, in the above equation (3.2), X_t represents the EWMA mean at time t i.e. μ_t, and Y_t represents the parameter being observed at time t i.e. x_t. Hence, the equation 3.2 can be rewritten as:

$$\boxed{\mu_t = \begin{cases} x_1 & \text{if } t = 1 \\ \lambda x_t + (1-\lambda) \mu_{t-1} & \text{if } t > 1 \end{cases}} \tag{3.3}$$

where $0 \leqslant \lambda \leqslant 1$. The current implementation of the change detection algorithm uses $\lambda = 0.02$ (unless otherwise specified). This is done to give more weight to recent observations while calculating the mean at a given time interval.

Instead of flagging a change in the parameter being observed with every single occurrence of the threshold condition (equation 3.1), the change detection technique has been slightly modified so that it triggers an alarm (an actual change) only after a minimum number of consecutive occurrences of threshold condition are detected. This is done to minimise the false alarms that would potentially arise due to erratic fluctuations in the parameters being observed. Thus, an alarm is triggered when there are k consecutive time intervals for which the change detection condition occurs before flagging an actual change in the observed parameter.

Configurable parameters of the change detection algorithm As shown in Algorithm 3.3, the change detection technique used in the ensemble makes use of three configurable parameters (η, λ and k) which can be fine-tuned to different values. These parameters are:

- η: this parameter is used to adaptively calculate the threshold value used for change detection. The chosen value of η is partly based on the expected sensitivity of the parameter being observed. Unless otherwise specified, $\eta = 0.5$ has been used for the observed parameter i.e. new or NSP source IPs. Later on in Chapter 6, different values of η are chosen for different parameters, however, they are kept constant for any given parameter across all the datasets used in the evaluation.

- λ: this parameter determines the relative weight given to the most recent values of the mean parameter value computed before the current time interval. The value of λ lies between 0 and 1. The experiments conducted for this chapter use a low value (0.02) of λ, thereby giving weight to the recent means computed before the current time interval.

- k: this parameter is used to determine the number of consecutive changes to detect before flagging an actual change in the parameter being observed. This is done to filter out erratic fluctuations and thereby reduce the false alarms. Unless otherwise specified, all experiments have been conducted using $k = 3$.

Algorithm 3.3: Change Detection Algorithm.

Input: Time series file of the parameters of interest, η, λ and k
Output: String of 1's (change) and 0's (no change)

```
/* First instance of the time series */
if (t = 1) then
    /* current mean using equation 3.3 */
    μ_t = x_1
    Print output
else
    /* subsequent instances of the time series */
    μ_{t-1} = μ_t
    /* change condition (equation 3.1) true */
    if (x_t >= (1 + η)μ_{t-1}) then
        /* Increment change counter */
        ChangeCounter++
        /* k consecutive change condition occurrences */
        if (ChangeCounter >= k) then
            Change detected
            Print output
            /* Decrement change counter */
            ChangeCounter--
        end
    else
        /* change condition (equation 3.1) false */
        No Change detected
        /* Reinitialise change counter */
        ChangeCounter=0
    end
    Print output
    /* current mean using equation 3.3 */
    μ_t = λx_t + (1 − λ)μ_{t-1}
end
```

An additional parameter used in the change detection algorithm is t, indicating the time resolution (in seconds) or the duration of the sampling interval over which the individual parameters are measured. The results presented in this chapter use 10 seconds as the sampling interval.

The chosen values of these configurable parameters are based on previous work done in change detection using EWMA, specifically in DDoS related scenarios [105]. Douglas [76] found that in general the values of λ within the interval $0.05 \leqslant \lambda \leqslant 0.25$ work well in practice. Khoo et al. [58] presented a study on the

false alarm rates of various change detection algorithms including EWMA, with $\lambda = 0.15$. Paul [88] employed EWMA for detecting saturation attacks against web-servers, using $\lambda = 0.15$. However, he noted that using λ values between 0.1 and 0.2 also brought good results. Siris et al. [105] applied EWMA to anomaly detection of SYN flood attacks with $\eta = 0.5, \lambda = 0.02, k = 4, t = 10$ for the configurable parameters.

The EWMA technique was applied on the rate of new source IPs in this chapter using different values for η, λ and k within these suggested ranges, which were experimentally adjusted to improve accuracy. The value of η directly affected the *sensitivity* of the change detection algorithm. Using low values for η such as 0.2 made the change detection prone to small fluctuation in the observed parameter, thereby giving rise to false positives, whereas assigning high values such as 1.0 caused EWMA to miss the changes, thereby giving rise to false negatives. $\eta = 0.50$ was used in the experiments, although in general, values between 0.25 and 0.50 gave good results.

The chosen value for $\lambda = 0.02$ (the smoothing constant) was based on the reviewed literature [76, 105], and determined the assigned *weight* or *importance* of recent mean values in comparison to older values. In general, values of λ ranging from 0.2 to 0.6 were found to give good results. Values lower than 0.02 resulted in a delay in detecting the changes, and for values higher than 0.06 EWMA was unable to accurately detect the end of a change, although the main focus in this work is to detect the onset of a DDoS attack indicated by a change in the observed parameter.

The k value determined the *robustness* of the EWMA technique to erratic fluctuations in the observed parameter. The value of $k = 3$ used here is once again based on the reviewed literature [105]. A larger value for k such as 4 or 5 unnecessarily delays the change detection, as also indicated by the small gap between the increase in the number of new source IPs and the decision function's output around the 260^{th} sampling interval, as shown in Figure 3.5. On the other hand, a small value of k, such as 1, can give rise to false alarms, depending on the nature of the parameter being observed. The datasets used in this chapter were not adversely affected by lower values of k, and gave good results even with $k = 1$. However, $k = 3$ was chosen, for the NSP source IPs parameter analysed in this chapter, and also for most of the parameters used later in Chapter 6, to minimise the false alarms that potentially may arise due to fluctuations.

Thus, the configurable parameters chosen for EWMA, as applied to NSP source IPs were $\eta = 0.50$, $\lambda = 0.02$, $k = 3$, and $t = 10$.

3.4 Experimental Results and Analysis

The proposed DDoS attack detection algorithm uses the change in the rate of new source IP addresses as the key parameter to detect the onset of DDoS attacks. In order to validate the use of this technique for the detection of HRF DDoS attacks, a comparative study of two different network traffic traces has been conducted. The analysis conducted by Peng et al. [91] compared daily traces collected by the University of Auckland (Auckland VIII dataset) [1] with the previous two week's data to observe the persistence of source IP addresses. Their analysis concluded that a majority of the IP addresses appearing in a network traffic under normal or non-attack conditions have appeared previously.

In this work, a similar analysis has been conducted using network traffic traces collected from a dedicated block of unused IP addresses (a so-called Darknet) and also the Auckland VIII traffic. In contrast to the used address space in the Internet where there are interconnected production hosts, the unused address space (Darknet) comprises routable IP addresses which are not connected to any production hosts. Hence, due the absence of real hosts in the Darknet, the observed traffic is by definition unsolicited and likely to be either malicious or opportunistic, or due to mis-configuration of the source software. Table 3.1 shows the results of our analysis by comparing the two network traffic traces – Auckland VIII and the Darknet – for the persistence of source IP addresses. It provides a comparative analysis of the percentage of source IP addresses appearing within a fortnight of the listed dates for both Auckland VIII and Darknet network traffic traces. It should be noted that the Auckland VIII trace used in this analysis is preprocessed to remove all TCP flows indicative of an anomaly i.e. any TCP flow with less than 3 packets. Thus, the Auckland VIII trace used in the analysis is representative of network under normal traffic conditions.

Table 3.1 shows a marked dissimilarity between the percentage of source IP addresses that appeared in the last fortnight for the two datasets. In the Auckland VIII dataset, a large percentage (about 88-90%) of source IP addresses reappeared when compared with the traffic observed in the last fortnight – a normal network traffic behaviour. In contrast, a very low percentage of source

Table 3.1: Persistence of source IPs in the Auckland VIII and Darknet Trace.

Auckland VIII Trace		**Darknet Trace**	
Date	Percentage	Date	Percentage
2001-Mar-26	88.7%	2009-Dec-25	1.25%
2001-Mar-27	90.3%	2009-Dec-26	1.05%
2001-Mar-28	89.1%	2009-Dec-27	1.03%
2001-Mar-29	89.2%	2009-Dec-28	0.91%
2001-Mar-30	90.2%	2009-Dec-29	0.94%
2001-Mar-31	89.9%	2009-Dec-30	1.07%
2001-Apr-01	88.1%	2009-Dec-31	7.99%

IP addresses re-appear in the case of likely malicious network traffic collected from the Darknet. Thus, it can be concluded that the change in rate of new or NSP source IP addresses is relatively high (low persistence) for a malicious activity such as DDoS attack, and low (high persistence) for a normal network behaviour. This can thus be used as a key parameter to detect the onset of a DDoS attack.

3.4.1 Datasets Used

In order to experiment with and evaluate the proposed DDoS attack detection technique it is important to have a dataset with two distinct components, viz., normal (background) traffic and the anomalous traffic. The normal traffic component is drawn from the data collected by the University of Auckland (Auckland VIII dataset) [1] and the anomalous (attack) traffic component is constructed using the *fudp*[3] utility.

Background Traffic

The Auckland VIII dataset is a continuous 2 week (Wednesday, 1^{st} December 2003 – Monday, 15^{th} December 2003) GPS synchronised IP header trace collected using a DAG3.5E Ethernet network measurement card at the University of Auckland [1]. The entire trace consists of only TCP, UDP and ICMP based traffic and all the non-IP traffic has been discarded. All TCP and UDP packets have been truncated to 20 bytes after the start of transport header. All UDP payloads, except for port 23 (DNS) packets which are not altered, have been zeroed.

[3]fudp – http://sourceforge.net/projects/usoft/

To ensure anonymity and privacy, the IP addresses in the captured traffic trace have been mapped to 10.*.*.* using a one-to-one hash mapping. The one-to-one mapping is performed sequentially so that the first real IP address observed is replaced by 10.0.0.1, and the next new IP address is mapped to 10.0.0.2, and so on. The sequential mapping pattern is preserved across the entire traffic trace.

For the purpose of experimentation, a subset of the entire dataset was used, comprising nearly 1 hour of continuous bidirectional real-world network traffic collected on the 1^{st} December 2003. Before using this data, it was pre-processed. The dataset was analysed to extract traffic destined for the busiest web-server (destination IP address 10.0.0.63). The selected traffic was then processed to remove TCP SYN attacks, which were the most common type. In order to achieve this, TCP flows with less than 3 packets (and represents TCP SYN attacks) were considered malicious and were removed. TCP flows with no data from the web-server were also ignored. The processed (clean) data was then converted into a one-way incoming traffic target at the destination web-server 10.0.0.63. The cleaned data was reproduced over the testbed (Figure 5.2) using the *Tcpreplay* utility[4]. The cleaned network traffic trace was replayed at around 300 packets per second.

Table 3.2: Statistics of the Attack Traffic.

Number of Unique Source IPs	**Number of Packets**	**Attack Duration (seconds)**	**Traffic Rate (Mbps)**
35	342,141	134.38	1.01
40	328,579	129.35	1.01
45	321,433	129.29	1.01
50	313,747	123.23	1.01
100	237,831	93.49	1.01
150	201,020	86.83	1.01

Attack Traffic

The attack traffic, superimposed on the cleaned background traffic, was generated using the *fudp* utility. It was used to flood the target host (Figure 5.2) with UDP packets having between 35 and 150 different source IP addresses. The

[4]Tcpreplay – http://tcpreplay.synfin.net/

rationale for using this variation was to analyse the proposed algorithm under different attack conditions. It is acknowledged that attacks with a large number of different source IP addresses (e.g. UDP flooding with 150 different source IP addresses per second) are relatively easier to detect compared to those with a small number of different source IPs. The average packet rate for the attack traffic is approximately 3000 packets per second (10 times the normal background traffic rate). The attack traffic used for experimentation is summarised in Table 3.2.

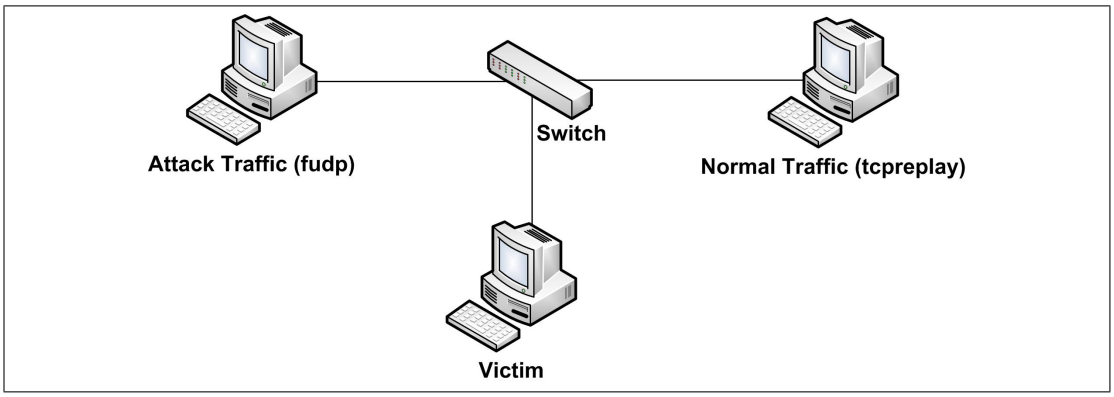


Figure 3.3: The testbed architecture for evaluating the NSP algorithm

3.4.2 Experimental Set-up

A proof-of-concept evaluation of the proposed NSP algorithm using EWMA as the change detection technique and bit vectors for IP address classification (as previously described in Section 3.2.1) has been performed. Figure 5.2 shows the testbed architecture used for evaluating the proposed technique.

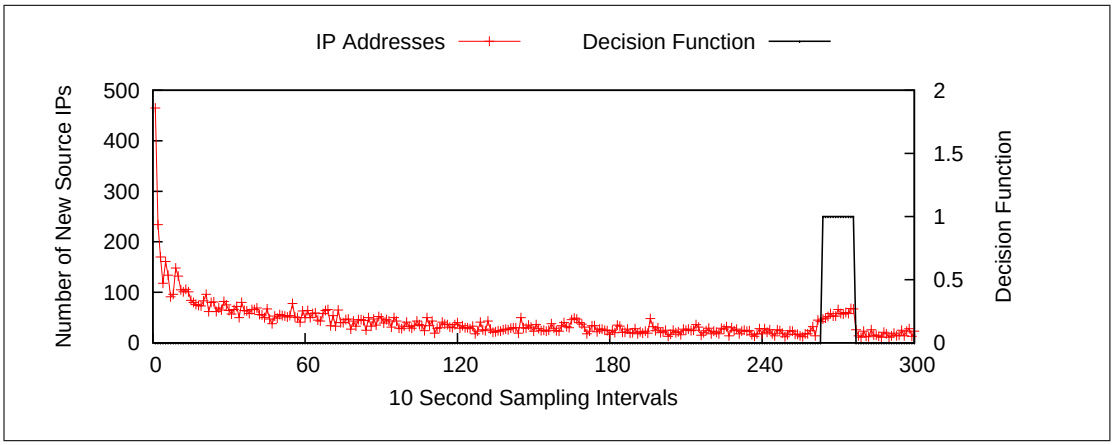


Figure 3.4: UDP flooding attack with 35 new IPs per second.

In the simple experimental set-up, three desktop machines were connected via a layer-2 switch. Two of the three machines were used to generate network traffic, one each for attack and normal (background) traffic. A mix of attack and normal traffic was directed against a target host (victim) machine. All the incoming traffic (normal and attack) to the victim was captured and used for an off-line analysis and evaluation of the proposed NSP algorithm using the change in the rate of source IP addresses to detect the onset of the HRF DDoS attack.

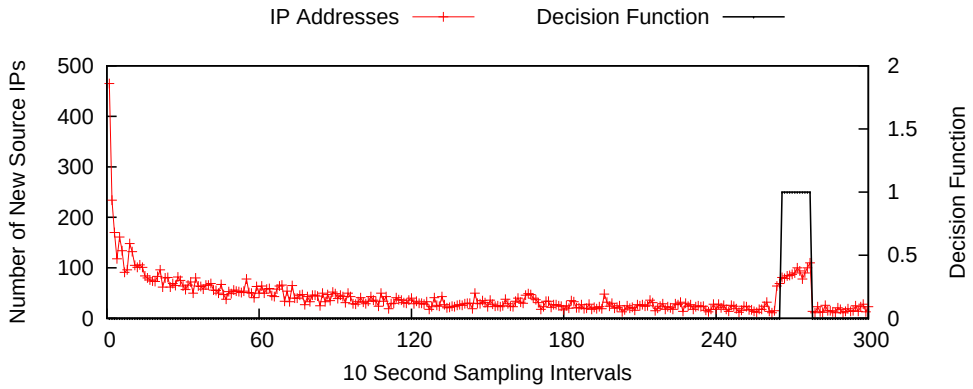
3.4.3 Performance Evaluation

In order to evaluate the performance of the proposed algorithm, several separate UDP flooding attack traffic datasets were configured with varying numbers of new or NSP source IP addresses – 35, 40, 45, 50, 100, and 150 new IPs per second. This traffic was then embedded in the normal (background) network traffic in several separate experiments which replayed the composite traffic streams against the target. The results of the experiments were analysed using 10 seconds as the sampling interval. Figure 3.4 shows the result of the UDP flooding attack using 35 new source IP addresses per second.

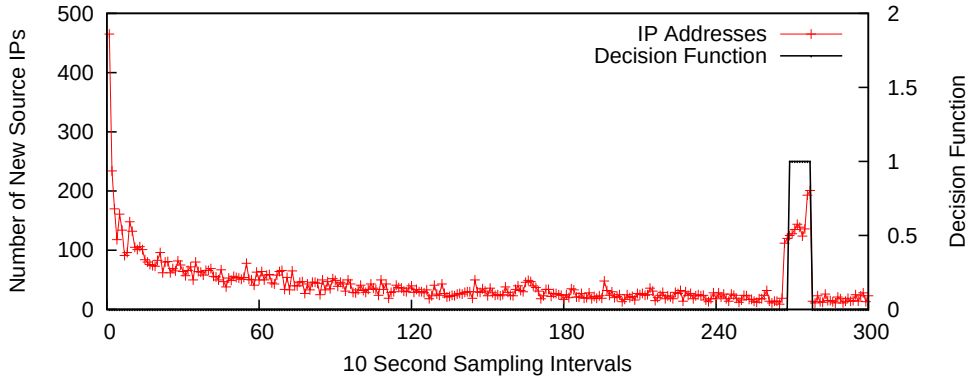
In Figure 3.4, the horizontal axis represents the sampling intervals (10 second), the left vertical axis is the number of new or NSP source IP addresses in the measurement interval and the right vertical axis represents the EWMA decision function with 1 being attack and 0 being no attack. The number of new source IP addresses in the 10 second measurement interval was calculated using the algorithm described in Section 3.1.

The UDP flooding attack with 35 source IPs started at measurement interval 260 and ended at interval 280, and was detected by the EWMA technique. Figure 3.4 naturally shows a large increase in the number of new IPs at the start. The NSP algorithm was able to generate a list of legitimate IP addresses or white list. A manual analysis of the generated white list was performed post attack to check for the presence of any attack sources in the list. No attacking source IP addresses were found in the white list (in any of the experiments). It should be noted that the detection delay is bounded by the measurement interval, which is set to 10 seconds for this experiment. All the attacks listed in Table 3.2 have been detected in less than 10 seconds, which can be further reduced using smaller measurement intervals.

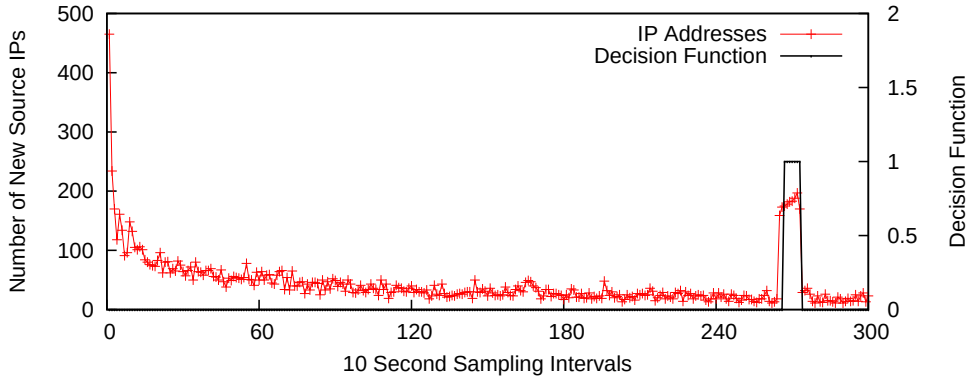
Experiments using a higher number of source IP addresses per second for the



(a) UDP flooding attack with 50 source IPs per second.



(b) UDP flooding attack with 100 source IPs per second.



(c) UDP flooding attack with 150 source IPs per second.

Figure 3.5: UDP flooding attack with varying source IPs per second.

UDP attack traffic were also conducted. Figures 3.5a, 3.5b and 3.5c show the results of the UDP flooding attack using 50, 100 and 150 source IP addresses per second respectively. A small gap (detection delay) observed around the 260^{th}

sampling interval, between the increase in the number of new source IPs and the decision function's output is due to the chosen value of k which looks for 3 consecutive changes in the observed parameter before flagging an actual change. This is mainly done to minimise false detection which may potentially arise due to erratic fluctuations in the parameters being observed. A lower value of k will further reduce this detection delay; however, it would also make the detection process more vulnerable to small fluctuations in the observed parameter. The proposed NSP algorithm successfully identified the attack instances in all the UDP flooding attacks with varying number of new source IP addresses per second.

3.5 DDoS Detection and Mitigation Model (D2M2)

A workable solution to protect against DDoS attacks requires an accurate and timely detection of the onset of attack. However, once a DDoS attack has been successfully identified, the next step is to protect the system by mitigating the impact of the attacks.

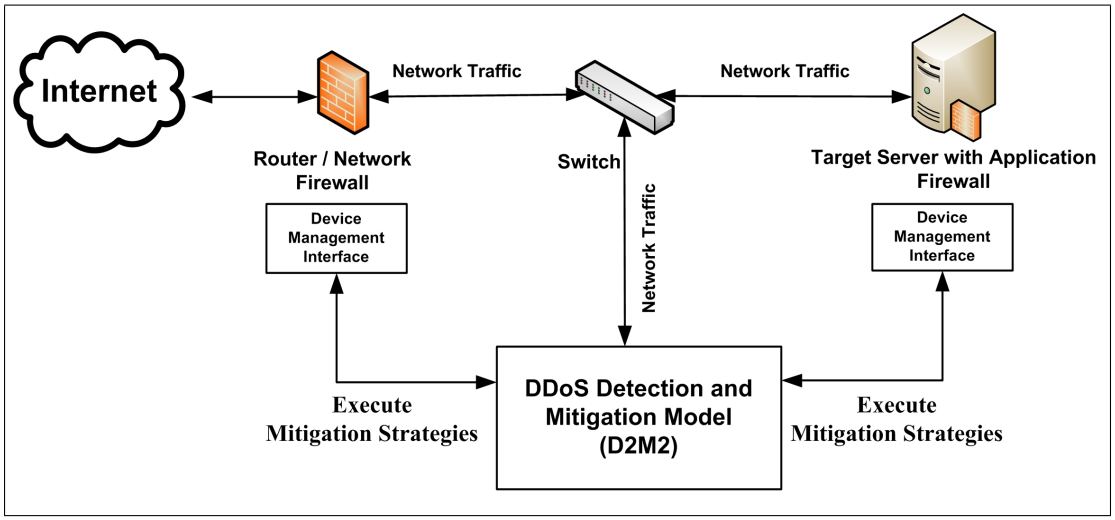


Figure 3.6: Deployment architecture of DDoS Detection and Mitigation Model.

Figure 3.6 presents a security architecture that has been designed as part of a larger project[5] to which the candidate has contributed. The architecture intends to protect such a security device such as an application firewall, and

[5]A joint research project into DoS and DDoS attacks undertaken under the auspices of the Australian and Indian Governments as part of the Australia-India Strategic Research Fund (AISRF) 2008 – 2012.

to ensure continuous service availability during DDoS attacks. Importantly, it includes a DDoS Detection and Mitigation Model (D2M2) – a conceptual model which can employ the different attack detection techniques developed as part of this research, deployed in an off-line mode to protect a target server and its application firewall.

The DDoS attack detection algorithm developed in this chapter (see Section 3.2 for details) identifies a client white-list (a list of legitimate IP addresses) which can then be dynamically downloaded to an application firewall. Preliminary work in this regard was carried out to demonstrate the feasibility of the design. In this proof-of-concept work, Apache was used as the web-server and ModSecurity [2] as an application firewall. The white-list generated by the attack detection technique was downloaded to the directory containing ModSecurity's main configuration file and the web-server gracefully restarted, thereby allowing traffic only from the white-listed IPs. This demonstrated, at the conceptual level the overall design goal of D2M2 to have a single component with integrated DDoS attack *detection* and *mitigation* capabilities to protect a target host running an application firewall against HRF DDoS attacks. The future work section in Chapter 7 expands on this work and suggests how it may be fully automated.

3.6 Conclusion

Notwithstanding the research conducted in DDoS attack detection over the past decade, timely and reliable detection of such attacks remains a challenging problem. So too does the formulation and implementation of an effective attack mitigation strategy.

The main contribution of this chapter is the design of a DDoS attack detection technique based on new or NSP source IP addresses and its proof-of-concept implementation and evaluation. The technique exploits a key characteristic of DDoS attacks, viz., a marked increase in the number of new source IPs accompanying the onset of a HRF DDoS attack than can transcend protocol-specific defences, such as those that protect against specific types of flooding attacks, such as SYN flood [49, 121]. The current implementation of the detection technique, although *post hoc*, uses a single network traffic feature and a lightweight data structure, and thus facilitates the detection of DDoS attacks at 'wire speed', compared to computationally intensive machine learning-based

models such as Hidden Markov Models (HMMs) [126] and Artificial Neural Networks (ANN) [66].

Finally, this chapter has presented the conceptual design of a mitigation strategy, which can employ the various detection strategies designed and developed in the thesis, including that presented in this chapter.

Chapter 4

Identifying, Classifying, and Modelling Flash Events

The previous chapter described the design, development and evaluation of a DDoS attack detection technique based on the rate of arrival of new or Not Seen Previously (NSP) source IP addresses, and provided a proof-of-concept implementation using bit vectors. However, this technique focused on detecting DDoS attacks without considering Flash Events (FEs), a network anomaly sharing a number of features with DDoS attacks. An FE refers to the situation that arises when a large number of legitimate users concurrently send access requests to a web-server. Examples of FEs include popular websites experiencing a surge in incoming traffic during major world events (e.g., earthquakes, 9/11 attacks, sporting events, newsworthy elections, etc).

Both DDoS attacks and FEs represent *anomalies* in the Internet traffic and cause performance degradation. However, each of these network activities results from a different section of the web-community with varied intentions. A precise distinction between DDoS attacks and FEs is important because each requires an entirely different set of actions to be undertaken by a network administrator after they have been successfully identified. In order to understand how FEs may be differentiated from similar looking DDoS attacks, a detailed study of FEs and their characteristic features is needed. However, the relative lack of public domain FE datasets hinders research in this area. The work presented in this chapter addresses these issues by proposing a model that provides the foun-

dations for both identifying and synthesising FEs. Thus, the chapter addresses two research questions identified earlier – Research Question 2: *Is it possible to differentiate DDoS attacks from FEs?* and Research Question 3: *What are the essential characteristics of different types of FE and is it possible to use those characteristics to model and subsequently generate different types of synthetic FE traffic?* In the process it focuses on:

- the development and evaluation of a technique for distinguishing DDoS attacks from FEs.
- the development of an FE model, which enables the generation of FE traffic datasets.

The characteristic features of FEs explored in this chapter will be used in tandem with some server-load based features, such as CPU and memory utilisation, to develop an ensemble-based DDoS attack and FE detection technique in Chapter 6. The ensemble-based technique is designed to detect a variety of DDoS attacks and to differentiate them from similar looking FEs.

The rest of this chapter is organised as follows. Section 4.1 presents a set of parameters viz., the change in rate of incoming traffic, the change in rate of new source IP addresses, and the distribution of requests among source IP addresses, which can be used to differentiate DDoS attacks from FEs. Section 4.2 provides a detailed classification of FEs into three categories: predictable, unpredictable and secondary. Section 4.3 presents a server-side FE model based on three characteristics: change in Rate of Incoming Traffic, change in rate of Source IP Addresses, and change in pattern of resources accessed. Section 4.4 presents the experimental results and validates the proposed model using real-world datasets. The model presented in this section facilitates the generation of synthetic FE traffic, using the traffic generation and testbed framework described in Chapter 5, and is used where applicable in this thesis. Finally, Section 4.5 summaries the research presented in the chapter.

4.1 Distinguishing Flash Events and DDoS Attacks

Although both DDoS attacks and FEs are accompanied by a surge in the incoming traffic volume, FEs are caused by legitimate clients, whereas DDoS attacks

result from malicious activities usually carried out by a set of compromised machines. DDoS attacks can be crafted to look very similar to FEs, with only the *intent* separating them from FEs, and not the actual traffic *content*. However, this criterion does not help in differentiating between the two. In order to exploit the behavioural differences between these two class of events, in this section a set of parameters are proposed which, while not completely orthogonal, do capture different aspects of the incoming traffic, and can be used to efficiently differentiate DDoS attacks from FEs.

4.1.1 Change in the Rate of Incoming Traffic

In a DDoS attack, the attacking machines or bots are often compromised via malware (malicious software), which can be configured to send packets at a predefined rate. In order to cause maximum damage, the bot-master triggers the machines simultaneously. On receiving control commands from the bot-master, the bots start sending huge amounts of data. Hence, the victim server often experiences a sudden burst in incoming traffic over a relatively short period of time. This unexpected surge causes the server to exceed its maximum pre-defined sustainable limits, thereby slowing it down considerably, and in some cases forcing it to shut down. In contrast, when a newsworthy event occurs, triggering an FE, it is highly unlikely that the entire web community is simultaneously informed. Instead, it takes times for the news to spread. Therefore, the incoming traffic to the web server increases less rapidly over time as compared to a DDoS attack, before reaching a maximum. It should thus be possible to use this difference in the rate of incoming traffic to differentiate DDoS attacks from FEs.

4.1.2 Change in the Rate of New Source IP Addresses

DDoS attacks are usually conducted by a finite set of compromised machines also known as a Botnet. Recent studies on Botnets define two terms related to their size: the Botnet's *footprint* and its *live population* [98]. A Botnet's *footprint* is the total number of infected machines at any point in its lifetime. A Botnet's *live population* is the number of bots currently available in the Command and Control (C&C) channel. The bot-master takes control over these live-bots, and issues commands, often using an Internet Relay Chat (IRC) server or a particular channel on a public IRC server. Rajab et al. [98] estimated that even though

the *footprints* of the Botnets tracked by them went as high as several tens of thousands, their effective size i.e. the *live population* at any given point in their lifetime was usually limited to a few thousand.

On the other hand, the number of web users accessing a popular website to obtain information related to a special event is substantially greater than the number of distinct bots. This difference in the total number of distinct machines or source IP addresses – bots in the case of a DDoS attack and legitimate clients in the case of a FE – forms the basis of the second parameter that can be used to differentiate DDoS attacks from FEs.

In the case of a DDoS attack, since only a limited number of bots are available to the bot-master, to maximise their utilisation the bots are simultaneously triggered. Thus, at the commencement of an attack, the victim host experiences a large number of previously unseen or new source IP addresses, along with an increase in incoming traffic. Due to a relatively small and finite set of compromised machines, the attacker uses nearly the same set of sources during the course of the attack. Thus, after an initial burst of new source IP addresses, the victim host receives traffic from relatively fewer new source IP addresses.

In contrast to a DDoS attack, during an FE, as the information related to the event travels across the web community, more and more new users come online to access the information. Hence, the web-server constantly experiences incoming traffic from previously unseen or new source IP addresses. It is conjectured, therefore, that change in the rate of new source IP addresses, as seen by the target, can be used as one of the parametric differences between DDoS attacks and FEs.

4.1.3 Distribution of Requests Among Source IP Addresses

In the case of a DDoS attack, since the attacker has control over a limited number of compromised machines, he/she tries to achieve the desired maximum load by forcing each bot to send a large number of requests. Hence, the sudden surge in incoming traffic to the victim is mainly attributed to a high number of requests per source IP address. In contrast to this, during an FE, the majority of the clients are interested typically in a limited amount of information, hence a relatively small number of requests originate from each client. It is conjectured, therefore, that the excess load that the server experiences during an FE is mainly due to an increase in the overall number of clients with a smaller request per client

rate than is the case in a DDoS attack.

The three proposed parameters viz. the change in rate of incoming traffic, the change in rate of new source IP addresses, and the distribution of requests among source IP addresses, suggest a way to distinguish a real-world DDoS attack from an FE. Section 4.1.4 presents the experiments conducted on the two publicly available datasets to validate the proposed parameters.

4.1.4 Experimental Results and Analysis

The proposed parameters for distinguishing DDoS attacks from FEs were investigated and validated by analysing two publicly available datasets, representing a real world FE and a real world DDoS attack. Each of the traces used in this research is available in the public domain and contains traffic pattern information relating to time, pseudonymised source and destination IP addresses, and the request type.

It should be noted that, because of legal and privacy issues involved in obtaining real-world data, only a very limited number of datasets, representing DDoS attacks and FEs, are publicly available. Thus, validation of the three parameters proposed to differentiate DDoS attacks from FEs is limited to these datasets. (Indeed, one of the research objectives of this thesis addresses this very issue, i.e. the lack of public domain datasets, by designing a traffic generation and testbed framework, cooperatively developed as a part of this research. This framework, discussed in Chapter 5, consists of a modest hardware platform and a software traffic generator, developed to synthetically generate different types of DDoS attack and FE traffic which approximates to real-world scenarios.)

Flash Event Dataset The '1998 FIFA World Cup' Dataset [12] is used as the FE dataset for validating the proposed parameters. This dataset is provided by the Internet Traffic Archive and contains all the requests made to the 1998 FIFA World Cup websites during a period of 92 days between 30^{th} April 1998 and 26^{th} July 26 1998. During this period, 33 different web servers were used to host information related to the World Cup. These servers were located at four different geographical locations: Paris (France), Plano (Texas), Herndon (Virginia) and Santa Clara (California). A time resolution of 1-second was used on all the servers. The time on each server was synchronised with the local time in France (GMT + 0200), which was the host country. The source IP addresses

in the traffic traces have been replaced by unique integer identifiers which are preserved throughout the dataset. To ensure privacy, the mapping file for the source IP address pseudonymisation is not publicly available.

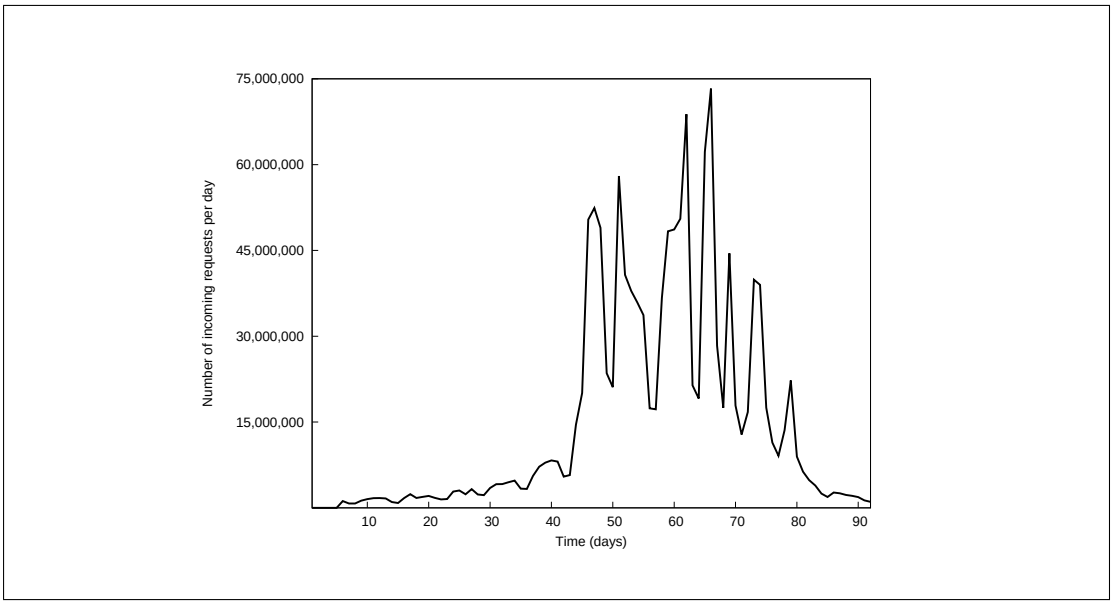


Figure 4.1: Incoming traffic profile of the 1998 FIFA World Cup dataset.

The FE dataset, available as web-server logs in Common Log Format (CLF)[1] contains only the incoming HTTP requests (GET, POST, HEAD and OTHER) made to one of the 33 servers hosting the 1998 FIFA World Cup content. Figure 4.1 shows the daily aggregates of incoming traffic volumes experienced by the websites hosting the 1998 FIFA World Cup. Each of the individual peaks or bursts of traffic is essentially a fine structure within the overall 92 day period, and represents individual FEs on a smaller scale than the overall event [13]. This chapter of the thesis focuses largely on the two FEs corresponding to the two semi-final matches on the 73^{rd} and the 74^{th} days of the tournament. Figure 4.2 presents the incoming traffic volumes seen during the semi-final matches. For this section, the incoming traffic around the 2^{nd} semi-final is used as an FE dataset.

DDoS Attack Dataset The CAIDA 'DDoS Attack 2007' Dataset [51], representing a real-world DDoS attack, was used to validate the chosen parameters. The dataset contains pseudonymised traces from a DDoS attack that occurred on August 4, 2007 for approximately one hour (20:50:08 UTC to 21:56:16). The dataset represents a type of attack in which the attacking machines attempt to

[1]http://httpd.apache.org/docs/current/logs.html

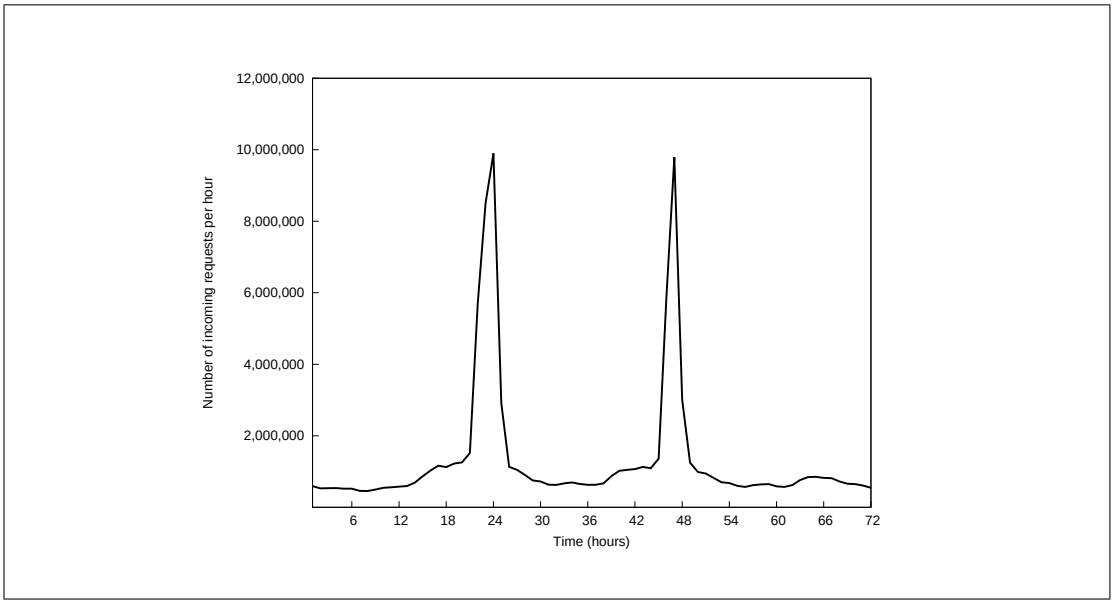


Figure 4.2: Incoming traffic profile around the semi-final matches of the 1998 FIFA World Cup dataset.

block access to the target server by sending a large number of access requests, and thus consume all the available computing resources, including the bandwidth of the server's Internet connection. The dataset contains the attack traces to the victim and its responses, pseudonymised using prefix-preserving CryptoPAn via a single key, and all packet payloads are removed. The entire CAIDA-dataset is divided into 5-minute packet capture (pcap) files. From the complete DDoS-dataset, only the one-way attack traffic going to the victim is used for analysis.

Figure 4.3 shows the one-way attack traffic of the DDoS-dataset. It shows variations in the number of incoming packets (per minute) against time. After the initial 25 minutes, the incoming packet rate rises abruptly. The maximum traffic rate is attained at around the 35^{th} minute, and is held nearly constant for the remainder of the attack. Table 4.1 summarises the macro-level characteristics of the two datasets (1998 FIFA World Cup and CAIDA) used for evaluation.

Using these two datasets, the following sections now present an evaluation and validation of each of the three parameters proposed to differentiate DDoS attacks from FEs – the change in rate of incoming traffic, the change in rate of new source IP addresses, and the distribution of requests among source IP addresses.

