LEABHARLANN CHOLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH | TRINITY COLLEGE LIBRARY DUBLIN
Ollscoil Átha Cliath | The University of Dublin

Terms and Conditions of Use of Digitised Theses from Trinity College Library Dublin

Copyright statement

All material supplied by Trinity College Library is protected by copyright (under the Copyright and Related Rights Act, 2000 as amended) and other relevant Intellectual Property Rights. By accessing and using a Digitised Thesis from Trinity College Library you acknowledge that all Intellectual Property Rights in any Works supplied are the sole and exclusive property of the copyright and/or other IPR holder. Specific copyright holders may not be explicitly identified. Use of materials from other sources within a thesis should not be construed as a claim over them.

A non-exclusive, non-transferable licence is hereby granted to those using or reproducing, in whole or in part, the material for valid purposes, providing the copyright owners are acknowledged using the normal conventions. Where specific permission to use material is required, this is identified and such permission must be sought from the copyright holder or agency cited.

Liability statement

By using a Digitised Thesis, I accept that Trinity College Dublin bears no legal responsibility for the accuracy, legality or comprehensiveness of materials contained within the thesis, and that Trinity College Dublin accepts no liability for indirect, consequential, or incidental, damages or losses arising from use of the thesis for whatever reason. Information located in a thesis may be subject to specific use constraints, details of which may not be explicitly described. It is the responsibility of potential and actual users to be aware of such constraints and to abide by them. By making use of material from a digitised thesis, you accept these copyright and disclaimer provisions. Where it is brought to the attention of Trinity College Library that there may be a breach of copyright or other restraint, it is the policy to withdraw or take down access to a thesis while the issue is being resolved.

Access Agreement

By using a Digitised Thesis from Trinity College Library you are bound by the following Terms & Conditions. Please read them carefully.

I have read and I understand the following statement: All material supplied via a Digitised Thesis from Trinity College Library is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of a thesis is not permitted, except that material may be duplicated by you for your research use or for educational purposes in electronic or print form providing the copyright owners are acknowledged using the normal conventions. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone. This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

Responsive Aggregate Defence for Denial of Service Attacks

Arkaitz Bitorika

A thesis submitted to the University of Dublin, Trinity College

for the degree of Doctor of Philosophy

March 2011

Declaration

I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work.

19 JUL 2011
LIBRARY DUBLIN
THOSIS
9104.

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Abstract

The Internet is the main provider of information, communication and media services for an increasing percentage of the world's population. Its architecture is based on packet switching principles and employs a layered, end-to-end design in its protocols. This architecture has demonstrated scalability and evolutionary resilience since its introduction thirty years ago.

Congestion responsive transport protocols based on an end-to-end design, such as TCP, coupled with best effort packet forwarding approaches have been the most widely used networking concepts on the Internet. However, these same design approaches are highly vulnerable to abuse from malicious traffic sources that do not comply with TCP congestion control principles. These non compliant traffic types can be used to mount Denial of Service (DoS) attacks to prevent legitimate users from accessing Internet services.

The work described in this dissertation provides a range of original contributions in the design of an effective router queueing mechanism for bandwidth-flooding DoS defence. The Responsive Aggregate Defence (RAD) scheme introduces a scalable, easily implementable Active Queue Management (AQM) scheme that accurately detects and penalises malicious congestion unresponsive DoS traffic at the router queue. RAD is effective against a wide variety of DoS attack vectors, including a range of attack time-scales and packet header spoofing techniques used to conceal their activity.

RAD enables the incremental deployment of improved queueing schemes on Internet routers. It provides enhanced DoS resilience to networks whilst ensuring transparent, AQM calibre TCP performance in typical usage. RAD is a valuable contribution towards improving the robustness of the Internet infrastructure against malicious usage.

Acknowledgements

I would like to start my acknowledgements by expressing my gratitude to my supervisor, Meriel Huggard, for her tremendous support in helping me on the long journey of my PhD. She has been a constant source of encouragement, direction and focus even when other aspects of my life made progress difficult.

My heartfelt thanks go to Ciarán McGoldrick, who has been a de-facto second supervisor in my work. Ciarán has in all these years always been happy to provide honest feedback, and kept me honest too.

I'd also like to especially thank Mathieu Robin, who has been my colleague and sounding board since the beginning of my research work.

To all the many undergraduate and graduate students, lecturers and support staff of Trinity College that I have interacted with over the years, thanks for making College the great place that it is.

Aloña has experienced the up and downs of my PhD work more than anyone else, while still providing me with warm support and well needed grounding in the real world, the one that happens outside of networks of computers. Eskerrik asko!

Patxi, Anttoni and Oiartza deserve the final and largest consideration, for always being there for me, understanding my choices in life and in many ways making me what I am. Milesker!

Arkaitz Bitorika

University of Dublin, Trinity College

March 2011

Contents

Abstract iv

Acknowledgements vi

List of Figures xi

Chapter 1 Introduction 1

1.1 Denial of Service 2

1.2 Active Queue Management 2

1.3 Research questions 3

1.4 Aims and objectives 4

1.5 Thesis outline 6

1.6 Publication record 8

1.7 Conclusion 8

Chapter 2 Literature Survey 9

2.1 Denial of Service Attacks 11

2.1.1 Software exploit DoS attacks 13

2.1.2 Bandwidth flooding DoS attacks 15

2.1.3 DoS detection 19

2.1.4 DoS defence 25

2.2 Active Queue Management 30

2.2.1 Random Early Detection (RED) 31

2.2.2 Fairness-oriented AQM 33

2.2.2.1 CHOKe 34

2.2.2.2 RED-PD 35

2.2.2.3 LRU-RED 36
2.2.2.4 Stochastic Fair Blue (SFB) 37
2.2.2.5 Self Adjustable CHOKe (SAC) 38
2.2.2.6 CARE 38
2.2.2.7 Approximate Fair Dropping (AFD) 39
2.2.2.8 Core-Stateless Fair Queueing (CSFQ) 40
2.2.2.9 Aggregate Congestion Control (ACC) 41
2.3 TCP 41
2.3.1 TCP Congestion Control 43
2.3.2 TCP modelling 44
2.4 Conclusion 50

Chapter 3 Characterising TCP Flow Aggregates 52
3.1 Overview and requirements 53
3.2 TCP flow behaviour 55
3.2.1 TCP data transfer period 57
3.2.2 Packets sent 59
3.2.2.1 Packets sent in total 59
3.2.2.2 Packets sent in Slow Start 61
3.2.2.3 Packets sent in Congestion Avoidance 61
3.2.3 Latency 62
3.2.3.1 Slow Start latency 63
3.2.3.2 First packet loss latency 64
3.2.3.3 Congestion Avoidance latency 65
3.3 TCP aggregate behaviour 68
3.4 Evaluation 70
3.4.1 Evaluation scenarios 71
3.4.2 Latency 72
3.4.3 Packets sent 72
3.4.4 Aggregate TCP rate 74
3.5 Conclusion 76

Chapter 4 Responsive Aggregate Defence (RAD) 78
4.1 Overview 80

4.2 Architecture 82
4.2.1 Aggregate Burstiness Filter 83
4.2.2 Aggregate Responsiveness Filter 84
4.2.3 Global Queueing 84
4.3 Measurement metrics 85
4.3.1 Metrics overview 85
4.3.2 Metrics implementation 86
4.3.3 Flow metrics implementation 88
4.3.4 Bloom filters 89
4.3.5 Implementing the flow measurement metric with Bloom filters 90
4.4 RAD DoS penalisation 92
4.4.1 Aggregate burstiness filter 92
4.4.2 Aggregate responsiveness filter 93
4.4.3 Summary of the parameters of RAD 96
4.4.4 False positives and negatives 96
4.4.4.1 False positives 97
4.4.4.2 False negatives 99
4.5 Conclusion 100

Chapter 5 RAD Performance Evaluation **101**

5.1 Methodology 102
5.1.1 Metrics 102
5.1.2 Topology 103
5.1.3 Traffic 105
5.1.4 Congestion responsive traffic 106
5.1.5 Malicious traffic 107
5.1.5.1 Persistent flooding traffic 108
5.1.5.2 Pulsing attack traffic 109
5.1.6 Base scenario 109
5.2 Evaluated aspects 112
5.2.1 Malicious traffic penalisation 113
5.2.1.1 Persistent flooding DoS attacks 114
5.2.1.2 Pulsing DoS attacks 115

5.2.1.3 Packet header spoofing 117
5.2.2 TCP traffic transparency 119
5.2.3 Parameter sensitivity 120
5.2.3.1 RTT 122
5.2.3.2 Average Connection size 123
5.2.3.3 The measurement period length (Delta) 125
5.2.3.4 Aggregate Responsiveness ratio thresholds . . . 126
5.2.3.5 Aggregate burstiness threshold 129
5.3 Conclusion 131

Chapter 6 Conclusion **133**
6.1 Summary of contributions 136
6.2 Future work 138
6.3 Conclusion 139

Definitions **141**

Bibliography **143**

List of Figures

2.1 RED drop probability function (gentle version) 32

3.1 Evaluation network topology 71
3.2 Evaluation of transfer latency prediction. 73
3.3 Average number of data packets sent by TCP connections. 74
3.4 Evaluation of aggregate TCP rate prediction with single RTT. . 75
3.5 Evaluation of aggregate TCP rate prediction with different RTTs. 75

4.1 RAD Architecture 82

5.1 Dumbbell scenario topology. 104
5.2 Instantaneous TCP throughput before, during and after a persistent DoS attack with RED and RAD. 113
5.3 Average TCP throughput comparison without DoS traffic and with persistent DoS traffic of a range of offered loads with RED and RAD. 114
5.4 Instantaneous TCP throughput before, during and after a pulsing DoS attack with RED and RAD. 115
5.5 Average TCP throughput comparison without DoS traffic and with pulsing DoS traffic of a range of offered loads with RED and RAD. 116
5.6 Average TCP throughput comparison for RED and RAD without DoS traffic and with persistent DoS traffic under a range of packet header spoofing strategies. 117

5.7 Average TCP throughput comparison for RED and RAD without DoS traffic and with pulsing DoS traffic under a range of packet header spoofing strategies. 118
5.8 Average throughput achieved by TCP traffic comparison between DropTail, RED and RAD for a set of traffic mixes composed of a range of active closed-loop TCP sources from 50 to 600. . . . 119
5.9 Average TCP throughput comparison for RED and RAD without DoS traffic and with persistent DoS traffic under a range of values for the RTT parameter between 0.03 and 0.42 seconds. 121
5.10 Average TCP throughput comparison for RED and RAD without DoS traffic and with pulsing DoS traffic under a range of values for the RTT parameter between 0.03 and 0.42 seconds. 122
5.11 Average TCP throughput comparison for RED and RAD without DoS traffic and with persistent DoS traffic under a range of values for the average connection size parameter between 25000 and 400000 bytes. 124
5.12 Average TCP throughput comparison for RED and RAD without DoS traffic and with pulsing DoS traffic under a range of values for the average connection size parameter between 25000 and 400000 bytes. 124
5.13 Average TCP throughput comparison for RED and RAD without DoS traffic and with persistent DoS traffic under a range of values for the Δ parameter between 0.01 and 0.1 seconds. 125
5.14 Average TCP throughput comparison for RED and RAD without DoS traffic and with pulsing DoS traffic under a range of values for the Δ parameter between 0.01 and 0.1 seconds. 126
5.15 Average TCP throughput comparison for RED and RAD without DoS traffic and with persistent DoS traffic under a range of values for the R_{min} and F_{min} ratio threshold parameters between 1 and 3. 127

5.16 Average TCP throughput comparison for RED and RAD without DoS traffic and with pulsing DoS traffic under a range of values for the R_{min} and F_{min} ratio threshold parameters between 1 and 3. 127
5.17 Average TCP throughput comparison for RED and RAD without DoS traffic and with persistent DoS traffic under a range of values for the R_{max} and F_{max} ratio threshold parameters between 3 and 5. 128
5.18 Average TCP throughput comparison for RED and RAD without DoS traffic and with pulsing DoS traffic under a range of values for the R_{max} and F_{max} ratio threshold parameters between 3 and 5. 128
5.19 Average TCP throughput comparison for RED and RAD without DoS traffic and with persistent DoS traffic under a range of values for the in_{th} ratio threshold parameter between 0.1 and 2. 130
5.20 Average TCP throughput comparison for RED and RAD without DoS traffic and with pulsing DoS traffic under a range of values for the in_{th} ratio threshold parameter between 0.1 and 2. 130

Chapter 1

Introduction

It is remarkable how, in a relatively short time span, the Internet has become the primary means of accessing information and communication services for a large part of the world's population. The Internet architecture has proven to be resilient and of extremely scalable design. It is an architecture rooted in packet switching principles, which uses the Internet Protocol (IP) [41] and the Transmission Control Protocol (TCP) [128] together as its main data transmission protocols.

The Internet and its main transport protocol, TCP, adopt an end-to-end principle within their design. This principle implies that protocol operations should, as far as possible, occur close to the end points that are communicating. It is an approach that advocates incorporating functionality close to the source and destination hosts, as opposed to building it in the network itself. This provides considerable advantages as TCP/IP can be made to run in a wide variety of physical networks with minimal assumptions made about them. Within TCP, this principle is applied such that all flow and congestion control mechanisms are located on the end hosts. However, end-to-end TCP congestion control has been shown to be vulnerable to abuse from sources that aggressively generate traffic that fails to comply with TCP congestion control principles [51].

1.1 Denial of Service

Denial of service (DoS) attacks are one source of malicious traffic on the Internet. DoS attacks can be defined as any attack whose goal is to render the victim's services unavailable to other parties. In the Internet, a DoS attack would make a host, or its network services, unavailable to other hosts. Internet DoS attacks can be split in two categories: *software exploits* and *bandwidth flooding attacks* [101].

Software exploit DoS attacks take advantage of software implementation vulnerabilities on network connected hosts and applications to cause service unavailability. The standard defence against software exploit DoS attacks is the continuous updating of networking software to deliver solutions to known vulnerabilities. This is necessary given that software exploits mainly use human implementation errors to their advantage, as opposed to protocol or architecture weaknesses.

Bandwidth flooding DoS attacks use the congestion responsive and end-to-end nature of TCP congestion control to obtain a large share of the bandwidth available to the victim, with the goal of exhausting the bandwidth or computing resources available to it. As these DoS attacks take advantage of the core architecture of the Internet and its main transport protocol, TCP, it is necessary to deploy additional functionality in the network routers and other intermediaries to provide bandwidth flooding DoS defence to current and future TCP implementations.

1.2 Active Queue Management

Active Queue Management (AQM) schemes are a class of Internet router queueing algorithms designed to surmount some of the shortcomings of classic Drop-Tail queues in best-effort networks with congestion responsive TCP traffic. The widely deployed Drop-Tail queueing employs a first-in first-out (FIFO) mechanism to buffer packets on incoming or outgoing links, dropping any arriving packet when queue buffer space is full. Most AQM algorithms are primarily designed to improve congestion control through early notification.

A subset of existing AQM proposals have been designed to provide better

fairness and protection for flows that use end-to-end congestion control, usually by limiting the rate of unresponsive flows. These fairness-oriented AQM schemes are not necessarily effective against DoS traffic that is able to spoof IP packet headers to conceal its presence. There is a clear need for AQM-based algorithms that are effective against DoS traffic that uses spoofed IP packet headers.

An important component of any system that intends to defend congestion-responsive TCP traffic from non congestion-responsive (potentially malicious) traffic is an accurate characterisation of congestion-responsive traffic. A thorough understanding of the expected behaviour of responsive TCP traffic is needed to be able to detect and control malicious DoS attacks. Moreover, knowledge of responsive traffic behaviour must adapt and be useful in the resource constrained, high-bandwidth, Internet router environment, where information avaiable about the traversing traffic is typically of low granularity and accuracy.

There is extensive literature documenting TCP modelling efforts that can be drawn upon to form the basis of such a characterisation of expected congestion responsive behaviour. Historically the focus has been on considering infinite transfers and low numbers of TCP connections. On the current Internet, dominated by Web traffic usage patterns, large numbers of relatively short flows are much more prevalent. Thus, TCP models need to be chosen carefully to be relevant to the expected traffic mix that is common in the modern Internet. Furthermore, complex parameterised models must be tractable in environments such as network routers, where the accuracy and availability of measurement information about the traversing traffic is limited.

1.3 Research questions

There are many unresolved research questions that are pertinent to the challenge of providing a DoS defence for Internet users and service providers. The following questions motivate this dissertation and drive the key design decisions behind the approach presented in this work:

- Scalable and efficient architectures for the capture of detailed measurement metrics at network routers. The spectrum of traffic metrics collected at network routers can vary from simple global state to detailed

per-flow information. There is a need for flexible metric collection designs that provide better than global traffic information, while not maintaining expensive per-flow traffic or degrading accuracy under normal operation.

- Accurate characterisation of realistic congestion responsive TCP traffic. Thus techniques that can operate at multiple time scales and in environments that can capture only partial information about the traffic are required. The state of the art in TCP modelling provides a good starting point but no practical implementation is available and existing router-based implementation mechanisms commonly use less-accurate models that do not characterise the finite and relatively short nature of TCP flows in real Internet traffic.

- Effective detection of malicious DoS traffic. There are multiple existing proposals to characterise and classify known and theoretical DoS attacks. However, there is no general framework that is able to accurately detect the wide spectrum of malicious traffic behaviours that are characterised as part of a DoS attack. Furthermore, such a scheme must also be able to operate effectively on the many systems that form part of an end-to-end TCP connection.

- An architecture and design for DoS attack defence that can be incrementally deployed and requires minimal ongoing management. Many of the existing DoS defence schemes require redesign of network architectures and protocols, and thus provide no simple route to incremental deployment. There is a need for tractable, credible DoS defence architectures that can be effectively employed independently of their location in the network and their deployment status in other systems.

1.4 Aims and objectives

DoS defence mechanisms can be designed for deployment on either intermediate or end systems in the traffic flow between sources and destinations. They can also be classified by the type of DoS attacks that they are designed to counteract. In this dissertation the high-level goal is to address the design of bandwidth

flooding DoS attack defence mechanisms that augment network router queueing logic to provide improved performance to congestion responsive TCP traffic. An additional goal is to ensure there is no negative impact in the absence of a DoS attack.

The design of such a network router based DoS defence mechanism requires traffic metric information about the flows traversing a given link. Most Active Queueing Management (AQM) proposals in the literature use either global state or sampling techniques to limit the amount of information that must be maintained. The amount of state required by the scheme is a key factor in its applicability within different router architectures and deployment scenarios. High bandwidth core network router environments must minimise the amount of state maintained in order to be scalable, whilst edge routers that handle lower amounts of traffic may not have the hardware resources required for full per-flow state maintenance.

An additional requirement for such a measurement metrics architecture is to perform well in the presence of potentially malicious traffic sources, and to avoid vulnerabilities such as packet header spoofing. This dissertation proposes a design for such a measurement metrics system; it is able to provide accurate information about the traffic traversing a network router, whilst being effective in the face of malicious bandwidth flooding DoS traffic.

It is the congestion responsive nature of TCP traffic that makes it vulnerable to bandwidth flooding DoS attacks. The lack of congestion control on the DoS traffic allows it to obtain an unfair share of the bandwidth, ultimately preventing the TCP traffic from receiving services from the victim. The congestion responsive behaviour of TCP traffic is thus a key aspect that differentiates it from most malicious DoS traffic. Thus the characterisation of TCP traffic is necessary for a DoS mitigation scheme that intends to protect the congestion responsive traffic.

This dissertation addresses this need by developing a practical method of characterising aggregates of TCP flows, focusing on their packet and session generation aspects. While there is an extensive research literature on TCP modelling, in practise most widely implemented models do not accurately capture the finite nature of common TCP flow types in the Internet.

The spectrum of behaviour for DoS traffic is extremely wide, from single packet software exploits to massively distributed flooding DoS traffic. This dissertation addresses the problem of detecting and characterising DoS traffic that exploits congestion responsive TCP behaviour for its malicious impact. Effective detection of this type of DoS traffic is an unsolved problem, especially considering the range of traffic generation patterns that can be used by attackers. Bandwidth based DoS attacks have the ability of using different time scales and packet header information to negatively impact TCP traffic, while being difficult to detect by simple methods. An effective DoS detection method must take these factors into account and be able to accurately detect DoS traffic independently of its stealthiness.

The final aim for this work is to develop a next generation network router queueing scheme that will provide transparent protection to TCP traffic when in the presence of bandwidth flooding DoS attacks. Such scheme must use state of the art measurement metrics to accurately characterise TCP traffic and detect DoS non-congestion responsive malicious traffic. It must consist of an architecture that can be deployed and be effective at a variety of locations on the network, whilst requiring minimal management. It is also important that it is incrementally deployable and allows for good performance independently of the queueing schemes deployed in other routers in the traffic path.

1.5 Thesis outline

This chapter presents an overview of the Denial of Service (DoS) defence area and relevant research themes, introduces the principal research questions and main objectives for the work described in this dissertation, and concludes with an outline of the dissertation and an enumeration of related publications that have arisen from the work described herein.

Chapter Two presents a survey of the state of the art in the DoS defence area, and identifies the topics that are most relevant to the proposed DoS defence solution. Denial of Service attack types are classified and concrete examples described in detail. Existing research proposals for DoS detection and defence are introduced and their limitations established. Given that the stated aim of

this dissertation is the development of a network router queueing scheme that is able to protect TCP traffic from malicious DoS attacks, the Active Queue Management (AQM) and TCP modelling research fields are synopsised. The AQM overview provides a clear insight into the main schemes, from the seminal RED proposal to more recent schemes that target flow fairness issues. TCP modelling is highly relevant to the area to allow differentiation of congestion responsive traffic from malicious DoS traffic. Both traditional steady-state models as well as more complex finite TCP modelling efforts are considered.

Chapter Three provides a practical method for characterising TCP flow aggregates at the router queue. This method is based on state of the art finite TCP flow modelling techniques, and is focused on being an accurate predictor of expected behaviour from traffic aggregates composed of finite congestion responsive TCP flows. This chapter also provides an evaluation of the predictive abilities of the proposed method.

Chapter Four introduces Responsive Aggregate Defence (RAD), a novel queueing scheme that provides an effective defence against DoS attacks. This scheme utilises efficient measurement metrics together with the TCP characterisation methods developed in Chapter Three. RAD is able to accurately detect and penalise bandwidth flooding DoS attack traffic at the router queue by focusing on the packet and session characteristics of traversing traffic. It is a DoS defence scheme that is effective at a wide variety of time scales and is robust against packet header spoofing techniques.

Chapter Five carries out an in-depth evaluation of the RAD DoS defence queueing scheme. This evaluation is performed using simulation studies of a variety of different scenarios that explore the space in which RAD is expected to operate in. The three key high-level aspects evaluated are the ability to penalise DoS traffic, the transparency provided to TCP and the impact of system parameterisation.

Finally, Chapter Six concludes the dissertation by highlighting the core contributions of this work and establishing future directions that offer interesting and challenging avenues for extension of this work.

1.6 Publication record

- Arkaitz Bitorika, Ciarán Mc Goldrick and Meriel Huggard. "Responsive Aggregate Defence for Denial of Service Attacks". *In Proceedings of the 2nd IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM07)*, October 2007.

- Meriel Huggard, Mathieu Robin, Arkaitz Bitorika and Ciarán McGoldrick. "Performance Evaluation of Fairness-oriented Active Queue Management Schemes". *In Proceedings of the 12th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, MASCOTS'04. October 2004.

- Arkaitz Bitorika, Mathieu Robin and Meriel Huggard. "An Evaluation Framework for Active Queue Management Schemes". *In Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, MASCOTS'03. Pages 200– 206. October 2003.

- Arkaitz Bitorika, Mathieu Robin and Meriel Huggard. "A Framework for Evaluating Active Queue Management Schemes". Technical Report TCD-CS-2003-34, Trinity College Dublin, Department of Computer Science, July 2003.

1.7 Conclusion

This chapter has provided an introduction to the area of Denial of Service (DoS) attack defence and established the main objectives for this dissertation work. High level research questions have been identified for the effective protection of TCP traffic from malicious DoS traffic. From these questions more concrete objectives have been defined that will guide the work in the following chapters. These objectives provide a road-map to an effective and flexible network router based bandwidth flooding DoS defence scheme. The Chapter concludes by listing relevant publications by the author and summarises the different Chapters in the dissertation.

Chapter 2

Literature Survey

The overall focus of this work is on the design of a network router based Denial of Service (DoS) attack defence mechanism. This chapter provides a detailed overview of the three key areas that need to be brought together in the design of such a system, namely Denial of Service attacks and existing DoS defence mechanisms, Active Queue Management techniques and TCP characterisation. This chapter provides a detailed insight into the problem of DoS attacks, the nature of congestion responsive TCP traffic that is penalised during a bandwidth flooding DoS attack, and past work on scalable router queueing algorithms.

The first relevant literature topic for the focus area of the dissertation is that relating to DoS attacks. This includes their characteristics and existing proposals for DoS detection and defence mechanisms. The term "DoS attacks" covers a wide range of traffic generation patterns that share the common goal of impacting the availability of a chosen victim service. This chapter provides a state of the art review of past research efforts that focus on detection and defence from DoS attacks that must be considered in order to provide the reader with the appropriate context within which to site this work.

Categorising and defining DoS attacks is the first aspect that should be addressed. There are a wide range of network traffic generation behaviours that can be considered to be part of a DoS attack. At the extremes these range from a single, carefully crafted packet that exploits a software implementation vulnerability to distributed attacks that generate massive amounts of traffic to

the intended victim. In the next sections existing DoS attack categorisation proposals are introduced, and many of the most important known DoS attack types described in detail.

Even though the onset of widespread DoS attacks is relatively recent, there is an extensive literature relating to DoS detection and defence mechanisms. The popularity of this topic in the research literature is closely correlated to the severity, and impact, of these attacks on Internet services and the lack of reliable detection and defence mechanisms for some of the attack types. A survey of DoS traffic detection proposals is provided in the next section, as well as a thorough discussion on DoS defence research relating to different types of DoS traffic.

When considering the design of next generation network router queueing mechanisms, Active Queue Management (AQM) approaches are at the core of many recent advances on scalable queueing mechanisms that improve upon the traditional, and simple, drop-trail queueing mechanism that is still the *de-facto* standard on the Internet. To provide the reader with the necessary background information on the field, a detailed review of AQM research activity from the initial RED proposal to more recent schemes that intend to provide better fairness and congestion-responsive traffic protection is provided.

The seminal RED [54] scheme that initiated the AQM research field is reviewed in detail. The design of RED includes many of the aspects that other, more recent, AQM proposals are based on and it can be considered the template for a whole class of scalable network router queueing mechanisms. Since the introduction of RED, there have been many other AQM proposals that seek to either improve flow fairness or provide some level of defence against aggressive traffic. The most relevant schemes in this area of AQM are detailed in this chapter.

The Transmission Control Protocol (TCP) is the main transport layer protocol on the Internet. The goal of bandwidth flooding DoS defence can be articulated as the protection of congestion responsive TCP traffic from malicious DoS traffic that does not operate in a congestion responsive manner. Advancing towards this goal requires a deep understanding of TCP, its congestion control mechanisms and past work on characterisation of TCP behaviour under a variety of scenarios.

The TCP congestion control mechanism is the main characteristic that differentiates TCP from bandwidth flooding malicious DoS traffic. TCP congestion responsiveness is the very same aspect that makes TCP flows vulnerable to bandwidth attacks by other traffic, such as DoS attack traffic, that does not maintain congestion responsive behaviour. This chapter will provide an overview of the main congestion control algorithms used in TCP.