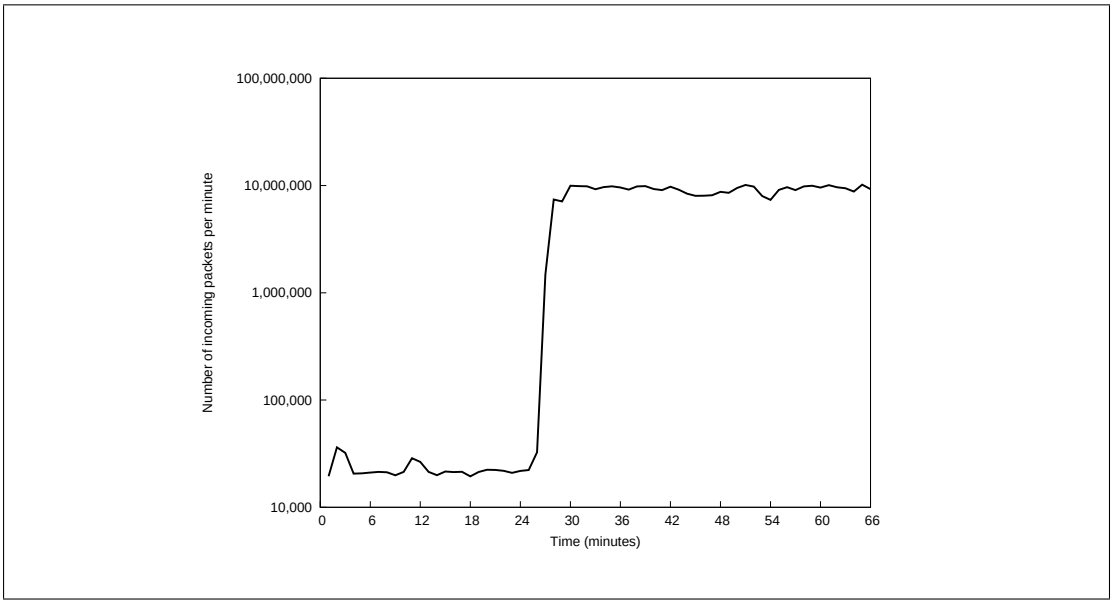


Figure 4.3: Incoming traffic profile for the CAIDA dataset

Table 4.1: Macro-level Statistics of the 1998 FIFA World Cup and the CAIDA dataset.

Characteristic Features	**1998 FIFA World Cup Dataset**	**CAIDA Dataset**
Activity type	Flash Event	DDoS Attack
Duration	92 days	66 min
Packets type	HTTP	ICMP ECHO
Number of target(s)	33	1
Number of packets sent	1,352,804,107	359,655,826
File format	Common Log Format	pcap
Size (uncompressed)	8.1 GB	5.3 GB

Change in the Rate of Incoming Traffic

This section discusses the rate at which the incoming traffic appears at the server or victim host in the cases of an FE or DDoS attack. The number of requests or packets in both cases is considerably greater than in the normal situation i.e. non FE and non DDoS traffic. The focus is on analysing the manner in which the number of these incoming requests or packets increases in both cases. In order to achieve this, the increasing part of the peaks for each of the two events are compared. In the case of the DDoS attack, this period is represented by a ten minute window starting at the 25^{th} minute and ending at the 35^{th} minute of the CAIDA dataset (see Figure 4.3). Figure 4.4 shows the increasing part of the

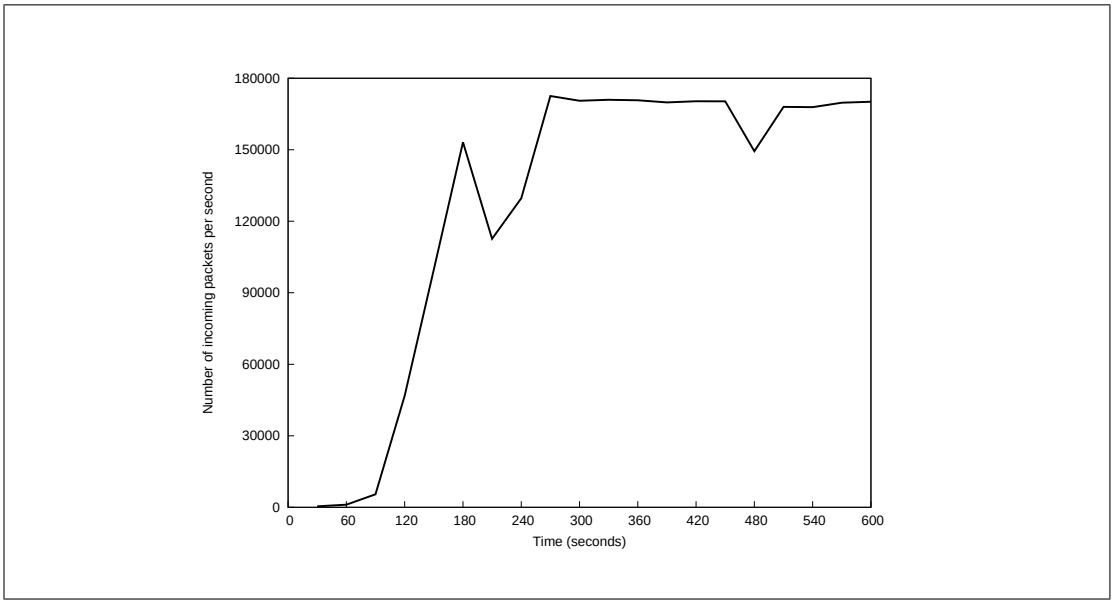


Figure 4.4: Incoming traffic profile for one 10 minute period (25^{th} – 35^{th} minute) within the CAIDA dataset.

CAIDA dataset in terms of number of incoming requests per second. In the case of the FE, the two hour interval (46^{th} and 47^{th} hours) during the 2^{nd} semi-final match of the World Cup i.e. the 74^{th} day, is shown in Figure 4.5, which shows the incoming traffic volume during these two hours. Table 4.2 gives the micro-level statistics during the observation period for the FE and DDoS datasets.

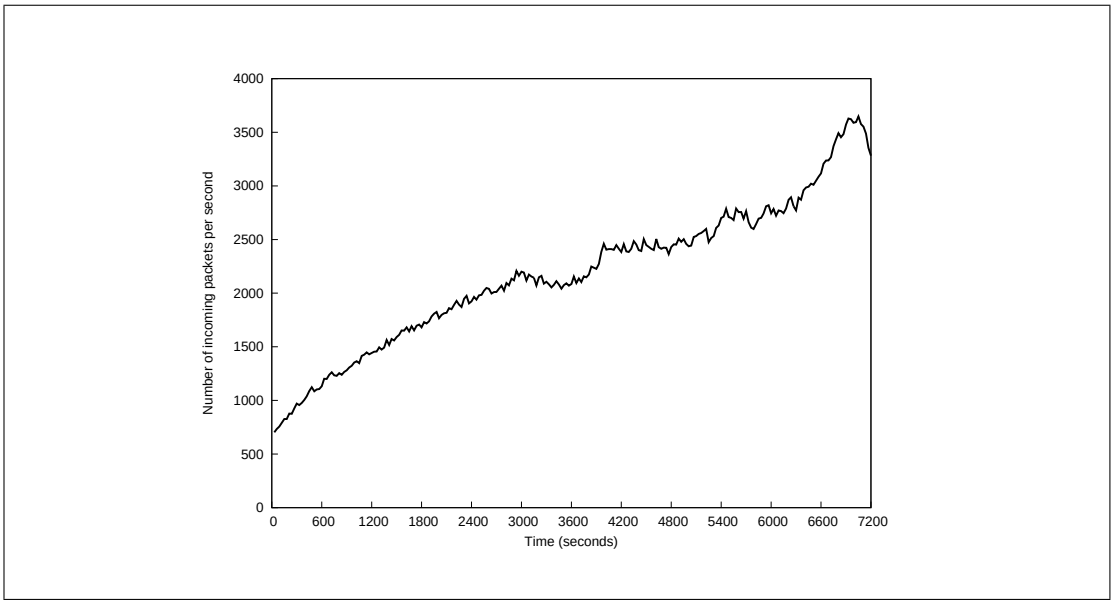


Figure 4.5: Incoming traffic profile for a two hour period (46^{th} – 47^{th} hour) during the 2^{nd} semi-final match of the 1998 FIFA World Cup dataset.

The Figures 4.4 and 4.5 and the Table 4.2 demonstrate a clear difference in the slopes for the two traffic profiles, thus validating the proposal that this feature can be used as a distinguishable parameter to separate a real-world DDoS attack from an FE. (Figure 4.4 also shows a sudden drop in the number of incoming packets at time 210 seconds and 480 seconds, which may be due to the measurement frequency (30 seconds) used in this analysis. The nature of the data and lack of additional information makes it difficult to identify the exact cause of such behaviour.)

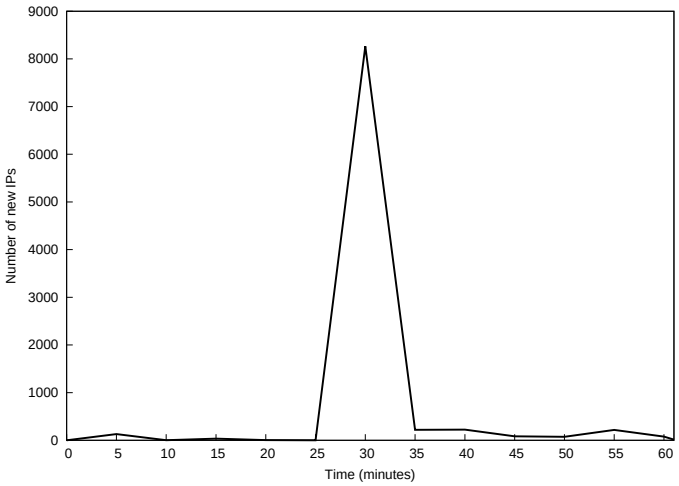
Table 4.2: Micro-level statistics of the 1998 FIFA World Cup and the CAIDA dataset during the observation period.

Characteristic Features	**2nd Semi-final of the 1998 FIFA World Cup Dataset**	**CAIDA Dataset**
Observation time	2 hours	10 mins
Number of Requests/Packets	15,698,000	74,478,486
Number of Source IPs	41,566	8,585
Requests per source IP	377.66	8675.42

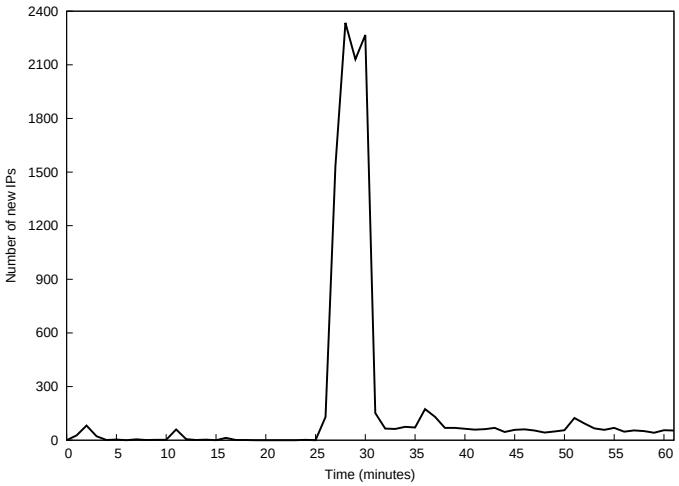
Change in the Rate of New Source IP Addresses

This section presents an analysis of the rate of increase in new source IP addresses as a characteristic feature which can differentiate a DDoS attack from an FE. As discussed in Section 4.1, in the case of a DDoS attack, the attacker has control over a finite set of compromised machines. Activation of the bots produces a gigantic traffic load on the target host coming from a set of sources which have not been previously seen. Therefore, it produces a sharp increase in the new source IP addresses synchronised with the onset of the attack. This limited set of compromised machines is used continually during the attack to produce the desired effect. In other words, this means that once the attack starts, the target host rarely sees any new source IP addresses.

Figure 4.6a shows the changes in the new source IP addresses, as seen by the target host, for the CAIDA dataset over five minute sampling intervals or history periods. Similar experiments were conducted using a smaller sampling interval of one minute, and the results are shown in Figure 4.6b. Both these figures, 4.6a and 4.6b, support the conjecture that with the onset of the DDoS attack, the



(a) New source IP addresses using a five minute history period.

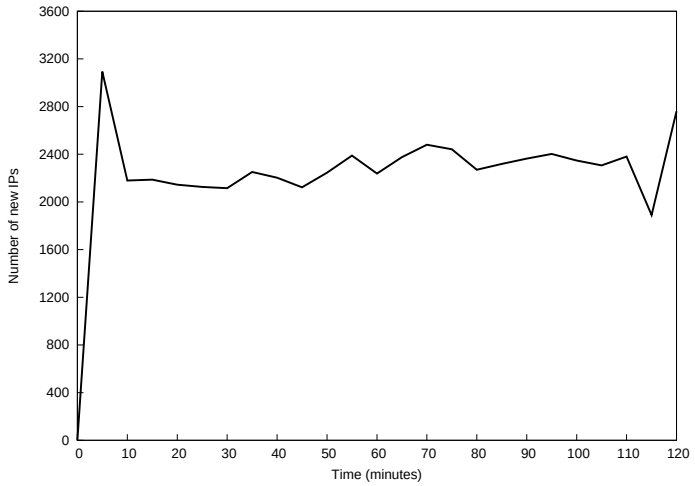


(b) New source IP addresses using a one minute history period.

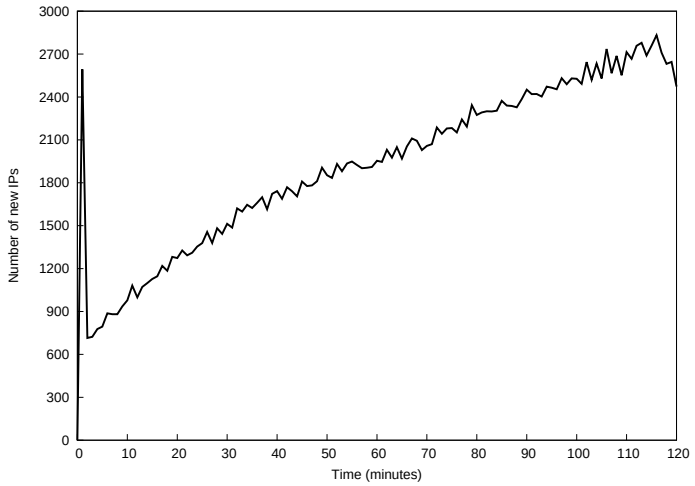
Figure 4.6: New source IP addresses for the CAIDA dataset.

number of previously unseen or new source IP addresses increases abruptly, and remains fairly low and constant for the remaining duration of the attack.

By contrast, during an FE, the web server observes a relatively constant or slowly increasing rate of new IP addresses, quite unlike a DDoS attack. Figure 4.7a shows the change in rate of new source IP addresses, using a five minute sampling interval, as seen by the target host, for the FE dataset. It shows that,



(a) New source IP addresses using a five minute history period.



(b) New source IP addresses using a one minute history period.

Figure 4.7: New source IP addresses for a two hour period (46^{th} – 47^{th} hour) during the 2^{nd} semi-final match of the 1998 FIFA World Cup dataset.

apart from the initial five minutes, the number of new sources accessing the server remains fairly constant. Using a small sampling period of one minute, the experimental results in Figure 4.7b show that the gradual increase in the number of new IP addresses is still very different from the DDoS attack, where the increase is more abrupt. Hence, the change in the rate of new source IP addresses can be used to distinguish between FEs and DDoS attacks.

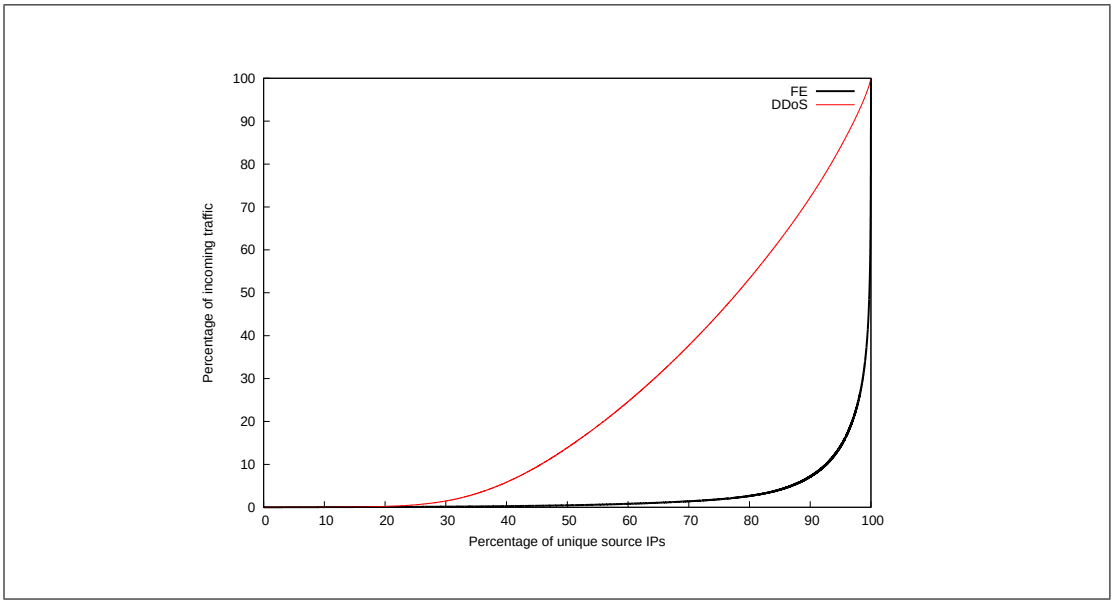


Figure 4.8: Requests per source IP distribution for the CAIDA and a subset of the 1998 FIFA World Cup dataset.

Distribution of Requests Among Source IP Addresses

Table 4.2 shows a marked difference in average requests per IP and an analysis of the distribution of the requests per IP appears in Figure 4.8.

Figure 4.8 shows that in the case of an FE, 90% of the total clients contribute less than 10% of the total outgoing traffic, which is distinctly different from a DDoS attack, where the distribution of traffic is more uniform amongst the source IP addresses. Thus, the distribution of requests per source IP clearly distinguishes DDoS attacks from FEs.

An important observation that can be drawn from this analysis is that the majority of traffic in the case of an FE is generated by a small percentage of the overall participating clients, less than 10% in this case. The source IPs sending large numbers of requests are suspected to be network proxies hiding a large number of private source IPs behind a comparatively small set of public IPs. Another possible explanation could be the presence of web-crawlers or web-spiders, which often send very large numbers of requests within a short time interval.

One of the differences between a proxy and a client, as pointed out by Krishnamurthy et al. [59], is the *think time*. A proxy has a lower *think time* than a client and hence it issues more requests per time interval compared to a client. In other words, the inter-arrival-time (IAT) for packets coming from a proxy is

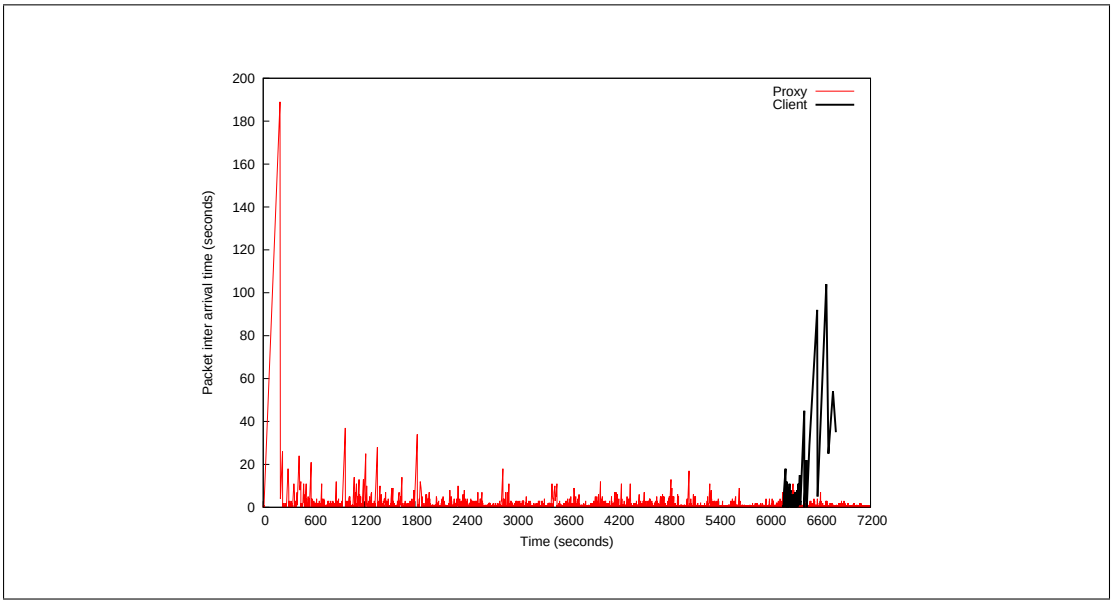


Figure 4.9: Packet IAT comparison for proxy and client from the the 1998 FIFA World Cup dataset.

smaller and stays fairly constant compared to packets coming from clients which have a large and variable *think time*.

This behavioural difference is shown by Figure 4.9, which compares the inter-arrival-time for packets from two different IPs within the FE dataset used in this analysis i.e. the two hour interval (46^{th} and 47^{th} hours) during the 2^{nd} semi-final match of the World Cup. Packets arriving from two different IPs were extracted from the dataset, one IP hypothesised to be a proxy, corresponding to an IP address sending a large number of requests, and the other hypothesised to be a client, corresponding to an IP sending a relatively small number of requests. The inter-arrival-time of packets originating from these two source IPs was computed and compared, and this comparison is shown in Figure 4.9. The client was 'alive' for just 624 seconds, from the 6167 second mark to the 6790 second mark.

It is conjectured that the evolution of botnet behaviour to mimic the variable IATs characteristic of FEs is just around the corner, and presents a possible threat to the successful use of IATs to differentiate botnet traffic from FEs. Future work may explore both this threat and possible counter-measures based on identifying IAT distributions per IP.

This led to the analysis of traffic during the first 25 minutes of the CAIDA dataset. The packet rate for this period is considerably lower than the rest of the dataset, therefore it is considered as non-attack or normal traffic. Figure 4.10

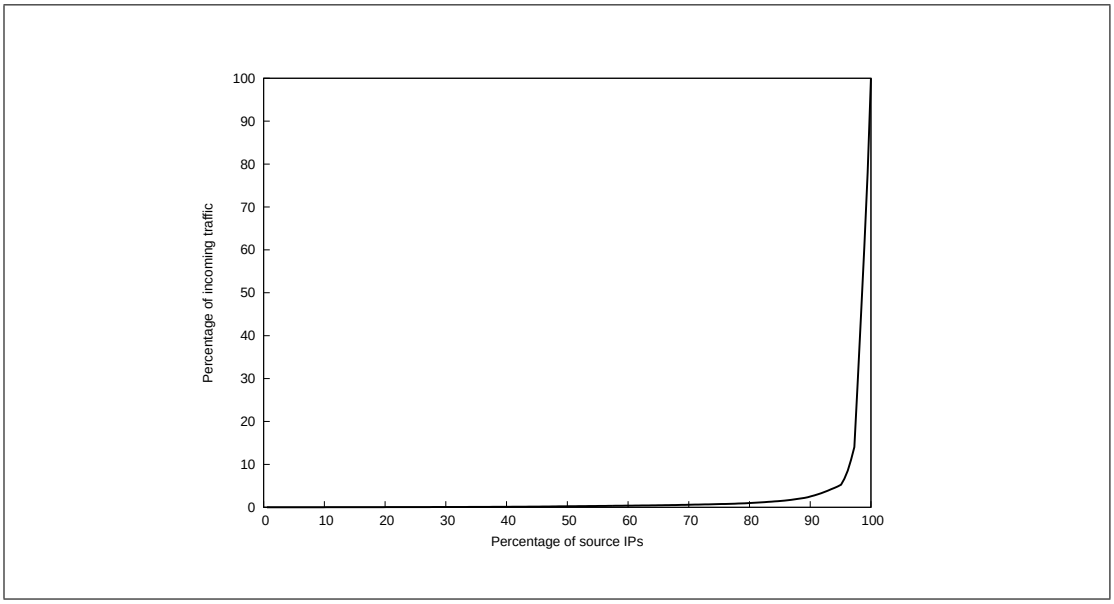


Figure 4.10: Requests per source IP distribution for traffic during the first 25 minute of the CAIDA dataset.

shows the distribution of outgoing traffic among the clients for the non-attack traffic of the CAIDA dataset. The experimental results look very similar to those observed during the FE dataset (Figure 4.8). The total number of unique sources responsible for the normal traffic (183) is very small as compared to the ones responsible for DDoS attack (8585). In summary, Figure 4.8 shows a noticeable difference between the distribution of outgoing requests among the source IPs which can be used to distinguish between these two class of events.

4.2 Flash Event Classification

It is argued in this section that FEs may be divided into three broad categories: *predictable*, *unpredictable* and *secondary*. Research presented by Jung et al. [54] briefly describes the predictable and unpredictable nature of FEs. This section of the chapter extends that work by presenting a detailed classification of FEs based on a few real-world examples, and then uses that classification in Section 4.3 to propose a general server-side FE model.

4.2.1 Predictable Flash Events

A *predictable* FE (henceforth pFE), is one whose expected occurrence is known *a priori*, thus allowing network administrators to prepare for it by using various provisioning techniques, such as the use of load-sharing mechanisms or Content Distribution Networks (CDNs) [54]. Some popular examples are product releases (e.g. by hi-tech companies like Apple), widely followed sporting events such as the Olympics, or online play-along websites for popular television programs, where the expected time of the incoming traffic burst is well known in advance. The time when the incoming traffic will hit its peak can also be fairly accurately estimated, so permitting better handling and provisioning for such events. Moreover, most pFEs are directed against servers owned by big companies, who can afford the necessary load or content-sharing techniques to mitigate their effects. The 1998 FIFA World Cup dataset [12], described in Section 4.1.4, is one of the few datasets available in the public domain representative of a pFE. Figure 4.2 shows incoming traffic of two such pFEs, the semi-final matches of the 1998 FIFA World Cup.

4.2.2 Unpredictable Flash Events

Events that are totally unexpected can, if sufficiently newsworthy, also cause a sudden and dramatic surge in network traffic to a site that is supposed to describe the event or provide further leads. The term *unpredictable* FE (henceforth uFE) is used to describe the ensuing burst of network traffic. Provisioning for these events in advance is akin to preparing for natural catastrophes like a tsunami or an earthquake. Designing systems to handle them is possible but may be economically infeasible due to their unpredictability and rarity. The 9/11 terror attack led to such an uFE when major news websites like CNN and MSNBC were overwhelmed by the amount of incoming traffic, pushing their availability close to 0% within minutes after its occurrence [52]. The start and peak-load time of such events is unpredictable and sometimes difficult to identify even *post hoc*. Their frequency of occurrence is relatively lower than for pFEs. Figure 4.11 is an example of an uFE when the popular website Wikipedia experienced a sudden increase in its hourly hits following the death of Steve Jobs [73]. Similar traffic was also observed following the death of Michael Jackson [73].

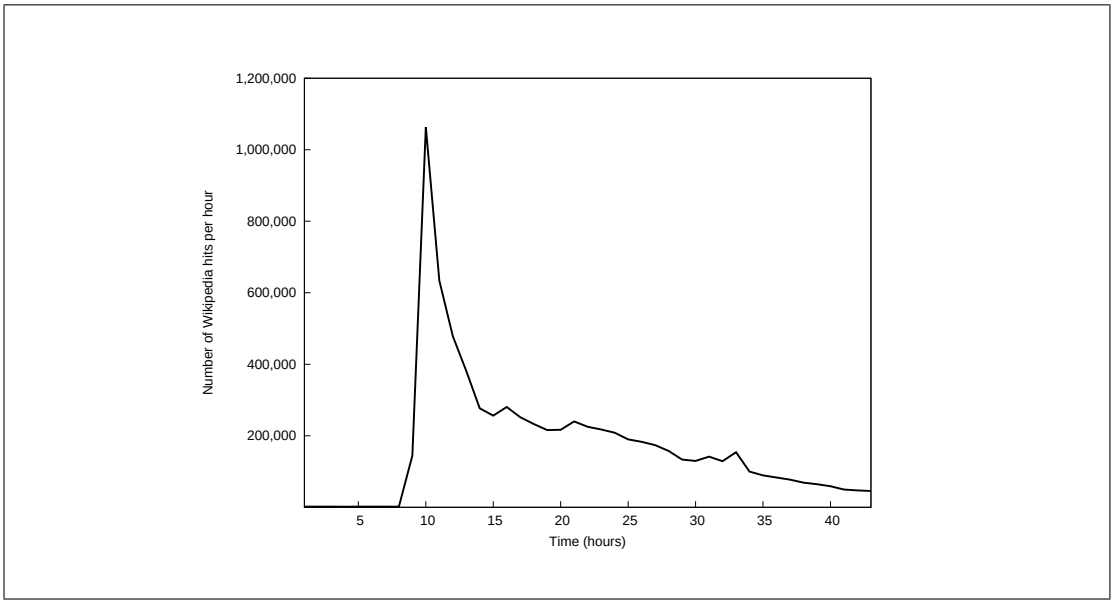


Figure 4.11: Incoming traffic profile for an unpredictable FE dataset: hourly hits on Wikipedia following the death of Steve Jobs.

4.2.3 Secondary Flash Events

The third category of FE is what this work calls a *secondary* FE (henceforth sFE). These events usually occur when a brief article, along with a web-link is posted on widely followed websites like Slashdot or Facebook. This link is often related to an interesting news item, although not as newsworthy an event on a world-wide scale as a uFE. This can capture the attention of a large number of followers and redirect a high percentage of them to another website in search of additional information. When these (usually user-posted) articles contain links to poorly resourced websites, they can easily result in the redirection of an unprecedented amount of traffic to those small websites, which exceeds their available resources and eventually cripples them. Once again, the event (article posting) is unpredictable, and the peak-load time is likewise relatively difficult to predict. Provisioning for such events can be challenging but is more feasible than for uFEs, due to the smaller nature of the event.

Anderson [10] shows the incoming traffic patterns of an sFE when the geological department web-server at Southern Methodist University was listed on the front page of Slashdot on June 15, 2003 and August 10, 2003. While the former posting on Slashdot briefly described their research on a two-wheel balancing robot, the later posting was on 'two-headed' virtual web servers, thus attracting a large number hits from people interested in technology. The receiving server,

on both the occasions, experienced a *'phase-transition'* from virtually no traffic to heavy traffic in a short time. Thus, in the case of this sFE, the incoming traffic pattern was similar to that of a uFE, as shown in Figure 4.11.

Table 4.3: Flash Event Classification.

FEs	**Predictable Start-time**	**Predictable Peak-load Time**	**Typical Peak Value**	**Provisioning**	**Occurrence Frequency**
pFE	Yes	Yes	Medium	Feasible	High
uFE	No	No	High	Not Feasible	Low
sFE	No	No	Medium	Not Feasible	Moderate

Table 4.3 summarises the classification of FEs in terms of Predicted Start-time, Predicted Peak-load Time, Typical Peak Value, Provisioning and Occurrence Frequency. Most pFEs are against web-servers which can afford the cost of web-content outsourcing though CDNs and other load sharing mechanisms. Hence, even during peak load they are not as badly affected as those subjected to sFEs and uFEs, e.g. a Slashdotted website or various news websites after a large scale natural calamity.

4.3 Flash Event Model

The server-side FE model presented in this section is based on the hypothesis that an FE can be described in terms of the following three components: change in the rate of Incoming Traffic, change in the rate of Source IP Addresses, and change in the pattern of resources accessed during the FE. The model is a variation of the three parameters proposed in Section 4.1 to differentiate FEs from DDoS attacks. The differences are the use of changes in the number of source IPs per sampling interval, instead of NSP IPs per sampling interval, and the use of changes in the web-resource access pattern, instead of the traffic distribution across IPs. The web resource access pattern was not used in the set of parameters used earlier to differentiate DDoS attacks from FEs (Section 4.1) because it is not present in the DDoS attack datasets, although it can be derived from (the FE) web-server logs. While this chapter proposes a general server-side FE model, it is acknowledged that not all components used in the model apply equally to all types of FE.

In the following section the model is developed with respect to these three components. The relation of each component to the overall FE model is discussed

in the following sections. However, a full mathematical model is only developed for the change in the rate of incoming traffic.

4.3.1 Change in the Rate of Incoming Traffic

In contrast to a DDoS attack, during an FE (or more precisely during a secondary or unpredictable FE) it is unlikely that the entire web community is simultaneously informed and goes online to get additional information. It usually takes time for the news that is the root cause of an FE to spread across the world. On the other hand, during a predictable FE, the section of the web community interested in that event has prior information about its occurrence, and hence concurrently goes online to access the related information. In each case, however, even though the incoming traffic to the web-server increases sharply, the results show that it is generally not as immediate as in a DDoS attack, assuming that it does not mimic an FE. Figure 4.3 shows the rate of incoming traffic for the CAIDA 'DDoS Attack 2007 Dataset'[51]. Contrasting this with Figures 4.2 and 4.11 shows that the rate of increase in incoming traffic for this DDoS attack at least is much higher than for an FE. (Note that whereas Figure 4.3 shows Time in 'minutes', Figures 4.2 and 4.11 show it in 'hours').

Let A_f or the 'traffic amplification factor' be the ratio of the 'peak request rate' (R_p) to the 'average normal request rate' (R_n), as observed by a web-server during an FE, similar to the *shock_level* parameter of [11]. R_n and R_p are expressed as the number of incoming requests per unit time. The traffic amplification factor (A_f) may thus be written as:

$$\boxed{A_f = \frac{R_p}{R_n}} \tag{4.1}$$

A_f can be set to different values to simulate various FE scenarios. An FE is considered to have two major phases: a *flash-phase* and a *decay-phase*. A further 'plateau period' or *sustained traffic phase* between these two phases, as proposed in [11, 24], appears to be extremely short-lived and orders of magnitude smaller than that of the other two phases in the observed data. The analysis of real-world datasets in Section 4.4 also shows the absence of any such plateau period. The *flash-phase* is defined as the part of an FE where the web-server experiences a significant increase in the incoming traffic, at the end of which incoming traffic reaches its maximum. The duration of the flash-phase is denoted by ΔT_f and

represents the time it takes for the normal request rate (R_n) to reach the maximum i.e. R_p. *Decay-phase* is defined as the period of the FE when the incoming traffic to the web-server subsides and goes back to 'normal' at the end of the phase, and the time taken to do so is written as ΔT_d.

There appears to be a difference in the relationship between the durations of these two phases for the three different types of FEs. In the case of a pFE such as a football match (and indeed many sporting events), the event has a duration, from the start through to its completion. So while the 'event of interest' is the final score, which should arguably correspond to the peak load, there is arguably nonetheless an increasing load throughout the whole of the event. At a point close to event completion, one would expect the traffic volume to peak, and the rate of incoming requests to the web-server to quickly decline and return to 'normal'. If the web-server has difficulty in handling the peak volume of requests, then increases in the server response time owing to network congestion may force users onto secondary sources of information, thus causing a still faster decline in the incoming traffic rate. For these reasons it is posited that for a pFE the decay time is comparable and possibly less than the build-up time. Thus, for a pFE

$$\Delta T_f \geqslant \Delta T_d$$

However, in the case of uFEs and sFEs, there is no such time-based 'end of event'. Long after the actual occurrence of the event, some section of the web-user community, previously unknowing of it, will become interested in it. Thus, there is a slow and gradual decay in the incoming request rate for sFEs and uFEs as compared to pFEs. Therefore, it can be said that the *flash-phase* has a much smaller duration than the *decay-phase* in sFEs and uFEs. Hence, for a sFE and uFE

$$\Delta T_f \ll \Delta T_d$$

The relationship between the durations of these two phases – *flash-phase* and *decay-phase*, especially for unpredictable and secondary FEs, is similar to that proposed by Ari et al. [11]. Their proposed model focused on modelling the duration of each phase (ramp-up, sustained and ramp-down) for unexpected FEs directed against 'poor' websites that cannot afford CDNs. Their model (Figure 4.12) defines the duration of ramp-down phase ($t_3 - t_2$, ΔT_d in our case) as 'n' (constant) times the sustained-phase duration ($t_2 - t_1$).

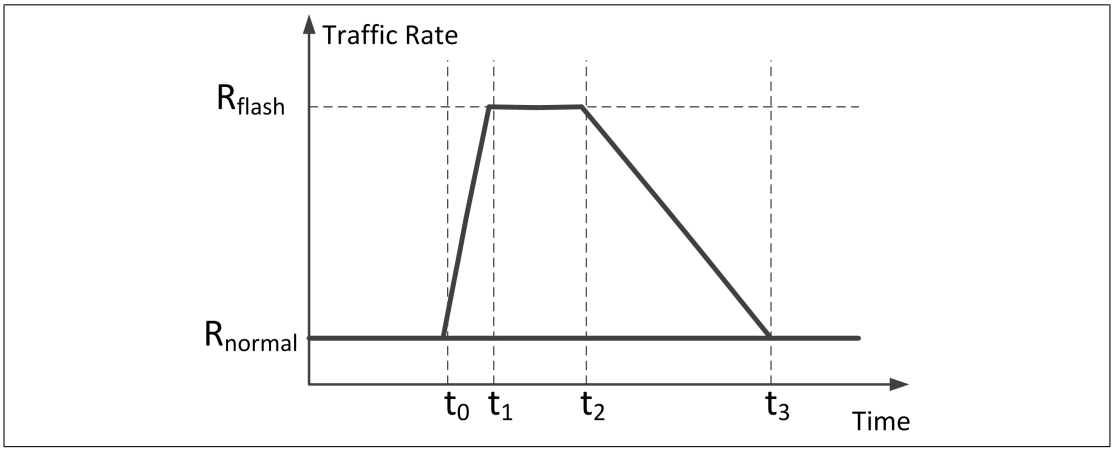


Figure 4.12: Traffic profile of an FE model proposed by Ari et al. [11]

The two phases of an FE, the *flash-phase* and the *decay-phase*, are now discussed in detail, and separately modelled.

Flash-phase

During an FE, the excess load on the server is mainly due to an increase in the overall number of clients accessing web-resources rather than in the number of requests per client [54]. Apart from a small percentage of enthusiastic clients, the majority of the clients participating in an FE are mostly interested in a very specific set of information related to that event [54]. Thus, it is expected that the average number of requests per participating client remains relatively constant during an FE.

When a newsworthy event occurs, as in the case of a uFE or an sFE, while it takes finite time for the news related to the event to spread across the world, it does so as a result of more and more people getting interested and propagating the information. This leads to a dramatic increase in the interested user population, which can be represented in terms of the classic exponential growth model used in many other domains such as worm propagation [132]. The exponential growth in user population appears intuitive for uFEs and sFEs more so than for pFEs. However, the proposed model has been tested against all three types of FE, and the experimental results are presented in Section 4.4.

These two propositions of relatively constant per client request rate and exponential growth in the interested client population lead to the increase in the rate of incoming requests being proportional to the current rate i.e. the rate of

incoming requests varies with time (t) as:

$$\frac{dR}{dt} = \alpha R \tag{4.2}$$

where α is defined as the *flash-constant*.

Integrating both sides w.r.t time (t) yields the following equation:

$$\ln R = \alpha t + C \tag{4.3}$$

where C is the constant of integration.

Assuming that the flash-phase starts at time $t = t_o$ where $R = R_n$ and substituting these values into 4.3 we get:

$$\ln R_n = \alpha t_o + C \tag{4.4}$$

Using 4.3 and 4.4 we get:

$$\begin{aligned} \ln R - \ln R_n &= \alpha(t - t_o) \\ R &= R_n e^{\alpha(t-t_o)} \end{aligned} \tag{4.5}$$

The end of the flash-phase is marked by time $t = t_f$ where the incoming request rate reaches the peak, i.e. $R = R_p = A_f \times R_n$ (using Equation 4.1). Substituting these values gives the flash-constant (α) as:

$$\alpha = \frac{\ln A_f}{(t_f - t_o)}$$

Substituting α in 4.5, the value for the incoming traffic during the *flash-phase* (i.e. for $t_o < t \leqslant t_f$) is obtained as:

$$\boxed{R = R_n e^{\frac{\ln A_f}{(t_f - t_o)}(t-t_o)}} \tag{4.6}$$

Flash-phase.

The start of the flash-phase i.e. t_o and the time of the peak request rate (R_p) can be determined *post hoc*, visually or in real-time using a change detection technique discussed in Chapter 2. The exponential increase of the incoming traffic during the flash-phase differs from the research presented in [11, 24] which

models the ramp-up phase as a linear function, while [123] defines flash crowds using a quadratic growth model.

Decay-phase

At the start of this phase, the number of incoming requests is at its peak and starts to decline. Three possible reasons are speculated for this behaviour.

1. by this time most interested users have the information they were looking for and have thus moved on.

2. the main web-server hosting the information reaches its serving capacity and thus starts rejecting new connections.

3. a prolonged response time from the primary web-server starts to annoy users, forcing them either to return later or to look for other sources of information (secondary servers).

The last point intuitively suggests that a growth, possibly exponential, in the number of secondary servers, providing the same information as the primary server, contributes to the exponential decay in the number of incoming requests to the main web-server. Hence, it is argued that during the *decay-phase*, the rate of incoming requests (R) decreases with time (t) as follows:

$$\frac{dR}{dt} = -\beta R \tag{4.7}$$

where β is a *decay-constant*. Integrating and substituting the boundary conditions (at $t = t_f, R = R_p$ and at $t = t_d, R = R_n$) the decay-constant (β) can be specified as:

$$\beta = \frac{\ln A_f}{(t_d - t_f)}$$

and, the incoming traffic model during the *decay-phase* i.e. for $t_f < t \leqslant t_d$ as:

$$\boxed{R = R_p e^{\frac{-\ln A_f}{(t_d - t_f)}(t - t_f)}} \tag{4.8}$$

Decay-phase.

where $R_p = A_f \times R_n$ (from Equation 4.1). The exponential decay model of incoming traffic during the decay-phase (Equation 4.8) differs from [11, 24],

which show a linear ramp-down and [123], which models only the ramp-up phase. For the duration before the *flash-phase* ($0 < t \leqslant t_o$) and after the *decay-phase* ($t > t_d$) it is assumed that the rate of incoming requests remains constant, i.e.

$$\boxed{R = R_n} \tag{4.9}$$

Pre Flash-phase and Post Decay-phase.

In summary, the model describes the incoming traffic volume for each of the two main phases of an FE i.e. *flash-phase* and *decay-phase* in terms of exponential growth and exponential decay. An FE is represented as a set of three equations (4.6, 4.8 and 4.9) with few configurable parameters (R_n, R_p, t_o, t_f and t_d), which can be tuned to represent different FEs.