Modelling TCP traffic behaviour is an area that has received considerable research focus. From the point of view of the work described herein, TCP modelling is highly relevant as it enables the understanding of expected TCP traffic performance and TCP traffic generation behaviour. This chapter will provide a survey of the most important TCP modelling proposals, including models for infinite steady-state TCP flows as well as more complex models that address finite TCP connection behaviour.

The literature survey in this chapter is organised as follows: the first section will provide an overview of Denial of Service (DoS) attack types, as well as the most relevant research proposals for DoS attack detection and defence. The next section will consider the topic of Active Queue Management (AQM). The seminal RED scheme, as well as more recent fairness oriented schemes, will be discussed. The following section will address TCP congestion control and existing proposals for TCP performance modelling. Finally, the conclusion will provide an unified overview of the content for this thesis.

2.1 Denial of Service Attacks

Denial of Service (DoS) attacks produce some of the most significant types of malicious traffic on the Internet. They are perpetrated by an attacker (or attackers) who cause their victim to receive malicious traffic that renders it unable to provide its intended network services. In short, they may be defined as any attack whose goal is to render the victim's services inaccessible to other parties.

The Internet has become the main communications infrastructure for businesses and private individuals, thus it has increasingly become a target for DoS attacks. Network, system and application level vulnerabilities have all been

exploited to create network DoS attacks. Moreover, the *end-to-end* principle [118, 18] at the core of the Internet architecture coupled with the permissiveness of network protocols to behaviour such as source address spoofing, make attacks easier to perpetrate. The Internet does not police end-to-end traffic, so any host is potentially able to damage its peers by sending malicious traffic.

DoS attacks are related to both the security and the availability of the core Internet infrastructure, making them an important issue for researchers, network operators and government bodies alike. The Computer Emergency Response Team (CERT) published [66] the outcomes of the Distributed Systems Intruder Tools Workshop in 2001. This provided a time-line of the evolution of known attack tools from July 1999 to September 2001 and a discussion on what forms future attack tools might take.

There is anecdotal evidence that DoS attacks are widespread and frequent [1]. The actual prevalence of DoS attacks on the current Internet was studied by Moore et al. [105]. The technique they employed used packet sampling and thus were only able to detect attacks with uniformly distributed, random spoofed source addresses. Their results showed that DoS attacks are widespread on the Internet, distributed among many ISPs and domains. The study also showed that attack size (number of bytes) and length (time duration) followed a heavy-tailed distribution with a small number of large and long attacks and many more smaller, shorter attacks.

In an effort to develop a shared language for research on DoS attacks, Mirkovic et al. propose a classification of DoS attacks [101] based on the degree of automation, exploited vulnerability, source address validity, attack rate dynamics, possibility of characterisation, persistence of agent set, victim type and impact on the victim. They also provide a simple taxonomy of defence mechanisms. While exhaustive, this taxonomy is very complex due to the many classification factors considered, and we have opted for a simpler subset below when considering DoS attack and defence categorisation.

Two broad categories can be used to categorise DoS attacks: *software exploits* and *flooding attacks* [67]. Software exploits take advantage of specific application or system level protocols to make the victim's services unavailable. Flooding attacks abuse the current Internet architecture to attack their targets

by overwhelming them with large amounts of artificially generated traffic.

The novel AQM based DoS defence system described herein may be considered as a router-based defence against flooding attacks. However, for completeness, both software exploits and flooding DoS mechanisms are described in detail in the following sections.

2.1.1 Software exploit DoS attacks

Software exploits take advantage of specific vulnerabilities in the software running at the victim to make its services unavailable. These exploits can be based on vulnerabilities in network or application protocol designs, as well as in their software implementations. These kinds of attacks may be extremely powerful as they can potentially disable a machine with a single packet. Given that users can install arbitrary software on Internet connected hosts, prevention of software DoS exploits must be carried out by the combination of proper software design processes and timely updates of the installed base of software systems. Examples of well known software exploit DoS attacks are given below. It must be noted that these are historical examples and that most systems on the Internet have already been patched against them.

Ping of death[90] consists of sending a malformed IP packet to the victim, causing it to crash. When a packet is larger than the physical layer frame size allows, it will be divided in multiple IP fragments [41]. These fragments will be assembled at the receiver to form the original packet. IP packets have a maximum size of 65,535 bytes, so if the original packet was larger than 65,535 bytes, it is possible that it will overflow receiver memory buffers at reassembly time, potentially causing it to crash. Despite the attack name, it is not related to the ICMP echo packets used for the *ping* application.

Smurf attack[28] uses broadcast ICMP echo (ping) traffic to flood the victim with traffic. There are three parties in a Smurf attack: the attacker, the intermediary and the victim. The attacker sends an ICMP echo request directed to the IP broadcast address of the intermediary network. If the intermediary does not filter ingress broadcast ICMP traffic, all hosts on

the network will receive the ICMP echo request and will respond with an ICMP echo reply packet. The attacker can spoof the source address of the original ICMP echo request packet to be the IP address of the victim, effectively causing the intermediary to send a flood of ICMP echo responses to the victim. Networks can avoid being intermediaries on Smurf attacks by disabling IP-directed broadcasts for incoming traffic on their edge routers.

LAND attack[91] works by sending a TCP SYN packet with the same IP source and destination addresses, and same TCP ports. This causes the victim to enter an infinite loop in which it continuously sends the malicious packet to itself. Most modern firewall software filters out LAND attack packets.

SYN flooding[26] also known as a half-open connection attack. This involves the malicious client sending the first SYN packet of a three-way TCP connection establishment handshake, but not replying with the final ACK packet after the server has sent the SYN-ACK packet. A common implementation of the TCP/IP protocol stack allocates new memory resources for connection-related state between when the initial SYN and the final ACK are received. This allows the host to recognise the half-open connection when the ACK arrives. A flood of SYN packets can cause a denial of service as the server's internal data structures are exhausted. SYN flooding can be prevented by using a mechanism called SYN cookies, which are used when the SYN queue is full at the server. When using SYN cookies, the server sends a specially crafted TCP sequence number in the SYN-ACK packet, and only adds the connection to the SYN queue once the final ACK has been correctly received from the client.

Teardrop[27] exploits a bug in the IP fragment reassembly code of some operating systems. Some network stacks do not properly handle the situation of overlapping IP packet fragments and can crash when receiving such traffic. Teardrop attacks have been prevented by applying operating system vendor patches.

Software exploit DoS attacks can be extremely dangerous as they take advantage of the specific network protocol implementations and thus do not need to generate very large amounts of traffic to be effective. However, due to the maturity and stability of current TCP/IP implementations, most recent software exploits target the application software layer instead of the networking stack.

2.1.2 Bandwidth flooding DoS attacks

Flooding DoS attacks are much more prevalent than software exploits in the current Internet [105]. Active research efforts are trying to address this issue from multiple perspectives [67, 87, 85, 73, 144]. In one of the first accounts of flooding DoS attacks, Gibson [58] details how his site grc.com was attacked in May 2001, moreover he provides an analysis of the tools and technologies used by the attackers.

Flooding DoS attacks consist of sending traffic to the victim with the goal of exhausting the bandwidth or computing resources available to it. These attacks can be categorised based on two main aspects: the structure of the network that generates them and the bandwidth pattern followed by the attack traffic.

We will focus on flooding denial of service (DoS), distributed denial of service (DDoS) and distributed reflector denial of service attacks from the attacking structure point of view. The two main attack traffic patterns we will study are Persistent and Pulsing flooding traffic. We will discuss these categories in further detail below.

Attack network structure DoS attacks can be perpetrated using everything from a single attacking host to a complex hierarchical network of attackers. Compromised host machines, known as *agents*, *zombies* or *bots [127]* are commonly used for multi-attacker or distributed DoS (DDoS) attacks. Even though a DoS attack from a single attack host is theoretically possible, most attacks are comprised of multiple attackers. Distributed DoS attacks (DDoS) typically use multiple agents controlled by one or more handler nodes. The agents are the hosts that send the flood traffic to the victim, while the handler is just a host that controls an agent network. Depending on the sophistication of the attacker and its tools, this architecture can be expanded to contain more than

one handler, each one with its own agent network.

DDoS attacks increase the attack intensity relative to single-host DoS attacks and lower the possibility of effective response. There are several readily-available DDoS attack tools available on the Internet. These provide simple mechanisms for the control of agent networks and the launching of attacks through them. The sophistication of these attack tools varies greatly and they are still evolving *in the wild*, with features such as hierarchical agent networks, encrypted handler to agent communication and other security measures to defend agent networks from being caught or "stolen" by rivals. However, in general the basic attack strategy of using several hosts to flood a target is kept.

DDoS attacks require many compromised hosts that can be controlled to launch their attacks. Staniford et al. [127] studied the spreading mechanisms of Internet worms and how they could be used to rapidly gain control of millions of hosts. These hosts, once under the control of the attacker, could be used to launch DDoS attacks. The use of aggressive infection techniques could allow for the building of a DDoS bot network in the order of tens of seconds. This would prevent any kind of human-mediated counter-response to the spread of the bot network.

Another variant of DoS attacks are distributed reflector DoS (DRDoS) attacks, in which agents send requests with the source address of the victim to hosts that respond to requests. The reflector hosts flood the victim with response packets to requests that it didn't make. The use of reflectors can potentially multiply the impact of the attack, depending on the size of responses that reflectors return for certain requests. In addition, the use of reflectors makes it much more difficult to trace the real source of the attack, as the attack traffic that the victim receives is generated by the reflectors, who only unknowingly collaborate in the attack. Paxson analysed [110] the danger presented by Distributed Reflector DoS (DRDoS) attacks, identifying three type of reflectors and proposing some defence strategies that could be used against these attacks.

Traffic pattern Flooding attacks that use generated traffic patterns can be divided into two categories: Persistent and Pulsing attacks.

Persistent flooding attacks try to make the victim unavailable by directing

large amounts of traffic to it. This causes congestion on the network links that lead towards the victim, thus preventing legitimate clients from accessing its services. Internet services typically use TCP as transport protocol for their application payloads (eg. HTTP [49]). However, the Internet packet switching infrastructure does not enforce any behaviour on the transport protocol used.

TCP is a congestion responsive protocol: TCP end hosts will lower their traffic rate when congestion is detected either from packet losses or ECN notifications [50]. This behaviour provides an approximation of flow fairness when TCP flows are competing for a fixed bandwidth link. However, when non-congestion responsive traffic is added to the mix (typically UDP), the unresponsive traffic is able to obtain an unfair share of the available bandwidth [51]. This unfairness can lead to congestion collapse, where most of the traffic sent by the sources is dropped by an intermediate router before reaching its destination.

Persistent flooding attacks are an extreme example of congestion collapse. The victim receives large amounts of non-congestion responsive traffic which fills the links that lead towards it, causing congestion. The legitimate clients that want to access the victim's services have to compete with the attack traffic for the available bandwidth. However, if the clients are using a TCP-based congestion controlled protocol, they will end up backing off and starving themselves of network access, leaving most of the link bandwidth available to the attack traffic.

A newer variant of flooding DoS attacks are pulsing or low-rate attacks. These types of low-rate TCP targeting DoS attacks were introduced by Kuzmanovic et al.[80, 81]. They are also known as "shrew" attacks due to their comparison with large, long-lived "elephant" flows [80]. These low-rate attacks take advantage of TCP congestion control behaviour to inflict DoS effects on the victim while generating a sufficiently low average transmission rate which prevents their detection by DoS defence mechanisms.

Shrew attacks generate an on-off "square wave" shaped traffic pattern. By matching the period of the DoS source to the approximate TCP Retransmission Time Out (RTO), these attacks can cause TCP sources to never leave the timeout state, thus obtaining close to no effective throughput. Shrew attacks are most effective when the victim traffic mix has relatively stable RTTs. With

highly variable RTTs they work as a high-RTT pass filter, with attack effectiveness decreasing as flow RTTs increase. These attacks can be considered as a hybrid between "classic" flooding DoS attacks and software exploits, as they generate traffic to compete with TCP sources for throughput, but take advantage of TCP congestion-control behaviour to amplify their impact.

Reduction of Quality (RoQ) attacks [61, 63] are based on a generalisation of the ideas behind shrew attacks. RoQ attacks target adaptation mechanisms on networked computer systems in general, instead of just trying to keep TCP in a timeout state like shrew attacks. By targeting the dynamic adaptation of systems like TCP and router queueing mechanisms, RoQ attacks can cause a variety of impacts that introduce instability and jitter to legitimate traffic [62]. A common theme of all the low-rate attack approaches is the use of periodic on-off (pulsing) traffic generation processes to maximise their impact on transient dynamic behaviour while minimising their "detectability" by keeping the long-term average generated traffic rates low.

While pulsing attacks have received widespread interest among the research community [81, 144, 63, 35], no known "in the wild" attacks having been recorded. Chertov et al. studied [35] low-rate attacks by evaluating their impact on a variety of simulation and emulation scenarios. They found significant differences between the results obtained through simulation and the ones obtained through emulation or in network testbeds. This highlights the need to consider the impact of the whole system, including hardware and network device drivers, when evaluating DoS attacks.

The relationship between pulsing low-rate attacks and router queueing configurations is studied in [119]. A promising result from this work is that the size of the router queue buffer, and the queueing algorithm used, have a considerable effect on the effectiveness of pulsing attacks. Larger buffers force attackers to increase their traffic load to cause multiple packet losses, thus causing them to be more easily detectable.

The interaction between different TCP flow types can also be used to cause a special kind of pulsing attack [45]. Introducing a very quick succession of very short TCP flows into a traffic mix with long-lived TCP flows can adversely impact the throughput achieved by the long-lived flows. This type of attack takes

advantage of the relatively aggressive nature of TCP flows during the initial slow-start phase, compared with "older", long-lived flows that are in the congestion avoidance phase. By carefully timing the short-lived TCP flow generation process, it is possible to cause what is effectively a shrew attack, in which the long-lived TCP flows enter a constant timeout state.

2.1.3 DoS detection

Detection of ongoing DoS attacks and the identification of their characteristics is a prerequisite for any DoS defence system. This is a very active research area [67, 87, 34, 76, 59, 111, 22, 120]. DoS detection mechanisms can be divided into three, broad categories depending mainly on the implementation approach taken: packet stream analysis, traffic profiling and traceback.

Packet stream analysis A variety of statistical, signal processing techniques can be applied to a stream of network packet events for the purpose of detecting DoS attacks.

Hussain et al. proposed [67] the detection and classification of DoS attack traffic as single-source or multi-source (DDoS and Reflector-DDoS). Their approach uses header analysis, ramp-up behaviour classification and spectral analysis. They examine packet headers for unusual header combinations, such as all TCP flags being set. Analysis of ramp-up behaviour and of the spectral characteristics of packet arrivals over the attack duration are used to discern whether it is a single-source or a distributed attack.

Temporal and spatial correlation of attack traffic information and machine learning have also been used to detect ongoing DoS attacks [87]. An architecture in which information is gathered from perimeter or edge routers and then aggregated for detection purposes was implemented, allowing for the detection of attacks, even if the attack traffic volume on individual links is quite low. A similar spectral analysis approach is taken in [34] to detect DoS attacks by focusing on the periodicity of TCP traffic. This approach requires small RTT ranges and relatively low aggregation levels; these prove to be significant limitations to this approach as they are not typical on normal Internet traffic [56, 3, 148].

Wavelet-based statistical techniques have proven popular for use in the ana-

lysis of traffic time series data for DoS detection. WADeS [114] uses an Least-Recently Used (LRU) cache filter of long flows, at any point in time this holds information about the most recent flows that the router has received traffic from. Wavelet variance analysis is applied to packets from flows that are not in the cache, heuristics are then used to detect attacks based on the changes in the wavelet variance caused by attack traffic.

Barford et al. [11] use wavelet time-frequency analysis to develop a *deviation score* that highlights signal variation on both high and medium frequency bands. This approach allows for the detection of both short and long-lived traffic anomalies. A hybrid of traditional statistical anomaly detection techniques combined with the continuous wavelet transform is used by Dainotti et al. [37] for DoS detection. An alternative approach involving the energy distribution of packet inter-arrival times and wavelet analysis is proposed by Li et al. [85] with the goal of detecting DDoS attacks. Changes in the energy distribution variance are used as an indicator of an ongoing attack in this case.

Seong Soo Kim et al. [76] took advantage of the different patterns in destination IP addresses between normal and malicious DoS traffic for DoS detection. This approach detects outgoing attacks at a network egress router by wavelet-based analysis of a weighted correlation of destination IP addresses. This system makes use of efficient data structures to allow for real-time detection of DoS traffic. The authors also showed how the additional analysis of port number correlation improves detection accuracy.

By applying a time series analysis approach to pulsing DoS attacks, Luo et al. [88] model low-rate pulsing attacks and propose a detection scheme that uses wavelet transforms to extract the desired frequencies from packet arrival time information. The detection mechanism is focused on traffic pattern anomalies, looking at both the pulsing behaviour of attack traffic as well as at fluctuations in the TCP transmission rate as it enters into the timeout state as a consequence of an attack.

Traffic pattern heuristics MULTOPS [59] uses the ratio of the observed packet rate going to and coming from a host, or subnet, to identify bandwidth flooding DoS attacks. Its authors acknowledge that it performs poorly in the

case of spoofed, or randomised, source IP addresses, a common traffic type on DoS attacks.

Another group of DoS detection proposals can be formed from those systems that focus on the behavioural characteristics of DoS traffic relative to that of well-behaved TCP traffic as generated through normal, non-malicious use of the network.

Wang et al. [140] used the discrepancy between the number of TCP SYN and FIN packets traversing through a router to detected SYN flooding attacks using a change point detection technique. However, this approach is limited by the fact that it does not detect any other kind of possible attack and does not work when asymmetric routing is in use or if the attacker changes the flood traffic mix to include FIN packets. Due to this the authors recommend against using it in core network routers.

Peng et al. [111] proposed monitoring the arrival rate of new source IP addresses as a way to detect DoS attacks. This approach should be more effective at detecting attack traffic close to the source and be able to differentiate flash crowds from attacks. The scheme seems effective for attacks using a large number of randomised source addresses. However, it is not clear how it would perform if addresses were spoofed in some other, more sophisticated way, for example by choosing addresses from the current subnet.

Traceback Being able to discover the real source hosts of DoS traffic is an important component of many DoS detection systems. This task is called *traceback*. Traceback poses considerable challenges when, as is frequently the case, the attack traffic contains spoofed source addresses. Most of the mechanisms to solve this problem strive to find a set of routers that are common to the attack tree traversed by the DoS traffic. An overview of the most important traceback proposals in the literature is provided below.

Cheswick and Burch [22] proposed a traceback method that consisted of sending bursts of traffic from the victim's network towards the attacker, with the goal of finding the link path to the attacker by detecting rate changes in the attack traffic. This method assumes largely symmetric routing and single-source DoS attacks, therefore it does not work well against DDoS.

CenterTrack [130] is a system that solved the problem of identifying the origin of flood traffic within the boundaries of an ISP network. It uses an overlay network of *tracking routers*. Traffic for the victim is directed to these routers, which then perform hop-by-hop traceback until the ingress adjacency from where the malicious traffic comes is identified.

ICMP Traceback [14] is a work in progress draft IETF proposal that allows for the traceback of attack traffic and the identification of real sources. ICMP Traceback works by requiring transit routers between the source and the destination to send ICMP Traceback packets to the destination with very low probability. In the case of flooding DoS attacks, even an extremely low probability of ICMP Traceback packet generation allows for the identification of the path taken by the attack traffic to the victim.

Intention-driven ICMP Traceback [94] is an improvement on ICMP Traceback that allows hosts to express their "intention", or desire to receive ICMP traceback messages. This information is added to BGP routing tables. Forwarding routers, instead of just generating an iTrace packet for random destinations every 1/20K packets, choose the iTrace packet destination from the list of destinations that have shown "intention" in their BGP routing tables.

iCaddie [137] is an ICMP traceback scheme that generates ICMP *Caddie* packets that follow randomly selected IP "ball packets", recording the sequence of routers they traverse. Caddie packets contain a digest of the invariant portions of the ball packet as well as a Router List, authenticated with a keyed-Hash Message Authentication Code (HMAC[78]). Caddie-aware routers then append their IP address, the address of the next hop and the interface in which the ball packet arrived to the Router List. The designers of iCaddie have advocated [138] its use for the traceback of reflective DoS attacks. Their mechanism involves trace back to the reflectors border router and thence to the original attacker's router, both using iCaddie. However, they do not provide sufficient detail about the inter-router protocols that would be used to achieve this.

SPIE ("Source Path Isolation Engine") [124, 125] works in a similar way to a logging mechanism, but it stores 32-bit hashes of the first 28 invariant bytes of each IP packet and uses Bloom filters to minimise storage space requirements. The architecture of SPIE includes Data Generation Agents (DGA), SPIE Col-

lection and Reduction agents (SCAR) and SPIE Traceback Managers (STM). DGAs run at every router, storing packet digests in bitmap tables in consecutive time intervals. These tables are transferred to SCARs if there is interest in any particular time interval. SCARs are able to produce traceback information for packets that traverse their region, while the STM can aggregate all the tracebacks from SCARs to obtain complete attack graphs which may then be made available to IDSs or other interested parties.

A seminal proposal by Savage et al. introduced Probabilistic Packet Marking [120]; an alternative to ICMP Traceback that would provide traceback functionality without requiring the injection of new packets into the network. Probabilistic Packet Marking (PPM) uses a probabilistic marking algorithm and IP header fields for the transmission of traceback information. This traceback technique can be used to track down the real source node(s) of DoS attacks. It works by having intermediate routers add edge information fragments to packets, encoded in existing IP headers to reduce per-packet space requirements. This proposal has spawned a large number of alternative packet marking proposals that intend to improve on its shortcomings:

- A proposal for improved PPM [126] altered the methods in [120] to incorporate the hash functions of IP addresses as markers and to use a pre-obtained map of the upstream router graph. If the router graph had been obtained from the victim, rather than through ICMP iTrace packets for example, it would have the limitation that asymmetric routing would affect its accuracy. This proposal also included a method for avoiding forged markings made by compromised routers. This was achieved through the use of a message authentication code to sign the marking and a set of time-varying keys that avoids the sharing of private keys between the victim and routers.

- Dean et al. proposed [38] an alternative probabilistic packet marking scheme that used algebraic techniques to encode path information in the packets. Their approach encoded path information as points in polynomials and then used coding theory to reconstruct these polynomials at the destination. The main goal of this scheme was to improve on the reconstruction complexity of PPM [120].

- Goodrich proposed [60] a method for IP traceback with packet marking called randomise-and-link. In this scheme routers send a checksum cord with a fragment of the message they want to deliver to the destination. This cord is used to both associate fragments together and to verify the integrity of the messages. The use of specific traceback messages that could work well with attack-trees of up to 2000 routers was also proposed. In addition to router address information, these messages included randomness to prevent packet header spoofing by the attack sources.

- Probabilistic Pipelined Packet Marking (PPPM) [13] is a packet marking traceback proposal that seeks to reduce the number of packets required for source identification. PPPM uses TTL information alongside router IP addresses to prevent the spoofing of marking information. Intermediate routers are then used to propagate marking information they have received from upstream routers via subsequent packets to the same destination. The existing proposal has the drawback that it requires an unrealistic 57 bits of IP header space.

- Distributed Link-List Traceback (DLLT) [12] is a variant of PPM that keeps marking information in intermediate routers for later retrieval. The main principle behind DLLT is to have routers store their address in the marking field using space-efficient Bloom filters [17]. Thus the marking received by the destination is effectively the head of a linked-list of routers that is itself a sub-list of the whole attack path. This method requires significantly fewer packets than PPM [120] to obtain good traceback. It also prevents attackers from sending spoofed marking information to the victim.

Park et al. studied [84] the performance of PPM traceback when the attacker has the ability to spoof the marking field. They showed that in the case of single-source DoS attacks, PPM is effective at localising the attack origin. However, PPM traceback is not effective when applied to DDoS attacks, as attack traffic generation is scattered among the different sources. Adler analysed PPM schemes [2] in both single and multiple attacker scenarios. He studied efficient encoding techniques for marking information and the trade-off between the num-

ber of packet header bits used and the number of packets received for attack path reconstruction. This work does not address the computational complexity of reconstruction at the victim and limits its recommendations to very specific multiple attacker scenarios.

GOSSIB [136] was proposed as an attack on PPM [120], it aims to insert misleading edges into the attack graph with the goal of masking the attacker as an innocent transit router. An additional consequence of GOSSIB is an explosion in the size of the search state space for the victim. GOSSIB generates marking information in such a way that some of the markings could be used for reconstructing multiple edges, allowing for very efficient introduction of fake edges. This was done by exploiting limitations in the PPM hashing scheme.

2.1.4 DoS defence

In this section an overview of DoS defence proposal characterisations is provided. Defence approaches for both more "traditional," persistent flooding DoS attacks and newer, low-rate pulsing attacks are considered in detail. The growth in DoS defence techniques has, naturally, resulted in a proliferation of defence mechanism proposals. To assist the research community a number of authors have sought to characterise these into a number of subclasses [32, 29, 143].

A characterisation study of DDoS defences was performed by Chen et al. [32]. Their taxonomy divides DDoS defence proposals into three categories: *congestion-based*, *anomaly-based* and *source-based*. Congestion-based defences, like ACC [92], primarily work by using packet drop information as an indicator of an ongoing attack. Anomaly-based proposals use some kind of change-point detection algorithm to act when the traffic mix moves away from some specified "normal" behaviour; while source-based approaches aim to deal with anomalous source IP address spoofing by DoS attack traffic. Chen et al. classify a number of DDoS defence proposals based on these categories together with a set of characteristics that include the granularity of detection, the network information monitored and the possibility of false positives.

An alternative taxonomy of DoS countermeasures was put forward by Champagne et al. [29]. Their work provides a very comprehensive tree of DDoS countermeasures, with a top-level categorisation divided into three classes: *pre-*

vention, *mitigation* and *deterrence*. Prevention mechanisms focus on improving network protocols and security to prevent compromised hosts from participating in attacks. Mitigation is the main category of DDoS countermeasures, which is divided further into network, server and client-based approaches. Deterrence oriented proposals have the goal of allowing the identification and prosecution of attackers.

Most DDoS attack and defence research effort focuses on empirical studies. However, an analytical model for a DDoS attack and defence mechanism was provided in [142]. This model captures the interaction between attack and defence mechanisms, with the goal of estimating optimal levels of defence investment based on the productivity of the defence mechanism. One of the more interesting contributions of this model is the possibility of configuring the optimal packet marking rate of a DDoS defence mechanism for a given productivity level.

The literature on DoS detection and defence research contains a variety of evaluation studies, but very little effort has focused on purely DoS defence proposal evaluation. In [143], DDoS defence mechanisms were evaluated and new attack types that would be successful against ACC[92] and RED-PD [93] were proposed. While typical flooding DoS type attack traffic was evaluated in this work, the authors studied the newer "periodical blasting" (pulsing) and "bandwidth soaking" attack types. These are very good examples of the smarter attacks that are able to successfully exploit limitations in current DoS defence proposals.

Persistent flooding defence This section provides a discussion of some of the most important DoS defence proposals that target persistent flooding DoS attacks. As these attacks have historically been the most common, and effective, the literature on how to defend against them is very large.

D-WARD [99, 100] was designed to be deployed at network egress routers to stop attacks originating from inside the network. It uses simple models of "normal" TCP, UDP and ICMP traffic and rate-limits flows that do not conform to them. It maintains per-connection information and does not work well with spoofed packets.