4.3.2 Change in the Rate of Source IP Addresses

The dramatic increase in the number of incoming requests during an FE can be attributed to either a substantial increase in the number of requests per participating client, or to an increase in the overall population interested in that particular event, or a combination of the two. Based on two FE datasets, Jung et al. [54] concluded that the per-client request rate does not increase, but rather drops and remains lower during FEs compared to other times. The analysis of a pFE dataset presented in Section 4.4 of this chapter shows somewhat different results. It shows a slight increase in the number of requests per client with the onset of a FE. But it also shows that the overall increase in incoming traffic volume is mainly due to an increase in the interested population.

The analysis of a publicly available dataset presented in Section 4.4 shows how the rate of source IPs increases with the onset of an FE (i.e. at $t = t_0$) and continues to increase during the *flash-phase* before starting to decline over the *decay-phase*. Thus, during the course of an FE, variations in the number of source IPs would be expected to closely resemble variations in the incoming traffic volume. Nonetheless, there are some reasons why the model for traffic rate may not necessarily be applicable to source IPs, there are some complicating factors. Firstly, the number of requests per client may not be constant in all cases, and secondly, IP mapping (NAT, DHCP) may have the effect of invalidating the assumption that the number of clients is equivalent to the number of source IPs, thus affecting the assumption of constant packets per IP. The latter reason is

similar to a previously discussed (see Section 4.1.3 for details) characteristic of FEs where the majority of traffic appeared to be generated by a small percentage of participating clients or source IPs, suspected to be network proxies hiding a large number of private IPs behind a comparatively small set of public IPs.

4.3.3 Change in the Pattern of Resources Accessed

The third component that has been used to understand and model FEs is the 'randomness of the resources' being accessed. It is hypothesised that during an FE, the majority of the participating population are normally interested in a very specific set of information relating to that event e.g. results of a soccer match (pFE), an article on a news website related to an unpredictable event like 9/11 attacks (uFE), or an interesting story on a website referred from Slashdot or Facebook (sFE). In other words, this translates into a greater concentration of resources being accessed during the event as compared to the pre-event or post-event period.

The analysis of a publicly available dataset presented in Section 4.4 confirms this hypothesis in the case of a predictable FE. The number of different resources being accessed during the course of a pFE decreases considerably compared to other times. Even though the trend in resources being accessed during unpredictable and secondary FEs is expected to be similar to that of predictable FEs, this cannot be verified due to the lack of publicly available datasets containing 'resource accessed' information for unpredictable and secondary FEs.

Entropy, or the degree of randomness, has been used as the metric to measure the resources accessed component of the proposed FE model. Section 4.4 presents the analysis of this component. The (normalised) resource entropy starts to decrease, from its average value with the onset of the FE, and continues to drop for the rest of the *flash-phase*, when most of the interested population start looking for specific information and thereby access a finite set of resources. Once the event peaks, at the start of the *decay-phase*, the majority of the interested population has already got hold of the information they were after, and hence the randomness of resources being accessed starts to increase until the end of this phase, before returning to normal.

4.4 FE Model Evaluation and Validation

Legal and privacy issues associated with the publication of real-world datasets results in only a very limited number of datasets being publicly available. Although some of these are old, they represent real FEs, mostly pFEs, and exhibit some, unfortunately not all, of the key characteristics that might be used to define an FE. This section uses the datasets described in Section 4.4.1 below to validate the FE model to the extent possible given the limitations of the datasets.

4.4.1 Datasets

The datasets used in the FE model analysis are described below and summarised in Table 4.4.

Predictable FE (pFE) Dataset: 1998 FIFA World Cup A subset of the 1998 FIFA World Cup Dataset [12] (henceforth referred to as the pFE-dataset) described in Section 4.1.4 is used as the base predictable FE dataset in this thesis in general. In this section, pFE-dataset contains approximately 20 hours of incoming traffic around the 1^{st} semi-final match of the 1998 FIFA World Cup is used for FE model analysis and evaluation. Similar results were obtained in analysing the 2^{nd} semi-final match of the same tournament.

Unpredictable FE (uFE) Dataset: Hourly Hits on Wikipedia The 'Hourly Hits on Wikipedia' (henceforth referred to as the uFE-dataset) is used as the base unpredictable FE dataset for the analysis. This uFE-dataset, briefly discussed in Section 4.2.2, was provided by Domas Mituzas, who maintains a repository of page view statistics for various Wikimedia projects[2]. The records following Steve Jobs' death were selected as representative of a uFE. The dataset provides hourly aggregates of requests and covers a period of 43 hours (starting 16:00 hours UTC, 5^{th} October 2011) as shown in Figure 4.11. Similar traffic was also observed following the death of Michael Jackson [73].

Secondary FE (sFE) Dataset: Slashdotted Website Since there were no public domain datasets representing an sFE, high-level statistics of the empirical

[2]http://dumps.wikimedia.org/other/pagecounts-raw/

Table 4.4: Datasets Used for Analysing Flash Event Model.

Datasets	Duration (hours)	Granularity	Number of Packets	Number of Source IPs	Number of Unique Resources	FE Type
1^{st} Semi-final (1998 FIFA World Cup)	20	seconds	42,373,729	160,704	38,892	pFE
Wikipedia Hits (Steve Jobs)	43	hours	7,417,070	N/A	N/A	uFE
Slashdotted Website (15^{th} June)	80	hours	1,042,751	N/A	N/A	sFE

data available from [10] was used. This dataset is incomplete, as it lacks information related to source IP addresses and the resources accessed, but it still shows the incoming network activity on the web-server of the Department of Geological Sciences at the Southern Methodist University after their website was listed on the front page of Slashdot on June 15, 2003 and August 10, 2003, following the posts related to a two-wheel balancing robot, and two-headed virtual web servers respectively. Hence, only the traffic volume component can be validated. Statistical information of the Slashdot event occurring on June 15, 2003 has been used for the analysis.

Analysis of the above datasets in terms of the three components that have been used to model an FE i.e. change in Rate of Incoming Traffic, change in rate of source IPs generating that traffic, and change in pattern of web-resources being accessed during the FE, is now presented.

4.4.2 Change in the Rate of Incoming Traffic

Figure 4.13 shows the actual incoming traffic volume over a 20-hour period around the 1^{st} semi-final match during the 1998 FIFA World Cup. The incoming traffic volume is represented as the number of packets received per one minute sampling interval. The rate of incoming packets before and after the FE is roughly constant. This feature is represented by equation (4.9) of the proposed model. The match starts around the 420 minute mark on the x-axis (t_o), continues for the next 162 minutes (ΔT_f), before ending at around the 580 minute mark (t_d). The 1^{st} semi-final match extended into a penalty shoot-out phase.

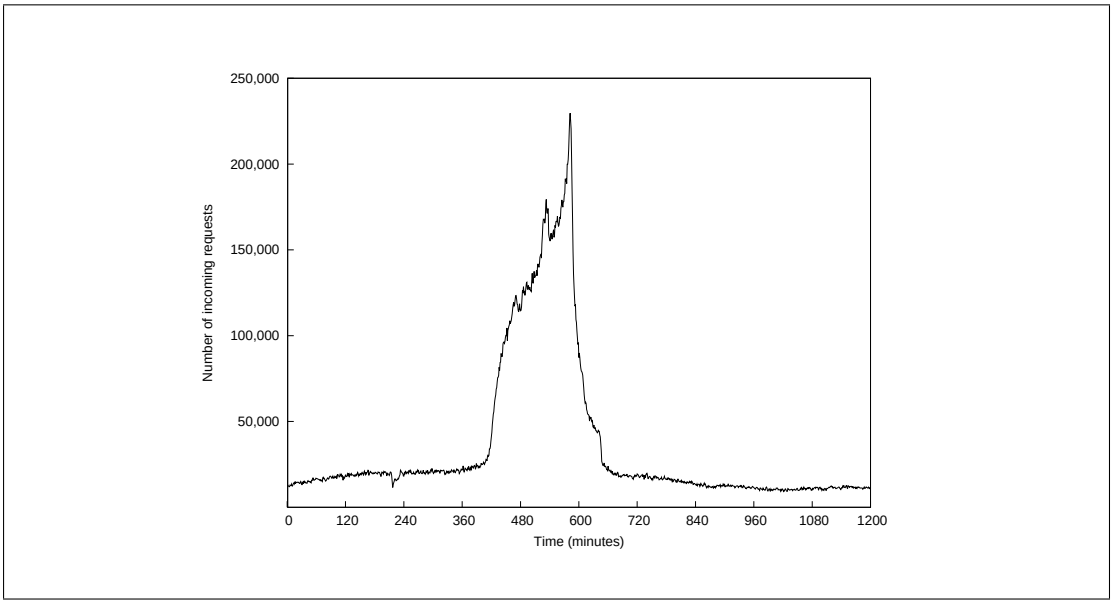


Figure 4.13: Incoming traffic during the 1^{st} semi-final match of the 1998 FIFA World Cup dataset.

During the *flash-phase*, R_n increased by a factor of nearly twelve.

Except for the FIFA World Cup dataset, the other datasets were only available with hourly time resolution. Hence, for comparison purposes, a sampling interval of one hour has been used for the FIFA World Cup dataset to find the values of the model parameters (R_n, R_p, t_o, t_f, t_d) as shown in Table 4.5. The time-based parameters (t_o, t_f, and t_d) indicate the instances of time (in hours) when different phases started and ended. Although these instances of time were determined visually, they could also be detected in real-time by using the Change Point Analysis (CPA) technique discussed in Chapter 2.

The request rates (R_n and R_p) were calculated (for pFE) and observed (for uFE and sFE) on a per-hour basis for comparative analysis. (Due to the nature of the dataset available for uFE (hourly aggregates) and sFE (high-level statistic of the empirical data), the request rates were observed for these FEs, as compared to pFE-dataset, where the rates were calculated from the data available as web-server logs in Common Log Format (CLF).) Table 4.5 also shows the values for derived parameters viz. the incoming Traffic Amplification Factor (A_f), α (flash-constant), β (decay-constant), and the duration of flash and decay phase i.e. ΔT_f and ΔT_d, for the three datasets used.

It is interesting to note that the incoming traffic amplifies (A_f) by nearly 600 times in case of the unpredictable FE (uFE) e.g. the hourly hits on Wikipedia

Table 4.5: Flash Event Model Parameters.

FEs	R_n (requests per hour)	R_p (requests per hour)	t_o (hours)	t_f (hours)	t_d (hours)	A_f	ΔT_f (hours)	ΔT_d (hours)	α	β
pFE: 1998 FIFA World Cup (1^{st} Semi-final)	1,167,621	9,898,273	7	10	12	8.47	3	2	0.71	1.06
uFE: Hourly hits on Wikipedia (Steve Jobs)	1,826	1,063,665	8	10	43	582.51	2	33	3.18	0.19
sFE: Slashdotted Website (15^{th} June)	1,000	78,000	13	15	71	78.00	2	56	2.17	0.07

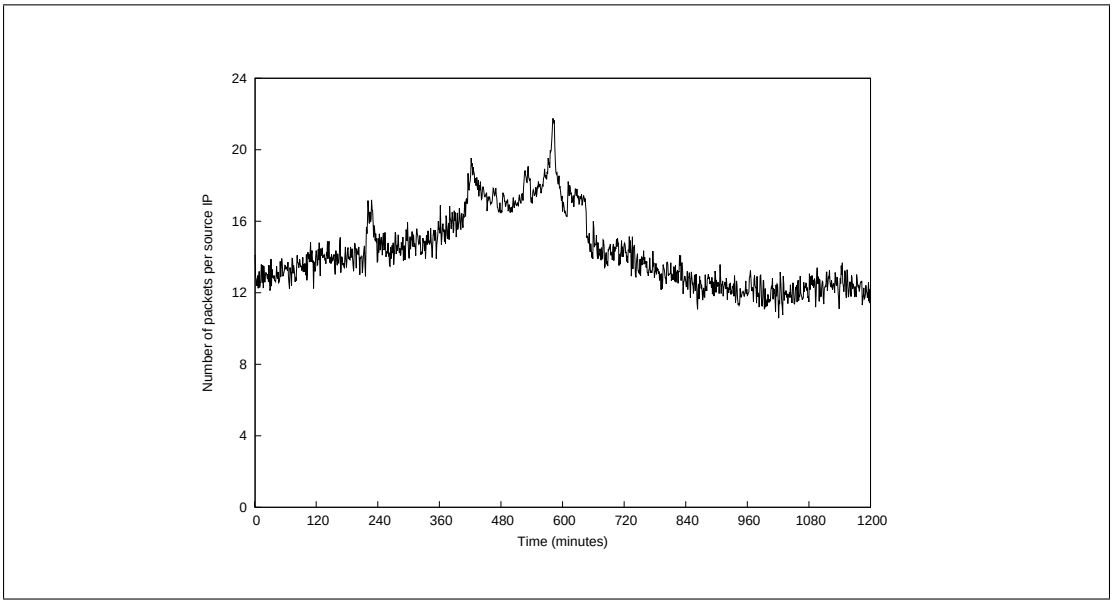


Figure 4.14: Packets per Source IP during the 1^{st} semi-final match of the 1998 FIFA World Cup dataset.

following the death of Steve Jobs. This substantial increase in incoming traffic, combined with the relative infrequency and unpredictability of such an event, makes it less feasible to have any provisioning for such events. Another observation is the rate at which the incoming traffic increases and decreases for each FE as determined by α (flash-constant) and β (decay-constant) respectively. These constants have comparable values for the two pFEs, whereas in sFEs and uFEs they differ substantially. In the latter cases the incoming traffic tends to fade-off at a much slower pace as compared to the build-up phase, whereas for the pFEs the incoming traffic appears to increase and decrease at similar speeds. These parameters are configured and used in the following chapter to generate different types of FE traffic, to mimic real-world scenarios.

4.4.3 Change in the Rate of Source IP Addresses

As mentioned previously, it is expected that the per-client request rate will be almost constant during the event. The analysis of the 1^{st} semi-final match dataset, however, somewhat deviates from this speculation, and instead shows an approximately 18% increase in the average per-client request for the duration of the FE (including both the *flash-phase* and the *decay-phase*), as compared to the pre-flash-event times i.e. before the start of the *flash-phase*. The average per-client request rate during the *flash-phase* increases even more, by nearly 24%, compared

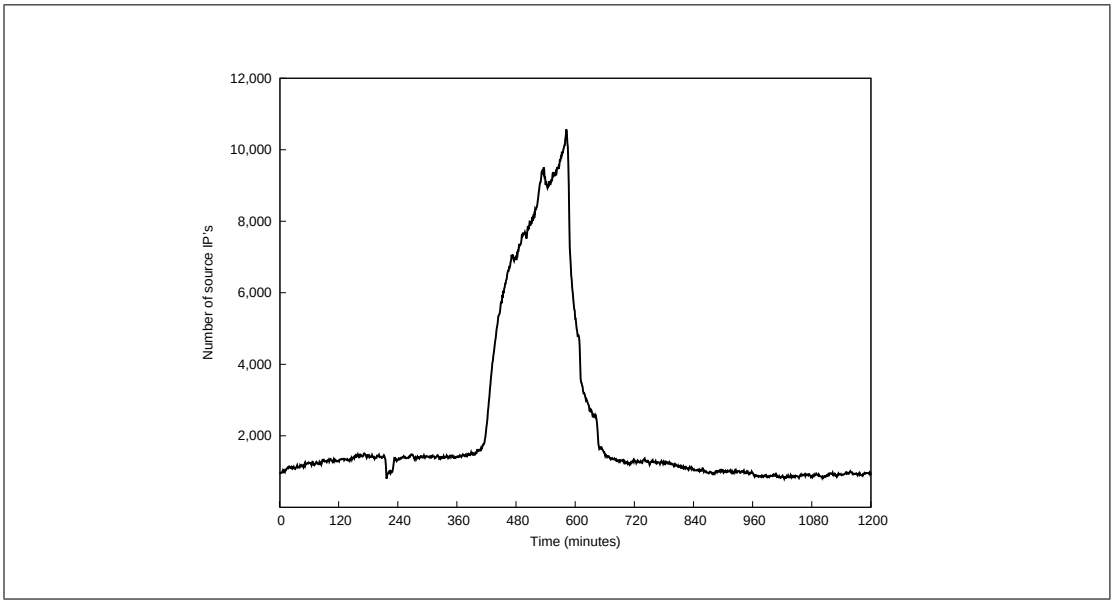


Figure 4.15: Source IPs during the 1^{st} semi-final match of the 1998 FIFA World Cup dataset.

to the pre-flash-event times. This increase in the average per-client request rate could perhaps be attributed to the 'enthusiastic section' of the interested population trying to get additional information regarding the event, as compared to others. Figure 4.14 shows the number of requests per source IP against time for the 1^{st} semi-final. Notwithstanding this increase, it is the overall increase in the participating population that causes the dramatic surge in the incoming traffic during the FE. Figure 4.15 shows the rate of increase and decrease of source IPs during the event. It varies in a similar fashion to the incoming traffic (Figure 4.13).

Thus it can be concluded that during an FE, the majority of the traffic increase is due to an increase in the overall participating population, although there is also a slight increase in the per-client request rate as compared to the non-flash period. This result differs from [54], which states that the per-client request rate drops and remains lower during the flash period as compared to other time periods.

4.4.4 Change in the Pattern of Resources Accessed

As the news causing the FE spreads across the Web, more and more people go online to get the related information. It is argued that a majority of the participating population is generally interested in obtaining a specific set of information

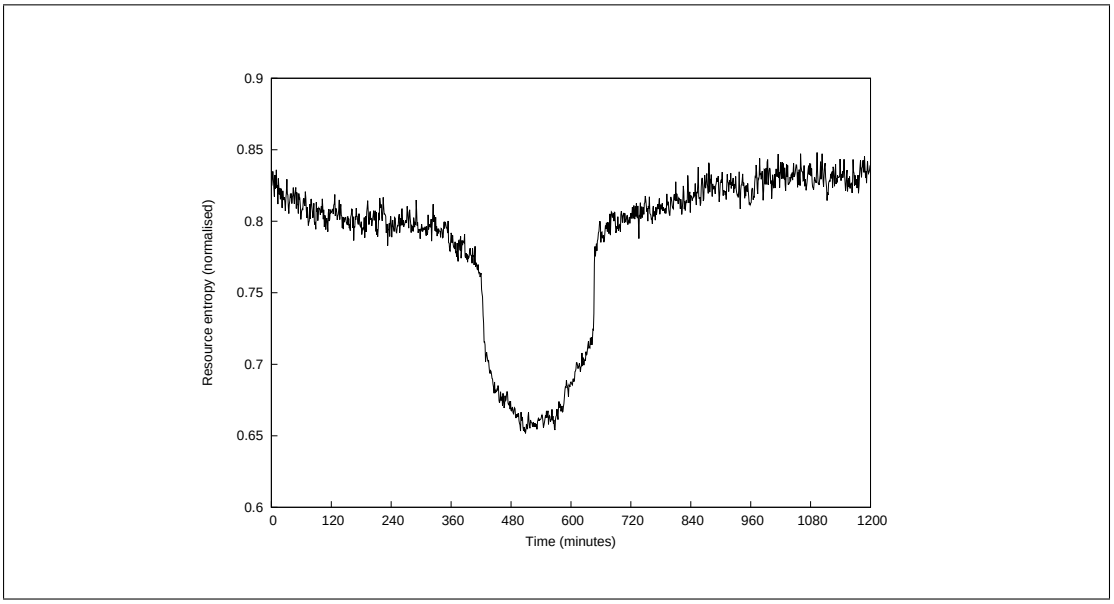


Figure 4.16: Resources access pattern during the 1^{st} semi-final match of the 1998 FIFA World Cup dataset.

from the web-server. Thus the number of different resources being accessed by the client during a FE should be lower than during non-flash-event times. One way to measure this attribute is Shannon's entropy [103], a measure of the uncertainty associated with a random variable. The entropy H of a discrete random variable X with n possible outcomes $x_1, x_2...x_n$, measured in *bits*, is given by

$$H(X) = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i)$$

where $p(x_i)$ is the probability associated with the outcome x_i. The entropy $H(X)$ is minimum at 0 when all the values x_i have a probability 1, i.e. there is no uncertainty with the outcomes of the random variable X. The entropy attains a maximum value of $log_2 n$ when all the values x_i have equal probability $1/n$ i.e. a maximal uncertainty about the outcomes of the random variable X. Dividing by $log_2 n$ gives the normalized entropy $H_0(X)$ with values ranging between 0 and 1, and is defined as:

$$H_0(X) = H(X)/log_2 n$$

For the analysis of the accessed resources component of the proposed FE model, random variable X represents a *unique web-resource* being accessed and the probability p_i of each resource represents its relative frequency of occurrence

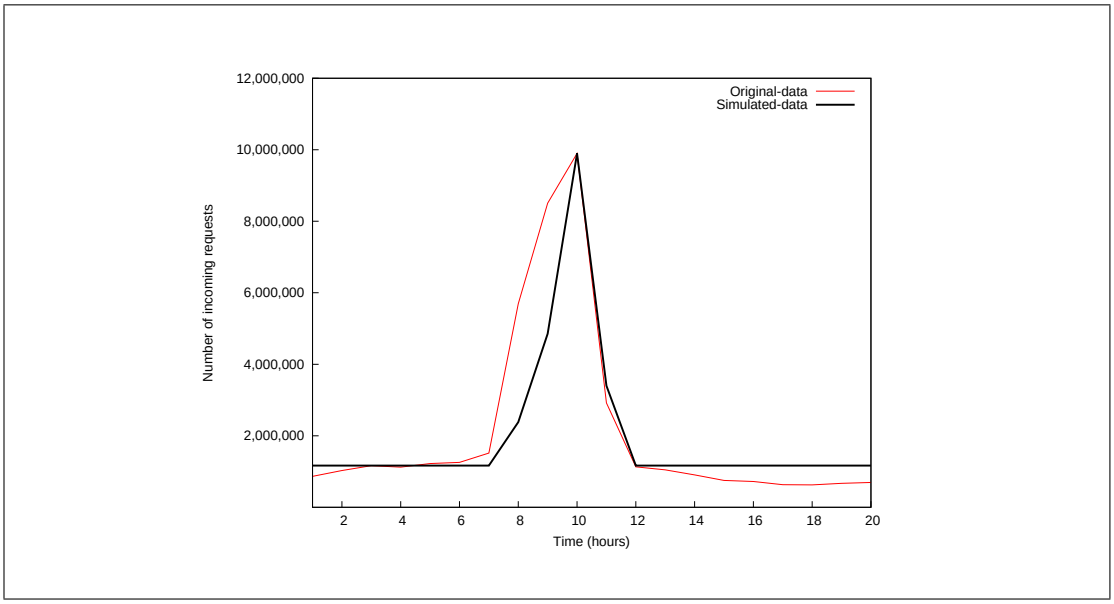


Figure 4.17: FE model validation for a predictable FE (1^{st} Semi-final of the 1998 FIFA World Cup).

in each one minute interval. Experiments conducted for one-second and five-minute intervals gave similar results. Figure 4.16 shows a drop at the onset of the event in the normalised resource entropy which stays low for the entire duration of the event.

4.4.5 Model Validation

This section reviews the above results in order to attempt to evaluate the extent to which they validate the proposed model. Unfortunately, the datasets do not provide all the information required to do this in any comprehensive fashion. Due to the fact that the sFE and uFE datasets provide neither IP nor resource information, the evaluation is necessarily limited to the first feature, namely the change in the rate of incoming traffic. Also since the dataset for an unpredictable FE (Wikipedia hits following Steve Jobs' death) had 'hours' as the granularity level, the same granularity level (hours) has been used for the predictable FE (1998 FIFA World Cup) dataset.

Figure 4.17 plots the number of incoming requests against time for the real pFE data and the synthetic data generated using the proposed FE model. During the *flash-phase* of the original pFE data, a 'belly' shape is observed between time 7 and 10 hours. It seems to reflect a faster initial increase in the incoming traffic in the original data than the proposed model. This belly shape is followed by a

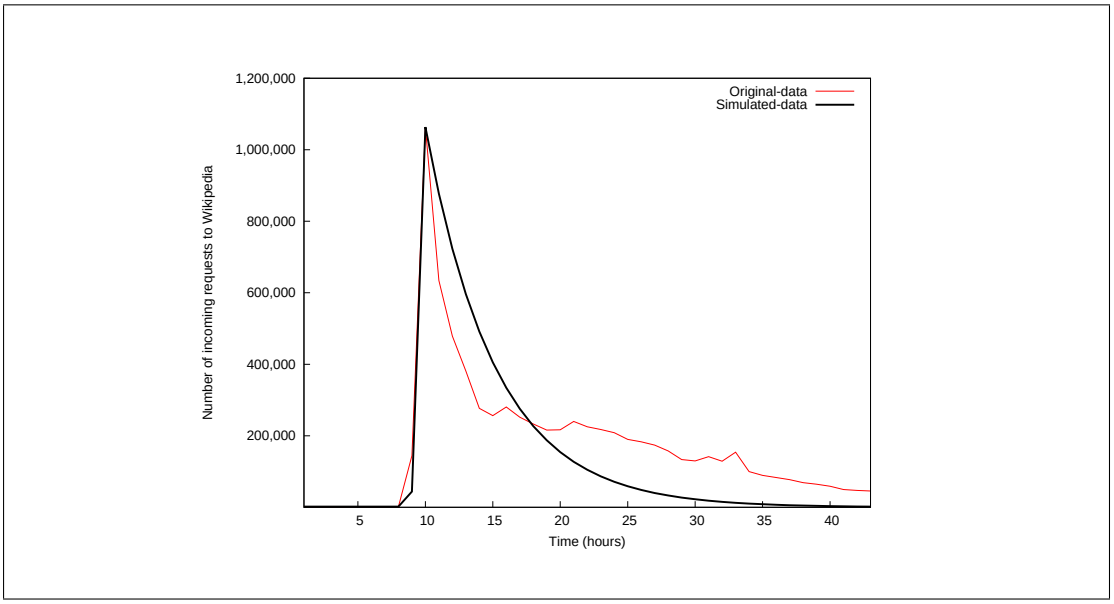


Figure 4.18: FE model validation for an unpredictable FE (hourly hits on Wikipedia following the death of Steve Jobs).

tailing off towards the peak. This deviation is discussed in the future work section of the final chapter of this thesis. The proposed model, however, mimics the decay-phase of the pFE more closely than the *flash-phase*. It also conforms with the speculation made earlier about the relative absence of the sustained-phase. The proposed model when applied to the unpredictable FE data (Wikipedia hits following Steve Jobs' death), interestingly, follows the *flash-phase* more closely than the *decay-phase*. Figures 4.18 compares real and synthetic data for uFE generated using the proposed FE model for incoming traffic.

Notwithstanding the limitations imposed on the evaluation by the datasets, it seems that a simple exponential model, with few configurable parameters, can be used to capture the essential characteristics of an FE. This, it is argued, can then be used to generate realistic FE traffic, thereby facilitating research in FE detection and their differentiation from DDoS attacks.

4.5 Conclusion

The existing problem of accurate and reliable DDoS attack detection is aggravated by the existence of FEs, a network anomaly sharing a number of similar characteristics with DDoS attacks. Hence an accurate detection of FEs and their distinction from DDoS attacks is important. The research presented in this

chapter has addressed this problem by exposing some subtle differences between these two network anomalies, and has presented an FE model. The two main contributions of this chapter are:

- The development and evaluation of a technique for differentiating DDoS attacks from FEs, and
- A server-side model of an FE which enables synthetic traffic generation of different types of FE.

The proposed FE model is more detailed and precise than earlier proposed models, and has been partially validated against public domain datasets. It differs from models proposed by Ari [11] and Bodik [24] which presented the ramp-up and ramp-down phase as a linear function, and included a sustained traffic phase, which was not found in our analysis. It also differs from Wendell's model [123], which only models the ramp-up phase using a quadratic growth function.

Regarding the first contribution, it is acknowledged that there are marked dissimilarities between the datasets used for evaluation. However, the datasets used in the analysis are possibly the only publicly available and complete datasets representing a DDoS attack and an FE. The future work section in Chapter 7 addresses this issue by proposing an evaluation on different dataset examples, that are not markedly dissimilar. The validation of the FE model presented in this chapter is also significantly limited by the availability of real datasets representing different types of FEs. However, within those limitations it has been demonstrated that the proposed server-side FE model with minimal parameters can capture various characteristic properties of different types of FE, which can then be used to synthetically generate realistic FE traffic.

Chapter 5 uses the FE model presented in this chapter to synthetically generate FE traffic. Traffic generated using the model is then used in the evaluation of the ensemble-based DDoS attack detection technique developed in Chapter 6, which can potentially detect not only a variety of DDoS attacks, but also successfully distinguish them from FEs.

Chapter 5

Synthetic Traffic Generation

The work presented in this chapter addresses Research Question 4 identified earlier (Section 1.2) i.e. *Is it possible to design and implement a simple experimental traffic generation and testbed framework, using only modest hardware for the testbed and a software-only traffic generator, to synthetically generate realistic network traffic representing DDoS attacks and FEs?* This is an important question because one of the key challenges in DDoS and FE research is the scarcity of *recent* and *realistic* datasets, reflecting real-world DDoS attacks and FE scenarios, for testing and evaluation purposes. The datasets that are publicly available such as the KDD Cup 1999 Dataset [50] are not only old but also lack sufficient comprehensiveness to be suitable for evaluating contemporary DDoS scenarios [70, 116]. Existing testbeds for generating and evaluating realistic DDoS and FE traffic, such as DETER [17] and Emulab [124] have technical limitations relating to realism and the number of attackers/clients they support, and they are also shared facilities. The chapter addresses the above issues by focusing on:

- the development and evaluation of a traffic generation and testbed framework, consisting of a hardware testbed set-up and software traffic generation tool, and
- using this framework to investigate and develop techniques to synthesise realistic emulation of real-world DDoS attack and FE traffic.

The synthetic generation of DDoS attack and FE traffic is based either on the characteristics of some real-world datasets, available in the public domain, or

on known methodologies. The traffic generation and testbed framework uses the FE model developed in Chapter 4 to synthetically generate similar FE traffic. The datasets generated in this chapter for DDoS attacks, FEs and background traffic are then used for testing and evaluating an ensemble-based technique for detecting a variety of DDoS attacks and FEs, developed in Chapter 6.

The rest of the chapter is structured as follows. Section 5.1 provides the detailed characteristics of the datasets needed for this research. Section 5.2 discusses the details of the traffic generation and testbed framework consisting of a hardware testbed and software traffic generator – Botloader, cooperatively developed as a part of this research[1]. It also provides details of synthetic traffic generated using this framework and evaluates the effectiveness of the obtained results to see how closely they mimic real traces. Section 5.3 provides the simulation results of synthesising a real-world FE. Section 5.4 presents the simulation results of synthesising a real-world DDoS attack using the traffic generation and testbed framework. Finally, the chapter is summarised in Section 5.5.

5.1 Dataset Characteristics

An integral requirement for a workable solution for evaluating DDoS attack detection is the availability of realistic traffic datasets, whether attack or benign. For the purpose of experimentation and evaluation, it is essential that the datasets being used have two distinct components, viz., attack traffic, such as DDoS and non-attack traffic, such as an FE or normal background traffic. This section describes the characteristic properties of datasets required for testing and evaluating techniques developed in this thesis for detecting DDoS attacks and differentiating them from FEs.

- **Non-spoofed source and destination IP addresses.** In order to simulate realistic scenarios, especially in the case of application layer DDoS attacks and FEs, it is important that the host machines *interact* with the applications running on the target server. This interaction is feasible only

[1]The testbed was developed as part of the the Australia-India Strategic Research Fund (AISRF) project investigating Denial-of-Service 2008–2012. The candidate's contributions were to the original concept of using interface aliasing, a preliminary implementation using Curl-loader, Wget and ifconfig, and significant contributions to testing, configuration and design, especially of the FE plugin, and to the scripts for pre/post processing of data.

when there is a *valid* TCP connection between the source and the destination machines. And to ensure a valid TCP connection it is necessary that both the source and destination IP addresses are real and non-spoofed.

- **A wide range of source IP addresses.** Real DDoS and real FE traffic datasets exhibit a wide range of source IP addresses, thus network traffic originating from a wide spectrum of IP addresses forms an essential characteristic of the required datasets.

- **Network packets with valid content.** In order to closely approximate the real traffic generated by application layer DDoS attacks and FEs it is desirable that packets have not only packet headers but also valid content (data). Packets with valid content may also be required at the application level in order to elicit a meaningful response from the targeted service. This processing of incoming packets consumes resources (memory, CPU) which can be measured and used in the proposed ensemble-based DDoS detection technique.

- **A mix of normal and anomalous network traffic.** In order to effectively test and evaluate the DDoS attack and FE detection techniques, datasets must be constructed with a *controllable* mix of two distinct components, viz., normal traffic and attack traffic. The normal traffic can represent the background noise for DDoS attack and FE simulations.

In order to obtain datasets with the aforementioned characteristics, four possibilities were identified: using the data captured from a live production network, using network simulators such as ns–2 or Opnet, replaying the existing network traces available in the public domain, and using some of the popular off-the-shelf hardware and software traffic generators.

Capturing and using live network capture from the research lab (ISI) or the university (QUT) had significant legal and privacy concerns, especially while sharing the data with other collaborators involved in the project[2]. Hence, the use of live network traffic capture was not considered to be a valid an option for obtaining the desired datasets. As previously discussed in Chapter 2 (see Section 2.5.3 for details), network simulation programs such as ns–2, have sometimes been used to simulate DDoS attacks and measure their effects [21, 101],

[2]The research conducted as part of the Australia-India Strategic Research Fund (AISRF) project on DoS and DDoS attacks.

but the realism of a method in which the attackers, targets and network devices are all simulated for HRF attacks in particular seems doubtful, and has recently been called into question [72]. Therefore, the use of network simulators, as an alternative for obtaining the desired datasets, was also discarded.

Chapter 2 highlighted the limitations identified in the direct replay of existing publicly available network traces, and in using some of the popular off-the-shelf traffic generators (see Section 2.5). A detailed review of the latter approach is given in Appendix B. The research work involved in these two approaches, however, laid the foundations for the development of the traffic generation and testbed framework. The framework uses modest hardware and a custom-built traffic generator, guided by traffic characteristics extracted from public domain datasets. The traffic generation and testbed framework will now be discussed in detail.

5.2 Traffic Generation and Testbed Framework

The work presented in this section addresses Research Question 4. This relates to designing and implementing a simple experimental traffic generation and testbed framework, comprising the testbed facility (aka the hardware platform) and the software traffic generator, which can together be used to synthetically generate realistic network traffic for DDoS attacks and FEs.

Given the limitations of the various approaches investigated for obtaining realistic datasets, one obvious way around this problem was to develop such a customised traffic generation and testbed framework to meet the identified requirements. This framework had to be based on available hardware and a customised software traffic generator able to simulate a wide range of network and application layer DDoS attacks and FEs. This section describes both the hardware and traffic generation components of the testbed.

5.2.1 Framework Design Requirements

The following design-level requirements were identified for developing the framework.

- **Maximum resource utilisation.** A single attacking host should be able to generate sufficient traffic to impede a server of similar power from servic-

ing its clients. Thus, orchestrating a number of such host machines should have a substantial impact on a large server, used as the target host.

- **Traffic load coordination.** Each attacking host should send a relatively small portion of the overall traffic. Thus, an attacker would combine the strength of all participating hosts. This feature would cater for coordinated attacks, but at the same time would require a so-called 'control machine' for remotely issuing commands to the attacking machines and synchronising their activity.

- **Traffic aggregation.** Since each of the participating hosts would be a standard PC with limited capabilities, in order to simulate a variety of volumetric DDoS attacks and FEs, the outgoing traffic from all of the participating hosts (attackers or legitimate clients depending on the scenario) would need to be aggregated to a single network interface before arriving at the target server.

- **Modest hardware.** Given the financial constraints of the project, it was necessary to develop a testbed facility using only modest hardware. It is acknowledged that the scale of simulated attacks can be increased by using high-end networking devices. However, at the same time, many meaningful experiments can still be conducted using modest equipment and customised software, for example, by comparing the results of different configurations.

- **Reconfigurability.** The framework is required to simulate a variety of DDoS attacks, both at the network and application level, and a variety of FEs, and as a result must allow for reconfigurability of the network topology.

- **Monitoring capabilities.** The framework should make it possible to monitor various intermediate nodes such as switches, firewalls, and end nodes such as applications running on the target hosts in real time.

5.2.2 Botloader: a software traffic generator

In order to simulate various types of DDoS attacks and FEs it was decided to make use of multiple autonomous agents. At the conceptual level, Botloader, running separately on every host machine (Figure 5.2), uses the technique of

IP-aliasing to create a large number of bots (or legitimate clients in the case of FEs), each with a unique source IP, and uses them to synthetically generate network traffic based on a user-specified profile (Figure 5.1).

IP aliasing is a well-known technique for associating multiple unique IP addresses to a single network interface card, and is available on most computing platforms. This use of IP-aliasing allows the creation of a large number of different data flows originating from a single physical machine, creating the impression (to the target) as if they were coming from different physical machines.

Bots, Clusters and Modules: the building blocks of Botloader

Each of the unique aliased-IP addresses created by Botloader is called a *bot*. The bots are referred to as 'compromised machines' (or real bots) when simulating DDoS attacks and as 'legitimate clients' during FE simulation scenarios. In software terms, a bot is like an 'object' that consumes a small amount of memory (approximately 46 bytes, plus whatever is allocated by the operating system for its socket and its alias), and performs a fixed set of functions via an Application Programming Interface (API). All bots, irrespective of their types, have the same API, thus allowing them to be uniformly controlled by the Botloader program.

A group of bots performing similar operations is called a *cluster*. All the bots within a cluster share the same user-specified configuration, such as the IP address of the target machine, the duration of the attack or FE, the data rate (kilobytes) of the attack, or the resource map to be used in case of HTTP-based DDoS attacks and FEs. The use of clusters presents two obvious advantages: first, aggregating bots with similar functionalities avoids duplication of code, and secondly it keeps the memory requirement of the Botloader program low. Each cluster is assigned a separate execution thread to facilitate the simultaneous execution of different types of attack or background traffic. The Botloader program itself has its own main execution thread and controls the various types of clusters.

A *module* is essentially a library or 'plugin', comprising of a set of functions that define a given attack type and are shared by all the bots within a cluster. Each attack type is carried out by a different module (also referred to as bot-type). On start up, Botloader reads the configuration file and searches for available modules or bot-types. Figure 5.1 shows an abstract level representation of Botloader with bots, clusters, and modules. Modules come in two basic

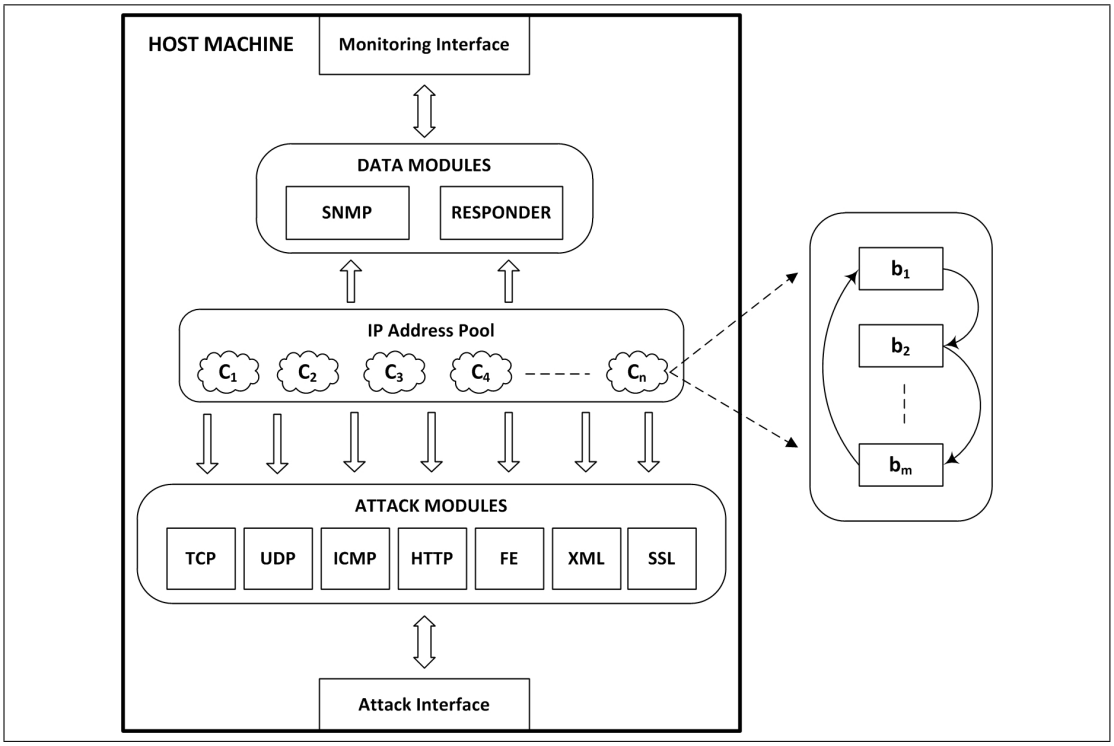


Figure 5.1: Schematics of Botloader

flavours: attack and data modules.

Attack Modules are responsible for sending the network traffic representing a user-specified attack-type, an FE, or normal background traffic. Each bot participating in an attack uses the characteristic properties of the relevant attack module and shares it with the other bots in its cluster. However, in an attack involving multiple clusters sharing the same functionality, clusters may be configured separately, as in case of a HTTP flooding attack with background HTTP traffic at a different rate. Botloader currently caters for seven attack modules, viz., `syn_flood`, `udp_flood`, `icmp_flood`, `http_flood`, `xml_flood`, `ssl_flood`, and `fe_flood` to simulate various types of DDoS attacks and FE scenarios.

Data Modules Apart from the modules responsible for conducting different types of attacks as specified in the configuration file, there are other modules designed for monitoring and data gathering. Two such data modules are currently implemented, viz., `snmp_bot` and `resp_bot`.

The `snmp_bot` is responsible for gathering data using the standard and custom MIB via the SNMP protocol. The `snmp_bot` when used with other

attack modules gathers information at 5 second intervals. The `snmp_bot` can gather both system-wide information such as the amount of data received on all the interfaces of the target, as well as application-specific information such as CPU and memory utilisation. The `resp_bot` is used for measuring the response time of a specific application running on the target host.

Both of these data modules – `snmp_bot` and `resp_bot` – use a separate 'management network' to perform monitoring and data collection operations. Using a separate network channel helps to avoid any potential interference between the operation of the data and attack modules. The data gathered by the `snmp_bot` is saved directly on the *control machine*, in a tab-delimited spreadsheet format.

SNMP data collection during UDP flooding attacks Even after using separate networks for attacking and monitoring the target host, UDP flooding attacks were found to interfere with the SNMP functionality, which also uses the UDP protocol for communication. UDP flooding attacks disable the UDP networking stack of the target machine and hence it was unable to respond to SNMP queries. In such cases a separate *monitor* program was used on the target machine. The monitor program performed the same data-gathering role as the `snmp_bot`, but was copied to the target at the start of the run. Once the experiment was complete, the gathered SNMP MIB data such as CPU utilisation, memory usage, etc., was copied back to the *control machine*. The monitoring program or SNMP daemon was given a higher execution priority on the target machine by increasing its 'niceness'[3] value. As the SNMP data was collected locally on the target machine, or via a separate monitoring network, any interference with the traffic on the attack network was eliminated. An insignificant amount of CPU was consumed on the target by the monitoring process which activates only once every 5 seconds.

IP address allocation

As discussed previously, the Botloader program consists of bots, clusters, and the attack and data modules. Each host machine in the experimental testbed

[3]nice is program available on most Unix or Unix-like operating systems, and is used to reschedule a program with a particular processing priority. Niceness values range from -20 (highest priority) to +19 (lowest priority).

runs an instance of Botloader. An abstract level design of Botloader is shown in Figure 5.1. Based on the number of IP addresses specified in the configuration file to simulate a given traffic scenario, Botloader reduces the size of the interface's netmask and so expands the available pool of addresses (initially 254 for a class C address) to a suitably large number. For example, if the netmask is expanded to a class B network then a total of 64,770 aliased addresses can be created (65,024 for class B, less the original 254 for the class C network, so as to avoid clashes with other machines). From this pool, Botloader selects *random* IP addresses. In practice only around 10,000 aliases per machine can be defined, without slowing down the simulation to the point where experimental fidelity begins to be compromised.

Figure 5.1 shows an instance of Botloader running on a host machine. Within each instance, the assigned IP address pool is further subdivided into n different clusters (C_i, $i \in [1, n]$) with non-overlapping IP addresses. Each of these clusters (C_i) is a collection of m bots (b_i, $i \in [1, m]$), each with a unique IP address and a network socket, and sharing the hardware address of the physical network interface. Before each data transfer cycle, bots are jumbled and then invoked in a sequential manner to send and receive traffic to/from the target. At the end of each data transfer cycle, the bots are jumbled again before repeating the cycle (see Figure 5.1). A typical traffic simulation scenario usually involves several data transfer cycles per second until the simulation times out. This ensures a *randomised* spread of source IP addresses (as seen by the target machine) that resembles the entropy of source IPs from real-world DDoS attack scenarios. As previously discussed (Chapter 4), Entropy H of a discrete random variable X is given by

$$H(X) = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i)$$

where n is the number of symbols and $p(x_i)$ is the probability of that symbol appearing in the message. The entropy H ranges between 0 and $log_2 n$. Dividing by $log_2 n$ gives the normalised entropy $H_0(X)$ with values between 0 and 1, and is defined as:

$$H_0(X) = H(X)/log_2 n$$

A normalised entropy of 1 implies that the symbol 'x_i' under consideration is purely random. Thus, $H_0 = 1$ if all bots transmit equally, and 0 if all traffic comes from a single address. Applying this formula to the source IP addresses of

packets in the CAIDA DDoS Attack 2007 dataset [51] resulted in $H_0 = 0.9405$, suggesting that the experimental value of $H_0 = 1$ is realistic.

5.2.3 Experimental Testbed Architecture

The experimental testbed environment contains two distinct networks: the *attack-network* for sending and receiving all the attack and normal background traffic, and the *management-network* for monitoring the target and collecting MIB data. The testbed architecture consists of ten dual-homed[4] host machines (Host 1 – Host 10), one layer 3 switch (attack switch), one layer 2 switch (monitor switch), and a target machine. In the attack-network, all hosts machines are connected to the target via the layer 3 switch. One of the 10 host machines in the attack-network is a *control* machine used for orchestrating the other hosts and for simulating various DDoS attacks or FE scenarios. Figure 5.2 given an abstract-level schematic of the testbed architecture.

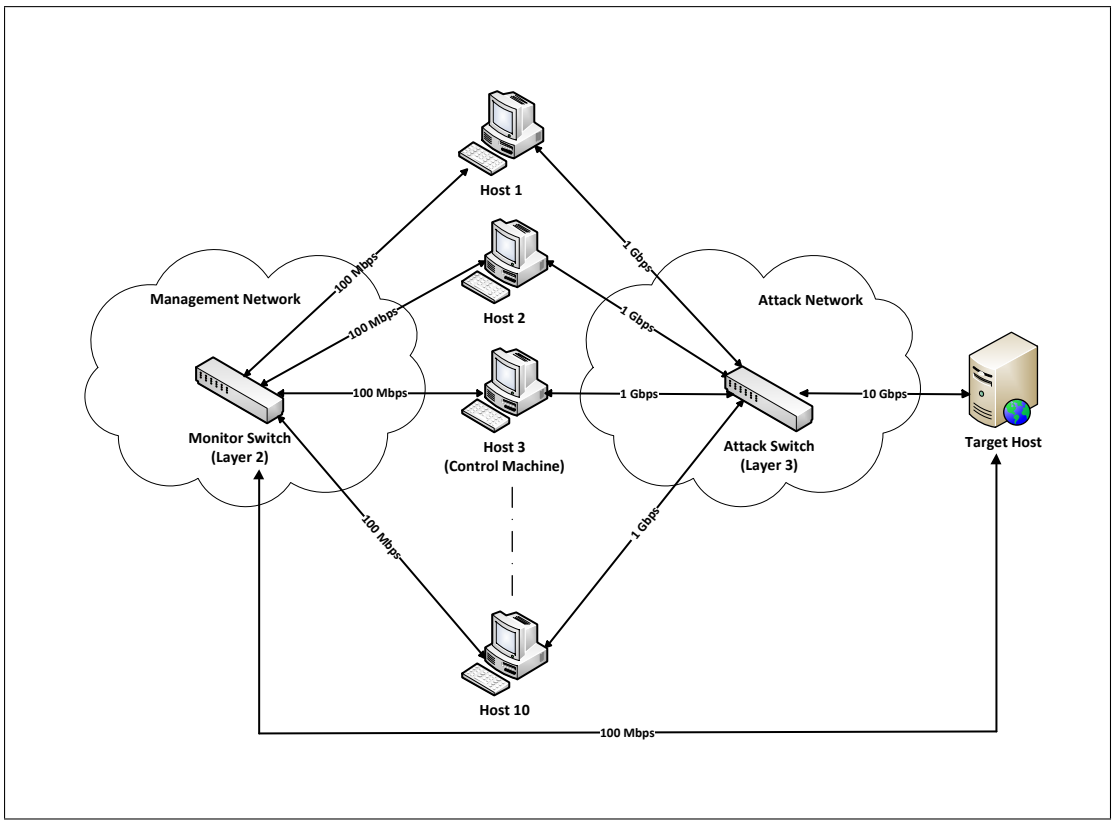


Figure 5.2: Testbed architecture

[4]A dual-homed host is a system equipped with two networking interfaces (Network Interface Cards (NICs)), generally sitting between an untrusted network and a trusted network in order to provide secure access.

All host machines in the attack-network use a 1 Gigabit per second (Gbps) link to connect to the attack switch, which is connected to the target machine via a 10 Gbps link. In the management-network, all connections between the target and the host machines to the monitor switch are 100 Megabits per second (Mbps). Every host machine runs an instance of the Botloader program and represents a 'group of attackers or bots' while simulating DDoS attacks, and a 'group of legitimate clients' during FEs scenarios. The control machine acts like a 'master' and issues commands to initiate the instances of Botloader in each of the participating hosts. All the traffic coming from the individual hosts is accumulated and sent through the 10 Gbps link connecting the attack switch to the target machine. All the host machines are standard PCs with 3.0 GHz Intel Core2 processors, 4 GB of memory, an integrated 1 Gigabit (Gb) NIC, an external 100 Megabit (Mb) Ethernet card, and are running Ubuntu 10.04 Desktop as the Operating System (OS). A small number of essential services like ssh, ntp etc. are also installed on each of the host machines. The target machine is a Dell PowerEdge R710 with six quad-core Intel Xeon 2.27 GHz processors (hyper-threaded), 32 GB of memory, and running Ubuntu 10.04 (Server) as the OS and Apache2 as the web-server. The monitor switch used in the management network is a layer 2 Cisco Catalyst 3560 switch while a Dell PowerConnect 6224 layer 3 switch is used to bind together the attack-network component of the testbed architecture.

The ARP Problem

In order to generate realistic DDoS attacks and FEs, it is important that the simulated traffic originates from a large number of IP addresses. In the case of an FE, such as the 1^{st} semi-final match of the 1998 World Cup, the number of IP addresses approaches 80,000. The problem with this requirement is that an Ethernet LAN is not designed to support that many IPs. On an Ethernet network, IP packets are sent within frames, with a source and destination MAC address. In order to identify which MAC address corresponds to a given IP address, the switch maintains an ARP table, which is of limited size. For example, the Dell switch in the testbed has a maximum ARP table size of only 1024 entries. When packets come back from the target (Figure 5.2), the switch must decide which of the 10 host machines receives the reply packets. If the IP-address is not in the table it broadcasts an ARP request packet, effectively asking 'who has IP

address x.x.x.x?'. While waiting for a reply the switch has no option but to drop the packet. When the reply comes back from the correct interface card the switch has nowhere to store it if the table is full, and hence cannot forward the packet when it is later retransmitted by the target (which by default takes 2 seconds). So the end result is that the network fills up with ARP requests and replies rather than with transmissions of desired traffic between attackers and target.

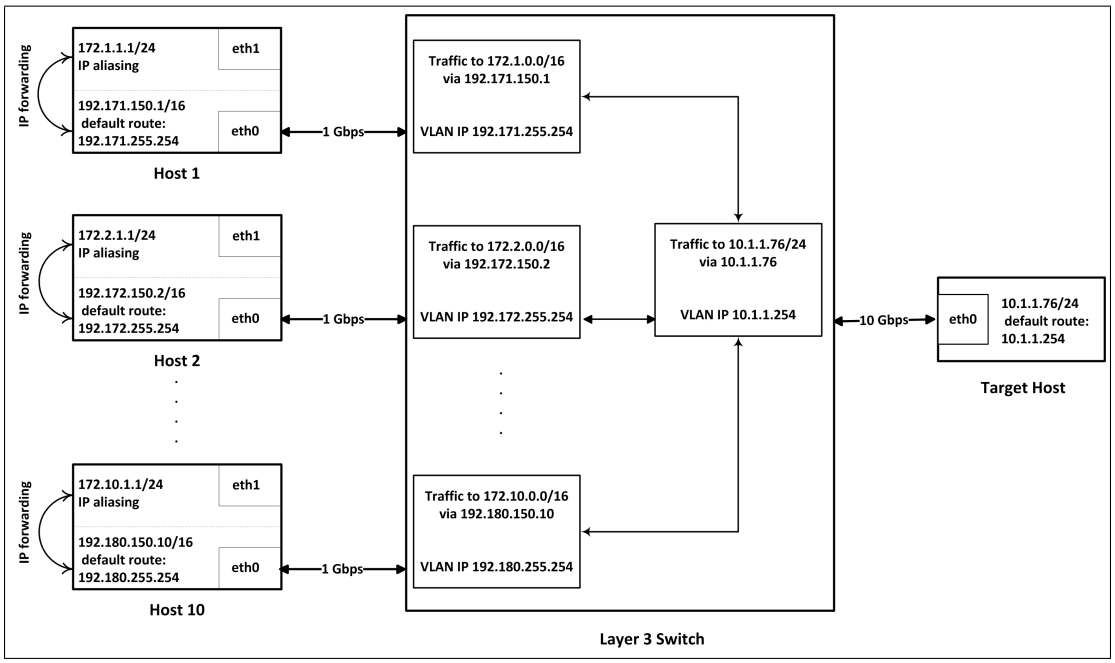


Figure 5.3: Multi VLAN with IP forwarding

The Work-around

In order to avoid this flood of ARP requests, it was necessary to change the overall infrastructure from a *switching-mode* to a *routing-mode*. Figure 5.3 shows the configuration of the testbed architecture. The process of turning the switch and each host machine into an effective router is achieved via the following steps:

1. The first step is to configure the main attack interface on each host machine as belonging to a separate network. On hosts 1 – 10, the network interfaces connected to the attack-switch (`eth0`) are assigned IP addresses in the range `192.171.150.1 - 192.171.150.10`, and the network is specified as Class B.

2. The second step is to re-use the secondary interface, normally used for monitoring, as the repository of all the aliased addresses. In this way they will be hidden from the switch, which can only see 11 address in total – well within the limit of switch's ARP table size.

3. The third step is to set up each host machine to act as a router by enabling IP-forwarding on every machine (on Linux). This allows the network packets on one interface (`eth1`) to be forwarded to another interface (`eth0`) and vice versa. IP-forwarding bridges the two (or several) interfaces into a single network topology and routes the packets between them extremely fast using internal pathways of the computer, without the need of any ARP (IP to MAC address mapping) information.

4. The final step is to sub-divide the switch into 11 VLANs, one of which is connected to the target, and the other 10 to the attacking host. Each attacker-VLAN has its next-hop address set to the IP address of the main attack interface. For example VLAN 1 has its IP set to 192.171.255.254 and its default route to 192.171.150.1. Likewise, the main attack interface on attacker 1 has its default route set to VLAN 1's address, 192.171.255.254. Thus all traffic between VLAN 1 and attacker 1 destined for the aliased addresses for eth1 (172.1.1.1/24) will be *routed* via the eth0 interface without the need for any ARP resolution. A similar set-up is applied to other 10 VLAN's.

This solution to the ARP problem allows the creation of a large set of 'real' IP-addresses (bots or legitimate clients) that can communicate and exchange data with the target, which responds to network traffic as if from a large number of physical machines. Unlike Network Address Translation (NAT), where the source or destination IP address is modified for local delivery, the packets sent to (and received by) the target host *exactly resemble the packets sent by an external machine*. The packets have unmodified source and destination IP address and the target responds to them as if they were physically located in a real network. But all the packets remain within a controlled and isolated test environment. This set-up of the testbed has so far sustained around 80,000 different IP addresses without any ARP related issues.

5.2.4 Performance Evaluation

This section presents the performance evaluation of the traffic generation and testbed framework including the Botloader program in terms of its *ability to mirror* a given network trace. In order to test the effectiveness of the testbed facility, a subset of the 1998 FIFA World Cup Dataset [12] was used as sample input trace. This network traffic trace was preprocessed and was then given to the Botloader program in a specific format to generate similar traffic. It is acknowledged that the validation is limited by a lack of datasets representing different DDoS attacks and FE scenarios available in the public domain.

Dataset Processing

A subset of the 1998 FIFA World Cup Dataset [12] was used for the experiments described in this section. The dataset is a six hour network trace around the 1^{st} semi-final match of the 1998 FIFA world cup. The original dataset was first filtered to remove all traffic except HTTP GET requests. Secondly, only packets containing 200, 206 and 400 HTTP status codes were extracted and used as the input trace (processed dataset) for the experiments. Table 5.1 shows the high-level characteristics of the original dataset and the processed dataset used in the simulations.

Table 5.1: Characteristic features of the dataset used for simulations.

Parameters	**Original Dataset**	**Processed Dataset**
Number of requests	29,662,465	19,615,109
Number of unique source IPs	79,033	78,646
Number of unique resources accessed	11,885	11,839
Peak packet rate (requests per second)	4,089	2,376
HTTP request type	GET, POST, HEAD	GET
HTTP status codes	all	200, 206 and 400

In order to ensure that the same web-resources are downloaded during the simulation as were present in the original trace, a fake directory structure was created and installed on the target machine's web-server. All the files within this directory structure used the same file name and contained the same amount of data as in the original traffic trace, although the contents were junk data.

Experimental Results and Analysis

The processed dataset was further characterised by four parameters: the number of packets (HTTP GET requests), the number of unique source IP addresses, the number of unique resource accessed, and the entropy of the accessed resources. All four parameters were computed for a 1 second time sampling interval, which reflects the time resolution of the original FE dataset. However, the sampling interval is a configurable parameter and can be changed to any value depending on the available data.

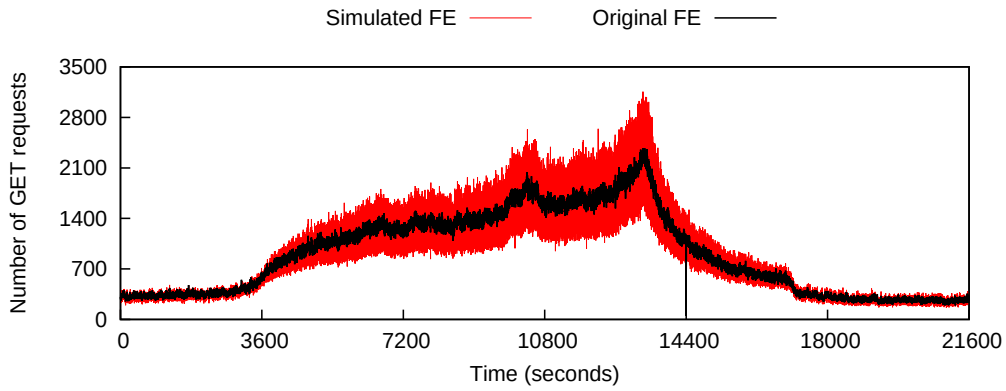
The sampled input file with per second values of the four parameters was provided to the Botloader program, which ran on every host machine in the testbed set-up described in 5.2.3. All the host machines were controlled by a single machine, which also synchronised the simulated FE traffic. In order to test the effectiveness of the traffic generation and testbed framework the simulated FE traffic was compared to the original FE traffic in terms of three parameters viz., the incoming traffic, the source IPs generating the traffic, and the entropy of the resources being accessed. The following section presents and discusses the experimental results obtained for each of these parameters.

Incoming Traffic The incoming traffic for the simulated and the original pFE trace was compared in parts: the total traffic volume, and the traffic shape. The original FE traffic trace here refers to the processed data i.e., per second values for the number of packets, the number of unique source IP addresses, the number of unique resources accessed, and the normalised resource entropy given as input to the Botloader program.

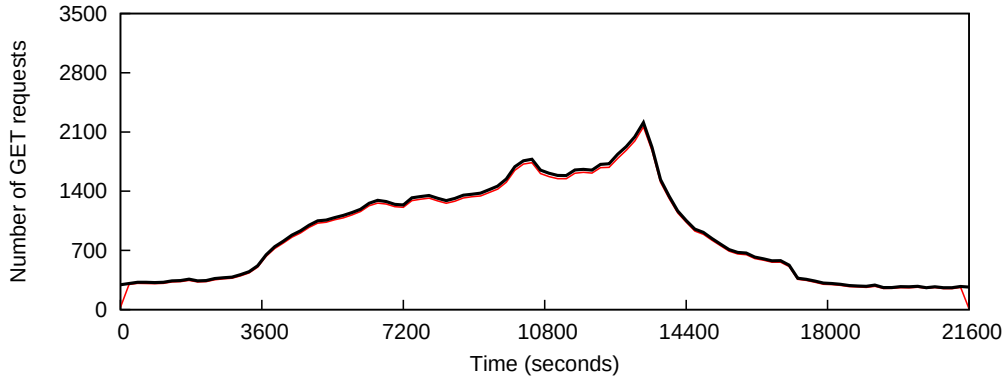
The total traffic volume of the simulated FE traffic trace was found to be 2.6% short of the original traffic trace. The original traffic has 19,615,109 GET requests, although at the end of the 6 hour simulation 19,103,258 GET requests were registered in the web-server logs.

The experimental results for the second part of the incoming traffic validation i.e. comparing the shape of the incoming traffic for the simulated and the original dataset are shown in Figure 5.4, which plots the number of incoming requests per second against time. Figure 5.4a compares the incoming network traffic in a raw format, while 5.4b uses a built-in smoothing technique (bezier) within the gnuplot[5] utility.

[5]Gnuplot is a command-line graphing utility available on many computing platforms in-



(a) Incoming traffic comparison using raw data.



(b) Incoming traffic comparison using smoothed data.

Figure 5.4: Incoming traffic comparison for the simulated and original FIFA dataset.

Although there are large variations in the simulated incoming traffic as compared to the original, these variations are very fine grained, and the simulated traffic closely mimics the original trace in terms of the overall traffic shape.

Source IP Addresses The source IP address parameter of the original trace is compared with the simulated traffic in two forms: the total number of unique source IPs, and the variations in the number of unique source IPs over time. The Botloader program is designed to distribute the overall traffic being sent to the target host equally divided among all the participating host machines.

cluding Linux.

So effectively each host machine sends 1/10th of the total traffic and also uses 1/10th of the total aliased-IPs assigned to the experiment.

The original FE dataset used in the simulation has 78,646 unique source IP addresses (see table 5.1). Conducting the experiment using 7,865 bots per host machine, nearly 1/10th of source IPs found in the original dataset (Table 5.1), all of 78,650 unique source IPs were found in the web-server logs. This demonstrates the capabilities of the traffic generation and testbed framework to simulate traffic scenarios involving a large number of distinct source IP addresses.

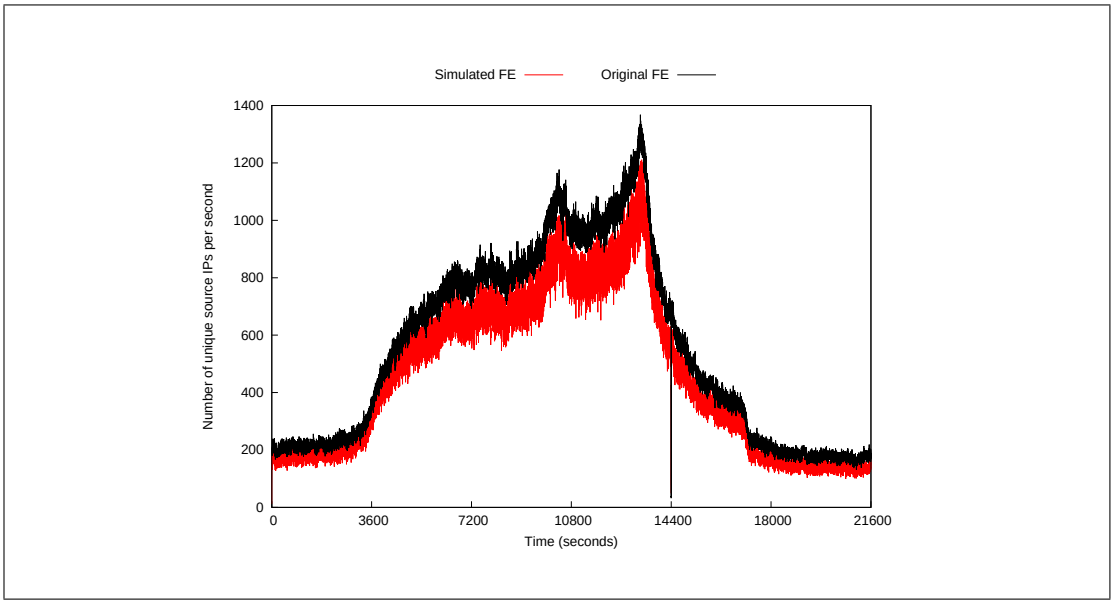


Figure 5.5: Source IP address comparison for the simulated and original FIFA dataset.

The variations in the unique source IP addresses for the original and the simulated traces are shown in Figure 5.5. The figure plots the number of unique source IPs per second against time for the two datasets. As for the incoming traffic comparison (Figure 5.4), the shape or variations in the unique source IPs in the simulated trace closely mimics the original traffic trace. However, the number of unique source IPs observed at any given time in the simulated traffic is somewhat less than in the original.

Resource Entropy The last parameter used to test how closely the traffic generation and testbed framework can mirror a given network trace is resource entropy. Figure 5.6 compares the normalised entropy of the resource being accessed for the original and the simulated network trace. The normalised resource

entropy for the original FE traffic is calculated using Shannon's entropy formula [103], as discussed above in Section 5.2.2.

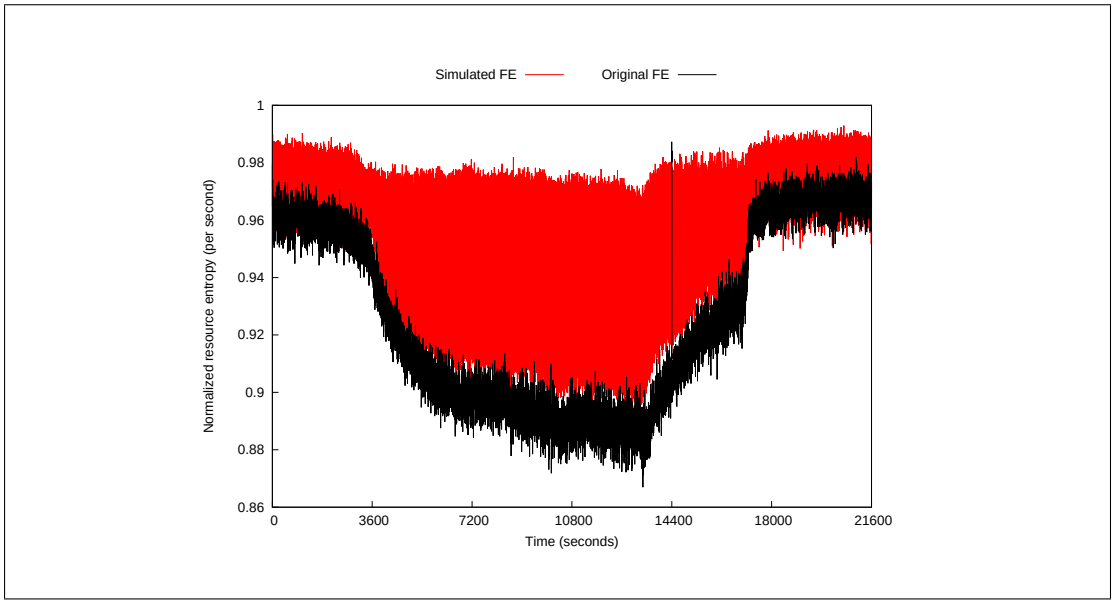


Figure 5.6: Resource entropy comparison for the simulated and original FIFA dataset.

In the simulated FE traffic, large variations are observed in the entropy of the accessed resources. These variations are mainly attributed to the overall design of the Botloader program. Instead of *directly replicating* the resource access pattern of a given input trace, Botloader has been designed to take the number of unique resources to be accessed in any given interval and the *desired normalised entropy* as the input parameters and performs *reverse entropy* calculations to compute the frequency distribution of resources that can generate the desired entropy. This approach gives a rather realistic feel to the synthetic traffic generation by not simply duplicating the exact resource access pattern from the original trace and at the same time keeping the overall access pattern close to the original. The approach is also light on memory usage as each host machine is not required to load the entire resource access map into available memory. All it needs is to load a comparatively small text file with the total number resources and the desired resource entropy per sampling interval.

Another observation made during the analysis of the web-server logs was that all of 11,839 different resources in the original trace (Table 5.1) were found in the simulated network traffic, demonstrating the ability of the traffic generation and testbed framework to mirror a given network trace.

The Botloader program is able to create a large number of bots and use them to generate traffic volume comparable to the original traffic trace. However, the variations in the simulated network traffic were much larger than the original trace. Reducing these variations and improving the simulation results for resource entropy have been identified as future work.

5.3 Simulating Flash Events

In this section, the traffic generation and testbed framework described above uses the FE model developed in the previous chapter (Section 4.3) to generate different types of FE scenarios. In order to generate FE traffic, the model uses information extracted from the few existing FE datasets, but due to their limited number and types, is limited to predictable and unpredictable FE scenarios. For similar reasons, as described in the previous chapter, the simulation of synthetic FEs is limited to representing traffic volume.

5.3.1 Datasets

In this section two publicly available datasets, representing a predictable FE (pFE) and an unpredictable FE (uFE) respectively, are used as base models to generate similar FE traffic. For simulating a pFE, a subset of the 1998 FIFA World Cup Dataset [12] is used as the basis. This dataset contains six hour long network traffic around the 1^{st} semi-final match of the 1998 FIFA world cup. The dataset containing the hourly hits on Wikipedia following Steve Jobs death is used as the base dataset for simulating the uFE, which provides hourly aggregates of requests over a period of 43 hours [73].

Model Parameters

The FE model developed in the previous chapter characterises FEs via three key components: the volume of incoming traffic, the related source IP-addresses, and the resources being accessed. Since the FE model provides a mathematical representation only of the incoming traffic, the values used in the simulations for other two components i.e., source IP addresses and the resources being accessed, are reproduced from those present in the original datasets. The information relating to source IP addresses and the resources being accessed is not available

for the uFE dataset (hourly hits on Wikipedia following Steve Jobs death), and the same applies to the case of the pFE dataset. The FE model developed in the previous chapter makes use of five key parameters to mathematically represent the incoming traffic volume during an FE. The essential parameters used by the FE model are:

- R_n: average normal incoming request rate
- R_p: peak incoming request rate
- t_o: start time of the flash-phase
- t_f: end time of the flash-phase or start time of the decay-phase
- t_d: end time of the decay-phase

Using these key parameters, the FE model derives the following parameters to model the incoming traffic volume during FEs:

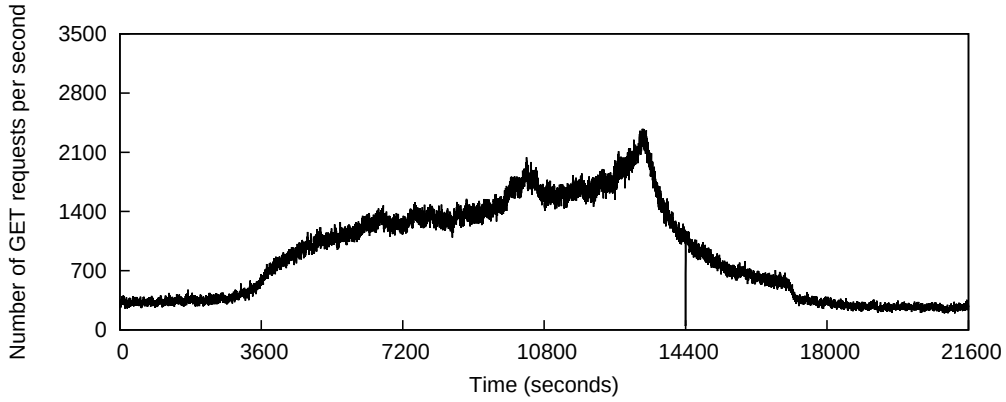
- A_f: amplification factor or the ratio of the peak request rate (R_p) and the average normal request rate (R_n)
- ΔT_f: duration of the flash-phase i.e., time take for the average normal request rate (R_n) to reach the peak request rate (R_p)
- ΔT_d: duration of the decay-phase i.e., time taken for the peak request rate (R_p) to reach the average normal request rate (R_n)
- α: flash-constant defined as $\alpha = \frac{\ln A_f}{(t_f - t_o)}$
- β: decay-constant defined as $\beta = \frac{\ln A_f}{(t_d - t_f)}$

Table 5.2: FE model parameters for datasets used in simulations.

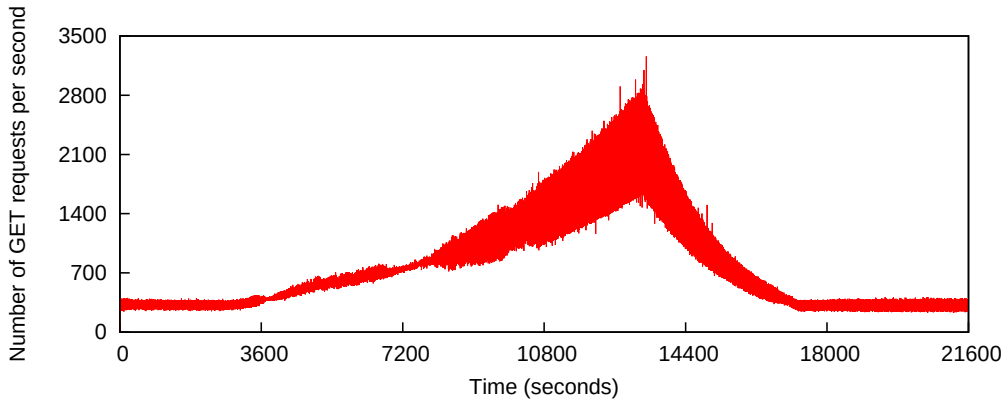
FE Datasets	$\mathbf{R_n}$	$\mathbf{R_p}$	$\mathbf{t_o}$	$\mathbf{t_f}$	$\mathbf{t_d}$
pFE (1^{st} Semi-final)	391	2,376	2,880	13,320	17,280
uFE (Wikipedia hits)	1,826	1,063,665	8	10	43

Table 5.2 provides the values for all the essential parameters used in the simulations. All the values are computed using 'seconds' as the time unit for simulating the pFE and 'hours' for simulating the uFE. Therefore for the pFE

dataset, $\mathbf{R_n}$ and $\mathbf{R_p}$ denote respectively the average number of incoming requests per second and the maximum number of incoming requests per second observed over a period of six hours. Values for $\mathbf{t_o}$, $\mathbf{t_f}$, and $\mathbf{t_d}$ denote the instances of time (in seconds) which mark the start and end of the flash-phase and the decay-phase. Similarly, the values for the uFE are computed using 'hour' as the time resolution. Different time resolutions were chosen for the pFE and the uFE because the original datasets were only available with different time resolutions.



(a) Original predictable FE traffic.



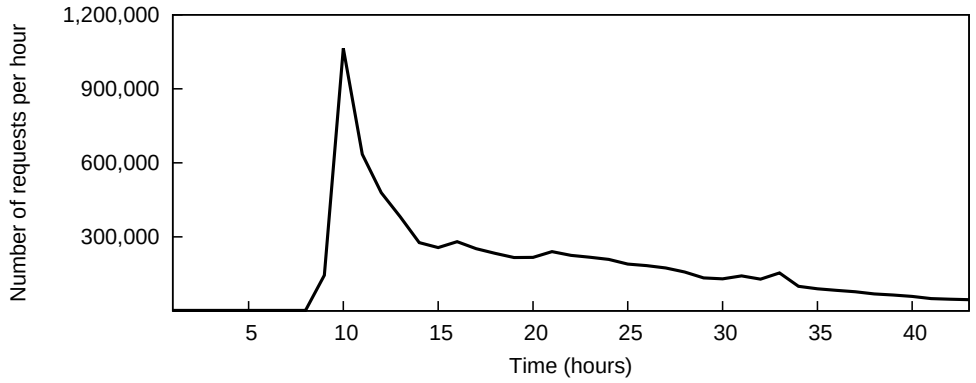
(b) Simulated predictable FE traffic using the FE model.

Figure 5.7: Incoming traffic profile for the predictable FE.

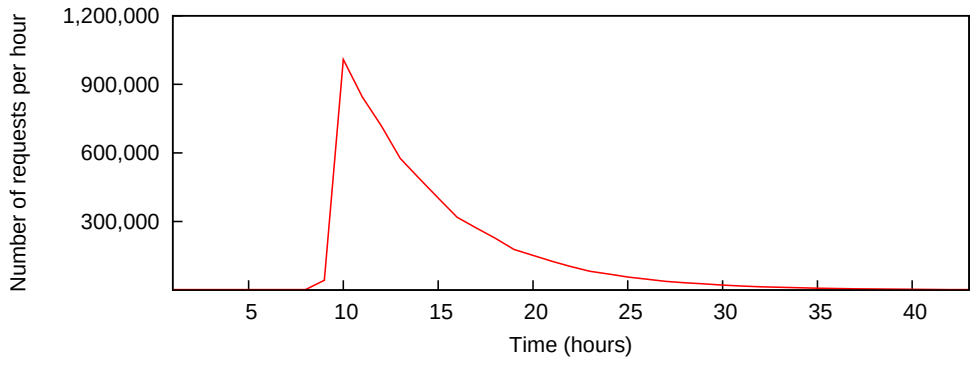
5.3.2 Experimental Results and Analysis

In this section, the FE model parameters presented in Table 5.2 are used to generate an 'input file' which is then loaded to the Botloader program (running

on each host machine within the testbed architecture) to synthetically generate the traffic. This input file consists of tab separated values for the number of packets (HTTP GET requests) to be sent, the number of unique source IPs to use, the number of unique resources to access, and the resource entropy to simulate. All of these values are supplied on a per-sampling-interval basis to the Botloader program.



(a) Original unpredictable FE traffic.



(b) Simulated unpredictable FE traffic using the FE model.

Figure 5.8: Incoming traffic profile for the unpredictable FE.

Figure 5.7a shows the incoming traffic profile for the original pFE dataset i.e. six hour traffic around the 1^{st} semi-final match of the 1998 FIFA world cup. The predictable FE traffic, based on the model developed in Chapter 4, is shown in Figure 5.7b. The simulated traffic is able to mirror the shape of the actual dataset, with exponentially increasing flash and decreasing decay phase.

However, there are variations in the traffic that are noticeably different from the original data. The time-synchronisation of the 10 host machines, each sending $1/10^{th}$ of the total traffic, is speculated to be the reason for these variations. Such variations can also be seen in the simulated traffic of the original FIFA dataset in Figure 5.7a. Figure 5.8a shows the incoming traffic profile for the original uFE dataset i.e. hourly hits on Wikipedia following Steve Jobs' death. The simulated uFE traffic using the FE model is shown in Figure 5.8b. The simulated traffic is able to closely mimic the original traffic. Chapter 7, in the section on future work, discusses directions for further work to investigate the possible reasons for such variations in the simulated pFE traffic.

5.4 Simulating DDoS Attacks

In this section, the traffic generation and testbed framework is used to emulate a real-world DDoS attack. In order to generate network traffic, attack or benign, the Botloader program takes user specified parameters such as the duration of the simulation, the number of bots (aliased-IP addresses) to use, and the traffic or data rate (specified in packets per second and kilobytes per second), via a configuration file. So each of the attack types has its own configuration file with some features that are common across all attacks, such as the duration of the simulation and the number of bots to create, and some custom features, such as a resource file containing a list of resources to download during a HTTP flooding attack or an FE. For example, the configuration file for a HTTP flooding attack contains the following line of code, with the user specified values for various parameters:

```
http_bot 100 debug=0 time=60 port=80 res_file=web-resources.txt
rates="100" dest_ip=192.168.200.76
```

This configuration file creates `100` bots and uses them to direct HTTP traffic towards the web-server running at port `80` on `192.168.200.76`. During the simulation, the resources are downloaded at `100` kilobytes per second, spread across `60` seconds from the `web-resources.txt` file. In this section, simulation results for the CAIDA "DDoS Attack 2007" Dataset [51] are presented. The following chapter presents the simulation results for application level DDoS attacks.

5.4.1 Datasets

In order to simulate a real-world DDoS attack, the CAIDA "DDoS Attack 2007" Dataset [51] has been used as the base traffic model for extracting features required by the Botloader program to generate similar traffic. The CAIDA dataset contains pseudonymised traces from a DDoS attack that occurred on August 4, 2007 for approximately one hour. From the complete dataset, one-way attack traffic is used for simulation. Table 5.3 shows the high-level statistic of the dataset used in the simulation.

Table 5.3: Characteristic features of the DDoS datasets used for simulations.

Parameters	**CAIDA Dataset**
Duration	66 minutes
Attack Type	Ping flood
Packet Type	ICMP
Number of Packets	359,655,765
Number of unique source IPs	9,311
Peak packet rate (packets per second)	175,010

5.4.2 Experimental Results and Analysis

The incoming traffic profile (packets per second) of the CAIDA dataset is shown in Figure 5.9. The one-way traffic profile clearly shows two rather distinct traffic rates, as experienced by the target victim. The first one is the packet rate during the first-half of the trace i.e. from the start of the trace until around the 1500 second mark. For simulation purposes, the packet rate during this period is referred to as the *normal packet rate*. The average packet rate for this period was approximately 385 packets per second in the original dataset. The second distinct traffic rate is seen during the second half of the trace i.e., starting from approximately 1500 seconds until the end of the trace. During this period, the packet rate is referred to as the *attack packet rate*, and contains approximately 125,705 packets per second on average.