Pushback/Aggregate Congestion Control [92] included a router mechanism to detect and control high bandwidth aggregates and a *Pushback* mechanism to ask upstream routers to control aggregates. It identifies high bandwidth aggregates by analysing the drop history of RED and rate-limits them to reduce the total arrival rate at the queue. Its main disadvantage is that it requires cooperation between routers to be effective against DDoS as it only detects the attack, or flash crowds, close to their destination, where most of the damage has already been done.

DefCOM was proposed in [117]. It is based on a set of design principles which the authors considered necessary for an effective DDoS defence system. DefCOM is a distributed system, with *alert generator* nodes that detect attacks, *core* routers that rate-limit the identified traffic and *classifier nodes* that differentiate between legitimate and attack flows and cooperate with core routers to provide good service to legitimate traffic.

Hop-Count Filtering [71] detects and filters out spoofed traffic close to the victim, thus proving an effective mechanism for the removal of malicious traffic. It estimates a hop-count for each packet that arrives based on the value of the TTL field. A mapping of IP prefixes to hop-counts is kept, dropping any packet that has a different hop-count to previous packets from its alleged source network. This solution would not work in the case of attacks that utilise spoofed source addresses from the local subnets or with reflector DDoS attacks.

Traceback mechanisms can be an effective way of detecting the real sources of attack traffic, but they don't provide any means of actually mitigating the attack. Sung et al. proposed [132] a DDoS defence scheme that uses on-line IP traceback information to preferentially filter packets that arrive through an "infected" edge, that is a link that is used by the attack traffic.

Denial of service (DoS) and quality of service (QoS) can be seen as related aspects of the same concept, as arguably DoS is just an example of the worst possible quality of service that can be provided. Garg et al.[57] used the QoS facilities of the operating system to protect a server from DoS due to resource usage. This approach provides little, or no, protection from bandwidth attacks as the resource in question, bandwidth, has already been used by the time the attack traffic reaches the server. Wang et al. discussed [139] how transport-

aware IP routers with QoS policies that differentiate between TCP data and control packets can provide better service and protection from DoS attacks.

There have also been proposals that considerably change the current architecture and traffic forwarding behaviour of the Internet. SIFF [145] divides Internet traffic into privileged and unprivileged classes, with privileged traffic always having priority over unprivileged traffic. In order to use SIFF end-hosts and routers must be updated, however legacy traffic from non-updated hosts is then considered unprivileged. It also requires space for flags in IP packet headers. SIFF-enabled receivers can protect themselves from aggressive or malicious flows by dropping their unprivileged channels, causing packets sent using these channels to be dropped by routers close to the source.

Providing hosts with a means to control which packets reach them has been recommended as a solution for DoS flooding attacks [83]. With the goal of receiving packets only for desired services or connections, hosts can respond to packet floods by choosing which packets they would like to receive and at what rate. This could be implemented via an overlay network that sits on top of the core IP network (called i3 in [83]), or through the modification of the upstream edge router software to allow it to accept configuration changes from the host.

The success of CAPTCHAs [135] or graphical puzzles for authenticating that actual humans are using a service has inspired some DoS defence applications. KillBots[73] targets DDoS attacks that mimic Flash Crowds by using multiple HTTP requests to cause a heavy load on the higher level resources of the attacked system. This approach uses graphical puzzles that new users have to resolve as part of initial connection establishment when the server is under attack. KillBots works by sending a graphical CAPTCHA to each client at HTTP/TCP connection establishment, so that the user must solve the CAPTCHA before being able to making a normal HTTP request to the server.

Pulsing DoS defence Pulsing or low-rate bandwidth DoS attacks that target TCP behaviour have received much attention in recent years [81, 149, 122]. Even though there haven't been any known attacks that have followed this approach, their potential impact is significant and it is essential that DoS defence mechanisms are able to mitigate them.

Xu et al. proposed [144] a double horizon approach, one component of this used short time-scales to look for low-rate bursty attack traffic and the other component used much longer measurement periods, focusing on TCP-like, congestion-responsive behaviour. This design is intended to be effective at defending against both persistent and pulsing DoS attack traffic. Their evaluation results suggested that the best performance can be achieved with a hybrid approach that uses instantaneous regulation for short-term traffic spikes and a more delayed response for longer-term non-responsive traffic.

Another proposal against pulsing DoS attacks is Collaborative Detection and Filtering (CDF)[33]. CDF uses spectral analysis of traffic to compare incoming traffic with predefined templates of the spectral characteristics of attack traffic. The authors described [33] possible hardware implementations of CDF in network processors or FPGAs.

HAWK [133] is an AQM-like adaptive queueing algorithm that utilises a small flow-table to protect TCP traffic from low-rate pulsing attacks. HAWK uses an approach similar to that of the AQM scheme CHOKe [108], randomly sampling arriving packets depending on the moving average of the queue occupancy, then creating a flow table and comparing new packets with existing packets already in the queue. The flow table is used to keep track of flow statistics so that bursts caused by pulsing DoS attacks can be identified.

Some researchers have argued for a more light-weight approach to mitigating the impact of pulsing DoS attacks [146, 119]. Randomisation of the minimum TCP retransmission time-out (RTO) is proposed [146] as a way of avoiding global synchronisation of TCP sources in the timeout state due to pulsing attacks. The relationship between router queue buffer sizes and the effectiveness of low-rate TCP targeted pulsing DoS attacks was studied in [119]. The results showed that moderate increases in queue buffer sizes can mitigate the impact of low-rate attacks; forcing them to increase their offered load until they no longer can be considered low-rate and thus can be identified by mechanisms designed for more typical flooding DoS attacks.

As detailed above, a wide variety of approaches have been taken to create DoS defence schemes. Most of the older proposals for bandwidth attack defence mechanisms do not protect against newer, low-rate TCP-targeting pulsing at-

tacks. There have been some AQM-based proposals designed specifically to deal with pulsing DoS attacks [144, 133]. However, a common shortcoming of these proposals is that they rely on the validity of packet headers due to their use of flow-level information. This is an important issue, as DoS attack traffic can spoof packet headers easily, thus potentially affecting both the effectiveness and scalability of those proposals. There is clearly a need for a router-based, scalable, spoof-resistant DoS defence mechanism that does not have the limitations of existing proposals.

2.2 Active Queue Management

Active Queue Management denotes a class of algorithms designed to provide improved queueing mechanisms for routers. Research in this area was inspired by the original RED proposal [54] in 1993. These schemes are called *active* because they dynamically signal congestion to sources: either explicitly, by marking packets (e.g. Explicit Congestion Notification [50]) or implicitly, by dropping packets. This is in contrast to Drop-Tail queueing which is passive: packets are dropped if, and only if, the queue is full. The Internet Engineering Task Force (IETF) recommended the deployment of AQM in Internet routers in RFC 2309 [19]. The main motivations for this were cited as the improvement of performance due to smaller overall queueing delays, reduced packet drop rates and the prevention of synchronisation due to the drop-tail queueing process [19]. RFC 2309 also recommended the development of new queueing mechanisms to improve fairness on the best-effort based Internet, and this gave rise to a large body of work targeted at creating new AQM proposals.

Since this IETF recommendation, the main focus of the research community has been on the development of new AQM schemes. A cross-section of AQM algorithm proposals for best-effort IP traffic includes REM [7], ARED [52], AVQ [79], DRED [8], GREEN [141], PI [65], LDC [82], SVB [39], HRED [72], APACE [69] and PD [131]; to name but a few.

It is hard to estimate the state of AQM deployment today, as there is no tractable way to probe routers to determine if they implement some form of AQM. Moreover, ISP deployment information is commercially sensitive. While

RED (or some vendor-specific variant of RED, e.g. Cisco's WRED [36]) is available in most routers today, this is often disabled by default. It should be noted that RED has a number of drawbacks and, indeed, some even recommend against its deployment [97].

Almost all of the schemes mentioned above are presented as improvements on RED. While RED has clearly evolved from the original proposal (there are now gentle and adaptive [48] versions), other schemes propose alternate approaches to queue management in router buffers.

2.2.1 Random Early Detection (RED)

RED [54] is the AQM scheme that the IETF has recommended for deployment in Internet routers [19]. It is the seminal algorithm that drives current Active Queue Management research efforts. The motivations for, and approaches of, most of AQM schemes are, in some degree, derived from RED. Thus, RED is a very typical example of an AQM scheme.

The RED algorithm estimates the current congestion level on the network by keeping track of the average queue size. Sources are notified of congestion, typically through packet dropping or marking with ECN [50].

The following pseudo-code implements the general control flow applied to packets in a RED queue:

```
for each packet arrival:
    calculate the average queue size avg
    if min_th ≤ avg < max_th:
        calculate probability p_a
        with probability p_a:
            mark arriving packet
    else if max_th ≤ avg:
        mark arriving packet
```

RED uses the min_{th} and max_{th} parameters, which represent the boundaries for the average queue length avg within which RED will probabilistically mark arriving packets. The average queue length avg is updated at each packet arrival as follows:

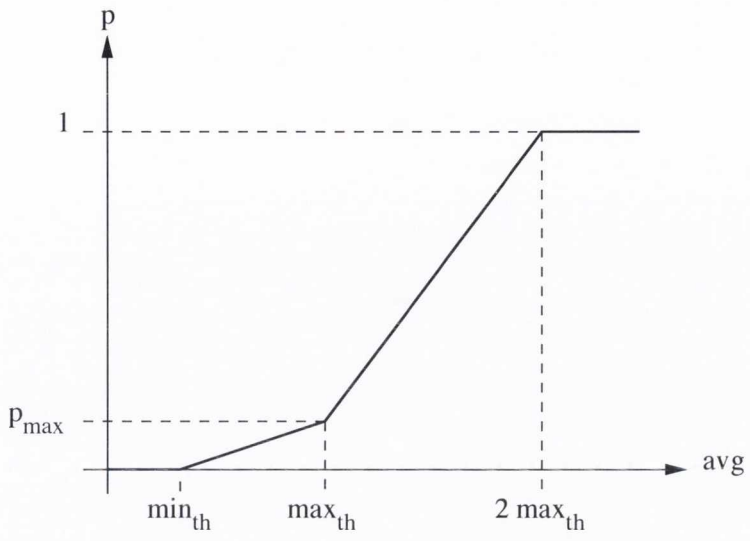


Figure 2.1: RED drop probability function (gentle version)

$$avg \leftarrow (1 - w_q)avg + w_q q \tag{2.1}$$

where q is the instantaneous queue length and w_q is a weighting parameter that affects how quickly avg tracks the instantaneous queue length.

The probability of notification is an increasing function of the average queue length. This average is computed using an exponential weighted moving average (EWMA), which acts as a first order low pass filter [103], so that short bursts of traffic do not affect the way RED drops packets. As avg varies from min_{th} to max_{th}, the intermediate packet marking probability p_b varies linearly from 0 to max_p, the maximum RED marking probability:

$$p_b = \frac{max_p(avg - min_{th})}{(max_{th} - min_{th})} \tag{2.2}$$

The final marking probability p_a increases slowly as the number of packets arrived since the last marked packet ($count$) increases:

$$p_a = \frac{p_b}{1 - (count \times p_b)} \tag{2.3}$$

The current most widely used version of the RED algorithm differs slightly from the original proposal [54]. The so-called *gentle* version of RED, shown in

figure 2.1, removes a discontinuity in the probability function when avg reaches max_{th}, increasing p_b to 1 as avg moves between max_{th} and $2 \times max_{th}$. Another improvement has been the development of better guidelines for RED parameter settings by the original authors [54].

RED's creators have acknowledged that their initial evaluation and design was incomplete [55] with regard to high-loss scenarios and the insufficient range of link bandwidths evaluated. These limitations have led to a lengthy literature on improvements to RED. Most of the post-RED AQM proposals share some common algorithmic aspects with RED:

- they are built on top of a drop-tail queue, if for some reason active dropping of packets fails to prevent full queues, packets are still dropped using the drop-tail queueing mechanism.
- they use at least one metric to estimate the current and/or incoming congestion levels of the network.
- the notification rate is computed as a function of the estimated congestion level.

2.2.2 Fairness-oriented AQM

While RED and many other proposed AQM schemes address some of the issues related to drop-tail queueing (such as global synchronisation); they do not prevent the problem of congestion collapse that may arise from the growth of non-congestion responsive traffic on the Internet [19, 51]. Flows that do not use TCP-compatible end-to-end congestion control can pose a threat to Internet performance because of their ability to obtain an unfair share of the network bandwidth. This is the key factor that enables flooding DoS attacks to impact on well behaved traffic.

Traditional AQM schemes have not prevented the impact of DoS attacks on responsive traffic. Luo et al. evaluated [89] several AQM schemes along with drop-tail queueing in scenarios where attacks occurred from both persistent and pulsating DoS traffic. Their results showed that the attack traffic pattern has a considerable impact on queueing scheme performance. AVQ [79] appears better than RED-like schemes for mitigating pulsing attacks, while RED-like

algorithms perform better in the case of severe flooding attacks. They also found that in the case of pulsing attacks, drop-tail queueing performs better than any of the AQM approaches considered.

One category of existing AQM proposals is those that have been designed to provide better fairness and protection for flows that use end-to-end congestion control. This is usually achieved by limiting the rate of unresponsive flows. A common characteristic of fairness-oriented AQM schemes is that they are designed to be lightweight; incorporating mechanisms that avoid costly per-flow tracking. Thus, these schemes typically use no per-flow tracking, allowing for their application on high-throughput backbone routers.

Seven different fairness-oriented AQM schemes will be described below: CHOKe [108], RED-PD [93], LRU-RED [115], SFB [47], SAC [70], CARE [30] and AFD [107].

All of these schemes share a common approach in that they use lightweight state and AQM-like techniques to identify and throttle aggressive traffic that does not conform to the behaviour expected from TCP congestion controlled traffic. However, the main limitations of these proposals is that even in the cases in which the maliciousness of the aggressive traffic is acknowledged, the schemes have a strong reliance on packet headers for flow and aggregate identification. This is a serious limitation as highly malicious distributed DoS traffic can easily spoof any packet header field, except the destination address, to subvert detection without decreasing its ability to reach the victim(s).