In order to simulate network traffic similar to the CAIDA dataset, the normal and the attack packet rates, calculated from the original dataset, are provided to the attack configuration file of the Botloader program to generate a 5 minute sample network trace. The configuration file is in the format given below where two simultaneous threads (clusters of bots) are started, with the first one for

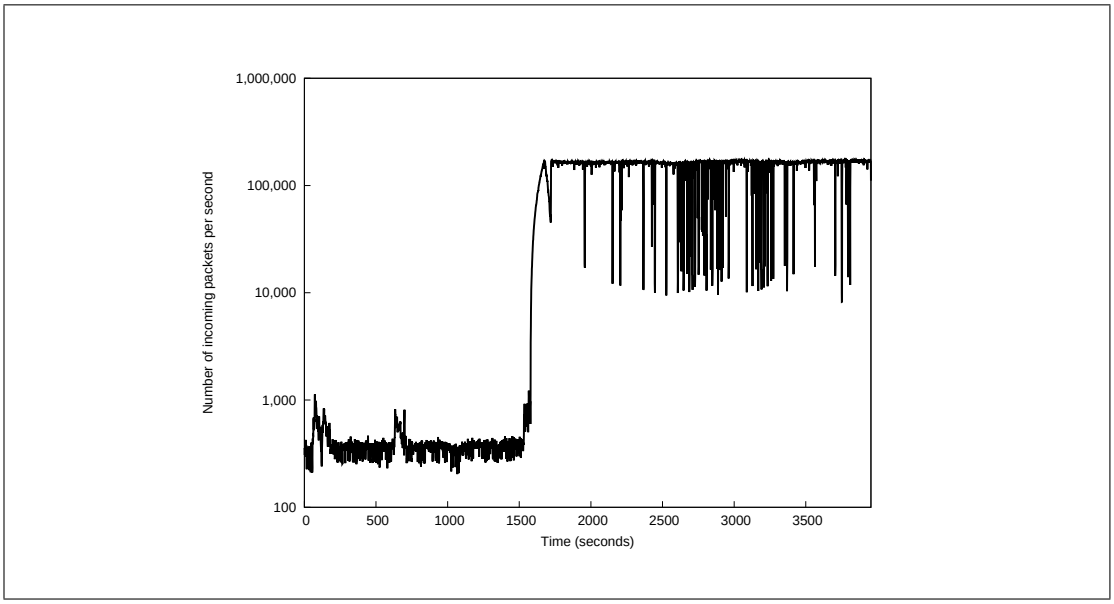


Figure 5.9: Incoming traffic profile for the CAIDA dataset.

simulating normal traffic part, and the second for simulating the attack part of the original CAIDA dataset.

```
ping_bot 73 debug=0 ip_version=4 time=120 packet_size=64
rates="46200" burst_default=1000 dest_ip=10.1.1.76

ping_bot 859 debug=0 ip_version=4 time=300 packet_size=64
rates="0 0 7542300 7542300 7542300" burst_default=1000 dest_ip=10.1.1.76
```

The first thread runs for 120 seconds, and sends 46,200 packets during this period, thereby achieving the desired rate of roughly 385 packets per second. The second thread is started simultaneously, and runs for the entire length of the simulation (300 seconds). However, this thread is *dormant* for the first 120 seconds, as specified by the two leading 0's in the `rates` section of the configuration file. During the remaining 180 seconds, the threads send ICMP traffic at a rate of approximately 125,705 packets per second to the target victim (`10.1.1.76`), as specified by the `dest_ip` parameter of the configuration file.

Figure 5.10 shows the simulation results of a 5-minute traffic trace synthetically generated using the traffic rates from the CAIDA dataset. The Botloader program was able to match up to nearly the normal and the attack packet rates from the original trace, thereby effectively emulating an existing real-world DDoS attack. Using different configuration files, application level DDoS attacks (HTTP

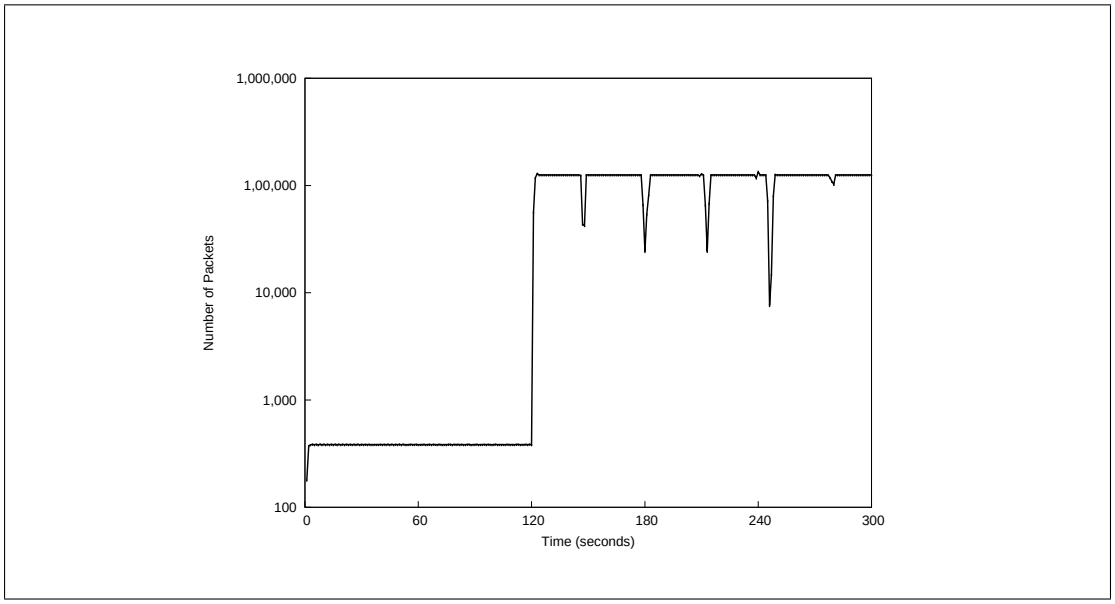


Figure 5.10: Incoming traffic profile for a simulated DDoS attack.

GET flood and SSL flood) and normal traffic trace have been also simulated and will be presented in the following chapter.

5.5 Conclusion

Research associated with network anomaly detection, particularly DDoS attack detection and FE separation, often suffers from lack of accurate evaluation, mainly because of the scarcity of recent and realistic datasets available in the public domain. The majority of the publicly available datasets are either too old to reflect current network trends, or are heavily anonymised to eliminate any associated privacy and legal concerns, and so cannot provide useful information about the traffic contents or origins, or are available only in a format such as Common Log Format (CLF), which limits their usability, since they cannot be replayed over the network. Commonly used off-the-shelf hardware and software traffic generators have their own limitations in generating realistic network traffic, as do simulation programs such as ns–2.

This chapter addresses the aforementioned challenges in obtaining realistic network traffic datasets, and in the process makes two related contributions:

- the development and evaluation of a traffic generation and testbed framework, based on only modest hardware and using a software traffic generator, cooperatively developed as a part of this research, and

- the development of techniques to realistically emulate real-world DDoS attacks and FEs.

These contributions provide a cost-effective and dedicate test environment which can emulate large-scale DDoS attacks, both at the network and application layers, and FEs, and to monitor them in real time, and at a scale not previously possible with existing emulation testbeds.

Chapter 6 uses the datasets generated in this chapter, along with some application level attack and normal traffic datasets, for evaluating the developed ensemble-based attack detection technique, which is potentially able to detect a variety of DDoS attacks and differentiate them from FEs.

Chapter 6

Ensemble Based DDoS Attack Detection

In chapter 3, a DDoS attack detection technique based on the rate of arrival of new or Not Seen Previously (NSP) source IP addresses was presented. That chapter provided the design, a proof-of-concept implementation using bit-vectors, and experimental evaluation of the proposed technique. The technique was evaluated against UDP attacks with varying numbers of source IP addresses. However, it focused only on detecting DDoS attacks without considering Flash Events (FEs), a network anomaly sharing a number of characteristics with DDoS attacks.

This chapter extends that work by including additional parameters – from both the incoming traffic and also the target server side, thereby extending the DDoS identification criteria beyond the simple property of NSP source IPs. It also addresses Research Question 5, i.e. *Can a more sophisticated ensemble-based technique based on both network traffic characteristics and server-load data (memory and CPU utilisation) improve the accuracy of DDoS attack detection and provide a means to identify and differentiate DDoS attacks, FEs and normal traffic?* This chapter thus focuses on the development and evaluation of an ensemble-based DDoS attack detection technique using interval-based sampling of various features. The technique combines two rather orthogonal attack detection strategies – network traffic analysis and server-load data analysis, and is expected not only to detect a variety of DDoS attacks, including application layer attacks, but also to differentiate DDoS attacks from FEs.

The attack detection technique developed in this chapter is evaluated against a variety of datasets, synthetically generated using the traffic generation and testbed framework developed in Chapter 5. The simulated datasets consist of network and application layer attacks, FEs, and normal traffic scenarios. Considerable care has been taken in generating and using representative datasets; however, the results presented here are undoubtedly influenced by the particular nature both of the datasets generated and the real world datasets on which they are modelled. This is an unavoidable aspect of such research, and while the results are clear and represent an important outcome, it would, as proposed in Chapter 7, also be desirable to apply these techniques to a wider variety of datasets.

The rest of this chapter is organised as follows. Section 6.1 describes the attack detection model and its two components i.e., network traffic analysis and server load analysis. Section 6.2 presents the set of features analysed by each of those components, which will be used in the proposed ensemble. Section 6.3 provides the technique used for detecting changes in each of the parameters belonging to the feature set. Section 6.4 describes the experimental set-up used for simulating different datasets, both for attack and non-attack (such as FE) traffic scenarios. Section 6.5 presents the experimental results when the change detection technique is applied to individual features used within the ensemble. Section 6.6 correlates changes in those different features to detect a variety of DDoS attacks and separates them from FEs. Finally, Section 6.7 summaries the research presented in the chapter and provides concluding remarks.

6.1 DDoS Attack Detection Model

The ensemble-based DDoS attack detection approach correlates two different attack detection strategies – network traffic analysis and server-load analysis. The reason for this is that different components of the system (the network infrastructure and the target server) react differently when subjected to different kinds of attack or FEs. Some attacks aim to consume the network bandwidth leading to the target host, whereas other attacks are directed against the resource allocation strategies of the victim. For example, during a high rate network layer flooding attack such as an ICMP attack, the available network bandwidth is more adversely affected than the processing capabilities of the target machine.

On the other hand, during an application level attack, it is expected that both the network bandwidth and the available resources such as CPU utilisation are synchronously targeted and consumed, as the target server is forced to process a large number of incoming requests directed at a specific application.

One example of an application level attack is the HTTP GET flooding attack. In this attack, compromised machines send conventional HTTP GET requests to the target web-server to download the complete response. This attack is typically used to exhaust the available bandwidth of the victim as well as consume the system and application resources. Such situations can at times be misinterpreted as FEs where a large number of legitimate users concurrently send access requests to a Web server, similar to a High-Rate Flooding (HRF) DDoS attack using HTTP GET as the attack vector. Both of these situations can often lead to service degradation, either due to a high network bandwidth consumption, or a high resource consumption, or a combination of both.

There are also DDoS attacks which work by exploiting a design or implementation level flaw in a service running on the target, so even a modest packet rate can potentially have severe effects on the available computing resources. A recent example of such attacks by suspected Iranian sources on US banks exploited the cryptographic protocols of the server [18]. Although the exact nature of the attacks is unknown, it was probably based on a recently published Secure Socket Layer (SSL) exploit [117]. Additional details of this attack are provided later in the chapter (Section 6.4).

Since both of the attack categories discussed here, network and application level flooding attacks, often target different components of the system, effective attack detection criteria in one domain (network level) may not necessarily work for attacks belonging to the other domain (application level) and vice versa. Thus, in order to effectively detect different types of DDoS attacks it is important to take both sets of attack detection criteria into account, and use them in tandem.

This reasoning forms the basis of the proposed ensemble-based DDoS attack detection technique, schematically represented in Figure 6.1, where the individual server-load and network traffic features are detected and correlated to decide whether the incoming traffic represents a DDoS attack, or an FE, or normal network activity.

Although the proposed ensemble-based technique is designed to detect dif-

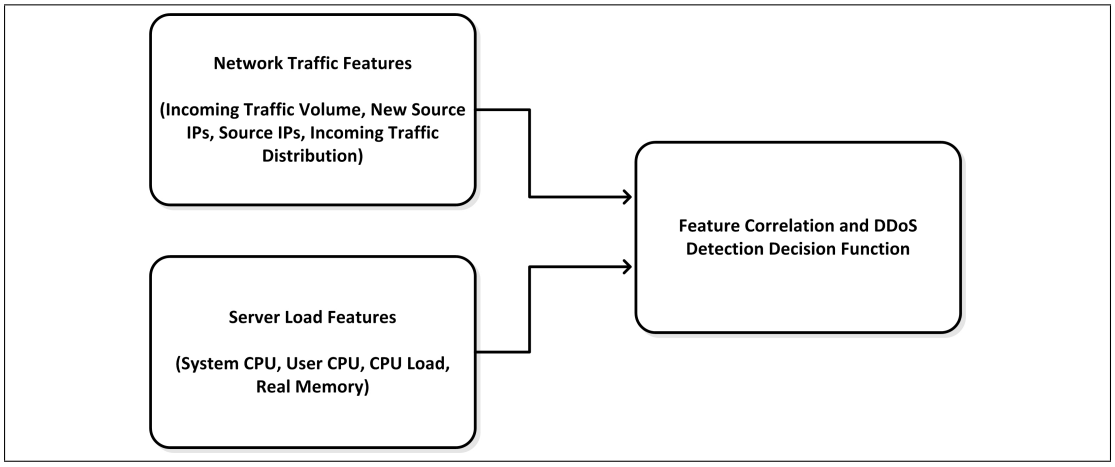


Figure 6.1: Schematic of the Ensemble-based DDoS Attack Detection Model.

ferent types of High-Rate Flooding (HRF) DDoS attacks, this approach is also believed to be able to detect a variety of low rate flooding DDoS attacks at the application level. However, such an application of this technique is beyond the scope of this thesis.

The following section describes the network traffic and the server-load based parameters used in the ensemble-based DDoS attack detection technique.

6.2 Feature Set Selection

As already mentioned, the features to be monitored and analysed fall into two major categories: incoming network traffic features, mostly used in detecting HRF flooding attacks, and server-load features, which are mostly used in identifying application level DDoS attacks and FEs. However, in certain attacks both the network traffic and the server-load may change, as in the case of a HTTPS GET flooding attack. The features chosen to represent each of these categories will now be described.

6.2.1 Incoming Network Traffic Features

The proposed ensemble based DDoS attack detection technique uses five characteristic features (however, for the reasons described below, the prototype implementation implements only four of these) of the incoming network traffic indicative of a potential HRF DDoS attack, both at the network or application level. These features are all calculated and used on a per sampling interval basis. For

the proof-of-concept implementation of the proposed attack detection technique, unless specified otherwise, a 5 second sampling interval has been used for all the network traffic based parameters. However, since the current implementation of the proposed ensemble based DDoS attack detection technique is *post hoc*, the time resolution for the individual parameters may vary.

Incoming Traffic Volume

One of the fundamental characteristics of a HRF DDoS attack is the dramatic increase in the incoming traffic volume. The incoming traffic volume also increases in the case of an FE; however, the increase is not as immediate as in the case of a DDoS attack, assuming that the attack is not mimicking an FE. This parameter is represented by the number of packets per sampling interval, and is referred to as the NumPack in the ensemble.

New Source IP Addresses

The onset of a DDoS attack is also accompanied by an increase in the number of unsolicited packets arriving at the target from previously unseen or new IP addresses. This characteristic property has been previously used in Chapter 3 to develop and implement a DDoS attack detection technique, and in Chapter 4 to differentiate DDoS attacks from FEs. The new source IP address parameter is referred to as NewSrcIP in the ensemble.

Number of Source IP Addresses

During a DDoS attack, the bot master commonly uses a set of compromised machines in concert to maximise the damage. This causes the victim to experience an abrupt increase in the number of source IP addresses per interval throughout the attack. This parameter is referred to as SrcIP in the ensemble.

Although the property is also a characteristic of FEs, the dramatic increase in incoming traffic is mainly due to an increase in the number of IP addresses (clients) rather than due to an increase in the number of requests per IP address (see Section 4.3.3 for details). On its own this parameter may not be sufficient to differentiate DDoS attacks from FEs, but when used in tandem with other network traffic parameters, it can potentially differentiate between these two network anomalies.

Incoming Traffic Distribution

To maximise damage during a DDoS attack, a bot master forces each of the available bots to send a large number of requests or packets, which then aggregate to the desired load before reaching the target server. This feature of the incoming network traffic translates to a high rate of requests or packets per source IP address, and can thus be used as a parameter to detect the onset of a DDoS attack.

This parameter is useful in distinguishing DDoS attacks from FEs, where the majority of clients are interested only in specific information related to the event, hence a relatively small number of requests originate from each accessing client (see Section 4.1.3). This parameter is referred to as PackPerIP in the ensemble.

Access Pattern of Web Resources

An additional parameter that can potentially be added to this list is the 'access pattern of Web resources' during different traffic situations. This parameter has been previously used to understand and model FEs (See Section 4.3), and varies in a distinct manner, especially in case of a predictable FE, and hence can be used to efficiently detect such events and differentiate them from DDoS attacks. However, the unavailability of this parameter in the public domain datasets representing DDoS attacks has led to its exclusion from the parameter feature set used in the ensemble.

6.2.2 Server-load Features

The proposed ensemble based DDoS attack detection technique makes use of four server-load characteristic features, features which are indicative of a DDoS attack, especially at the application level. These features are grouped into CPU based and memory based features, and while not completely orthogonal, are used to capture various aspects of different DDoS attacks. All the features are observed locally on the target machine by reading various `/proc`[1] files at 5 second intervals.

[1] `/proc` is a boot-time mount point name given to procfs or proc file-system: a special files-system for Unix-like operating systems containing information about the system and various processes in a hierarchical file-like structure. Thus, `/proc` acts like a real-time API to the kernel, providing functionality to set or retrieve kernel information.

CPU Utilisation

DDoS attacks force the target server to process the incoming requests at the network and/or application level, thereby consuming a considerable amount of its processing power. Thus, the percentage of CPU utilisation, both at the system level (henceforth referred to as SysCpu) and at the user level (henceforth referred to as UsrCpu) can potentially indicate the occurrence of a DDoS attack.

Using both the system and user level CPU utilisation within the ensemble may facilitate the detection of different types of DDoS attacks. For example, a network layer flooding attack such as ICMP flooding will consume more of system level CPU rather than at user level, since no application running at the target server is involved in the attack. However, an application level DDoS attack, such as a HTTP GET flooding attack, will affect the target machine's CPU both at the system and user level.

An additional parameter indicative of overall load on the target is CPU Load. This represents the number of processes running or waiting for execution. This parameter is particularly useful in detecting SSL based DDoS attacks, where even a modest traffic rate can overpower the target server's CPU resources by forcing it to perform computationally expensive calculations. This parameter is available in all Unix and Unix-like platforms as 'load average', computed every 1, 5 and 15 minute interval. The 1 minute average is used in the ensemble, and is referred to as CpuLoad.

Memory Utilisation

Since applications allocate and deallocate memory as part of their normal execution, during a DDoS attacks or an FE, the use of 'real-memory' – RAM – may potentially increase. This parameter is referred to as RealMem in the ensemble.

In summary, the proposed ensemble based DDoS attack detection technique thus makes use of various parameters from the incoming network traffic side (NumPack, NewSrcIP, SrcIP and PackPerIP) and the target server-load side (SysCpu, UsrCpu, CpuLoad, and RealMem). These parameters are then correlated to detect the onset of different types of network and application level DDoS attacks, and to differentiate them from FEs. The following section discusses the attack detection technique used in the ensemble.

6.3 DDoS Attack Detection Technique

As discussed previously, at the onset of a network anomaly such as a DDoS attack or an FE, it is expected that the statistical properties of the parameters indicating anomalous activity may change abruptly. Hence, the problem of detecting such anomalies can be formulated as a change detection problem. In order to detect such changes, various detection techniques have been proposed by researchers and were described earlier in Chapter 2. The change detection technique used in the proposed ensemble based DDoS attack detection uses Exponentially Weighted Moving Average (EWMA): a variation of the moving average technique, discussed previously in Chapter 3 (see Section 3.3 for details).

The attack detection technique relies on detecting changes in the individual 'parameters of interest', gathered via an *offline* analysis of datasets representing different type of DDoS attacks, FE and normal traffic. These datasets have been generated using the traffic generation and testbed framework developed in Chapter 5 (see Section 5.2 for details). As discussed in Section 6.6, the changes detected in the individual parameters are then correlated using the Feature Correlation Matrix (FCM) to decide whether the current network activity represents a DDoS attack or not.

6.4 Experimental Set-up and Datasets

The proposed ensemble-based DDoS attack detection technique has been evaluated against a number of datasets representing different type of DDoS attacks (both at the network and the application layer level), FEs, and normal background traffic. These datasets, described in Section 6.4.3, are synthetically generated using the traffic generation and testbed framework developed in the previous chapter (see Section 5.2). Wherever possible, the synthetic generation of these datasets is based on the statistical properties of real-world datasets available in the public domain. The network architecture of the experimental set-up used for traffic generation is discussed in the following section, and Section 6.4.2 describes the methodology used for the experiments.

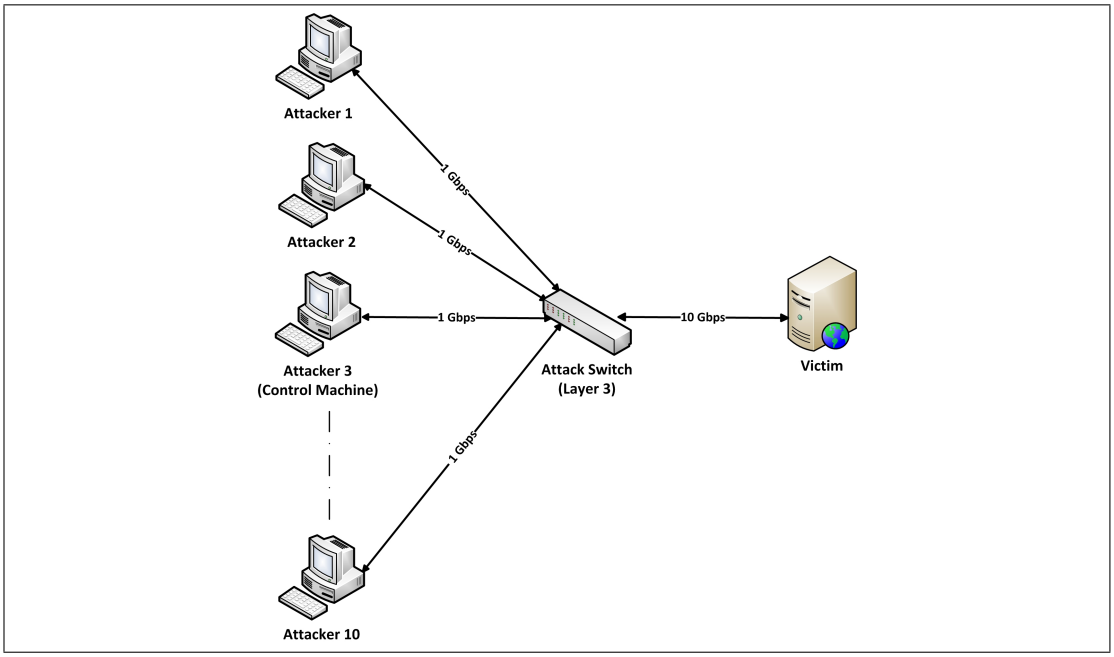


Figure 6.2: Experimental set-up

6.4.1 Network Architecture

The experimental set-up used for generating the required traffic (DDoS attacks, FEs, and normal traffic) consists of 11 interconnected Linux machines. The choice of operating system for all the host machines is governed by the traffic generator, Botloader, which is currently implemented for the Linux environment. From the 11 host machines used in the set-up, 10 machines behave as *attackers*, and are connected via a layer 3 switch, and while the remaining machine acts as a *victim*, as shown in Figure 6.2. The traffic generated by each attack machine is aggregated before being sent to the victim.

All of the 10 attack machines are standard PCs with 3.0 GHz Intel Core2 processors, 4 GB of memory, an integrated 1 Gb NIC, and run Ubuntu or Debian as the operating system. The internal configuration of all the attack machines, i.e. the default IP addresses of the interfaces and the IP forwarding (enabled), is identical to that described in Chapter 5 (see Figure 5.3 for additional details). All of the attack machines run their own instance of Botloader, along with some essential services such as SSH (Table 6.1). All the attacking machines are interconnected via a 1 Gbps link.

The victim machine is a Dell PowerEdge R710 with two six-core Intel Xeon 2.27 GHz processors (hyper-threaded), 32 GB of memory, running Ubuntu 10.04

(Server) as the OS, and Apache2 (with mpm worker module) as the web-server. Table 6.1 summarises the hardware and the software components of the experimental set up. The following section describes the datasets used in the experimentation.

Table 6.1: Hardware and the software components of the experimental set up

	Components	**Host Machines**	
		Attacker (x 10)	Victim
Hardware	Architecture	64-bit	64-bit
	Cores	4	24
	CPU (GHz)	3.0	2.27
	Memory (GB)	4	32
	NIC (Gb)	1	10
Software	Operating System	Ubuntu 10.04/Debian 6.0 (Desktop)	Ubuntu 10.04 (Server)
	Essential Services	Botloader, SSH	Apache 2.2.16, SSH

6.4.2 Methodology

As previously discussed in Chapter 5 (see Section 5.2.2 for additional details), the traffic generation and testbed architecture includes a *monitor* program running on the target machine. This program is executed locally to collect the server load based data by reading various `/proc` files, and copy back the collected data onto the control machine or Attacker 3 (see Figure 6.2), once the simulation is over. This approach of using a separate program, running locally on the target machine, to collect the server load-based data differs from an earlier strategy of probing the server via SNMP GET requests. The idea of the monitor program is to avoid any interference between SNMP packets and the actual simulated traffic in general, and in particular during UDP flooding attack simulation (since SNMP uses UDP as its transport protocol). Hence, before the start of every simulation, the monitor program is remotely copied to, and executed on the target machine with a high execution priority.

As discussed in the following section, the values of the incoming network traffic parameters used within the ensemble were extracted either from the collected pcap files (collected by Tcpdump) in the case of ICMP and SSL flood attack or from the Apache web-server logs in the case of a HTTP GET flood attack, FE, or normal traffic scenario. However, the use of Tcpdump to log the traffic was found to interfere with the monitoring of the server load parameters, especially those related to memory, thereby confounding the actual results. Figure 6.3

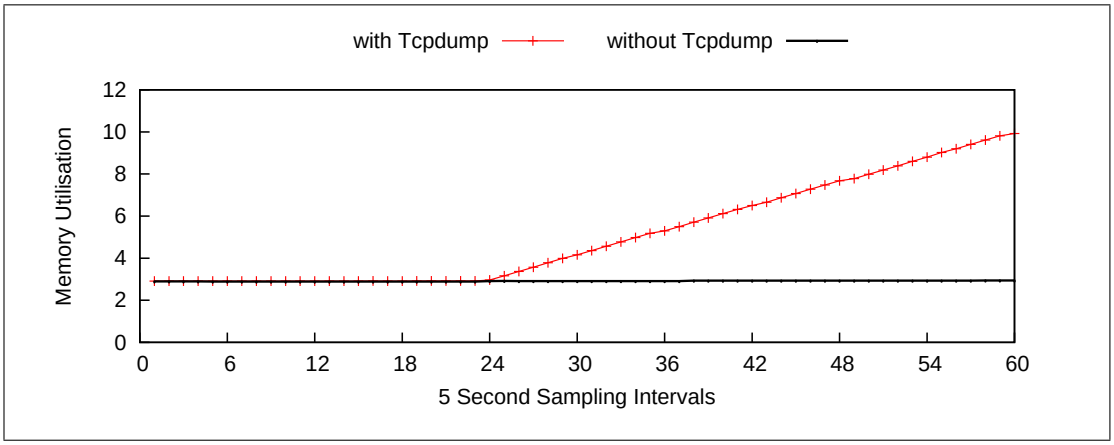


Figure 6.3: Tcpdump influence on memory utilisation.

shows this behaviour during the ICMP flooding attack simulation by comparing the percentage of memory utilisation of the target system when monitored with and without running a Tcpdump instance. Hence, in order to minimise any such interference with the monitoring of the target system, two iterations of every experiment were performed, one with Tcpdump on the target machine and one without. The iteration with Tcpdump was used to collect and analyse the network traffic parameters, and the iteration without Tcpdump was used for the server load analysis. The results obtained from each of the two iterations were then combined.

6.4.3 Datasets

The datasets used for testing and evaluation are divided into three broad categories: DDoS attack datasets, FE datasets, and normal background datasets. Within the DDoS attack category, different datasets representing network layer flooding and application layer flooding are used.

DDoS Attack Datasets

Three DDoS datasets are used for evaluating the proposed attack detection technique. One of these datasets represents a network layer flooding attack while the other two represent different types of application level attacks. Each of these attack datasets are simulated for a period of 5 minutes: first by two minutes of normal activity, followed by three minutes of flooding. Each of these datasets is now described.

1. The ICMP dataset is based on the CAIDA DDoS Attack Dataset, which represents a real-world DDoS attack and contains approximately an hour (20:50:08 UTC to 21:56:16) of pseudonymised traces from a DDoS attack that occurred on August 4, 2007 [51]. The traffic profile of the CAIDA dataset represents a flooding attack where the attacking machines send a large number of ICMP Echo Requests (ping) to the victim aimed at consuming the victim's networking bandwidth. The simulated attack uses an identical packet rate and the same number of source IP addresses, as in the CAIDA dataset, to generate a 5-minute attack dataset containing one-way traffic i.e., directed against the target host. Generation of this dataset was previously described in Chapter 5 (see Section 5.4.1 for details).

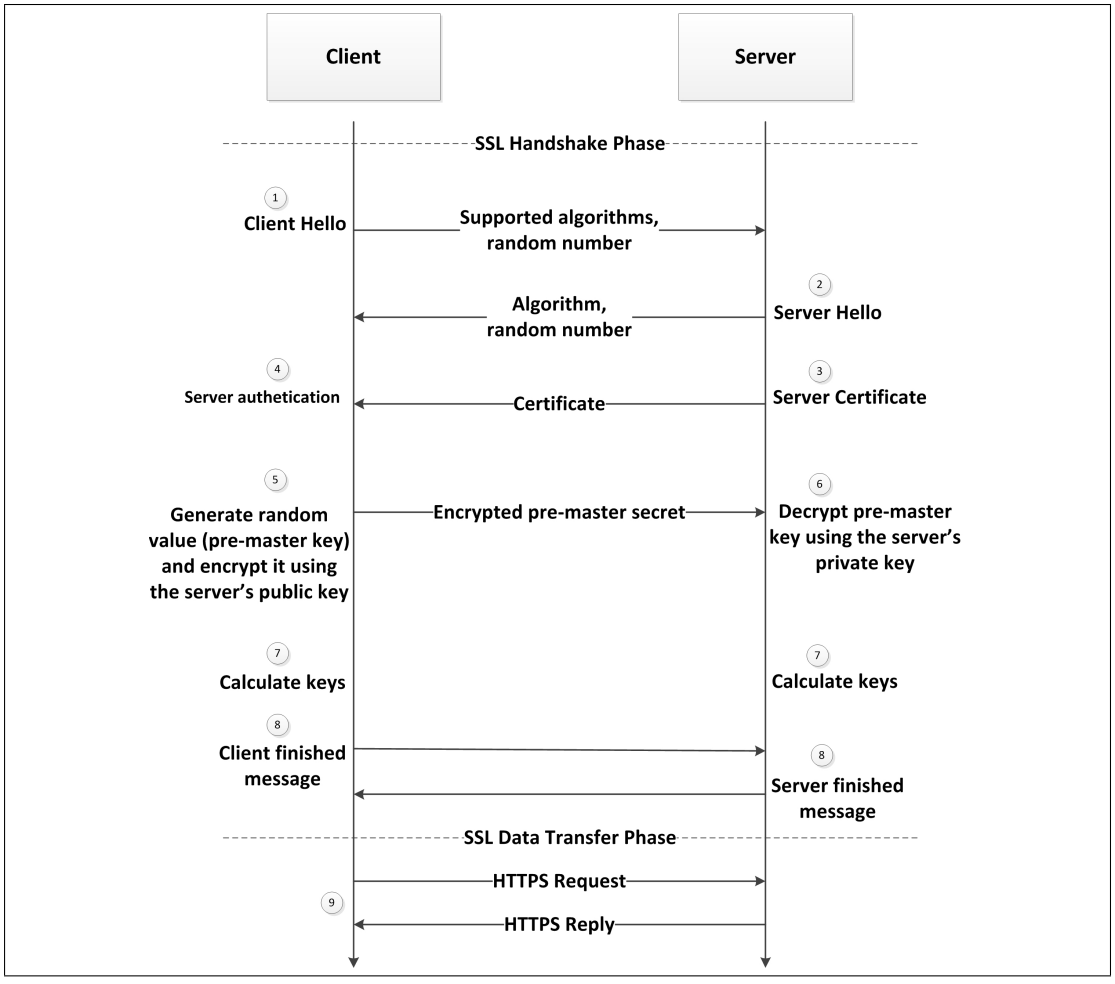


Figure 6.4: SSL protocol.

2. The HTTP dataset represents an application level attack where the participating bots (aliased IP addresses) send a large number of conventional

HTTP GET requests to the target server and download the complete response. The access requests are made by randomly choosing a Uniform Resource Locator (URL) from a file containing a list of all URLs available on the target web-server. This dataset contains one-way HTTP traffic to the web-server where the attack traffic starts at the two-minute mark into the traffic trace.

As pointed out by Gavrilis et al., [46] this type of attack is easy to launch, but can seriously affect a web-server's ability to service legitimate clients. This attack can also be difficult to repel, owing to a strong similarity between the legitimate and attack requests. A real-life application of this attack against a company in the U.S. allegedly caused their website to go off-line for a period of two weeks [95].

3. The SSL dataset represents a more recent application level attack that exploits the Secure Socket Layer (SSL) handshake protocol to exhaust server resources [117]. SSL, or its successor Transport Layer Security (TLS), is a widely used protocol over the Internet to provide a secure channel of communication between client and server. The SSL/TLS connection starts off with a 'client hello' 'server hello' exchange that chooses, among other things, an encryption algorithm acceptable to both sides, as shown in Figure 6.4. The server then sends its certificate. The client is supposed to verify the certificate and encrypt a random key using the server's public key. However, in the attack, the client merely sends a canned response (encrypted pre-master secret) to the server, so avoiding any cryptographic cost. If the RSA algorithm is chosen with a large keysize the server is forced to perform an expensive decryption, up to 40 times the cost of the client [19]. The attacker then breaks off the connection and starts a new one.

   In the original version of this attack, a normal SSL handshake is completed, but is immediately followed by a renegotiation of the keys. As soon as the first renegotiation is complete, another renegotiation is requested, and so on. If the SSL renegotiation is disabled on the server (the recommended security practice), the attacker simply closes the connection after the initial negotiation is complete and then opens up a new connection to restart the negotiation process.

The `ssl_bot` implemented within Botloader is based on Trojnara's sslsqueeze program [118]. Hence, this SSL dataset represents an attack where every participating bot asynchronously initiates a large number of SSL handshakes. Half-way through the handshake it sends some garbage data using a 2048 bit key, and immediately breaks off the connection and starts a new one. The one-way traffic of such an attack is used as the SSL dataset. A detailed description of the HTTP GET flood and SSL flood dataset generation is given in Appendix C.

Flash Event Dataset

This simulated dataset, henceforth referred to as the FE dataset, is based on a subset of the 1998 FIFA World Cup Dataset provided by the Internet Traffic Archive, containing 92 days of pseudonymised Web server logs of all the access requests made to the websites hosting the 1998 FIFA World Cup [12]. Each of the individual peaks or bursts of traffic represents individual FEs on a small scale as compared to the overall event [13]. A detailed description of this dataset is provided in Section 4.1.4. From the complete 1998 FIFA World Cup Dataset, a six hour trace of incoming traffic around the 1^{st} semi-final match was used for generating the FE dataset.

Normal Traffic Dataset

This dataset, henceforth referred to as the NT dataset, contains a simulated traffic trace of a 5 minute non-flash-event activity before the 1^{st} semi-final match of the 1998 FIFA World Cup Dataset.

Thus five datasets (ICMP, HTTP, SSL, FE and NT) representing different type of network activities viz., network level flooding, application level flooding, flash event, and normal background traffic are used for evaluating the proposed ensemble based DDoS attack detection technique. Table 6.2 summarises the high-level characteristics of these datasets.

Table 6.2: High-level characteristics of the datasets used for evaluation.

Characteristics	**DDoS Attacks**			**Flash Events**	**Normal Traffic**
	CAIDA	HTTP	SSL	FIFA	NT
Duration	5 minutes	5 minutes	5 minutes	6 hours	5 minutes
Activity	ICMP Echo Flood	HTTP GET Flood	SSL Flood	Flash Event	Normal Activity
Packet Type	ICMP	HTTP	TCP	HTTP	HTTP
Number of Packets/GET requests	21,636,393	2,822,932	5,691,482	18,032,403	89,959
Number of Source IPs	9,320	1,500	1,813	78,650	29,845
Peak packet rate (packets per second)	134,116	17,624	31,825	3,996	423

Table 6.3: Parameter values used in Change Detection Algorithm.

Domain	Features	Parameters			
		η	λ	k	t
Network Traffic	NumPack	0.50	0.02	3	5
	NewSrcIP	0.50	0.02	1	5
	SrcIP	0.25	0.02	3	5
	PackPerIP	0.25	0.02	3	5
Server Load	SysCpu	0.5	0.02	3	5
	UsrCpu	0.50	0.02	3	5
	CpuLoad	2.00	0.02	3	5
	RealMem	0.15	0.02	3	5

6.5 Analysis of Change Detection Results for Individual Features

In this section, the change detection algorithm using EWMA, as described earlier in Chapter 3 (see Section 3.3 for details), is applied to the network traffic and server-load features discussed previously.

6.5.1 Incoming Network Traffic Features

The EWMA technique is applied to individual network traffic features by setting the values of its tunable parameters. The values chosen for these parameters are based on previous work in change detection using EWMA, specifically in DDoS related research [105], as previously discussed in Chapter 3 (see Section 3.3 for details). Table 6.3 summarises the optimal values, which were experimentally adjusted to improve detection accuracy.

Incoming Traffic Volume

The abrupt change in the incoming traffic for the three different DDoS attack datasets is shown in Figure 6.5, where the attacks start at 2 minutes (mark 24 on x-axis) into the traffic trace. The change detection technique identifies this sudden change in the incoming packets at the start of attack with the decision function showing 1, within each dataset. The decision function continues to show 1 for the remainder of the attack, for the HTTP and SSL datasets, implying that

the current packet rate is greater than the mean packet rate until that time interval.

In the ICMP dataset, which emulates the CAIDA traffic, the incoming traffic rate shows a few dips at various time intervals. The decision function picks the biggest of these dips and drops back to 0 at around 50 second mark, staying at 0 for the next couple of intervals before going back to 1. It is speculated that this behaviour is due to heavy traffic that the server is constantly subjected to, which leads to some packets being dropped by the kernel. However, it is to be noted that such behaviour is also seen in the original CAIDA dataset (see Figure 5.9 for details), where the incoming traffic is erratic and often drops to lower levels during the attack phase.

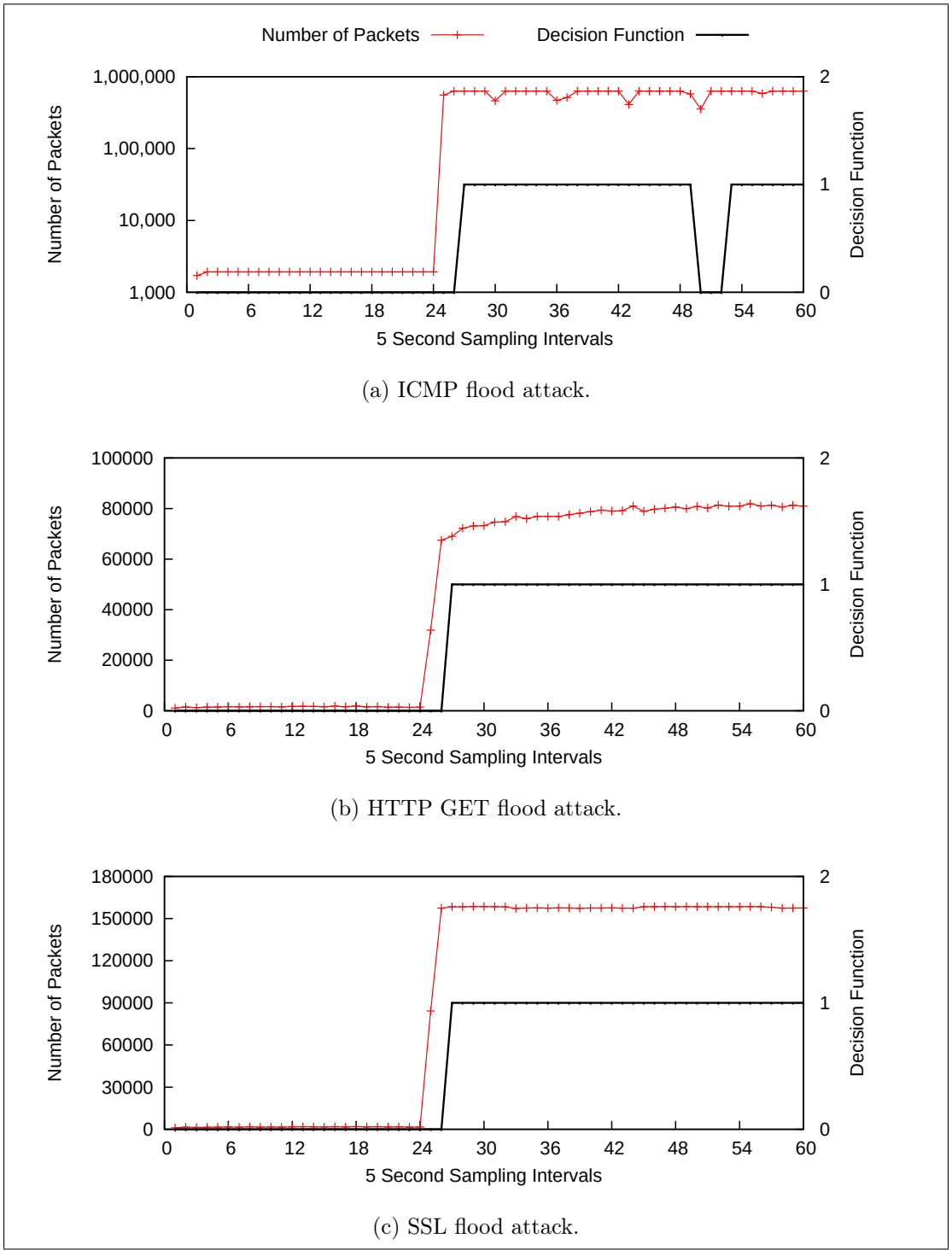
A small *phase-shift* between the attack traffic and the decision function is due to the value of k used in the change detection technique which flags a change (indicated by 1) after 3 consecutive occurrences of the detection condition have been identified.

The changes in incoming traffic for non-attack traffic datasets (FE and NT) is not as immediate and significant as in the case of the DDoS attack datasets. This characteristic is shown in Figure 6.6, where a negative result from the decision function indicates that the change in the incoming traffic is much more gradual and not of the same magnitude as in the case of the DDoS attacks.

New Source IP Addresses

As discussed previously in Chapter 3 (Section 3.1), DDoS attacks are accompanied by a sudden change in the new or previously unseen IP addresses, as demonstrated by the results illustrated in Figure 6.7. This characteristic i.e., change in rate of new source IPs, is different for non-attack traffic scenarios such as FEs, where the target server constantly observes new source IPs with the incoming traffic, as shown in Figure 6.8a. These variations in the number of new source IPs, both for DDoS attacks and non-attack traffic scenarios, are shown in Figures 6.7 and 6.8 respectively. The new source IP feature in the original FIFA dataset (Figure 6.8a) is distinctly different from any of the DDoS attack datasets (Figure 6.7) and thus can be used to differentiate between these two network anomalies.

For this particular feature, the original FIFA dataset (Figure 6.8a) has been used instead of the simulated FE dataset (Figure 6.8b). This is mainly because



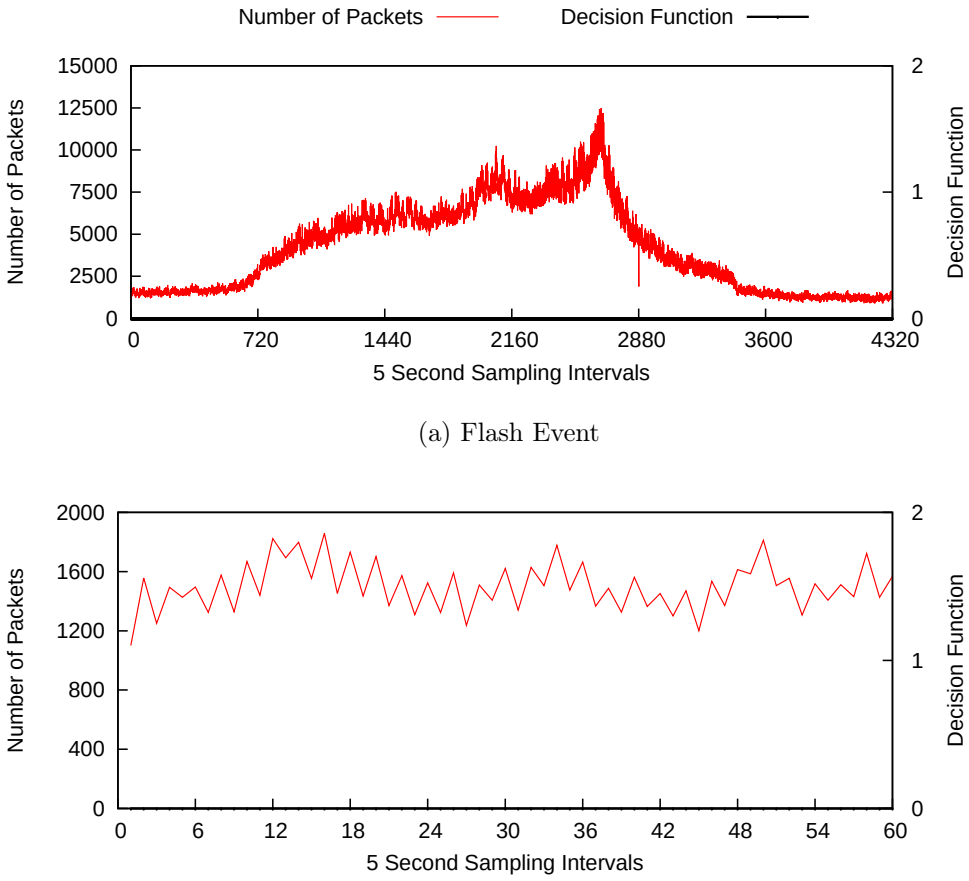
(a) ICMP flood attack.

(b) HTTP GET flood attack.

(c) SSL flood attack.

Figure 6.5: Incoming traffic volume variations for different DDoS attack datasets.

of an artefact of the Botloader program which tends to use all the available bots (aliased IPs) at the start of the simulation, and reuses them during the course of the simulation. This artefact can be seen in the simulated FE dataset shown



(a) Flash Event

(b) Normal Traffic

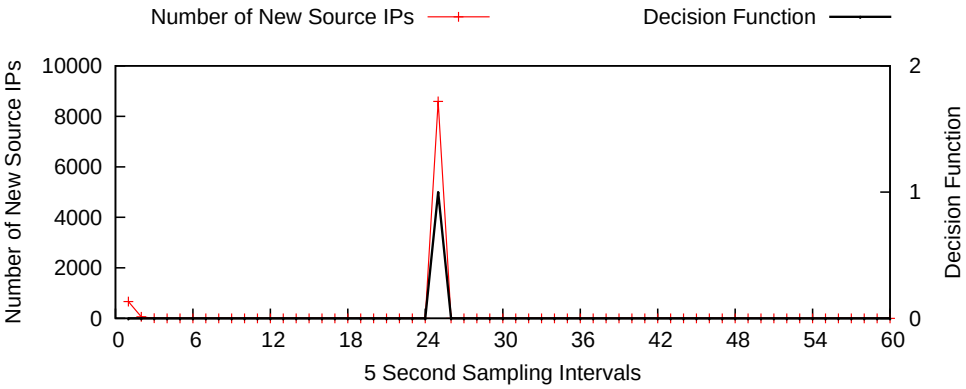
Figure 6.6: Incoming traffic volume for non-attack datasets.

in Figure 6.8b where all the available bots are used (or seen by the target host) within the first hour of the simulation, resulting in no new IP addresses for the rest of the simulation.

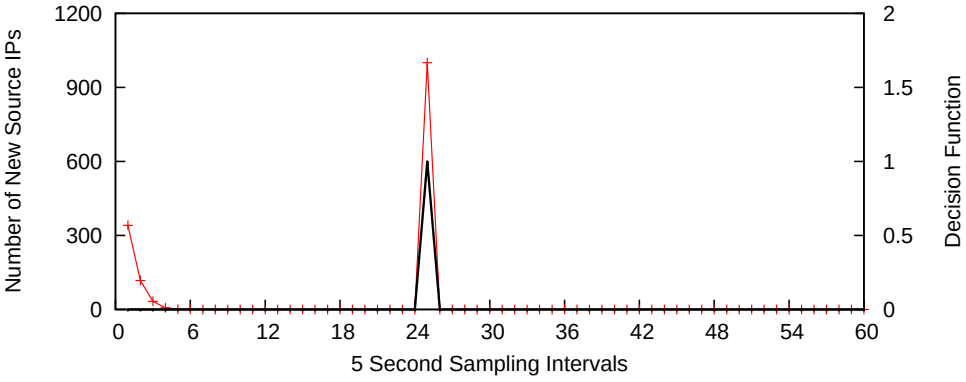
Number of Source IP Addresses

In DDoS attacks, as the bot master synchronously uses the set of compromised machines or bots, the target machine experiences a sudden increase in the number of source IPs. As these bots are reused during the course of attack to continue to send the desired traffic, the target server continues to experience a high number of source IPs per interval throughout the attack. This characteristic is shown in Figure 6.9, which shows an increase (or a change) in the number of source IPs with the onset of attack, which persists for the rest of the attack.

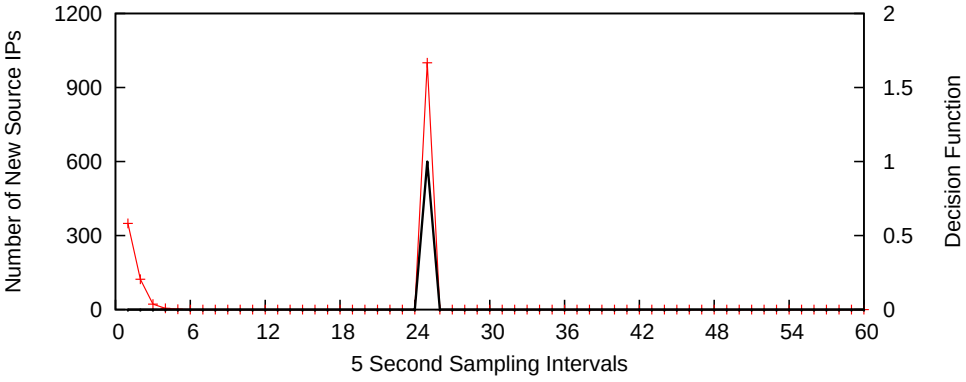
In case of an FE scenario, the number of IPs per sampling interval also



(a) ICMP flood attack.



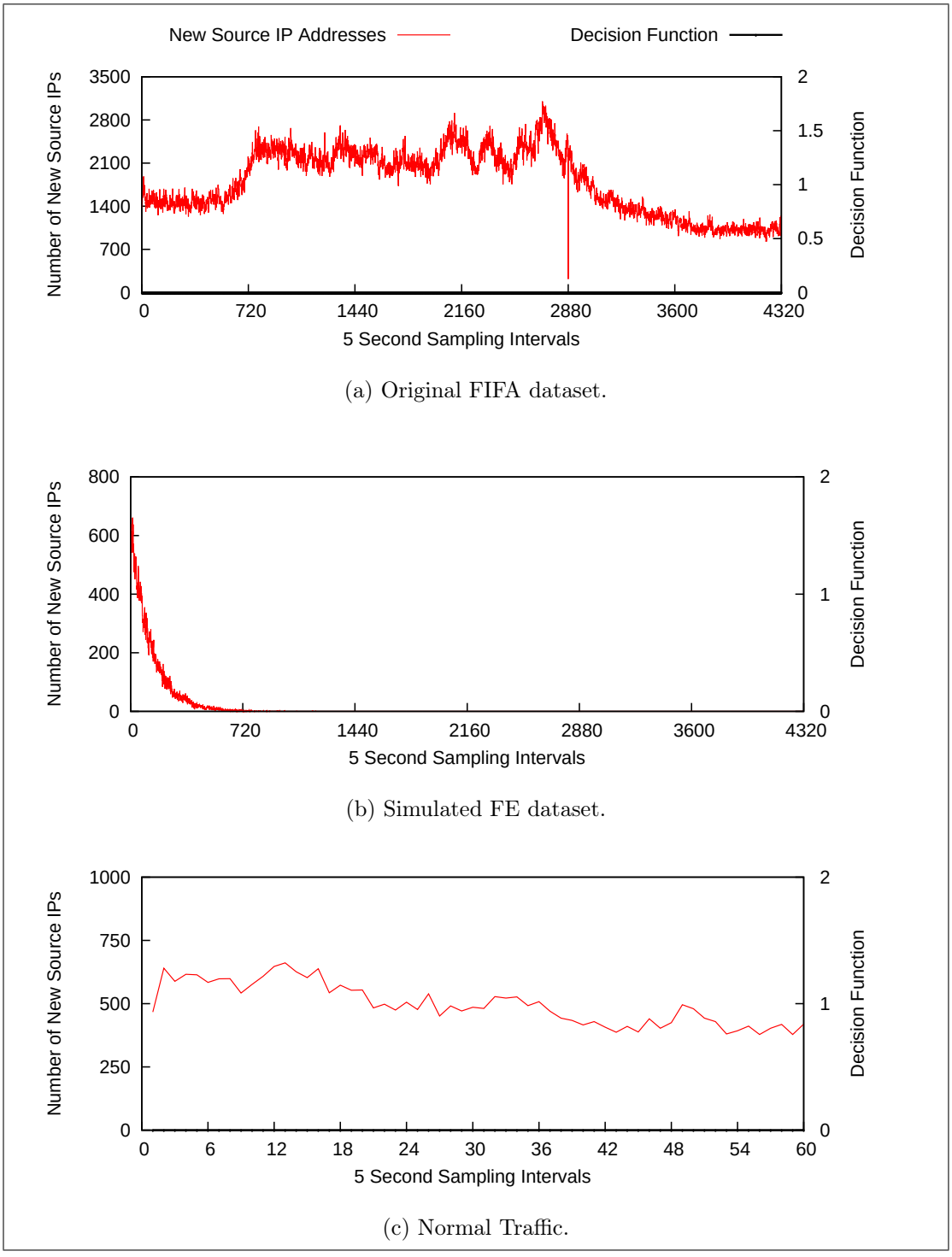
(b) HTTP GET flood attack.



(c) SSL flood attack.

Figure 6.7: New source IPs for different DDoS attack datasets.

increases, but more significantly at the start of the event i.e., at the start of the *flash-phase*, just as news of the event has probably reached the masses, pushing them to go online to get additional information. For additional details regarding



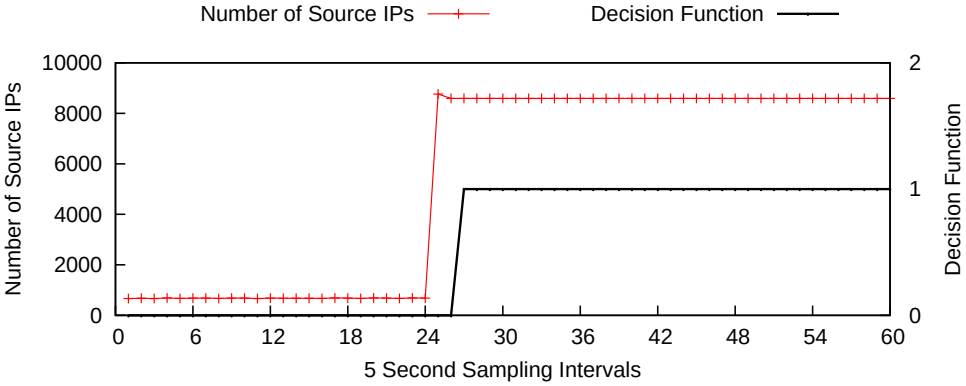
(a) Original FIFA dataset.

(b) Simulated FE dataset.

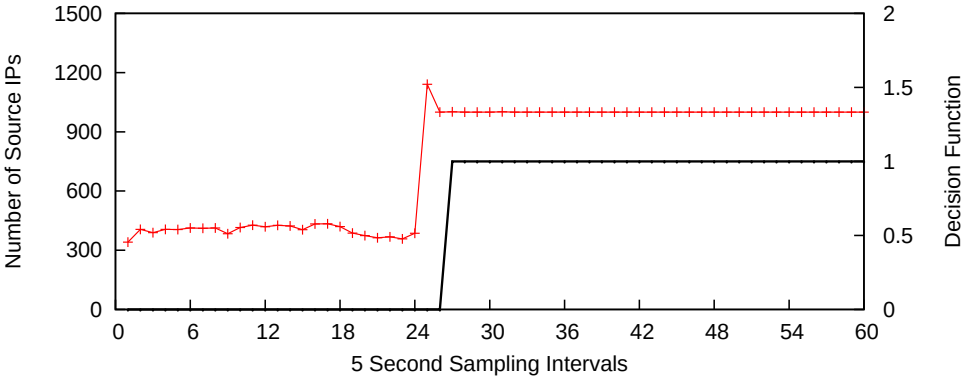
(c) Normal Traffic.

Figure 6.8: New source IP addresses for non-attack datasets.

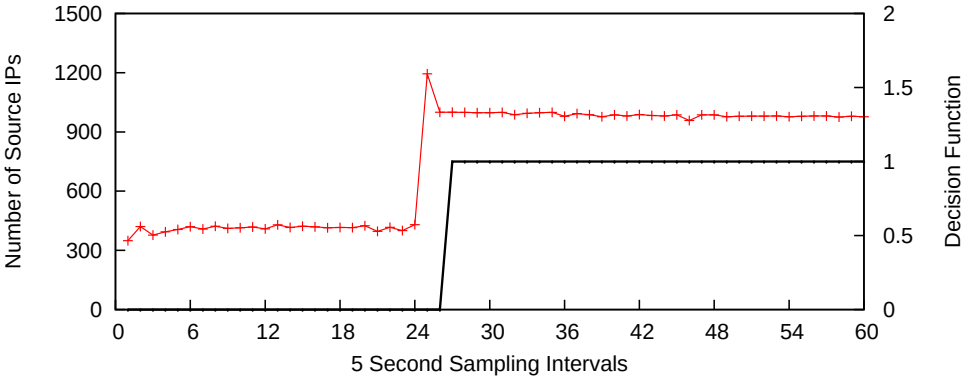
the *flash-phase*, please refer to Chapter 4 (Section 4.3). This feature is captured in the simulation and is shown in Figure 6.10, where a change in the number of source IPs is triggered at the start of *flash-phase* i.e., at around the 720 second



(a) ICMP flood attack.



(b) HTTP GET flood attack.



(c) SSL flood attack.

Figure 6.9: source IP addresses for different DDoS attack datasets.

mark, and is successfully detected by the decision function.

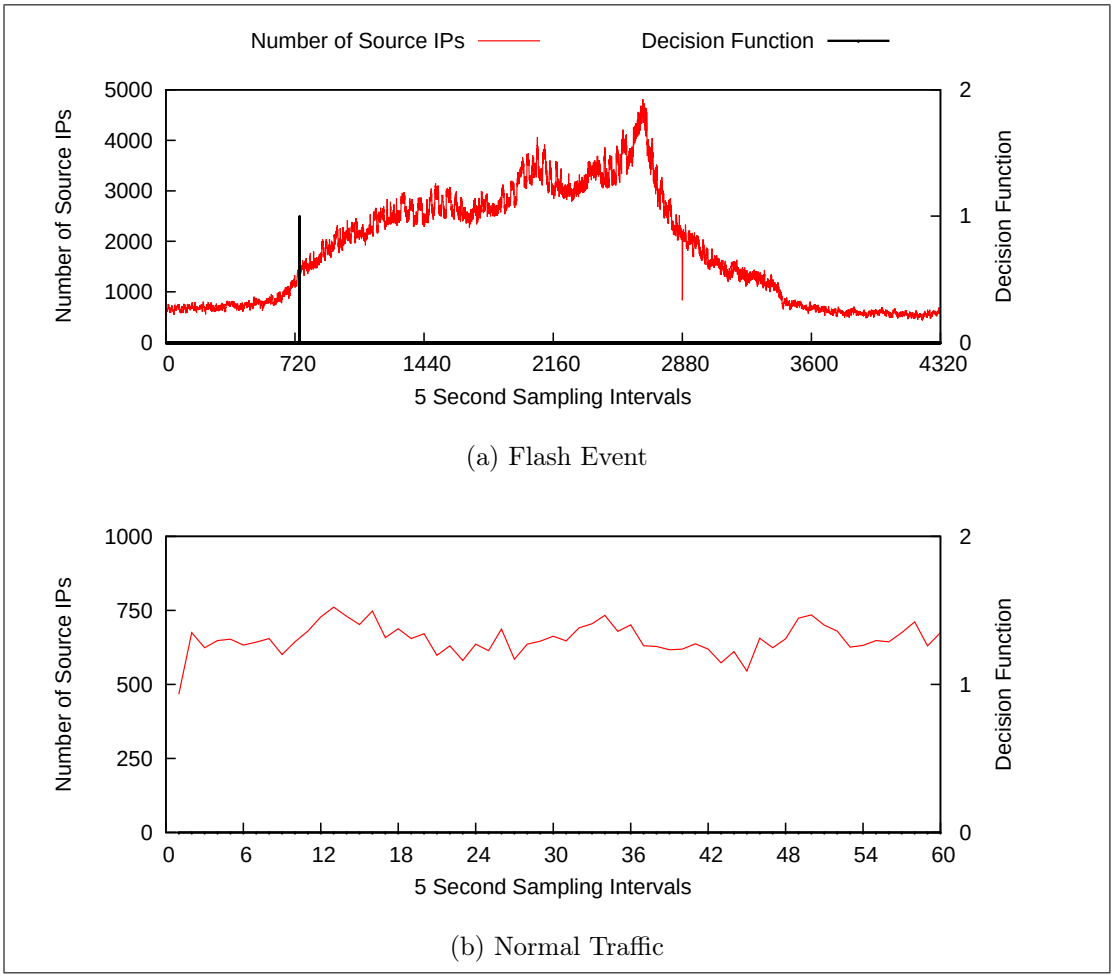


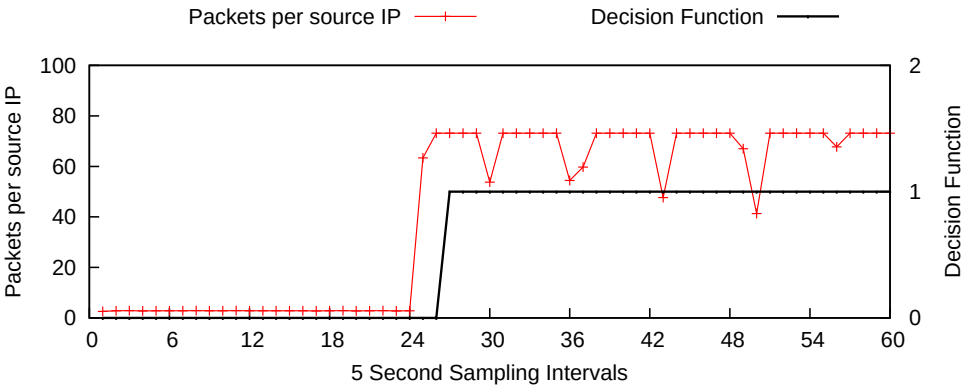
Figure 6.10: Source IP addresses for non-attack datasets.

Incoming Traffic Distribution

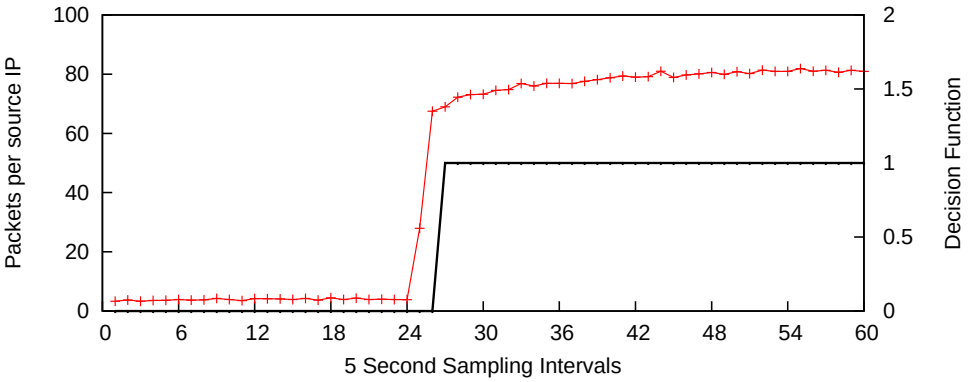
The incoming traffic distribution in terms of the number of packets per source IP address is much higher in the case of DDoS attacks as compared to FEs. During DDoS attacks, a limited number of compromised machines (bots) forces the bot master to send a large number of packets from each bot to maximise their utilisation and the damage they can cause on the target. However, this characteristic is different during FEs, where the legitimate clients are usually interested only in a specific set of information related to that event, thereby having a relatively low number of requests per client (source IP address) as compared to DDoS attacks.

The number of packets per source IP feature for DDoS attacks is shown in Figure 6.11. The dips observed at fairly regular intervals during the ICMP flooding attack correspond to the dips observed in the incoming traffic volume

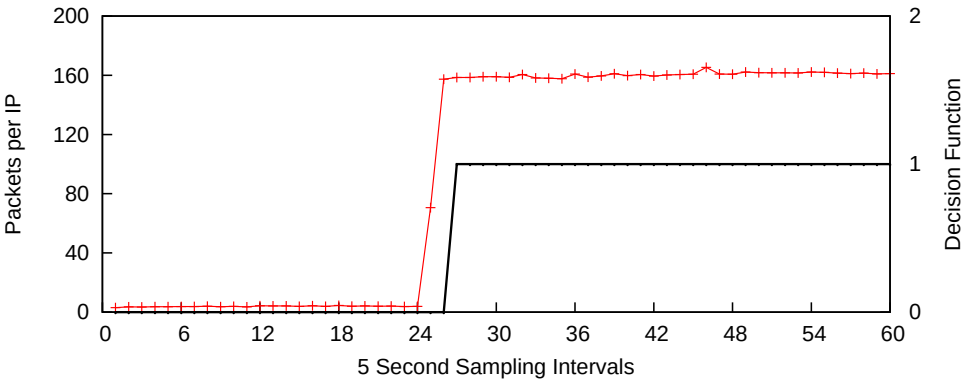
for this particular attack, as previously shown in Figure 6.5.



(a) ICMP flood attack.



(b) HTTP GET flood attack.

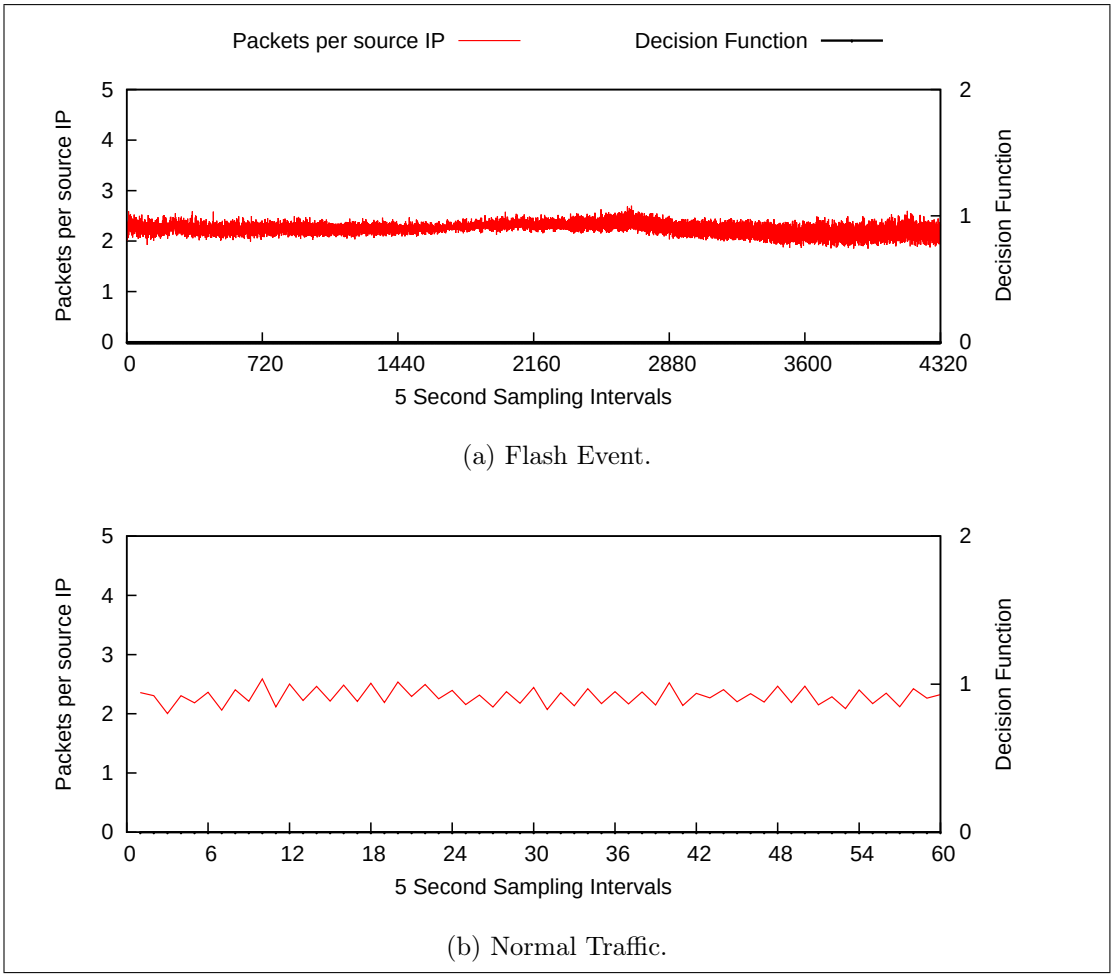


(c) SSL flood attack.

Figure 6.11: Incoming traffic distribution for different DDoS attack datasets.

In the case of the non-attack (FE and normal traffic) scenarios, the number of packets per source IP remains almost constant, resulting in no change notified

by the decision function, as shown in Figure 6.12. The differences in this feature i.e. number of packets per source IPs, can therefore be used to distinguish DDoS attacks from FEs.



(a) Flash Event.

(b) Normal Traffic.

Figure 6.12: Incoming traffic distribution for non-attack datasets.

6.5.2 Server-load Features

The change detection technique is now applied to individual server-load features viz., system level CPU utilisation (SysCpu), user level CPU utilisation (UsrCpu), CPU load (CpuLoad) and real memory utilisation (RealMem), to identify the parameters which could potentially indicate the onset of a DDoS attack and differentiate them from similar looking FEs.

System CPU

DDoS attack datasets, especially the HTTP and the SSL flood attack datasets, show a considerable increase in the percentage of system level CPU utilisation as shown in Figure 6.13. For SSL flooding attacks this increase is mainly attributed to the expensive decryption operations involved during the SSL handshake, resulting in a near-capacity (approximately 90%) utilisation of the available processing capability.

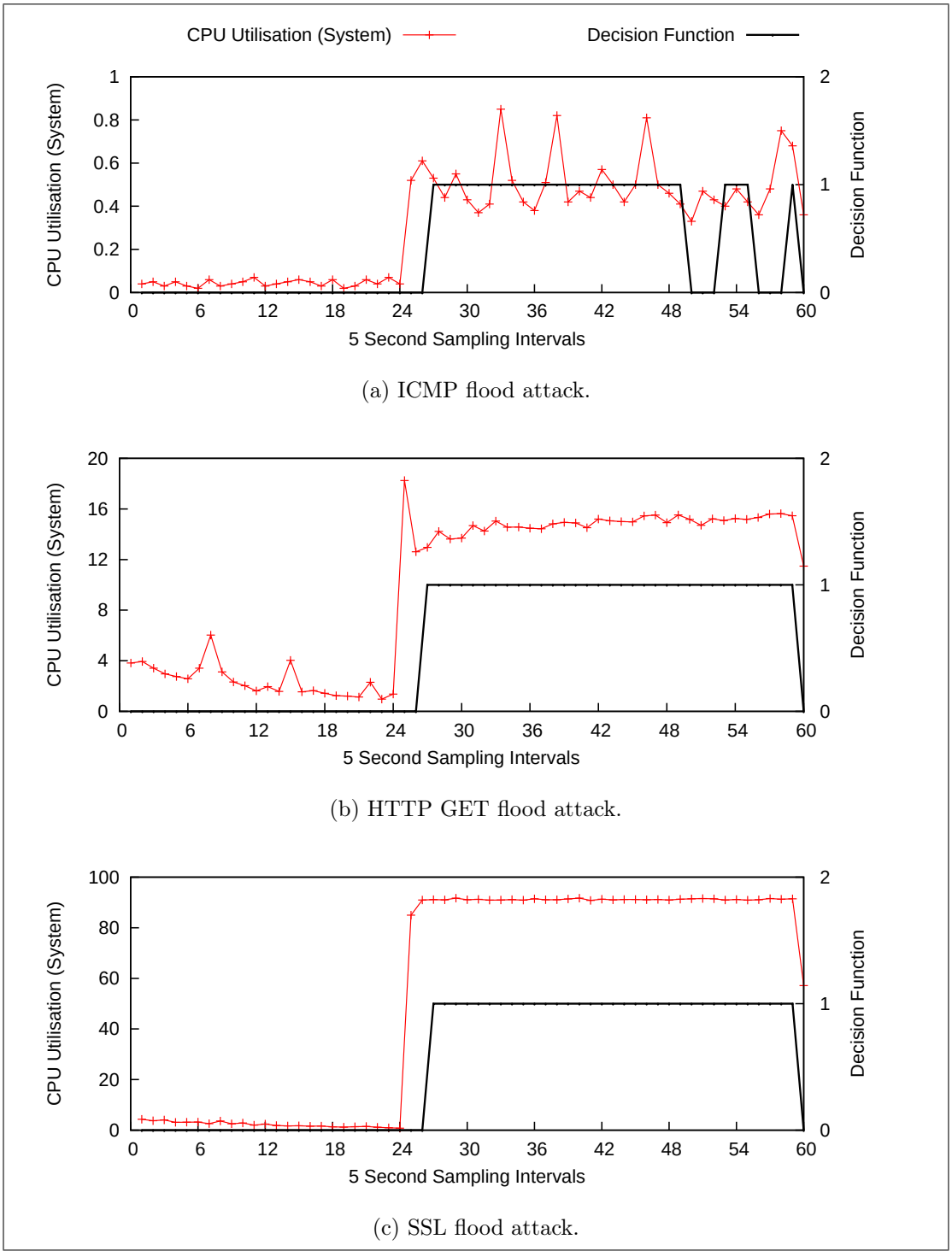
During the HTTP GET flood attack, as each incoming request is made to an existing resource on the target server, it has to be processed by the server, resulting in high CPU utilisation. It is, however, interesting to note that during the ICMP flooding attack, CPU utilisation does increase but only slightly, as compared to the pre-attack period. This is possibly due to the heavy traffic load of nearly 125,000 packets a second, that the target is subjected to for a period of 3 minutes.

During the FE traffic scenario, the system level CPU increases rather gradually from approximately 0.1% to a maximum of 2.66% (excluding the start-up value of 4.85%) over a period of 3.6 hours, and hence is not picked by the change detection technique. These variations in the system level CPU utilisation for FE dataset are shown in Figure 6.14.

For the normal traffic dataset, the system CPU starts-off with a relatively high value, gradually decreasing and stabilising at nearly 0.5% half-way through the simulation. This initial high value of the system level CPU, also observed in other simulated traffic traces, is likely due to the background processes that are activated at the boot time. However, in any case it does not affect the overall behaviour being observed.

User CPU

The user level CPU utilisation varies as expected for the ICMP flooding attack i.e., near 0% utilisation, as no user level processing is involved in such an attack. In the case of HTTP GET and SSL flooding attacks, as the target server is forced to process the incoming requests that target a running application, this results in a higher user level CPU utilisation as compared to the network layer flooding attack. For these attacks (HTTP GET and SSL) the CPU utilisation shows a dip, towards the end of their simulations resulting in a 0 value for the decision function. However, this behaviour in either case does not interfere with



(a) ICMP flood attack.

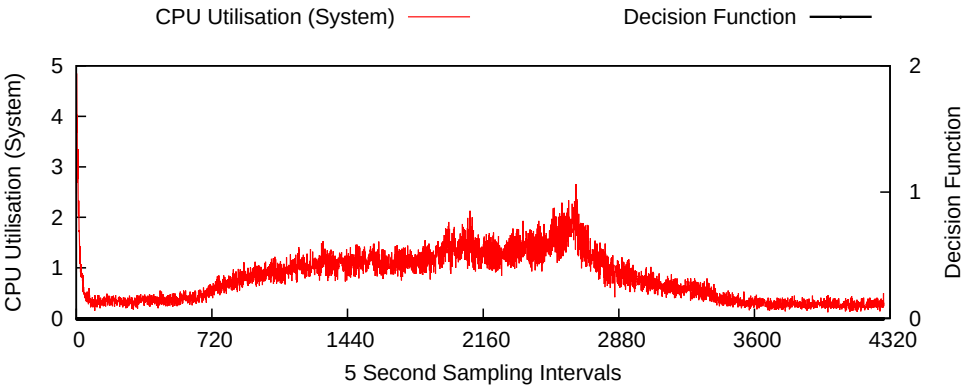
(b) HTTP GET flood attack.

(c) SSL flood attack.

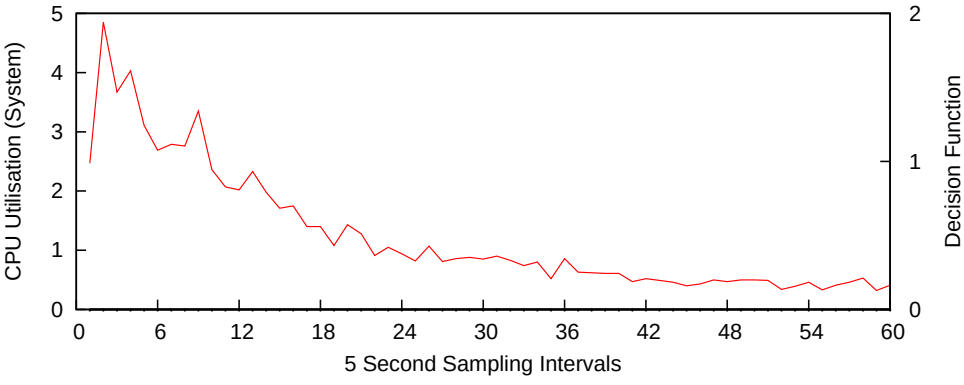
Figure 6.13: System CPU for different DDoS attack datasets.

the overall trend of the parameter under observation.

In the case of the FE dataset, it is interesting to note the similarity in shape for different load-based parameters such as user CPU, system CPU, with the



(a) Flash Event.



(b) Normal Traffic.

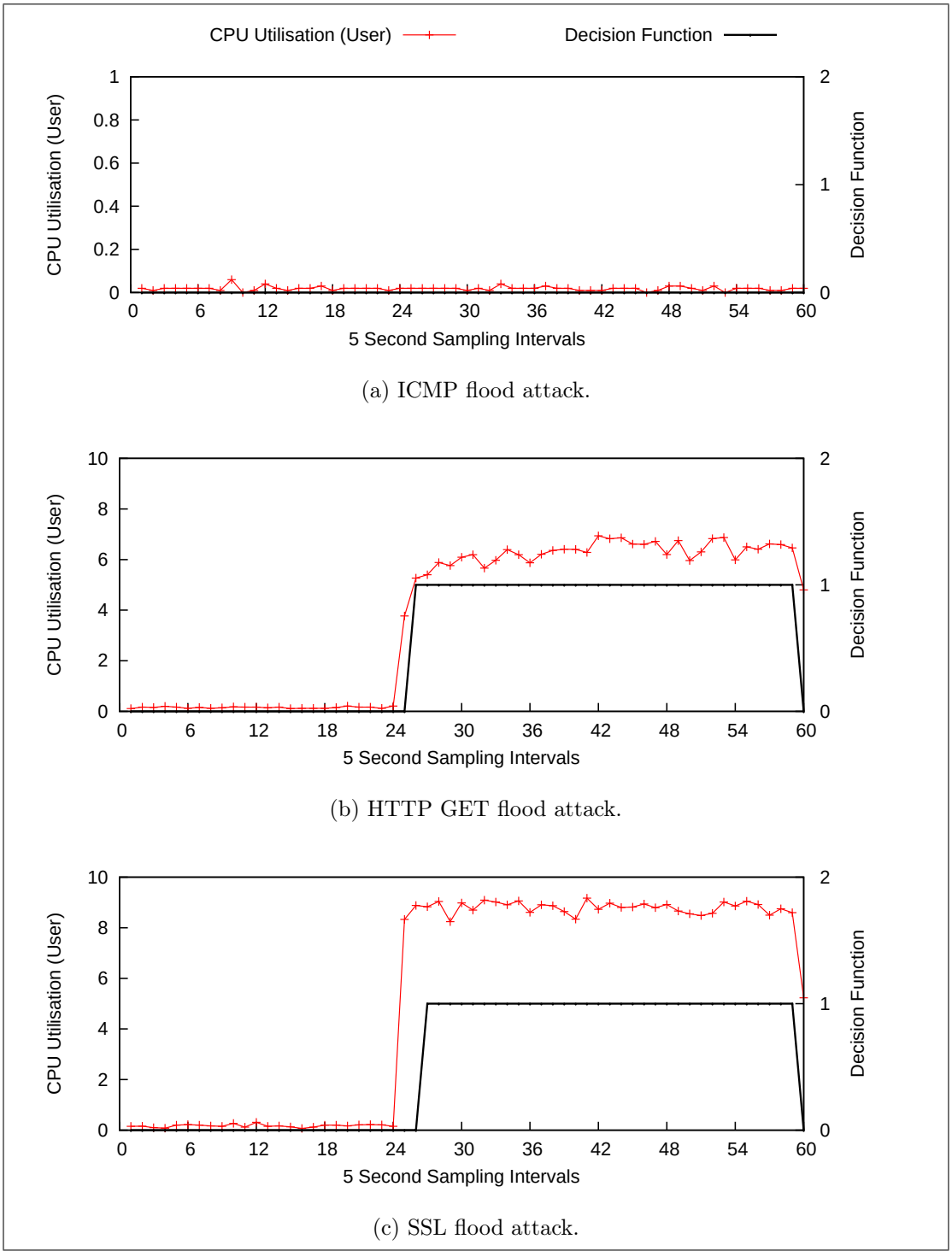
Figure 6.14: System CPU for different non-attack datasets.

network traffic based parameters, such as incoming traffic volume and source IPs. The user level CPU utilisation for the FE dataset is again gradual (as in the case of system level CPU utilisation) and is therefore not identified as a change. This behaviour is shown in Figure 6.16a. A relatively low packet and data rate for the normal traffic dataset has resulted in an average 0.15% of user level CPU utilisation, as shown in Figure 6.16b.

The results of applying the change detection to various server-load based parameters are now discussed.