2.2.2.1 CHOKe

CHOKe [108] augments RED by providing a stateless mechanism for approximating fair bandwidth allocation to the traffic traversing a link. CHOKe extends the probabilistic dropping algorithm from RED as follows: an arriving packet is compared with a random packet drawn from the queue and they are both dropped if they belong to the same flow. If they are from different flows, the arriving packet is subjected to the standard RED drop probability. CHOKe is more effective against unresponsive flows when more than one candidate packet is compared for each packet arrival. The number of comparisons made can be adjusted depending on the congestion level.

The following pseudo-code implements the CHOKe algorithm when applied to packets arriving at the queue:

```
for each packet arrival:
    if avg ≤ min_th:
        admit packet
    else:
        draw random packet from queue
        if both packets from same flow:
            drop both packets
        else:
            if avg ≤ max_th:
                admit packet with probability p
            else:
                drop new packet
```

CHOKe maintains an average queue length avg using an exponential moving average of the instantaneous queue size, as used by RED in equation 2.1. The parameters min_{th} and max_{th} specify whether a packet will be unconditional accepted by CHOKe when $avg <= min_{th}$ or dropped when $avg > max_{th}$. The drop probability p applied to packets that do not match with random samples from the queue is calculated in exactly the same way as p_a in RED equation 2.3.

2.2.2.2 RED-PD

RED-PD [93] is a variant of RED that uses the RED packet drop history to identify high-bandwidth flows and then apply preferential dropping to them. This approach is based on the idea is that the RED drop history is a reasonably random sample of incoming traffic and that a flow with more drops in the past is very likely to have a higher sending rate.

RED-PD uses the following simplified version of Mathis' TCP throughput model [96] (see equation 2.10 below), which gives the throughput, T for round-trip time RTT and average loss probability p as:

$$T \approx \frac{\sqrt{1.5}}{RTT\sqrt{p}} \tag{2.4}$$

The *congestion epoch length* CL in RED-PD is defined as a period that should contain exactly one packet drop, and thus contains $\frac{1}{p}$ packets. From 2.4 and for a flow with round-trip time RTT and loss probability p the CL can be calculated as:

$$CL = \frac{1}{Tp} = \frac{RTT}{\sqrt{1.5p}} \tag{2.5}$$

Considering the packet drop history as a random sample of the traffic at the link, RED-PD keeps its packet drop history over $K \times CL$ seconds. It identifies flows with K or more drops in this history as sending at a rate larger than T. Instead of keeping a single drop history list, RED-PD partitions it in M consecutive lists, where the length of a list is:

$$\frac{K}{M}CL = \frac{K}{M}\frac{RTT}{\sqrt{1.5p}} \tag{2.6}$$

RED-PD identifies flows with losses in at least K of M lists. A drop probability is maintained, and periodically updated, for each of the monitored flows that appear in the packet drop history. This flow-specific dropping probability is applied before the RED drop decision is taken. This pre-filter provides relative fairness among the monitored flows and limits the bandwidth they obtain.

2.2.2.3 LRU-RED

LRU-RED [115] extends RED by keeping a cache of recently seen flows. It uses the Least-Recently-Used (LRU) replacement policy: For any packet arrival at the queue, its flow is added unconditionally to the cache if there is space to store it, while it will replace the least seen flow with probability p if the cache is full.

High-bandwidth flows are identified because they are more likely to remain in the cache. A packet count is associated with each flow in the cache. If the count associated with a flow's entry in the cache exceeds a given threshold, its RED drop probability is increased with the goal of penalising high bandwidth flows.

The final drop probability p_{drop} applied to arriving packets is calculated as

follows:

$$p_{drop} = p_{red} \frac{count}{threshold} \tag{2.7}$$

where p_{red} is the RED drop probability, *count* is the packet count associated with the flow and *threshold* is a parameter.

2.2.2.4 Stochastic Fair Blue (SFB)

Stochastic Fair Blue (SFB) [47] is an extended and improved version of the BLUE [31] algorithm. The BLUE algorithm uses packet loss and link utilisation indicators to maintain a single drop probability for all packets arriving at the queue.

SFB applies a similar mechanism to adapt the drop probability differently for every flow. Using a Bloom filter [17], it maps the different flows traversing the buffer to their respective drop probabilities. The pseudo-code for the SFB algorithm is:

```
B[1][n]: L x N array of bins
enqueue():
    Calculate hashes h_0, h_1, ..., h_{L-1}
    for i = 0 to L - 1:
        if B[i][h_i].qlen > bin_size:
            B[i][h_i].p_m += Δ
            drop packet
        else if B[i][h_i].qlen == 0:
            B[i][h_i].p_m -= Δ
    p_min = min(B[0][h_0].p_m .. B[L][h_L].p_m)
    if p_min == 1:
        ratelimit()
    else:
        drop with probability p_min
```

The principle behind the SFB algorithm is that a non-congestion responsive flow will increase p_m to 1 in all the L bins it is hashed into. Responsive flows may share a number of bins with non-responsive ones, up to the maximum of the number of non-responsive flows; however they will be mapped into at least

one bin that is not polluted by non-responsive flows. This approach relies on packet header information for flow identification, thus making it vulnerable to packet header spoofing.

2.2.2.5 Self Adjustable CHOKe (SAC)

Self Adjustable CHOKe (SAC) [70] is an extension of CHOKe that aims to improve on CHOKe. CHOKe causes the unfair treatment of different unresponsive flows as higher rate flows achieve lower throughput. Higher input rate flows tend to have most of their packets at the end of the queue, causing them to be over penalised by CHOKe. SAC is an improvement on this as, once a random CHOKe match has been done and it has been decided to drop packets from a given flow, it picks packets to drop from the tail of the queue .

Another CHOKe limitation that SAC improves upon is the selection of the parameter k that determines the number of samples chosen for comparison. SAC maintains a pair of moving average values: P determines if the incoming packet has the same flow number as the randomly selected one, and R is the probability that an incoming packet is part of a UDP flow. The value used for k is then updated using an exponential moving average:

$$k \leftarrow (1 - w_k)k + w_k \frac{2\sqrt{R}}{P^{0.75}} \tag{2.8}$$

where w_k is the weight parameter that specifies how quickly k tracks changes in P and R.

2.2.2.6 CARE

CARE [30] is an algorithm based on a CApture-REcapture model to approximate a fair distribution of the available bandwidth. The capture-recapture approach was originally used to estimate the number of animals in a population. Initially a random set of animals are captured, marked and released. A later time recapture of another random set allows the estimation of the population by looking at the proportion of marked animals in the recaptured set.

CARE estimates the amount of flows sending traffic at the router queue with the following process:

1. Select incoming packets with probability p_{cap} and store them in a "capture list".

2. Build a capture frequency data set based on the flow ID of packets in the "capture list".

3. Estimate the total number of flows at the queue by using the jackknife estimator [30].

The next metric CARE needs to estimate is the individual incoming packet rate for each flow. Based on the capture mechanism used for estimating the number of flows, CARE applies the following steps to estimate the rate for captured flows:

1. Select incoming packets with probability p_{cap} and store them in a "capture list".

2. Keep a count of the number of packets in the "capture list" for each flow ID.

3. Estimate the packet rate for each flow in the "capture list" given a virtual buffer size limit for the "capture list" buffer.

Once estimates of the number of flows and their incoming packet rates are calculated, CARE adjusts their throughput using the following equation:

$$rate_i \times (1 - d_i) = fairshare \tag{2.9}$$

where $rate_i$ is the incoming rate for flow i and d_i is the ideal dropping probability at the queue for the same flow. For each packet arrival, CARE applies the drop probability $d_i = 1 - \frac{fairshare}{rate_i}$.

2.2.2.7 Approximate Fair Dropping (AFD)

Approximate Fair Dropping (AFD) [107] builds on the RED model of probabilistic dropping of arriving packets, it bases the applied drop probability on the recent history of a packet's flow. This is based on the observation that the distribution of flow rates on the Internet is long-tailed, with most flows generating

quite low packet rates (*turtles*) while a few send most of the bytes at high packet rates (*cheetahs*).

Thus AFD takes advantage of the long-tailed distribution of Internet flow rates to maintain a rate history of the largest (highest rate) flows. This approach is manageable as the amount of information to be stored is proportional to the number of large flows, rather than to the total number of flows traversing the link. AFD applies probabilistic differential dropping to limit the throughput of the large flows to their fair share of the bandwidth.

The implementation of AFD includes a "shadow buffer" into which arriving packets are sampled with probability $\frac{1}{s}$ for a sampling interval of s. Thus, approximately one in s packets are sampled into the shadow buffer. If this buffer reaches its maximum size b, a newly sampled packet will cause the eviction of a randomly chosen packet from the existing set. The s and b parameters are critical to the performance of AFD and require fine-tuning to ensure that reasonable rate estimates are calculated without increasing the used memory space considerably.

2.2.2.8 Core-Stateless Fair Queueing (CSFQ)

Core-Stateless Fair queueing [129] is a router queueing scheme with the goal of approximating fair bandwidth allocation while minimising the need to maintain per-flow state information. The main novelty in CSFQ lies in its architecture rather than the details of the proposed algorithm. Its author's, Stoica et al. [129], acknowledged it to be an example of one of many possible options.

CSFQ divides all routers in a contiguous network section or *island* into *core* and *edge* routers. Edge routers maintain per-flow state, estimating incoming packet rates for flows and labelling packets with this estimate. Core routers do not maintain per-flow state, they apply FIFO queueing with the addition of probabilistic dropping that uses the packet labels and an estimate of the aggregate traffic at the router. CSFQ allocates bandwidth within the island of routers to be approximately fair.

2.2.2.9 Aggregate Congestion Control (ACC)

Aggregate Congestion Control (ACC) [92] is a proposal that utilises AQM ideas to design a system for DoS defence. ACC augments an existing RED-like AQM queueing scheme by using drop metrics from the AQM queue to identify high-bandwidth aggregates. A heuristic clustering algorithm is used to obtain the aggregate information from the dropped packets. The identified high-bandwidth aggregates are then rate-limited to approximate a fair bandwidth allocation.

The ACC architecture includes a second element that provides a mechanism to extend the rate-limiting of aggressive aggregates beyond the local router queue where the aggressive behaviour is experienced. *Pushback* is the mechanism by which the congested router requests adjacent upstream routers to rate-limit a given aggregate. This is implemented by sending *Pushback Messages* to the immediate upstream routers that generate most of the incoming traffic for an aggressive aggregate. Routers receiving these messages will throttle the aggregate's throughput and can recursively propagate this throttling to upstream routers.

The idea behind Pushback is that rate-limiting attack traffic close to the source prevents attack packets from wasting bandwidth when they will just be dropped closer to the destination at a later point in time. It also intends to minimise ACC's rate limiting impact on non-attack traffic. Pushback has the disadvantage of requiring cooperation between multiple routers, potentially under the administration of different organisations. Another limitation is that for highly distributed DoS attacks that use packet header spoofing ACC with Pushback is likely to be quite ineffective as the real attack sources are highly dispersed and can at the same time provide any arbitrary source information in the attack traffic, making it difficult for ACC to identify attack aggregates and their source networks.

2.3 TCP

The Transmission Control Protocol (TCP) is currently the dominant transport layer protocol on the Internet [25]. TCP provides the functionality of an ordered, full-duplex, congestion controlled and reliable end-to-end stream

of bytes between two entities or applications executing on hosts connected using the Internet Protocol (IP) [41]. TCP is built on top of the IP, a network layer best effort service that delivers individual packets from sources to destination. IP provides packet routing but does not provide any stream abstraction of flow control. TCP and IP form the Internet Protocol Suite, commonly called TCP/IP, when used together.

The main features the TCP protocol provides over basic IP packet communication are:

- De-multiplexing of communication between different applications running on the same host. These is implemented by the notion of "TCP ports" and allows applications to simultaneously establish multiple TCP connections with applications executing on remote hosts.
- Reliable and ordered delivery of information using send and receive buffers by using Acknowledgement (ACK) packet header fields to allow the receiver to signal what portion of the stream has been successfully received.
- Full-duplex communication, each fully initialised TCP connection provides two independent streams of bytes between the communicating applications.
- Flow control implemented using a sliding-window mechanism that allows the receiver to control the amount of data sent towards it. This prevents the sender from overloading the receiver with more data than it can actually process.
- Congestion control mechanisms that detect network congestion via duplicate ACKs and retransmission timeouts and prevent the sender from overloading the network. These congestion control mechanisms used by TCP play a crucial part in avoiding the congestion collapse of the Internet [51].

DoS traffic often uses its traffic generation behaviour to impact on TCP flows. It is able to affect its victims due to the fact that it does not use end-to-end congestion control mechanisms. Thus, a deep understanding of TCP congestion

responsive behaviour and dynamics is an important component of a system that intends to detect and defend against DoS attacks.

2.3.1 TCP Congestion Control

The TCP congestion control algorithms [4] were first introduced by Van Jacobson [68]. The approach taken by TCP congestion control is that sources determine how much capacity is available on the network and thus can be used without causing additional congestion. TCP uses an intertwined set of algorithms to determine the available capacity for each connection, in a manner that changes over time. These algorithms, which are Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery, work together as a whole to implement TCP congestion control.

Each TCP sender maintains a congestion window parameter (*cwnd*), with each receiver maintaining a receive window (*rwnd*). The congestion window *cwnd* determines the amount of data a sender can transmit into the network before receiving an acknowledgement (ACK), while the receiver's advertised window *rwnd* is a limit on the amount of outstanding data the receiver can receive. The minimum of *cwnd* and *rwnd* controls the rate of the TCP data transmission between the sender and the receiver.