CPU Load

Another server load-based parameter used in the proposed ensemble is the CPU load averaged over 1 minute intervals. The variations in this feature for different attack datasets is shown in Figure 6.17. The target machine used for these



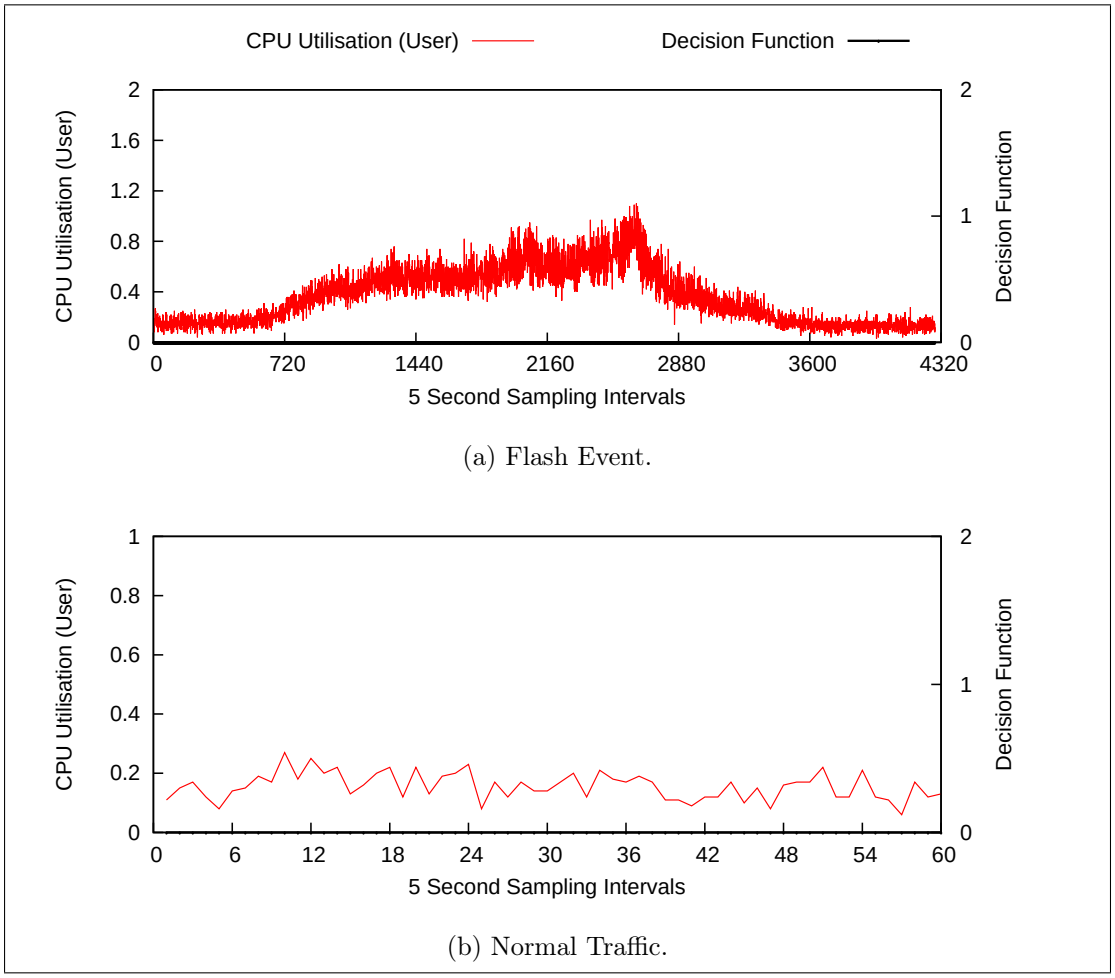
(a) ICMP flood attack.

(b) HTTP GET flood attack.

(c) SSL flood attack.

Figure 6.15: User CPU for different DDoS attack datasets.

simulations has two six-core processors (hyper-threaded), thus making a total of 24 available cores. Therefore, a value of 24 on the left-hand y-axis for the graphs presented for this feature indicates a 100% utilisation of the available



(a) Flash Event.

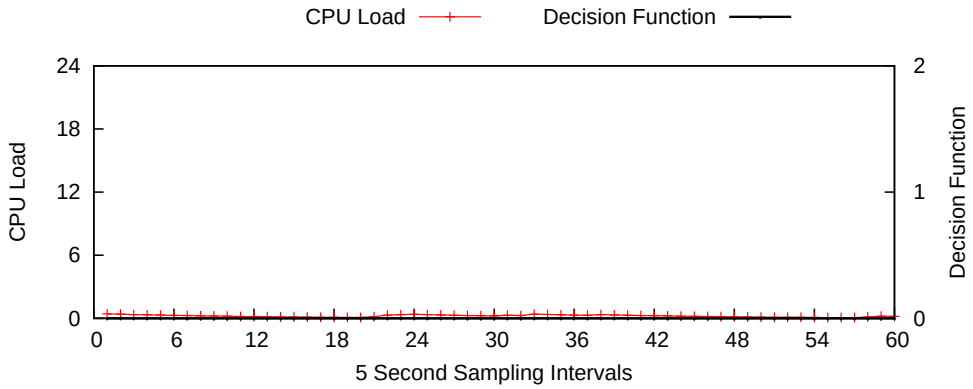
(b) Normal Traffic.

Figure 6.16: User CPU for different non-attack datasets.

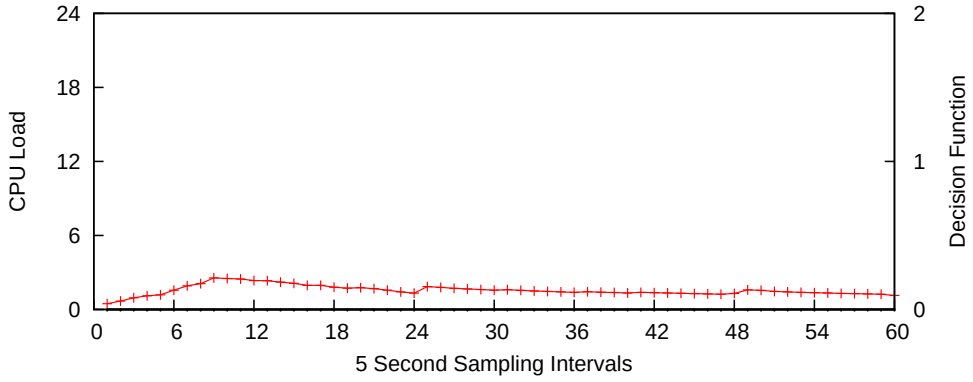
CPU power.

The CPU load is almost constant at 0 for ICMP flooding attack as seen in Figure 6.17a. In the case of the HTTP flooding attack, besides an unexpected increase at the 9^{th} interval, it increases slightly just after the onset of the attack i.e., at the 24^{th} observation interval, after which it remains fairly constant at around 1.42. This behaviour can be seen in Figure 6.17b.

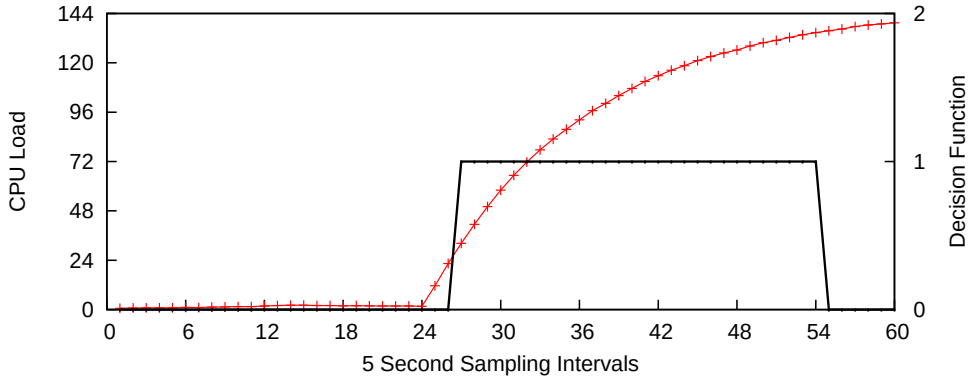
As shown previously in Figure 6.13, during an SSL attack, the target server's CPU is heavily loaded, due the computationally expensive decryption operations involved in the SSL handshake. This trend is also reflected in the CPU load parameter for the SSL attack dataset, where immediately following the onset of attack (at the 24^{th} sampling interval), the CPU load starts to increase and continues to increase for the remainder of the attack period as shown in Figure 6.17c. This increase in the CPU load is slowed down after the 54^{th} observation interval



(a) ICMP flood attack.



(b) HTTP GET flood attack.

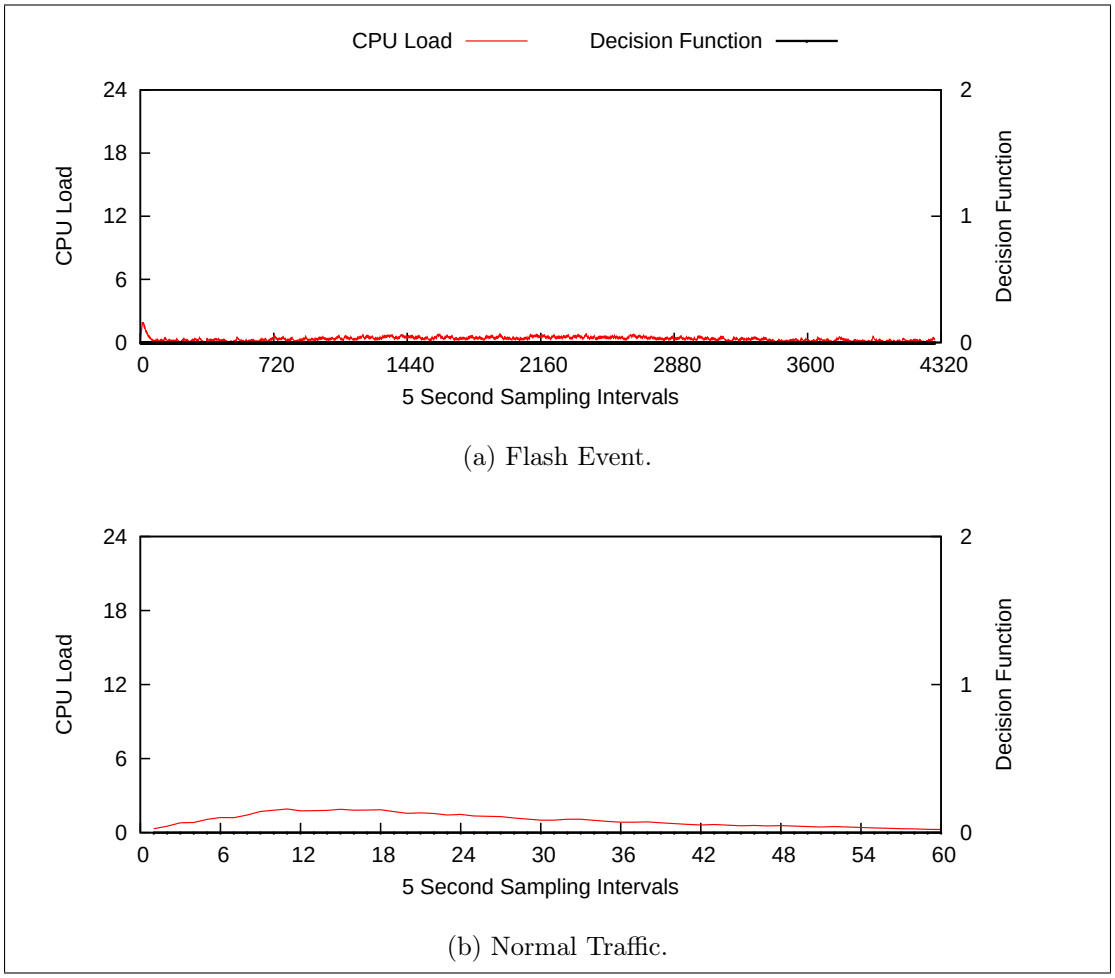


(c) SSL flood attack.

Figure 6.17: CPU Load for different DDoS attack datasets.

resulting in a 0 value from the decision function. The CPU load parameter stays at a relatively lower value for the non-attack datasets (Figure 6.18), apart from the initial increase which is likely to be due the background processes. How-

ever, these variations do not affect the decision making process related to this parameter.



(a) Flash Event.

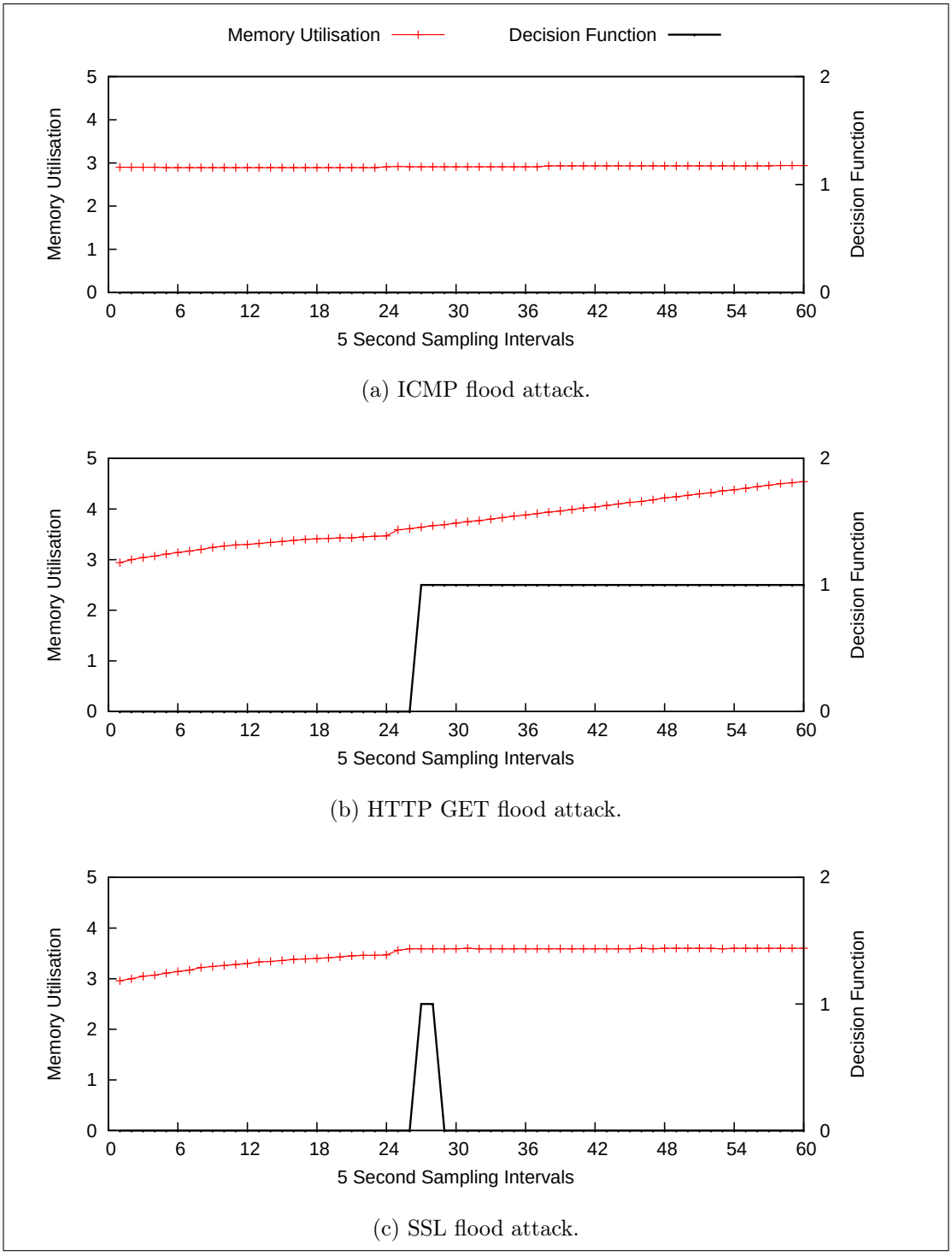
(b) Normal Traffic.

Figure 6.18: CPU Load for different non-attack datasets.

Memory Utilisation

The ICMP flooding attack, being a network level attack, has as expected, no affect on the target system's memory utilisation, which thus stays constant at its start-up value of 3% (Figure 6.19a). However, in the case of the HTTP GET flooding attack, as the web-server running on the target machine is expected to process the incoming requests and fetch the requested data from the disk, some memory is consumed for temporary holding of the data before it is sent back to the client. This behaviour is shown in Figure 6.19b, where following the start of the attack traffic, the memory consumption increases and continues to increase for the rest of the attack. It is to be noted that the real memory available on the

target machine is 32 GB, as previously mentioned in section 6.4.1. Therefore, in order to detect small variations in the total available memory, a lower value of $\lambda = 0.15$ is used for this parameter across all the datasets (Table 6.3).



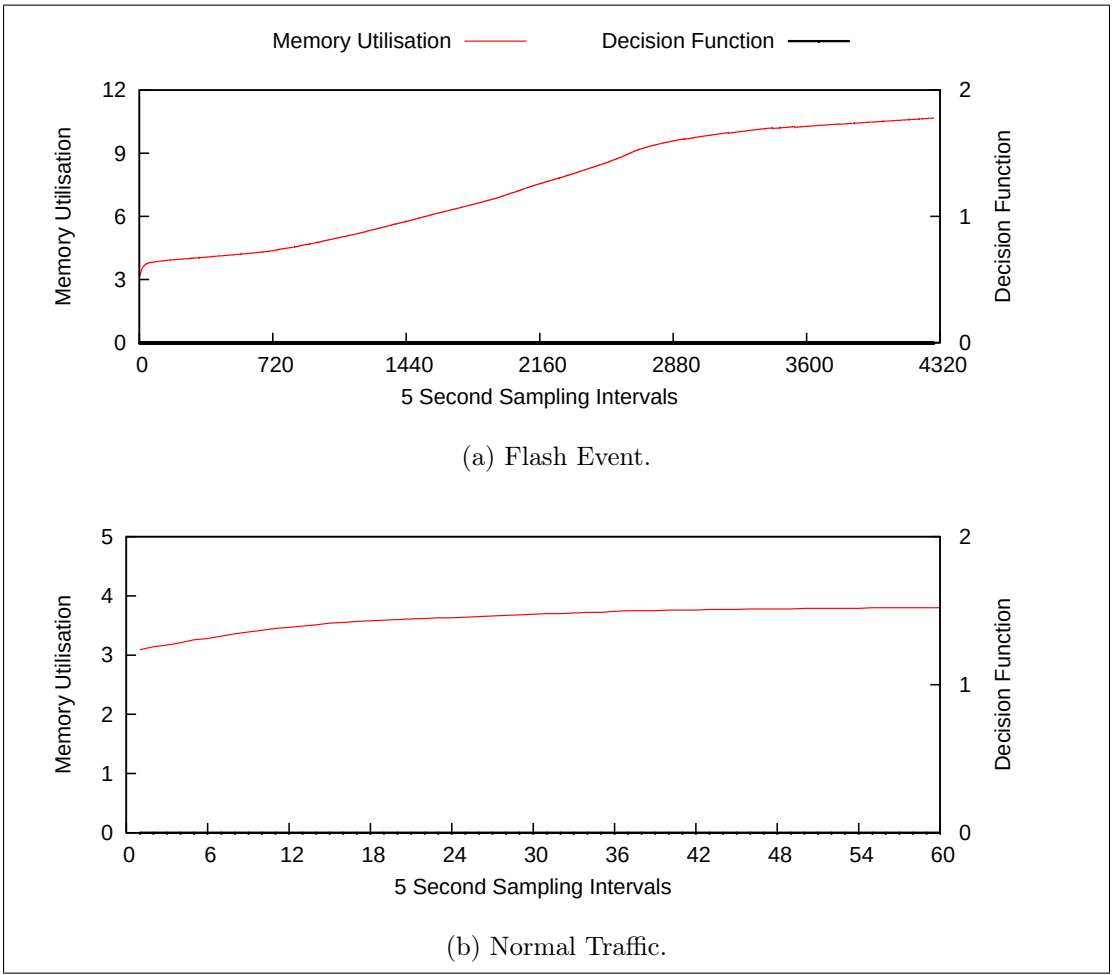
(a) ICMP flood attack.

(b) HTTP GET flood attack.

(c) SSL flood attack.

Figure 6.19: Memory utilisation for different DDoS attack datasets.

The SSL attack dataset also shows an increase in memory utilisation, corresponding to the attack traffic. However, after the increase around the 26^{th} sampling interval, there is no further substantial increase, resulting in a 0 from the decision function (Figure 6.19c). The change in memory being consumed per sampling interval is very gradual (Figure 6.20a) and negligible (Figure 6.20b) for the FE and normal traffic datasets respectively.



(a) Flash Event.

(b) Normal Traffic.

Figure 6.20: Memory utilisation for different non-attack datasets.

6.6 Feature Correlation

In the previous section, the change detection technique described in Chapter 3 (see Section 3.3 for details) was applied individually to every single feature, both from the incoming network traffic side and the server-load side. This resulted in identifying the significance of each parameter when used in isolation for indicat-

ing a possible DDoS attack. This section makes use of the decisions made by the change detection technique on individual parameters and correlates them to increase both the variety and the accuracy of DDoS attack detection, and their differentiation from FEs.

Feature Correlation Matrix

The outcomes of the change detection technique when applied to individual features across all five datasets is illustrated by a matrix referred to as the Feature Correlation Matrix (FCM). Table 6.4 shows this matrix with all the five datasets used in the experiment listed in the first column and the individual features (both from network traffic and the server-load side) listed in the first row. The values or outcomes of the decision function for each feature is placed in the cell corresponding to that feature and the dataset. The FCM shows a 1 if a change was detected for that feature for that dataset, and a 0 if no change was detected.

6.6.1 Detecting DDoS attacks and FEs

The research presented in this section attempts to address Research Question 5 identified earlier i.e. *Can a more sophisticated ensemble-based technique based on both network traffic characteristics and server-load data (memory and CPU utilisation) improve the accuracy of DDoS attack detection and provide a means to identify and differentiate DDoS attacks, FEs and normal traffic?* This question can be addressed by correlating the changes detected in individual parameters for different datasets, as shown in the FCM.

As evident from the FCM (Table 6.4), a simple boolean AND of all the network traffic parameters ($NumPack$, $NewSrc$, $SrcIP$, and $PackPerIP$) is sufficient to identify DDoS attacks overall:

$$DDoS = NumPack \wedge NewSrc \wedge SrcIP \wedge PackPerIP \tag{6.1}$$

It is to be noted that at this stage, the detected attack traffic can be either a network level or application level attack. If it has been determined that the incoming traffic is non-DDoS traffic (represented by $\overline{DDoS}$), the traffic can further be classified as an FE or normal traffic (NT) by combining it with the $SrcIP$ feature. Thus, for non-DDoS traffic, a distinction between an FE and a normal

Table 6.4: Feature Correlation Matrix.

Datasets	**Features**								
	Network Traffic					**Server Load**			
	NumPack	NewSrcIP	SrcIP	PackPerIP	SysCpu	UsrCpu	CpuLoad	RealMem
ICMP	1	1	1	1	1	0	0	0
HTTP	1	1	1	1	1	1	0	1
SSL	1	1	1	1	1	1	1	1
FE	0	0	1	0	0	0	0	0
NT	0	0	0	0	0	0	0	0

traffic scenario can be made using:

$$\begin{aligned} FE &= \overline{DDoS} \wedge SrcIP \\ NT &= \overline{DDoS} \wedge \overline{SrcIP} \end{aligned} \tag{6.2}$$

where $\overline{DDoS}$ and $\overline{SrcIP}$ stands for 'NOT $DDoS$' and 'NOT $SrcIP$' respectively.

The network traffic features are thus sufficient to determine if the incoming traffic represents a DDoS attack or an FE or normal traffic. The server-load features are not needed for this classification. However, they will be required to further classify the DDoS attacks identified at this stage into network and application level attacks, and also to separate application level attacks into HTTP and SSL attacks, as will now be discussed.

Detecting Network level flooding attacks

Once the incoming network traffic has been identified as attack traffic using Equation 6.1, it can further be classified into a network or application level attack, represented by $NetAttack$ (ICMP flood) and $AppAttack$ (HTTP and SSL flood) respectively. These attacks are detected by combining two server load parameters ($UsrCpu$, and $RealMem$) with the AND of network traffic parameters (obtained using Equation 6.1), via the following correlations:

$$\begin{aligned} NetAttack &= DDoS \wedge \overline{UsrCpu} \wedge \overline{RealMem} \\ AppAttack &= DDoS \wedge UsrCpu \wedge RealMem \end{aligned} \tag{6.3}$$

where $\overline{UsrCpu}$ and $\overline{RealMem}$ stands for 'NOT $UsrCpu$' and 'NOT $RealMem$' respectively.

At this stage, in order to differentiate between network and application layer DDoS attacks, for the datasets used in this analysis, only two server load-based parameters – $UsrCpu$ and $RealMem$ – have been used. However, it is speculated that $SysCpu$ feature may play an important role in successfully identifying other attack types such as TCP SYN flood attack, which can potentially have an effect on system CPU, in addition to user CPU and real memory utilisation. Thus, future work in this direction, as discussed in Chapter 7, would be to use a wider range of datasets and to evaluate the proposed correlation function.

Detecting Application level flooding attacks

After identifying the incoming network traffic as an application level attack, HTTP and SSL flooding attacks, represented by *HttpAttack* and *SslAttack* respectively, can finally be distinguished. This can be achieved by combining the *AppAttack* value, obtained from Equation 6.3, with the *CpuLoad* feature of the target server. Hence, in order to differentiate an SSL attack from a HTTP GET flood, the following correlations are performed:

$$\begin{aligned} HttpAttack &= AppAttack \wedge \overline{CpuLoad} \\ SslAttack &= AppAttack \wedge CpuLoad \end{aligned} \tag{6.4}$$

where $\overline{CpuLoad}$ stands for 'NOT *CpuLoad*'.

The correlation performed between different parameter outcomes, as presented in the Feature Correlation Matrix (FCM), shows that it is possible to identify whether the incoming network traffic represents attack or non-attack traffic, simply by using the network traffic parameters (Equation 6.1). These network side parameters are also sufficient to decide whether the incoming traffic represents an FE or normal traffic (Equation 6.2). However, in order to detect (or differentiate between) different types of DDoS attacks, the server load parameters are required, as shown by Equations 6.3 and 6.4.

6.7 Conclusion

This chapter extends the work presented earlier in Chapter 3 by including additional parameters, both from the incoming network traffic side and the server load side, to identify the onset of DDoS attacks and to differentiate them from FEs. It extends the DDoS detection criterion, developed in Chapter 3, beyond the simple property of Not Seen Previously (NSP) source IP addresses. The main contribution of this chapter is the development and evaluation of an ensemble-based DDoS attack detection technique, which combines two rather orthogonal attack detection strategies, viz., network traffic analysis and server load analysis.

The network traffic features used within the ensemble are: incoming traffic volume (NumPack), new source IP addresses (NewSrcIP), source IP addresses (SrcIP), and the incoming traffic distribution (PackPerIP). The parameters used from the server load side are: system level CPU utilisation (SysCpu), user level

CPU utilisation (UsrCpu), CPU Load (CpuLoad), and real memory utilisation (RealMem).

The selected features are first individually detected using a customised change detection technique based on Exponentially Weighted Moving Average (EWMA). The outcome of detecting changes in individual parameters is fed into a Feature Correlation Matrix (FCM), and subsequently correlated to detect various types of DDoS attacks, both at the network and the application layer, and to differentiate them from FE and normal traffic scenarios. While the network traffic parameters proved to be sufficient for flagging the incoming network traffic as a DDoS attack, and for differentiating the non-attack traffic as an FE or normal traffic, the server load parameters play an important role in further classifying the DDoS attacks as network layer or application layer attacks.

To the best of our knowledge, although network MIB variables measured on the target host have been previously used to detect DDoS attacks, there have been no techniques proposed that use server performance parameters for this purpose, or that combine network parameters and server resource monitoring to differentiate DDoS attacks from FEs.

It is acknowledged that a limited number of simulated datasets have been used for analysing the ensemble-based attack detection technique proposed in this chapter. These datasets, although limited in their number and variety, are based on real-world datasets, except for the HTTP GET and SSL flooding attack datasets. In these cases, however, the attacks are still based on the known attack methodologies. The future work, as discussed in detail in Chapter 7, aims to validate the proposed technique against a more diverse variety of datasets.

Chapter 7

Conclusion and Future Directions

Distributed Denial-of-Service (DDoS) attacks utilise the power of thousands, and sometimes tens or hundreds of thousands of compromised, geographically distributed machines, to attack web-services, resulting in their degradation and consequent financial loss. Hence the early and reliable detection of such attacks is an important area of research.

The problem of detecting DDoS attacks, already complex in itself, is further complicated by Flash Events (FEs), which occur whenever a server experiences an unexpected surge of requests from legitimate clients, because these share some similar characteristics with DDoS attacks. Unfortunately, the required research in developing DDoS attack detection and FE differentiation techniques is hampered by the extreme scarcity of public domain data representing real traffic, whether attack or benign. This thesis addresses these challenges and makes a number of related contributions, as summarised in the following section.

The rest of the chapter is organised as follows. Section 7.1 presents a summary of the thesis contributions. Section 7.2 discusses the limitations of the work presented in the thesis and provides directions for future research. Finally, Section 7 presents the concluding remarks of this thesis.

7.1 Summary of Contributions

In this thesis, the challenges and limitations of efficient DDoS attack detection and also their differentiation from FEs, along with the challenges and limitations

of synthetic traffic generation of both types of anomaly have been addressed. The contributions relating to the research questions identified in Chapter 1 (see Section 1.2) are summarised below.

The first contribution has been the design of a DDoS attack detection technique based on single network traffic parameters viz., the change in rate of new or not seen previously (NSP) source IP addresses. The proposed technique exploits an important characteristic of DDoS attacks i.e. a marked increase in the number of NSP source IPs accompanying the sudden surge of the incoming traffic. The proof-of-concept implementation of the technique has demonstrated that this simple network traffic-based feature can be used to effectively detect HRF DDoS attacks. This directly addresses the scalability issues which apply at higher network speeds, and demonstrates through the use of bit-vectors that the approach may be used as part of a real-time DDoS attack detection strategy, in contrast to more computationally intensive machine learning based approaches.

The second contribution has been the design and implementation of an FE model, which laid the foundation for the detection and synthetic generation of an FE. Differences between DDoS and FE traffic were highlighted by analysing two publicly available datasets and identifying three parametric differences between these two network traffic anomalies: change in the rate of incoming traffic, change in the rate of new source IP addresses, and distribution of requests among source IPs. The proposed model further classified FEs into three broad categories: predictable, unpredictable and secondary, and described each of them via three key components: the volume of incoming traffic, the related source IP addresses and the resource being accessed. This characterisation has been used to develop a server-side mathematical model for FEs with a small set of configurable parameters to synthetically generate different types of FEs.

This model is to the best of our knowledge the most detailed model of Flash Events yet proposed, and is based on verifiable, publicly available datasets, unlike other models.

The third contribution has been the development of a traffic generation and testbed framework, developed as part of this thesis in a cooperative effort with other researchers to synthetically generate different types of realistic DDoS attacks and FEs in a controlled and isolated environment. This makes use of modest hardware and an in-house software traffic generator tool, *Botloader*, which exploits IP-aliasing, a well-known technique available on most computing plat-

forms, to generate a wide variety of attack and benign traffic originating from a wide range of valid IP addresses.

One advantage of using a small dedicated facility is the reduction in experimental complications that may arise from use of a large shared testbed. Its main advantages, however, are the increased realism of fully responsive TCP/UDP endpoints, and the possibility of simulating very large attacks of up to 100,000 clients using only modest hardware.

The fourth contribution has been the design and implementation of an ensemble-based DDoS attack detection technique, which combines two orthogonal attack detection strategies i.e. network traffic analysis and server-load analysis based attack detection. The proposed technique uses eight features, four from the incoming network traffic and four from the server load side. The network traffic features used in the ensemble were: incoming traffic volume, new source IP addresses, number of source IP addresses, and the incoming traffic distribution. The four load-based parameters used from the server side are: system level CPU utilisation, user level CPU utilisation, CPU load and real memory utilisation. Using the Exponentially Weighted Moving Average (EWMA) technique, changes in individual features were correlated to detect a variety of DDoS attacks, both at the network and application layer, to differentiate them from FE and normal traffic scenarios.

To the best of our knowledge, no DDoS detection technique has used such an accurate ensemble-based approach, which is able to detect and differentiate FEs from DDoS attacks and determine their type, potentially in real-time.

Finally, an additional contribution described in Chapter 1, has been the conceptual design of a DDoS attack detection architecture. The key component of this architecture is a DDoS Detection and Mitigation Model (D2M2) – a single device with attack detection and mitigation capabilities, designed to protect a target server with an application-aware firewall from DDoS attacks.

7.2 Limitations and Future Directions

During the course of the research, a number of limitations have been identified, which suggest directions for future research. These future directions are described in the following sections.

7.2.1 Improving Feature Correlation

The individual features identified and used in the ensemble are currently assigned equal weights and are thus correlated using simple boolean operations. This simple correlation, although effective in identifying the types of attack presented in the thesis, may not hold for a wider range of attacks where the variations in these features may be more subtle. An obvious step forward would be to assign different weights to the individual features and then perform a weighted correlation. This would potentially extend the attack detection capabilities of the proposed technique to semantic attacks, where the variations in the server-load based parameters are more significant than those in the network traffic parameters.

7.2.2 Evaluation against a Wider Range of Datasets

The attack detection techniques developed in Chapters 3 and 6 have been tested and evaluated against a limited number of real-world and simulated datasets, either based on exiting real-world datasets or known attack methodologies. The limited availability of public domain datasets representing different types of DDoS attacks and FEs has been one of the primary motivations for the work presented in Chapter 5. Thus, future work in this regard would be to apply the proposed techniques to a wider range of datasets, especially application level DDoS attacks, as well as secondary and unpredictable FEs. This will not only provide an opportunity to perform experiments using different sampling intervals, but would also help in answering questions related to generalising these techniques for detecting DDoS attacks and differentiating them from FEs.

7.2.3 Real-time DDoS Attack Detection and Mitigation

The current implementation of the attack detection techniques proposed in the thesis is *post hoc*. It is believed that these techniques can be implemented to work in real or near-real time. This would involve an investigation into the performance of the proposed techniques with respect to detection delays.

A real-time implementation would lead naturally to the investigation of mitigation strategies and their design and implementation, including, for example, further work on the D2M2 work described in Chapter 3.

7.2.4 DDoS Detection in the IPv6 Address Space

The DDoS attack detection technique developed in Chapter 3 has been designed for the IPv4 address space. In order to accommodate all of the 2^{32} IPv4 addresses, it uses bit vectors as the data structure, thus requiring 0.5 GB of space. Using the same data structure in IPv6 address space would be impossible due to the significantly increased storage space requirements in order to accommodate 2^{128} IP addresses. Thus, future work in this direction would focus on implementing the attack detection algorithm using a more efficient data structure, such as bloom filters [22]. This would not only help in extending the proposed technique to a wider address space (IPv6), but would also allow its performance to be compared with the bit vector implementation.

7.2.5 Improving the Traffic Generation and Testbed Framework

The traffic generation and testbed framework presented in Chapter 5 can be further improved in the following areas:

- as previously discussed in Chapter 6 (see Section 6.4.2 for details), using Tcpdump on the target machine to capture the incoming traffic was confounding the actual results, especially those related to the memory utilisation. This required two iterations for each simulation, one with Tcpdump running on the target machine and one without. This overhead can be minimised by adding a *traffic capture machine* to the existing testbed architecture (see Figure 5.2), and mirroring all the network traffic between the attack switch and the target host to that machine. The addition of a traffic capture machine, however, would also involve additional hardware, including a 10 Gb network interface module to be plugged into the attack switch, since the testbed set-up uses a 10 GB link connecting the attack switch and the target host.

- the current implementation of the FE module (`fe_bot`) is unable to closely simulate the 'new source IPs' characteristic of the original FE dataset. This resulted in the use of the FE original dataset for this particular feature when presenting the detection of individual features for the ensemble (see Section 6.5 for details). As previously mentioned in Chapter 6, the

deviation in this feature of the simulated traffic from that of the original dataset is mainly due to an artefact induced by the software traffic generator, which tends to use all the available bots (aliased IPs) at the beginning, and reuses them during the course of the simulation. Changing the traffic generation to get around this would result in a better and more controlled simulation overall.

7.3 Concluding Remarks

Correlating two orthogonal DDoS attack detection techniques based on network traffic analysis and server-load data (memory and CPU utilisation) has resulted in the development of an ensemble-based technique, capable of accurately and reliably detecting the onset of different types of DDoS attacks, as well as differentiating them from FEs, with which they share some similar characteristics. The thesis also addresses an intrinsic challenge in research associated with DDoS attacks, namely, the extreme scarcity of datasets representing real traffic, whether attack or benign, by investigating and developing techniques, using modest hardware and exploiting a well-known technique (IP aliasing), to synthesise realistic emulations of DDoS attack and FE traffic.

Given the serious challenge posed by DDoS attacks to our ICT-based society, it is important to find effective measures to detect such attacks at an early stage. The work described in this thesis has attempted to contribute to that effort. It remains to extend the evaluation of this work to a larger and more representative set of traffic data, and to enhance and deploy these ideas in the real time domain.

Appendix A

Change Detection using CUSUM

As discussed in Section 3.3, parts of Chapter 3 previously published [8, 75], used a sliding-window-based non-parametric CUSUM proposed by Ahmed et al. [6, 7] for change detection. This section presents the experimental results of applying the modified CUSUM to the observed parameter i.e. Not Seen Previously (NSP) source IPs used in the change detection technique developed in Chapter 3.

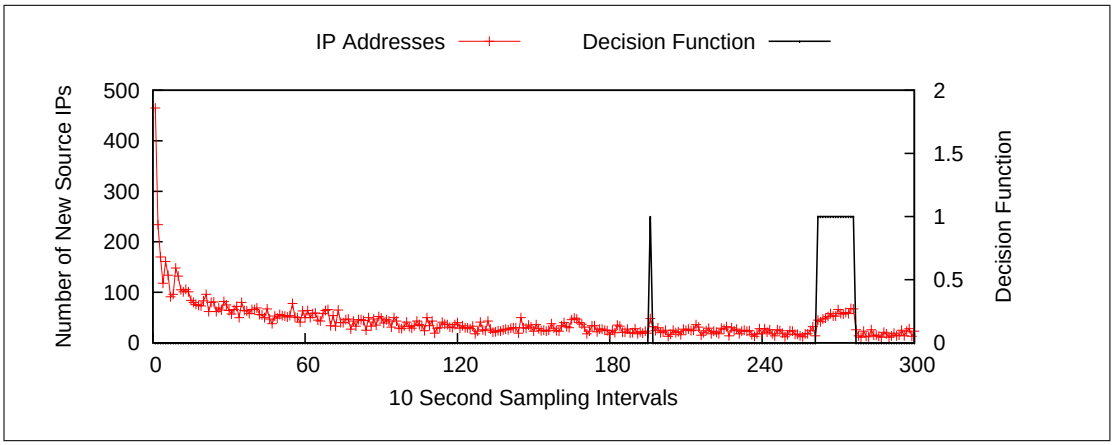
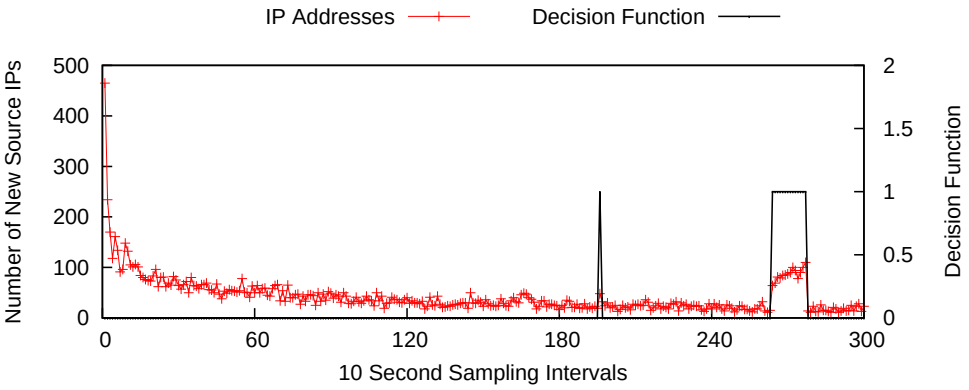


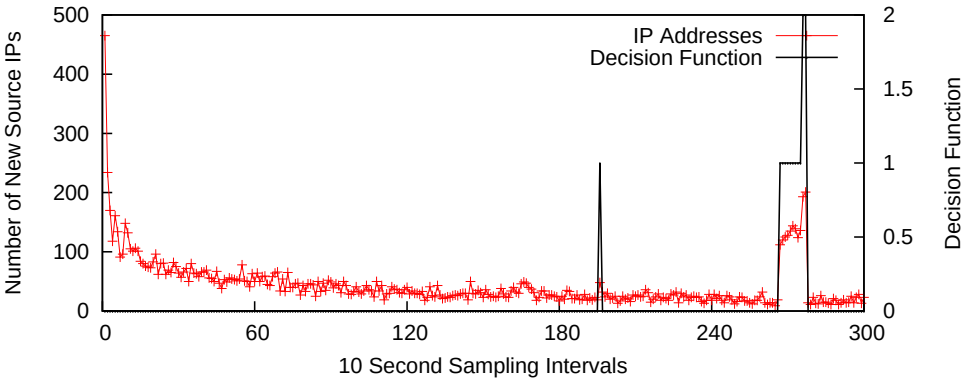
Figure A.1: UDP flooding attack with 35 source IP addresses (CUSUM).

In order to evaluate the performance of the NSP algorithm using CUSUM as the change detection technique, the UDP flooding attack datasets with a varying number of source IPs, including 35, 40, 45, 100, and 150 new source IPs per second, were merged with the normal background traffic. Details of the attack and background traffic have been previously discussed in Chapter 3 (Section 3.4.1). All the experiments were conducted using a 10 second sampling

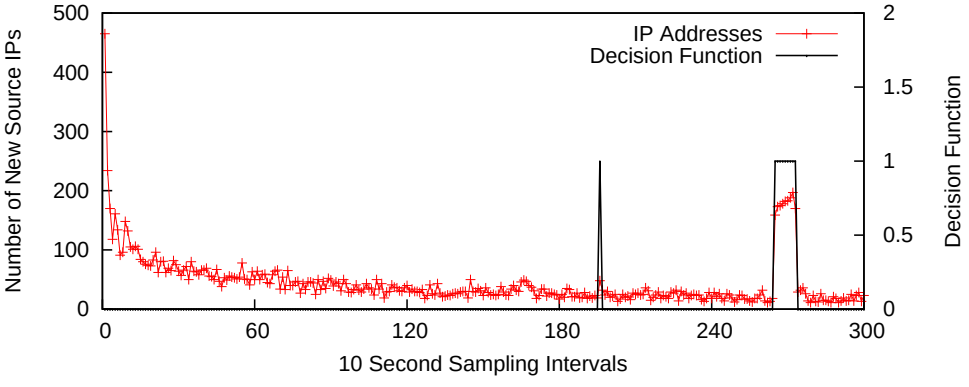
interval and a 100 second sliding window.



(a) UDP flooding attack with 50 source IP addresses.



(b) UDP flooding attack with 100 source IP addresses.



(c) UDP flooding attack with 150 source IP addresses.

Figure A.2: UDP flooding attack with varying source IP addresses (CUSUM).

Figure A.1 shows the experimental results when CUSUM was applied to the UDP flooding attack using 35 new source IPs per second. The x-axis in the figure

represents 10 second sampling intervals, the left y-axis the total number of new source IPs in the sampling interval, and the right y-axis is output of the CUSUM decision function with 1 indicating an attack and 0 indicating no attack. The time series for the number of new source IPs in the 10 second sampling interval was calculated using the *ipac* function of the DDoS attack detection technique developed in Chapter 3.

The UDP flooding attack was started at 260^{th} observation interval and ended at 280^{th} observation interval and the change in the number of NSP source IPs was detected by the CUSUM technique. As earlier mentioned in Chapter 3, the new source IP counter starts with a relatively high value that gradually decreases over time. This is mainly because the bit vector $\boldsymbol{W}$ is empty at the start of the training period, and is then populated with the initial source IP addresses. The change detection technique is not applied for the training period i.e. up to the measurement interval 230, and this allows the sliding window based CUSUM to learn the normal network behaviour.

Similar experiments were conducted using attacks with higher new source IP values such as 50, 100, and 150 to evaluate the performance of attack detection technique with various attack types. The results are presented in Figure A.2. The attack detection algorithm using CUSUM produced comparable results to EWMA as shown in Figure 3.5. An additional advantage of the CUSUM algorithm over the current implementation of EWMA was its ability to detect nested changes i.e. a change within a change, as indicated in Figure A.2b. This advantage, however, came with an associated cost of its complex implementation as compared to a rather simplistic EWMA, and a higher false detection rate as shown in all of the above figures at around 200^{th} measurement interval. This false detection can be minimised by using a longer sliding window, which can be different for different datasets. This characteristic feature, however, can be better controlled via the k value, as used in EWMA based change detection, and the same value can applied across different datasets. Moreover, the prime focus of the techniques developed in this thesis is to detect the *onset* of a wide variety of DDoS attacks and *differentiate* them from FEs, and therefore, a simpler change detection technique (EWMA) gave comparable results to relatively complex technique (CUSUM) was chosen.

Appendix B

Traffic Generators

B.1 Curl-loader

Curl-loader is a C based open-source tool used to simulate the behaviour of large number of HTTP/HTTPS and FTP/FTPS clients, each with its own unique source IP address. It uses a real C-based client protocol stack, namely HTTP and FTP stacks of libcurl and TLS/SSL of openssl to simulate user behaviour. Curl-loader has the capability to generate large number of so-called 'virtual clients' (VCs) each having its own valid unique source IP and MAC address. The multi-threading option of curl-loader allows a better use of available CPUs on platforms with multi-core processors.

In order to generate synthetic network traffic, comprising both malicious and normal data on the experimental testbed infrastructure, the scalability of this approach was tested and found to be governed by (a) processing capacity of the individual machines (i.e. the number and speed of CPU's and the amount of memory); and (b) the number of distinct platforms available in the experimental testbed. When curl-loader was being researched, the experimental testbed consisted of eight host machines and a high-performance Dell PowerEdge R710 as the target server. Each of these host machines were standard PCs with 2.60 GHz Intel Pentium-4 processor, 2 GB of memory and an integrated 1 Gigabit (Gb) Network Interface Card (NIC), and running Ubuntu 10.04 Desktop. The target server was equipped with two Six-core Intel Xeon 2.27 GHz processors (hyper-threaded) and 32 GB of memory and running Ubuntu 10.04 Server as the

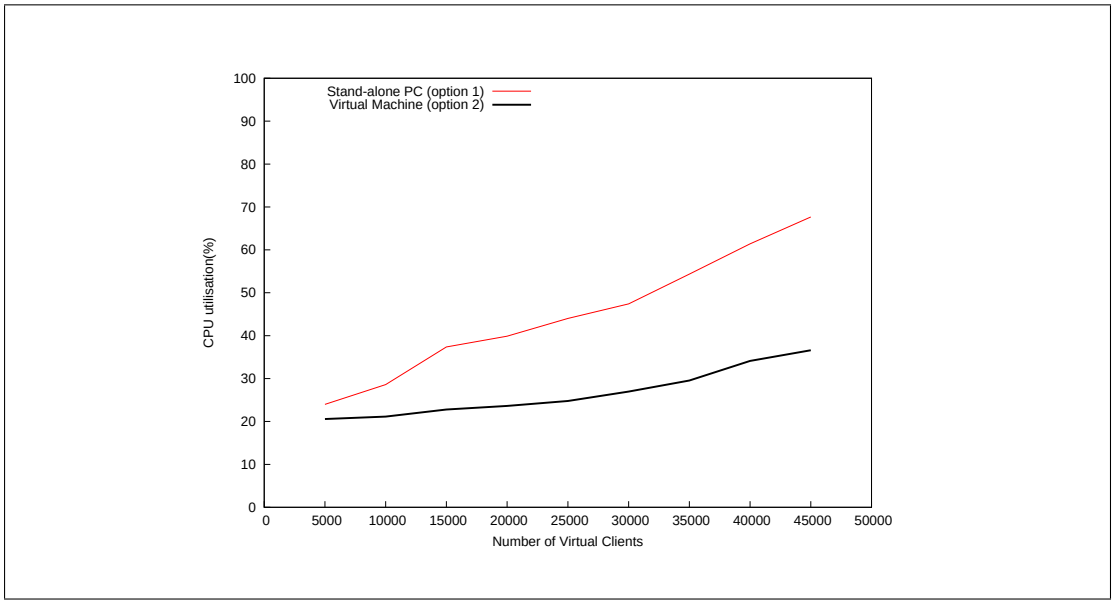


Figure B.1: CPU utilisation with varying number of clients

operating system. The virtualisation software (VMware ESXi) allowed the Dell platform to be used as a cluster of virtual machines.

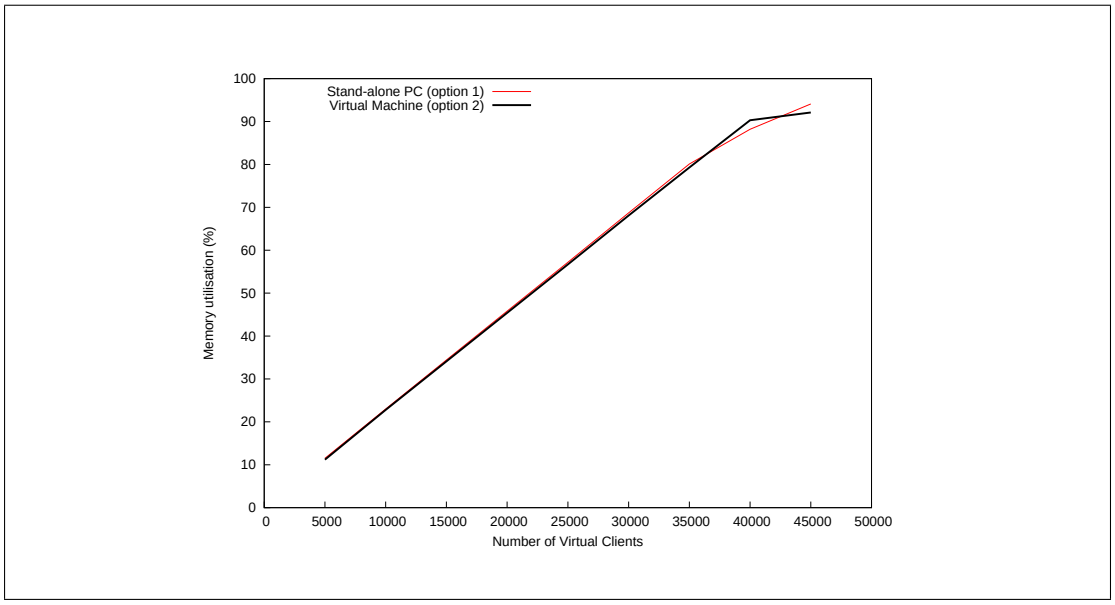


Figure B.2: Memory utilisation with varying number of clients

Notwithstanding the overheads introduced by virtualisation, the idea was to analyse and compare the performance of curl-loader running on a stand-alone PC (option 1) with the one running on virtual machine (option 2) using a high-performance Dell platform. The stand-alone PC used in option 1 was one of the

host machines (2.60 GHz Intel Pentium-4 processor and 2 GB of memory) from the experimental testbed. For option 2 a virtual machine was created on the Dell platform with one dedicated CPU (Intel Xeon 2.27 GHz) and 2 GB of memory.

Two options were compared on the basis of CPU utilisation, memory utilisation and the number of requests per second versus the number of virtual clients as shown in Figures B.1, B.2 and B.3 respectively. The maximum number of VCs that could be supported by curl-loader running on either a stand-alone PC or a single virtual machine was approximately 45,000. The difference between CPU utilisation (Figure B.1) in both the cases is attributable to a better family of processor in option 2. The asymptotic behaviour of the memory utilisation graphs (Figure B.2) in both cases indicates the availability of memory as the limiting factor. A dip in the number of requests generated per second by curl-loader running on a stand-alone PC (Figure B.3) is consistent with the increase in CPU utilisation.

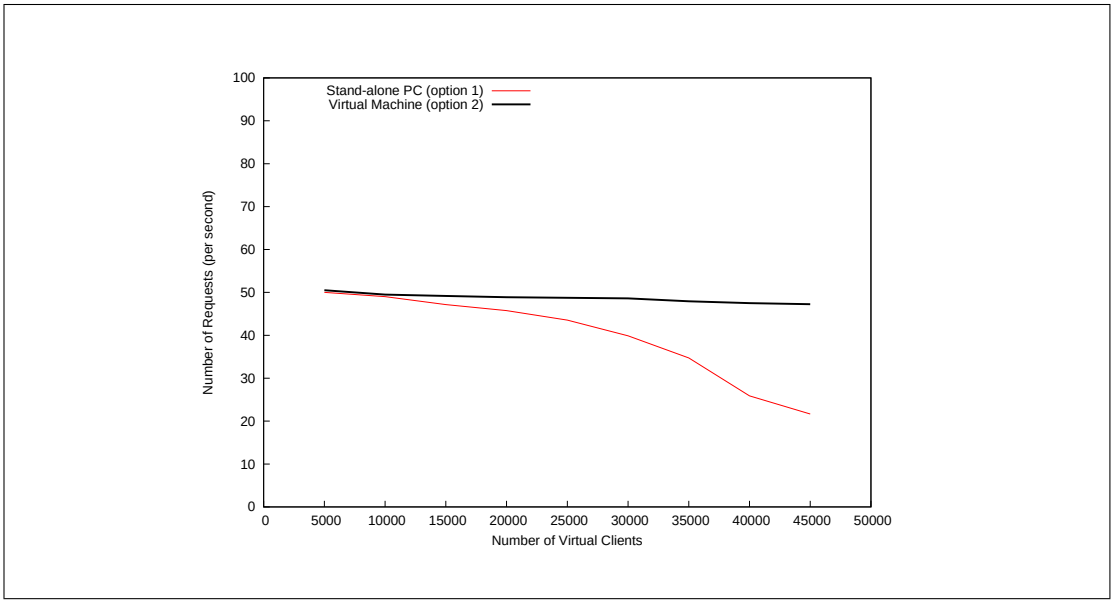


Figure B.3: Number of requests with varying number of clients

Thus, curl-loader was found to place a heavy load on host systems' resources (CPU and memory) and proved far from an ideal tool for generating synthetic traffic at the desired rate, originating from large number of source IPs. Each client generated by curl-loader consumed approximately 30-40 KB of user-space memory and some non-paged memory in kernel space, thus imposing an effective limit on the number of VCs that could be created. Curl-loader was modified to incorporate randomness in source IP addresses and to create ramp-up, ramp-

down and pulsing attacks. However, these modifications led to instabilities in the tool's performance, and combined with the need for further modifications to incorporate different types of attacks, this led to eventual abandonment of this approach. However, the research involved in using curl-loader in this way led to the idea of using *IP-aliasing* along with wget[1], a command line Linux utility for fetching HTTP files, to generate application layer network traffic.

B.2 IP-aliasing and Wget

A physical network interface on any networking device including a computer is identified by its assigned name and IP address. However, this is just a software construct, which can be expanded by creating 'sub-interfaces' for each available physical interface. Each sub-interface, also referred to as *aliased IP address*, can be assigned a unique IP address and can act independently to establish TCP connections with the target host or with other sub-interfaces. The technique of creating multiple sub-interfaces of a physical network interface (NIC) and assigning sub-interface a distinct IP address is called *IP-aliasing*. Using this technique, traffic originating from a single host can be divided into multiple *network-flows* [2] with each flow having at least a unique value for source IP address, and thus can be made to appear (to the target) to be coming from many separate machines. IP-aliasing is available on most computing platforms. On Linux and BSD, for example, thousands of unique IP addresses can be assigned to a single Ethernet card. IP-aliasing on Linux platform can be achieved with the use of *ifconfig* utility.

GNU Wget is a non-interactive command line utility for retrieving files using HTTP, HTTPS and FTP protocol. As wget is an non-interactive tool, it can be easily integrated with custom scripts, cron jobs and other command line utilities available in Linux without the support of X-windows. Therefore, Wget when used with IP-aliasing can potentially create the appearance of an array of machines ('bots' for DDoS attacks and 'legitimate clients' for FEs), each with a unique source IP address and sharing the hardware address of the physical interface. This array of machines could then be used to synthetically generate realistic

[1] GNU Wget – http://www.gnu.org/software/wget/

[2] A TCP/IP network flow is defined as a sequence of packets from a source machine to a destination machine with unique values for source IP, source port, destination IP, destination port, and protocol

DDoS attacks and FEs.

The approach of using wget with IP-aliasing provided a method for crafting application layer DDoS attacks like HTTP flooding attack. However, the time it took to create aliases increased with the number of requested aliases. An increase in the number of aliases also resulted in an increase in the host machines' CPU and memory consumption. At one stage, 65,000 aliases were successfully created, however, this resulted in near exhaustion of available memory. Approximately 4,000 aliases per physical interface on a standard PC appeared to be an acceptable upper limit, taking into account the resource consumption. The variations in the memory and CPU utilisation with increasing number of aliases were comparable to the results obtained using curl-loader as shown in Figures B.2 and B.1. In addition to the resource consumption constraints, the approach of using IP-aliasing with wget was also limited by the variety of attacks that it was capable of generating. This approach could not be adapted to simulate different types of flooding attacks such as ICMP flooding or UDP flooding. Thus, it was limited to generating application layer DDoS attacks. Hence, the approach of combining IP-aliasing and wget proved to be too resource-heavy and lacked the desired scalability and flexibility essential for generating different types of attacks and FEs. However, it did lay the foundations for a customised software traffic generator Botloader, which was cooperatively developed as a part of this research.

Appendix C

DDoS Attack Dataset Generation

Three DDoS attack datasets have been used for evaluating the proposed attack detection technique. One of datasets represents a network layer flooding attack, while the other two represent different types of application level attacks. Each of these attack datasets are simulated using the network traffic generation framework described in Chapter 5 for a period of 5 minutes: first by two minutes of normal activity, followed by three minutes of flooding. Each of these datasets are now described.

The generation of network layer flooding attack has been previously discussed in Chapter 5 (see Section 5.4.1 for details). This section describes the synthetic generation of application level DDoS attacks.

C.1 HTTP GET flooding attack dataset

The HTTP GET flooding attack dataset is simulated using the `http_flood.conf` file given below, consisting of two threads (clusters of bots) started simultaneously. The first thread (`fe_bot`) represents the normal traffic and runs for two minutes, as indicated by `time=120`. The attack thread, represented by `http_bot` is started simultaneously with the normal traffic thread is inactive for the first 120 seconds, as indicated by two leading 0's in the `rate` parameter of the configuration file.

```
fe_bot 50 debug=0
res_file=/home/ddos/botloader/botloader/fe_compacted_modified.txt
```

```
interval_file=/home/ddos/botloader/botloader/per-1sec-stats time=120
interval=1 port=80 dest_ip=10.1.1.76

http_bot 100 debug=0 time=300 port=80
res_file=/home/ddos/botloader/botloader/fe_compacted_modified.txt
rates="0 0 -1 -1 -1" dest_ip=10.1.1.76
```

The `res_file` indicates the resource file, containing a list of URLs being accessed by each bot during the simulation. The `interval_file` consists of per second samples of the number of HTTP GET requests to fetch, the number of source IPs to use, the number of unique resources to access from the Web server, and the desired resource entropy. `dest_ip` specifies the IP address of the target host, running a Web server, listening on port `80`. During the normal traffic part of the simulation, 50 bots or aliased IPs are used on each of the 10 attack machines. The `rates` parameter for the `http_bot` governs the rate at which the HTTP traffic is being sent. `rates="0 0 -1 -1 -1"` implies that the total simulation duration given by `time=300` is divided into 5 equal intervals of 60 seconds. For the first two intervals, no traffic is being sent, and for the remaining three intervals (or the last 180 seconds of the simulation), all the bots are sending traffic at the maximum possible rate, indicated by `-1`.

C.2 SSL breakoff attack dataset

The SSL breakoff attack dataset is generated in a similar fashion to the HTTP GET flooding attack dataset. The configuration file used to generate this dataset is given below.

```
fe_bot 50 debug=0
res_file=/home/ddos/botloader/botloader/fe_compacted_modified.txt
interval_file=/home/ddos/botloader/botloader/per-1sec-stats time=120
interval=1 port=80 dest_ip=10.1.1.76

ssl_bot 100 debug=0 keysize=2048 rates="0 0 -1 -1 -1"
time=300 dest_ip=10.1.1.76
```

As in the HTTP GET flooding attack, this dataset is also generated using two concurrent threads, `fe_bot` and `ssl_bot`, for normal and attack traffic respectively. The normal traffic thread runs for two minutes, followed by the attack

thread for the next three minutes. The `keysize` specifies the size of the key used in the SSL handshake. The generated dataset uses a 2048 bit key.

Bibliography

[1] Waikato Applied Network Dynamic Research Group. `http://wand.cs.waikato.ac.nz/wits/auck/8/`. [Online; accessed 26-Sep-2012].

[2] ModSecurity: Open Source Web Application Firewall. `http://www.modsecurity.org/`, 2009. [Online; accessed 26-Sep-2012].

[3] Prolexic Quarterly Global DDoS Attack Report – Q4 2012. Technical report, Prolexic, 2012.

[4] The DDoS that knocked Spamhaus offline. `http://blog.cloudflare.com/the-ddos-that-knocked-spamhaus-offline-and-ho`, 2013. [Online; accessed 2-Apr-2013].

[5] E. Addley and J. Halliday. Operation Payback Cripples MasterCard Site in Revenge for WikiLeaks Ban. `http://www.guardian.co.uk/media/2010/dec/08/operation-payback-mastercard-website-wikileaks`, Dec 2010. [Online; accessed 23-Sep-2012].

[6] E. Ahmed, A. Clark, and G. Mohay. A Novel Sliding Window Based Change Detection Algorithm for Asymmetric Traffic. In *NPC 2008 IFIP International Conference on Network and Parallel Computing, 2008*, pages 168–175. IEEE, 2008.

[7] E. Ahmed, A. Clark, and G. Mohay. Effective Change Detection in Large Repositories of Unsolicited Traffic. In *Proceedings of the Fourth International Conference on Internet Monitoring and Protection*, May 2009.

[8] E. Ahmed, G. Mohay, A. Tickle, and S. Bhatia. Use of IP Addresses for High Rate Flooding Attack Detection. *Security and Privacy–Silver Linings in the Cloud*, pages 124–135, 2010.

[9] L. V. Ahn, M. Blum, and J. Langford. Telling Humans and Computers Apart Automatically. *Communications of the ACM*, 47(2):56–60, 2004. ISSN 0001-0782.

[10] David P. Anderson. Surviving Slashdot'ing with a Small Server. `http://www.geology.smu.edu/~dpa-www/attention_span/`, 2005. [Online; accessed 9-June-2012].

[11] I. Ari, B. Hong, E.L. Miller, S.A. Brandt, and D.D.E. Long. Managing Flash Crowds on the Internet. In *Proceedings of 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS)*, pages 246–249. IEEE, 2003.

[12] M. Arlitt and T. Jin. 1998 World Cup Web Site Access Logs. `http://www.acm.org/sigcomm/ITA/`, 1998. [Online; accessed 9-June-2012].

[13] M.F. Arlitt and C.L. Williamson. Web Server Workload Characterization: The Search for Invariants. In *Proceedings of ACM SIGMETRICS Performance Evaluation Review*, volume 24, pages 126–137. ACM, 1996.

[14] C.M. Bao. Intrusion Detection Based on One-class SVM and SNMP MIB Data. In *2009 Fifth International Conference on Information Assurance and Security*, pages 346–349. IEEE, 2009.

[15] P. Barford and D. Plonka. Characteristics of Network Traffic Flow Anomalies. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pages 69–73. ACM, 2001.

[16] M. Basseville, I.V. Nikiforov, et al. *Detection of Abrupt Changes: Theory and Application*, volume 104. Prentice Hall Englewood Cliffs, NJ, 1993.

[17] T. Benzel, B. Braden, T. Faber, J. Mirkovic, S. Schwab, K. Sollins, and J. Wroclawski. Current Developments in DETER Cybersecurity Testbed Technology. In *Conference For Homeland Security, 2009. CATCH'09. Cybersecurity Applications & Technology*, pages 57–70. IEEE, 2009.

[18] V. Bernat. Iran Blamed for Cyber Onslaught on US Banks. `http://www.france24.com/en/20130110-iran-blamed-cyber-onslaught-us-banks`, 2011. [Online; accessed 10-Jan-2013].

[19] V. Bernat. SSL Computational DoS Mitigation. http://vincent.bernat.im/en/blog/2011-ssl-dos-mitigation.html, 2011. [Online; accessed 22-Nov-2012].

[20] D. J. Bernstein. SYN Cookies. http://cr.yp.to/syncookies.html, 1996. [Online; accessed 10-Jan-2013].

[21] W.J. Blackert, D.M. Gregg, A.K. Castner, E.M. Kyle, R.L. Hom, and R.M. Jokerst. Analyzing Interaction Between Distributed Denial of Service Attacks and Mitigation Technologies. In *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, volume 1, pages 26–36. IEEE, 2003.

[22] B.H. Bloom. Space/time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 13:422–426, July 1970. ISSN 0001-0782.

[23] C. Bo, B.X. Fang, and X.C. Yun. A New Approach for Early Detection of Internet Worms Based on Connection Degree. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 4, pages 2424–2430. IEEE, 2005.

[24] P. Bodik, A. Fox, M.J. Franklin, M.I. Jordan, and D.A. Patterson. Characterizing, Modeling, and Generating Workload Spikes for Stateful services. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 241–252. ACM, 2010.

[25] A. Botta, A. Dainotti, and A. Pescapé. A Tool for the Generation of Realistic Network Workload for Emerging Networking Scenarios. *Computer Networks*, 2012.

[26] T. Bradley. Operation Payback: WikiLeaks Avenged by Hacktivists. http://www.pcworld.com/businesscenter/article/212701/operation_payback_wikileaks_avenged_by_hacktivists.html, 7 Dec 2010. [Online; accessed 23-Sep-2012].

[27] E. V. Buskirk. Denial-of-Service Attack Knocks Twitter Offline. http://www.wired.com/business/2009/08/twitter-apparently-down/. [Online; accessed 9-Aug-2012].

[28] S. Byers, A.D. Rubin, and D. Kormann. Defending Against an Internet-based Attack on the Physical World. *ACM Transactions on Internet Technology (TOIT)*, 4(3):239–254, 2004.

[29] J.B.D. Cabrera, L. Lewis, X. Qin, W. Lee, and R.K. Mehra. Proactive Intrusion Detection and Distributed Denial of Service Attacks: A Case Study in Security Management. *Journal of Network and Systems Management*, 10(2):225–254, 2002.

[30] J. Calvet, J.M. Fernandez, P-M. Bureau, and J-Y. Marion. Large-Scale Malware Experiments Why, How, and So What? In *Proceedings of Virus Bulletin Conference*, pages 241–247, Sept 2010.

[31] G. Carl, G. Kesidis, R.R. Brooks, and S. Rai. Denial-of-service Attack-detection Techniques. *Internet Computing, IEEE*, 10(1):82–89, 2006.

[32] J. Case, M. Fedor, M. Schoffstall, and C Davin. *A Simple Network Management Protocol (SNMP)*. Network Information Center, SRI International, 1989.

[33] CERT/CC. CERT Advisory CA-1996-01 UDP Port Denial-of-Service Attack. `http://www.cert.org/advisories/CA-1996-01.html`, 1997. [Online; accessed 26-Sep-2012].

[34] CERT/CC. CERT Advisory CA-1997-28 IP Denial-of-Service Attacks. `http://www.cert.org/advisories/CA-1998-01.html`, 1998. [Online; accessed 26-Sep-2012].

[35] J. Chan, C. Leckie, and T. Peng. Hitlist Worm Detection Using Source IP Address History. In *Proceedings of Australian Telecommunication Networks and Applications Conference*, 2006.

[36] J. Cheng, J. Yin, Y. Liu, Z. Cai, and M. Li. DDoS Attack Detection Algorithm Using IP Address Features. *Frontiers in Algorithmics*, pages 207–215, 2009.

[37] W.R. Cheswick and S.M. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley Professional, 1994.

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

[39] B.L. Dear. Perhaps the First Denial-of-Service Attack? http://www.platohistory.org/blog/2010/02/perhaps-the-first-denial-of-service-attack.html, 2010. [Online; accessed 26-Sep-2012].

[40] D. Dittrich. The 'Stacheldraht' Distributed Denial of Service Attack Tool, 1999.

[41] J. Dudley-Nicholson. Click Frenzy Website Crashes as Online Sale Set to Launch. http://www.news.com.au/technology/biztech/click-frenzy-website-crashes-as-online-sale-set-to-launch/story-fn5lic6c-1226520660974, 2012. [Online; accessed 22-Nov-2012].

[42] J. Dudley-Nicholson. Retailers on Failed Click Frenzy Website Begin Discussing Refunds. http://www.news.com.au/technology/retailers-on-failed-click-frency-website-begin-discussing-refunds/story-e6frfrnr-1226521002703, 2012. [Online; accessed 22-Nov-2012].

[43] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred. Statistical Approaches to DDoS Attack Detection and Response. In *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, volume 1, pages 303–314. IEEE, 2003. ISBN 0769518974.

[44] M.J. Freedman. Experiences with CoralCDN: A Five-year Operational View. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, pages 7–7. USENIX Association, 2010.

[45] L. Garber. Denial-of-service Attacks Rip the Internet. *Computer*, 33(4): 12–17, 2000.

[46] D. Gavrilis, I. Chatzis, and E. Dermatas. Flash Crowd Detection Using Decoy Hyperlinks. In *Networking, Sensing and Control, 2007 IEEE International Conference on*, pages 466–470. IEEE, 2007.

[47] T.M. Gil and M. Poletto. *MULTOPS: a data-structure for bandwidth attack detection*. Defense Technical Information Center, 2001.

[48] Corey Grice. How a Basic Attack Crippled Yahoo. Technical report, CNET News, 2000.

[49] Y. He, W. Chen, and B. Xiao. Detecting SYN Flooding Attacks Near Innocent Side. *Mobile Ad-hoc and Sensor Networks*, pages 443–452, 2005.

[50] S. Hettich and S. D. Bay. The UCI KDD Archive University of California, Department of Information and Computer Science. `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`, 1999. [Online; accessed 26-Sep-2012].

[51] P. Hick, E. Aben, K.C. Claffy, and J. Polterock. The CAIDA "DDoS Attack 2007" Dataset. `http://www.caida.org/data/passive/ddos-20070804_dataset.xml`. [Online; accessed 9-June-2012].

[52] J. Hu and G. Sandoval. Web Acts as Hub for Info on Attacks. *Retrieved April*, 26:2003, 2001.

[53] R. Iakobashvili and M. Moser. Welcome to Curl-loader. `http://curl-loader.sourceforge.net/`, 2009. [Online; accessed 22-Nov-2012].

[54] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web sites. In *Proceedings of the 11th International Conference on World Wide Web*, pages 293–304. ACM, 2002.

[55] S. Kandula, D. Katabi, M. Jacob, and A. Berger. Botz-4-sale: Surviving Organized DDoS Attacks that Mimic Flash Crowds. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 287–300. USENIX Association, 2005.

[56] M. Kenney. Ping of Death. `http://insecure.org/sploits/ping-o-death.html`, Jan 1997. [Online; accessed 26-Sep-2012].

[57] F. Khan. Botnet Economy. `http://dos-attacks.com/2010/10/26/botnet-economy/`. [Online; accessed 23-Sep-2012].

[58] M.B.C. Khoo, S.Y. Teh, L.F. Ang, and X.W. Ng. A Study on the False Alarm Rates of X, EWMA and CUSUM Control Charts when Parameters are Estimated.

[59] B. Krishnamurthy and J. Wang. On Network-aware Clustering of Web Clients. In *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 97–110. ACM, 2000. ISBN 1581132239.

[60] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based Change Detection: Methods, Evaluation, and Applications. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 234–247. ACM, 2003.

[61] A. Kuzmanovic and E.W. Knightly. Low-rate TCP-targeted Denial of Service Attacks: The Shrew vs. The Mice and Elephants. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, page 86. ACM, 2003. ISBN 1581137354.

[62] C. Labovitz. The Internet Goes to War. `http://asert.arbornetworks.com/2010/12/the-internet-goes-to-war/`, 14 Dec 2010. [Online; accessed 23-Sep-2012].

[63] Q. Le, M. Zhanikeev, and Y. Tanaka. Methods of Distinguishing Flash Crowds from Spoofed DoS Attacks. In *Proceedings of 3rd EuroNGI Conference on Next Generation Internet Networks*, pages 167–173. IEEE, 2007. ISBN 1424408571.

[64] A. Leach. Mr Sulu causes DDoS panic after posting link on Facebook. `http://www.theregister.co.uk/2012/06/08/takei_ddos_facebook_fans/`, 2012. [Online; accessed 22-Nov-2012].

[65] C. Lévy-Leduc. Detection of Network Anomalies Using Rank Tests.

[66] J. Li, Y. Liu, and L. Gu. DDoS Attack Detection Based on Neural Network. In *Aware Computing (ISAC), 2010 2nd International Symposium on*, pages 196–199. IEEE, 2010.

[67] G. Linden. Make Data Useful. *Presentation, Amazon, November*, 2006.

[68] W.Z. Lu and S.Z. Yu. An HTTP Flooding Detection Method Based on Browser Behavior. In *Computational Intelligence and Security, 2006 International Conference on*, volume 2, pages 1151–1154. IEEE, 2006.

[69] S. McCanne, S. Floyd, K. Fall, K. Varadhan, et al. Network Simulator ns–2, 1997.

[70] J. McHugh. Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):262–294, 2000.

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

[72] J. Mirkovic, S. Fahmy, P. Reiher, and R.K. Thomas. How to Test DoS Defenses. In *Conference For Homeland Security, 2009. CATCH'09. Cybersecurity Applications & Technology*, pages 103–117. IEEE, 2009.

[73] D. Mituzas. In Numbers. `http://dom.as/2011/10/07/steve-jobs/`, 2011. [Online; accessed 26-Sep-2012].

[74] OPNET Modeler. OPNET Technologies Inc. *Bethesda, MD. URL: http://www. opnet.com*, 2009.

[75] G. Mohay, E. Ahmed, S. Bhatia, A. Nadarajan, B. Ravindran, A. B. Tickle, and R. Vijayasarathy. Detection and Mitigation of High-Rate Flooding Attacks. In S.V. Raghavan and E. Dawson, editors, *An Investigation into the Detection and Mitigation of Denial of Service (DoS) Attacks*, pages 131–181. Springer, 2011.

[76] D.C. Montgomery. *Introduction to Statistical Quality Control*. John Wiley & Sons, 2007.

[77] D. Moore, C. Shannon, D.J. Brown, G.M. Voelker, and S. Savage. Inferring Internet Denial-of-Service Activity. *ACM Transactions on Computer Systems (TOCS)*, 24(2):115–139, 2006.

[78] G. Mori and J. Malik. Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA. 2003. ISSN 1063-6919.

[79] A. Moses. Click Frenzy a mega tech fail: the Online bonanza that became nothing of the sort. `http:`

//www.theage.com.au/technology/technology-news/click-frenzy-a-mega-tech-fail-the-online-bonanza-that-became-nothing-of-t html, 2012. [Online; accessed 24-Nov-2012].

[80] G. Münz and G. Carle. Application of Forecasting Techniques and Control Charts for Traffic Anomaly Detection. In *Proc. 19th ITC Specialist Seminar on Network Usage and Traffic, Berlin, Germany*, 2008.

[81] J. Nazario. Political DDoS: Estonia and Beyond (Invited Talk). In *USENIX Security*, volume 8, 2008.

[82] Arbor Networks. The Growing Threat of Application-Layer DDoS Attacks. Technical report, Arbor Networks, 2011.

[83] H. Niksic. GNU wget. *Available from the master GNU archive site prep.ai.mit.edu, and its mirrors*, 1998.

[84] L. Niven. *The Flight of the Horse*. Ballantine Books, 1973.

[85] E.S. Page. Continuous Inspection Schemes. *Biometrika*, pages 100–115, 1954.

[86] H. Park, P. Li, D. Gao, H. Lee, and R. Deng. Distinguishing between FE and DDoS Using Randomness Check. *Information Security*, pages 131–145, 2008.

[87] J.S. Park and M.S. Kim. Design and Implementation of an SNMP-based Traffic Flooding Attack Detection System. *Challenges for Next Generation Network Operations and Service Management*, pages 380–389, 2008.

[88] O. Paul. Improving Web Servers Focused DoS Attacks Detection. In *Proceedings of the IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM 2006), Tuebingen, Germany*, 2006.

[89] V. Paxson. Bro: A System for Detecting Network Intruders in Real-time. *Computer networks*, 31(23):2435–2463, 1999.

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

[91] T. Peng, C. Leckie, and K. Ramamohanarao. Protection from Distributed Denial of Service Attacks Using History-based IP Filtering. In *IEEE International Conference on Communications, 2003. ICC'03*, pages 482–486, 2003.

[92] T. Peng, C. Leckie, and K. Ramamohanarao. Survey of Network-based Defense Mechanisms Countering the DoS and DDoS Problems. *ACM Computing Surveys*, 39(1):3, 2007.

[93] T. Peng, C. Leckie, and R. Kotagiri. System and Process For Detecting Anomalous Network Traffic, may 2008. WO Patent 2,008,052,291.

[94] L. Peterson, A. Bavier, M.E. Fiuczynski, and S. Muir. Experiences Building Planetlab. In *Proceedings of the 7^{th} Symposium on Operating Systems Design and Implementation*, pages 351–366. USENIX Association, 2006.

[95] K. Poulsen. FBI Busts Alleged DDoS Mafia. `http://www.securityfocus.com/news/9411/`, 2008. [Online; accessed 22-Nov-2012].

[96] X. Qin, W. Lee, L. Lewis, and J.B.D. Cabrera. Integrating Intrusion Detection and Network Management. In *Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP*, pages 329–344. IEEE, 2002.

[97] C. Morales R. Dobbins. Worldwide Infrastructure Security Report. Technical report, Arbor Networks, 2011.

[98] M.A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. My botnet is Bigger than Yours (Maybe, Better than Yours): why size estimates remain challenging. In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, page 5. USENIX Association, 2007.

[99] S.W. Roberts. Control Chart Tests Based on Geometric Moving Averages. *Technometrics*, 42(1):97–101, 2000.

[100] M. Roesch et al. Snort-lightweight Intrusion Detection for Networks. In *Proceedings of the 13th USENIX conference on System administration*, pages 229–238. Seattle, Washington, 1999.

[101] M. Sachdeva, G. Singh, K. Kumar, and K. Singh. Measuring Impact of DDoS Attacks on Web Services. 2010.

[102] S. Sanfilippo. Hping–Active Network Security Tool. `http://www.hping.org/`, 2008. [Online; accessed 22-Nov-2012].

[103] C.E. Shannon. A Mathematical Theory of Communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[104] R. Singel. Operation Payback Cripples MasterCard Site in Revenge for WikiLeaks Ban. `http://www.wired.com/threatlevel/2010/12/web20-attack-anonymous/`, Dec 2010. [Online; accessed 24-Sep-2012].

[105] V.A. Siris and F. Papagalou. Application of Anomaly Detection Algorithms for Detecting SYN Flooding Attacks. *Computer Communications*, 29(9): 1433–1442, 2006.

[106] Spirent SmartBits. 600 Series Traffic Generators.

[107] J. Sommers, H. Kim, and P. 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.

[108] R. Stapleton-Gray and W. Woodcock. National Internet Defense—Small States on the Skirmish Line. *Communications of the ACM*, 54(3):50–55, 2011.

[109] Packet Storm. Tribe Flood Network 2000 (TFN2K) DDoS Tool, available on-line: http://packetstormsecurity. org/distributed. *TFN2k_ Analysis-1.3. txt*, 2000.

[110] W.W. Streilein, D.J. Fried, and R.K. Cunningham. Detecting Flood-based Denial-of-service Attacks with SNMP/RMON. In *Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection, Fairfax, Virginia, USA*, 2003.

[111] A. Sudduth. The What, Why, and How of the 1988 Internet Worm. `http://www.snowplow.org/tom/worm/worm.html`, 1988. [Online; accessed 26-Sep-2012].

[112] S. Suriadi, A.J. Clark, and D. Schmidt. Validating Denial of Service Vulnerabilities in Web Services. In *IEEE Computer Society Proceedings of the Fourth International Conference on Network and System Security*. IEEE Computer Society, 2010.

[113] H.H. Takada and A. Anzaloni. Protecting Servers Against DDoS Attacks with Improved Source IP Address Monitoring Scheme. In *Next Generation Internet Design and Engineering, 2006. NGI'06. 2006 2nd Conference on*, pages 154–159. IEEE, 2006.

[114] A.G. Tartakovsky, B.L. Rozovskii, R.B. Blažek, and H. Kim. Detection of Intrusions in Information Systems by Sequential Change-point Methods. *Statistical Methodology*, 3(3):252–293, 2006.

[115] A.G. Tartakovsky, B.L. Rozovskii, R.B. Blažek, and H. Kim. A Novel Approach to Detection of Intrusions in Computer Networks via Adaptive Sequential and Batch-sequential Change-point Detection Methods. *Signal Processing, IEEE Transactions on*, 54(9):3372–3382, 2006.

[116] M. Tavallaee, E. Bagheri, W. Lu, and A.A. Ghorbani. A Detailed Analysis of the KDD Cup 99 Data Set. In *Proceedings of the 2009 IEEE Symposium Computational Intelligence for Security and Defense Applications (CISDA 09). IEEE Computer Society*, 2009.

[117] THC. The Hackers Choice. `http://www.thc.org/thc-ssl-dos/`, 2010. [Online; accessed 22-Nov-2012].

[118] M. Trojnara. Sslsqueeze. `ftp://ftp.stunnel.org/sslsqueeze/`, 2011. [Online; accessed 22-Nov-2012].

[119] Uniontown. Student Accused of Trying to Crash School's Computer System. `http://www.wkyc.com/news/news_article.aspx?ref=RSS&storyid=45721`, January 2006. [Online; accessed 23-Sep-2012].

[120] US Committee on National Security Systems. National Information Assurance (IA) Glossary. Instruction 4009, CNSS, 2006.

[121] H. Wang, D. Zhang, and K.G. Shin. Detecting SYN Flooding Attacks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1530–1539. IEEE, 2002.

[122] H. Wang, D. Zhang, and K.G. Shin. Change-point Monitoring for the Detection of DoS Attacks. *Dependable and Secure Computing, IEEE Transactions on*, 1(4):193–208, 2004.

[123] P. Wendell and M.J. Freedman. Going viral: Flash Crowds in an Open CDN. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 549–558. ACM, 2011.

[124] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. *ACM SIGOPS Operating Systems Review*, 36(SI):255–270, 2002.

[125] Y. Xie and S.Z. Yu. Detecting Shrew HTTP Flood Attacks for Flash Crowds. *Computational Science–ICCS 2007*, pages 640–647, 2007.

[126] X. Xu, Y. Sun, and Z. Huang. Defending DDoS Attacks Using Hidden Markov Models and Cooperative Reinforcement Learning. In *Intelligence and Security Informatics*, pages 196–207. Springer, 2007.

[127] N. Ye, C. Borror, and Y. Zhang. EWMA Techniques for Computer Intrusion Detection Through Anomalous Changes in Event Intensity. *Quality and Reliability Engineering International*, 18(6):443–451, 2002.

[128] N. Ye, S. Vilbert, and Q. Chen. Computer Intrusion Detection Through EWMA for Autocorrelated and Uncorrelated data. *Reliability, IEEE Transactions on*, 52(1):75–82, 2003.

[129] J. Yu, H. Lee, M.S. Kim, and D. Park. Traffic Flooding Attack Detection with SNMP MIB using SVM. *Computer Communications*, 31(17):4212–4219, 2008.

[130] S. Yu, T. Thapngam, J. Liu, S. Wei, and W. Zhou. Discriminating DDoS Flows from Flash Crowds Using Information Distance. In *NSS'09 Third International Conference on Network and System Security, 2009*, pages 351–356. IEEE, 2009.

[131] B. Zhang, A. Iosup, J. Pouwelse, and D. Epema. Identifying, Analysing, and Modeling Flashcrowds in Bittorrent. In *Proceedings of Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on*, pages 240–249. IEEE, 2011.

[132] C.C. Zou, W. Gong, and D. Towsley. Code Red Worm Propagation Modeling and Analysis. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 138–147. ACM, 2002.

[133] M. J. Zuckerman. How the Government Failed to Stop the World's Worst Internet Attack. *USA Today*, 2000.