At the beginning of a TCP connection, or after a timeout loss retransmission, TCP utilises the Slow Start algorithm to probe the network to determine the available capacity. During Slow Start, TCP increments *cwnd* by one segment size unit of bytes for each ACK received in acknowledgement of the receipt of new data. The Slow Start operating mode is stopped when either *cwnd* reaches the Slow Start Threshold parameter, *ssthresh*, or congestion is detected. TCP increases its *cwnd* sending window exponentially during the Slow Start phase.

A TCP sender enters Congestion Avoidance mode when $cwnd > ssthresh$. During Congestion Avoidance *cwnd* is incremented by one segment size per round-trip time (RTT). Congestion Avoidance is applied until congestion is detected, which causes a reset of *cwnd* to 1. After retransmitting the lost segment, TCP goes back to Slow Start, but with *ssthresh* adjusted so that $ssthresh = \max(\frac{\min(cwnd, rwnd)}{2}, 2 \times SMSS)$, where $SMSS$ is the sender's maximum segment size, or the number of bytes in a full TCP segment from the

sender. The behaviour of TCP senders during Congestion Avoidance takes the name of additive increase / multiplicative decrease (AIMD), which refers to the way in which *cwnd* is incremented and decremented while in Congestion Avoidance regime.

The initial proposal to add congestion control to TCP used timeouts to detect packet loss and trigger retransmissions. This caused long pauses for connections while they were waiting for the timer to expire. The Fast Retransmit algorithm was added to alleviate this issue. Fast Retransmit can sometimes trigger the retransmission of a lost packet sooner than the timeout mechanism would, but it does not replace the timer, just enhances TCP's responsiveness to loss. Fast Retransmit works by retransmitting a packet when three consecutive duplicate ACKs are received by the sender. The TCP receiver sends an ACK for each received packet. In the case of lost packets, the receiver will ACK packets arriving after the lost one with the sequence number up to the sequence number of the lost packet. This will cause the sender to receive multiple ACKs with the same sequence number if a packet is lost.

A final improvement used in TCP congestion control is Fast Recovery. When Fast Retransmit happens, the TCP sender does not reset *cwnd* to 1 and enter Slow Start. Instead, it uses Fast Recovery, setting *ssthresh* to $\max(\frac{\min(cwnd,rwnd)}{2}, 2 \times SMSS)$ and *cwnd* to $ssthresh + 3 \times SMSS$. From then on, the TCP sender operates the additive increase algorithm of Congestion Avoidance.

The four algorithms presented above form the core of the TCP congestion control mechanism. They collaborate to give TCP the ability to both use the available capacity and also control its transmission rate at times of congestion.

2.3.2 TCP modelling

TCP modelling efforts have historically focused on the steady-state behaviour of the TCP congestion-control algorithm. This is the phase in which a TCP flow operates mostly in the Congestion Avoidance mode, as is more applicable to long-lived TCP connections. A short survey of analytical TCP model proposals was provided by Khalifa et al. [75].

Mathis et al. provided a well known, and relatively simple, model [96] for the

steady-state throughput of long-lived TCP flows. Their model only considers the congestion avoidance regime of TCP and assumes no losses with timeout recovery occur. This model expects a constant packet loss probability p, with $\frac{1}{p}$ segments being sent between each loss event. This defines the classical TCP sawtooth pattern. The TCP throughput T is given as

$$T = \frac{MSS}{RTT}\frac{K}{\sqrt{p}} \tag{2.10}$$

where MSS is the maximum segment size, RTT is the round-trip time and K is a constant, with $K = \sqrt{\frac{3}{2}}$ for a periodic loss process with no delayed acknowledgements.

Padhye et al. [106] developed a model for the steady-state performance of bulk transfer TCP Reno [128] flows that provides better accuracy by using a stochastic model of TCP. The model focuses on the congestion avoidance phase of TCP flows. It is based on the concept of "rounds", with each round starting with the transmission of W packets, for a current congestion window size of W. Once the W packets are sent, no more packets will be sent until the first ACK is received, which marks the end of the first round and the beginning of the next. In this model the duration of a round is equal to the round-trip time between the two connection end points, and is assumed to be independent of the window size. It assumes a correlated loss model in which all segments of a round are lost when a loss event happens. This model approximates the TCP throughput as:

$$T = \min(\frac{Wm}{RTT}, \frac{1}{RTT\sqrt{\frac{2bp}{3}} + RTO_0 min(1, 3\sqrt{\frac{3bp}{8}})p(1+32p^2)}) \tag{2.11}$$

where Wm is the maximum window size, RTT is the average round-trip time, RTO_0 is the initial value of the retransmission timeout (RTO) timer, b is the number of rounds necessary to increment the window by one and p is the average loss probability experienced by packets in the flow.

Altman et al. [5] provided a model of steady-state TCP throughput under a stationary ergodic loss process. This proposal models the TCP dynamics by using the instantaneous transmission rate $X(t)$, defined by dividing the number of

packets in the window size divided by the RTT at time t. The TCP throughput T is then computed as the time average of the $X(t)$ process:

$$\frac{1}{RTT\sqrt{bp}}(1-\lambda_{TO}E[t_{TO}]) \times \sqrt{\frac{1+\nu}{(1-\nu)}+\frac{1}{2}C(0)+\sum_{k=0}^{\infty}\frac{1}{2^k}C(k)} \tag{2.12}$$

where λ_{TO} is the number of timeout losses per unit of time, $E[t_{TO}]$ is the average duration of the timeout period, $C(k) = (E[S_nS_{n+k}] - E[S_n]^2)/E[S_n]^2$ is the normalised covariance of the inter-loss duration S_n and ν is the factor by which *cwnd* is reduced when a loss occurs.

Analytic models for TCP data transfer latency, including Tahoe and SACK and better accounting for slow-start phase, were included in a model by Sikdar et al. [123]. This model provides an estimate of the latency of sending N segments:

$$\begin{aligned} E[L(N)] = \quad & E[L_{CE}] + E[L_{delack}] + E[L_{ml}(N)] + (1-p)^N E[L_{nl}(N)] \\ & + p(1-p)^{N-1}E[L_{sl}(N)] \end{aligned} \tag{2.13}$$

where $E[L_{CE}]$ is the latency caused by the initial three-way TCP handshake, $E[L_{ml}(N)]$ is the expected latency with M loss events, $E[L_{nl}(N)]$ is the expected latency when no losses occur and $E[L_{sl}(N)$ is the expected latency when a single loss occurs.

Other authors, like Cardwell et al. [25], have focused on predicting the latency of finite-length TCP transfers. Their model extends the one in [106] to account for the time taken for connection establishment and close down. They provide models for the TCP connection establishment phase as well as for the data transfer phase. The expected latency of the connection establishment phase $E[L_{CE}]$ is defined as:

$$E[L_{CE}] = RTT + RTO_0\left(\frac{1-p_r}{1-2p_r}+\frac{1-p_f}{1-2p_f}-2\right) \tag{2.14}$$

where p_f is the average packet loss probability in the forward path from the sender to the receiver and p_r is the loss rate in the reverse path. RTT is the

average round-trip time experienced by packets in the connection and RTO_0 is the initial value of the retransmission timeout timer. The data transfer phase latency is modelled by Cardwell et al. as:

$$E[L] = E[L_{ss}] + E[L_{loss}] + E[L_{ca}] + E[L_{delack}] \tag{2.15}$$

where L_{ss} is the latency incurred during the initial slow start, L_{loss} is the latency caused by the time-out or fast recovery packet loss that occurs at the end of the slow start period, L_{ca} is the expected time required to transfer the remaining segments after slow start, and L_{delack} is the cost of the first delayed ACK. Cardwell's model was evaluated using real traffic traces by Arlitt et al. [6].

Li et al. [86] divided the TCP connection data transfer phase into slow-start and packet loss sub-phases, with the aim of improving the modelling of small Web-like TCP flows. They based their model on work in [25].

Ehsan et al. [46] used a Markov Chain model to more accurately model the TCP congestion window after the first packet loss. Their model improved on the modelling of the congestion window evolution after the initial loss event and as it converges towards the steady-state value. Their model for TCP window size evolution is:

$$W_{avg}(t) = \begin{cases} r\frac{t}{RTT} & \text{if } t \leq t_0 \\ W_f + (W_c - W_f)|\lambda_2|^{\frac{t}{RTT}} & \text{if } t > t_0 \end{cases} \tag{2.16}$$

where $W_{avg}(t)$ is the average TCP congestion window value at time t; W_c is the expected congestion window when congestion avoidance starts; t_0, which is the time instant when $W_{avg}(t) = W_c$; W_f is the steady-state congestion window; r indicates the number of packets sent before an ACK is sent due to delayed ACKs; RTT is the average round-trip time experienced by the connection end-points and λ_2 is the rate at which the congestion window converges towards steady-state.

Downey provided [42] a complex, empirical TCP model based on observed behaviour in traces. They considered both independent, or exogenous, loss processes as well as endogenous processes that depend on TCP behaviour. Where

other TCP models only consider the congestion avoidance steady-state behaviour, their model included buffer-limited and self-clocking as modelled TCP steady-states. Buffer-limited flows have their sending rates controlled by the minimum of the sending and receiving buffer sizes instead of the congestion window. In self-clocking steady-state, the rate limiting factor is the bottleneck bandwidth instead of the congestion window.

With HTTP being a very important part of the current protocol mix on the Internet, there have been many efforts to model the type of TCP behaviour caused by Web traffic. Web traffic can be considered a variant of finite-length TCP, with short request-response type connections. Mellia et al. [98] provided a short lived TCP latency model which is a function of average loss rate and RTT, focusing on very short flows of less than 10 TCP segments. This is not a complete model, as it does not take into account longer flows and how they evolve towards steady-state.

The evaluation of models has receive much attention in the literature. Mitzenmacher et al. [104] evaluated Markov process-based analytical TCP transfer latency models. This work focused not only on predictions of the expected TCP latency, but also on the cumulative function of the latency under different loss models and lost acknowledgements. Barakat et al. [10] provided a complete overview and evaluation of TCP modelling issues, highlighting the impact of different loss process and the need to separate TCP window evolution from the network model used.

An alternative approach to the analytical TCP models described above is that provided by history-based methods of TCP behaviour prediction. In [64] the model provided by Padhye et al. [106] is compared to simple linear history-based predictors. Their results showed that history-based predictors can be more accurate, but require information on the history of previous transfers and are very dependant on the characteristics of the underlying path. Machine learning techniques were used by Mirza et al. [102] to design a history-based predictor that uses path properties like queueing delay, packet loss and available bandwidth to provide higher accuracy than predictors based on time series forecasting approaches.

The definition of TCP-friendly protocols and the minimum requirements

that non-TCP protocols must follow to coexist with TCP traffic has also been considered. Many applications, such as streaming media delivery, do not require the reliability provided by TCP. In the past, these applications have not used any congestion control or have incorporated their own approach, one example of this is RTP [121].

Examples of unreliable datagram transfer protocols that are TCP-friendly would be TFRC [53, 9] and DCCP [77]. TFRC applies TCP steady-state modelling equations to achieve TCP-friendliness on longer time scales, using smoother congestion control that does not necessarily guarantee TCP-friendliness in the short term. DCCP [77] is a more recent transport protocol designed to provide an unreliable flow of datagrams in tandem with a choice of TCP-friendly congestion control mechanisms.

Yang et al. studied the dynamic behaviour of the TCP-friendly protocols GAIMD, TFRC and TEAR [147]. They focused on the metrics of fairness, smoothness, responsiveness and aggressiveness in their analysis of the transient behaviour of the protocols. Their results show how these algorithms can provide similar, or better, fairness and smoothness to TCP, while being much lower on responsiveness and aggressiveness by comparison. Overall, these unreliable TCP-friendly protocols have yet to achieve wide scale deployment, with most multimedia transfers using raw UDP.

It is clear from this section that modelling the behaviour of TCP has been the focus of much research activity. There are multiple TCP model proposal with varying complexity and accuracy. However, there has been little effort towards providing simplified, but still accurate, models that may be deployed at Internet routers. Existing TCP models focus mainly on accuracy and thus require a large number of parameters and exhaustive knowledge of the environment in which the end hosts operate. Later chapters of this work will focus on adapting existing TCP modelling proposals to minimise the number of parameters required with a view to their use for the characterisation of traffic in Internet routers.

2.4 Conclusion

Denial of Service (DoS) attacks provide a challenging research problem. Any network based traffic generation process that intends to degrade the level of service of a given provider can be considered a DoS attack. This definition covers a wide range of traffic source behaviour, from single packets that exploit networking stack implementation vulnerabilities and crash victim hosts, to large scale distributed bandwidth flooding DoS attacks that overload traffic links towards the victim.

This chapter has provided an overview of the most relevant research areas that must be considered when addressing the problem of DoS attacks in general, and network router based defence mechanisms in particular. An overview of DoS attacks as well as detection and defence mechanism proposals, a discussion on network router queueing AQM algorithms and their application in the defence of congestion responsive TCP traffic, and finally an overview of the congestion control mechanisms used by TCP and existing efforts to model TCP behaviour and performance has been provided above.

DoS attack behaviour and strategies have been divided into two major groups: software exploit and bandwidth flooding DoS attacks. This classification refers to the core approach used by the malicious attacker to degrade the victim's performance. Software exploit DoS attacks take advantage of implementation vulnerabilities in specific application network stack implementations. On the other hand, bandwidth flooding DoS attacks take advantage of the congestion responsive and end-to-end nature of the TCP protocol to obtain an unfair share of the bandwidth available to access the victim, preventing legitimate client traffic from accessing the victim's services.

While still an open problem, a well defined mechanism of protection against software exploit DoS attacks involves improving the quality of software implementation and ensuring timely software updates are deployed on network connected computers. However, bandwidth flooding DoS attacks rely on core aspects of the design of TCP to cause their impact. This means they are not amenable to a solution based on end host software and are more suited to defence mechanisms that augment network routers and other intermediaries to provide improved defence for congestion responsive traffic. With this approach

in mind, this chapter provided a deep overview of the research efforts on DoS detection and defence mechanisms for bandwidth flooding DoS attacks.

Active Queue Management (AQM) is a class of network router queueing mechanisms that improves upon the historical FIFO drop-tail queueing used by Internet routers. AQM intends to provide improved performance, lower loss rates and better utilisation than drop-tail for congestion responsive TCP traffic. The focus of AQM schemes is on scalable, lightweight designs that do not require expensive per-flow information. This makes AQM principles in general, and specific schemes such as RED in particular, very well suited for use as a base on top of which a network router based DoS defence mechanisms could be implemented. This chapter provided an overview of AQM research and of key AQM proposals, from the seminal RED scheme to more recent proposals that intend to protect TCP traffic by approximating fair flow bandwidth allocation and penalising aggressive traffic.

The third main section in this chapter covers the TCP transport protocol congestion control algorithms and analytical modelling of TCP behaviour. Bandwidth flooding attacks take advantage of congestion responsiveness in TCP to impact their victims. Thus, understanding TCP congestion control and being able to predict its behaviour is highly relevant to the task of protecting TCP traffic from malicious DoS traffic that is not congestion responsive.

The background knowledge on the three areas of DoS, AQM and TCP given in this chapter form a base on which the following chapters of this dissertation will build to develop a proposal for an improved network queueing mechanism that protects TCP traffic from bandwidth flooding DoS attacks.

Chapter 3

Characterising TCP Flow Aggregates

Defending well behaved traffic from bandwidth flooding Denial of Service (DoS) attacks at network routers is the high level motivation for this dissertation. The first requirement for any system that intends to provide such a DoS defence is to be able to differentiate, with as high an accuracy as possible, between malicious DoS traffic and congestion responsive TCP traffic. However, a full characterisation of DoS traffic is an extremely difficult task as the space of possible attack traffic parameters is vast and the attackers are highly motivated to diversify their modus operandi so as to prevent detection and penalisation.

An alternative approach to this problem would be to focus on characterising congestion responsive TCP traffic. An accurate reference model for the expected behaviour of TCP traffic would provide the ability to prioritise and defend it when a given network link is considered to be under a DoS attack. The behaviour of TCP flows is considerably better defined than that of DoS traffic due to the existence of the relevant standards and common implementations. Thus, being able to characterise the expected behaviour of traffic composed of TCP flows is a first step towards protecting TCP traffic from malicious bandwidth flooding DoS attack traffic.

This chapter will introduce, and answer, the following core question: what is the expected generated packet rate by a traffic aggregate composed of finite TCP

flows in their data transfer phase? Being able to answer this question allows for the design of a scheme that will discern whether a given traffic aggregate conforms to expected TCP behaviour. While TCP flows also consist of flow setup and teardown phases, we focus on the data transfer phase as this is the phase where the majority of the bytes generated by the flow are transmitted.

In the following sections of this chapter, a method for obtaining the average packet rate based on existing TCP modelling efforts for a finite TCP connection will be given. This result will then be used to characterise the expected latency of single TCP data transfers as well as the expected generated packet rate of traffic aggregates composed of finite TCP flows. Finally, an evaluation study will be carried out to compare the results obtained using this methodology with the actual performance of TCP flow aggregates.

3.1 Overview and requirements

This section provides an overview of the area of TCP characterisation and of the concrete requirements for TCP behaviour prediction for DoS defence at the network router. As discussed in the previous chapter, an extensive research literature on analytical TCP modelling exists. However, when practical applications of those models are considered it is found that most router-based queueing schemes implement solutions based on simple models that use only a few TCP characteristics as parameters or else they assume infinite steady-state flows.

In the past, AQM schemes designed to defend congestion responsive TCP traffic, such as RED-PD [93], have used relatively simple TCP models limited to infinite TCP transfers [96, 106]. However, as real Internet traffic is seldom composed of infinite flows better predictive characterisations are required that apply the recent improvements on finite-length TCP flow modelling to the limited-information environment of a high-bandwidth Internet router.

It is not possible to decide on what could be considered a reference distribution of TCP connection durations on the Internet. However, it is correct to state that most TCP connections are finite TCP connections that do not necessarily achieve steady-state during their lifetime. This argument is true for the traditional wired connections and personal computers that have historically gen-

erated Internet traffic, but it becomes even more evident as Internet connected devices become smaller, wireless and in many cases intermittently connected.

Thus, when applying TCP modelling to characterise traffic at a network router, finite TCP connection models should provide better accuracy and provide closer models of the expected behaviour of actual traffic. However, due to the implementation complexity of some of the most recent TCP models, traditional infinite TCP steady-state models are used for these applications. There is a pressing need for accurate characterisation of traffic composed of finite TCP flows with a level of complexity that allows for relatively easy implementation at a network router.

This work is motivated by the use of TCP characterisation for DoS defence. In this case, the goal is to be able to accurately provide an upper bound on how *aggressive* TCP traffic sources can be. The concept of traffic aggressiveness can be considered as being composed of two aspects:

Offered load This is the classical definition of aggressiveness, in which a flow or aggregate is considered to be more aggressive as it increases its offered load, given in packets per second.

Session initiation The number of concurrent sessions initiated is a less common but still important aspect for considering the aggressiveness of a traffic source. When analysing TCP traffic this means that, everything else being equal, the aggressiveness of a source or traffic aggregate is considered to increase as it opens more concurrent TCP connections. This is because TCP congestion control operates at the level of individual TCP connections.

Thus, when analysing whether a given traffic aggregate should be considered more aggressive than the equivalent aggregate composed of congestion responsive TCP connections, a characterisation will need to provide upper bounds for the expected offered load and the number of TCP connections.

Another requirement for the desired TCP aggregate characterisation is that it should be able to provide results when no detailed per-flow information is available for the traffic aggregate under study. This is a requirement that would enable use of the characterisation on scalable high-bandwidth network router

implementations that do not have the resources to maintain expensive per-flow state.

Specifically, only the following set of per-aggregate information values are expected to be available as inputs to the characterisation algorithm:

- A lower bound for the average packet loss probability experienced.
- The average packet rate generated by the traffic sources in the aggregate.
- An estimate of the number of distinct active flows in the aggregate.
- The average round-trip time experienced by the sources in the aggregate.
- The expected average number of bytes transferred by each TCP connection.

Given this set of limited metrics available to the TCP characterisation algorithm, it should be able to provide answers to the following questions:

- What is the expected upper bound on the incoming packet rate generated by an aggregate of finite TCP connections?
- What is the maximum expected number of flows that should be concurrently active to generate the observed incoming packet rate under the measured conditions?

The answers to these two questions are key when considering whether a given traffic aggregate can be considered more *aggressive* than an aggregate of congestion responsive TCP connection under equivalent conditions.

The following sections in this chapter will develop a methodology based on analytical TCP modelling that will be able to provide answers to these questions. From this, a characterisation for the expected level of *aggressiveness* of TCP flow aggregates will be derived and made available to DoS defence mechanisms at network router queues.

3.2 TCP flow behaviour

This section will develop a method for predicting the expected generated packet rate for a finite TCP source during its data transfer period. Rather than the

prediction of steady-state TCP behaviour, the goal is to capture the behaviour of relatively short-lived TCP flows that do not necessarily achieve a steady-state in their lifetime. This prediction should be obtainable with the following parameters:

- The average loss probability, p, experienced by the connection.
- The average round-trip time, RTT, experienced by the TCP source.
- The number of bytes, b, the connection is expected to transfer to the destination.

The average packet rate during the data transfer period of a finite TCP connection can be modelled as

$$B = \frac{W_{avg}}{RTT} \tag{3.1}$$

where W_{avg} is the average value of the congestion window maintained by the TCP source during the data transfer and RTT is the round-trip time experienced by the source in seconds. Variations due to queueing delays or other network conditions are not considered and so RTT may be approximated as being constant during the flow's lifetime.

Each TCP connection, or flow, can be divided into three phases: connection establishment, data transfer and connection tear-down. For the purposes of this work only the data transfer period is significant as this is where the majority of network traffic is generated by TCP flows. The connection establishment and tear-down phases add considerable latency to the duration of a small TCP connection, however the flow does not generate much traffic during these phases. Thus, excluding these phases will not have a negative impact on the accuracy of the model as its focus is on the rates generated by the flows during data transfer.

The primary mechanism used by TCP flows to regulate their offered load during the data transfer phase is the "congestion window" (*cwnd*) value that is maintained by TCP senders. During the transfer period of each connection, TCP data senders update *cwnd* periodically. The value of the window determines how many TCP segments (packets) the source will send in each round, where round is defined as a period of the size of a single round-trip time (RTT).

Given that RTT is an input parameter to the prediction algorithm, a result for the average packet rate B equation 3.1 can be obtained by estimating the value of the average congestion window, W_{avg}, during the flow's lifetime. The average congestion window W_{avg} can also be interpreted as the average packet rate, in packets per RTT, obtained by the flow,

$$W_{avg} = \frac{d_p}{L} \tag{3.2}$$

where d_p represents the amount of data (packets) sent by the connection during its data transfer period and L is the latency in RTTs of transferring the data.

Both the actual number of packets sent and the latency depend on the loss probability experienced by the flow during its data transfer period. Higher packet losses require the TCP source to retransmit the lost packets, which increases the actual number of packets transmitted by the source and the latency incurred by all the transferred data to reach the destination.

Given equation 3.1 for the packet rate of a single finite TCP flow during its data transfer period, if the average congestion window W_{avg} from equation 3.2 the average packet rate can also be obtained as:

$$B = \frac{W_{avg}}{RTT} = \frac{d_p}{RTT \times L} \tag{3.3}$$

where RTT is the parameter for the expected round-trip time experienced by the flow, d_p is the number packets sent with the experienced packet loss probability and L is the latency to complete the data transfer. The following subsections will address the structure of the TCP connection data transfer period and methods for obtaining estimates of both d_p and L.

3.2.1 TCP data transfer period

The duration of the data transfer period for a finite TCP connection has an inherent structure that is linked to how the TCP sender adapts its congestion window *cwnd*. Given that *cwnd* specifies how many TCP segments or packets the source sends for each round-trip time, its evolution affects directly both the packet generation rate as well as the incurred latency by the TCP transfer.

Structurally, a TCP data transfer can be divided in two distinct phases:

Slow Start This is the initial phase of the TCP data transfer. During Slow Start the TCP source grows its congestion window *cwnd* exponentially. This phase ends when either the sender detects a packet loss, *cwnd* reaches the *ssthresh* value or the data transfer completes.

Congestion Avoidance This phase is started when the TCP sender detects an initial loss event via duplicate ACK packets or the retransmission time-out mechanism. During Congestion Avoidance the TCP sender adapts its *cwnd* using an additive increase / multiplicative decrease (AIMD) algorithm in which the congestion window grows much slower and eventually converges to a steady-state that is mainly dependant on the average loss probability and the experienced round-trip time.

Historically, TCP modelling efforts have focused on providing analytical predictions to the performance of infinite TCP flows that achieve the steady-state [5, 96, 106]. These models do not consider the Slow Start phase, as the impact of this phase is negligible for infinite flows which by definition spend the overwhelming majority of their time in Congestion Avoidance. However, the Slow Start phase must be considered when the goal is to characterise TCP transfers of arbitrary sizes which have a finite number of bytes to transfer and may actually never even reach Congestion Avoidance.

The practical TCP characterisation discussed in this chapter is based on more recent modelling efforts that consider finite TCP connections as opposed to infinite ones. The work on this chapter draws from the transient model proposed in [46] and the study of TCP latency in [25]. These efforts follow the division provided above in which finite TCP data transfers are composed of distinct Slow Start and Congestion Avoidance phases.

It must be noted that the characterisation developed herein assumes there is no congestion, and thus packet loss, occurring on the reverse path from destination to source. This scenario is the most common on actual networks given the fact that the reverse path traffic consists mainly of ACK packets and their size in bytes will be orders of magnitude smaller than data payload packets.

3.2.2 Packets sent

The average packet rate generated by a single, finite TCP flow during its data transfer lifetime is dependant on the number of packets that the TCP source must send to achieve the reception of all the data to be transferred to the TCP receiver. This value is a major component of the average packet rate together with the expected latency L of completing the transfer, as introduced in equation 3.3.

This section addresses the prediction of the number of packets that a given TCP source will generate. This estimate will be composed of the sum of the number of packets sent during the Slow Start and Congestion Avoidance phases. The input parameters for this prediction will be the average experienced packet loss (p) and the number of bytes to be transferred (b).

3.2.2.1 Packets sent in total

This section describes a method for predicting the number of actual packets a TCP sender is expected to generate during its data transfer period. Intuitively, this number will increase as the packet loss experienced by the flow during its lifetime increases. This is due to the fact that the TCP source will need to retransmit dropped packets due to packet loss in the path between the sender and the receiver.

Let $P(n = 1) = 1 - p$ be the probability that only one packet is actually sent; $P(n = 2) = p(1 - p)$ the probability that exactly two packets are sent and so on successively. Thus we have $P(n = k) = p^{k-1}(1 - p)$ as the probability of exactly k packets being sent. This is a probability distribution as the sum of all P for all possible values of k equals 1:

$$\begin{aligned}\sum_{k\geq 1} P(n=k) &= \sum_{k\geq 1} p^{k-1}(1-p) \\ &= (1-p)\sum_{k\geq 1} p^{k-1} \\ &= (1-p)\sum_{k\geq 0} p^{k} \\ &= \frac{1-p}{1-p} \\ &= 1\end{aligned}$$

Let A be the average number of times each data packet will have to be sent, then

$$\begin{aligned}A &= \sum_{k\geq 1} k\, P(n=k) \\ &= \sum_{k\geq 1} k\, p^{k-1}(1-p) \\ &= (1-p)\sum_{k\geq 1} k\, p^{k-1} \\ &= \sum_{k\geq 1} k\, p^{k-1} - \sum_{k\geq 1} k\, p^{k} \\ &= \sum_{k\geq 0} (k+1)\, p^{k} - \sum_{k\geq 0} k\, p^{k} \\ &= \sum_{k\geq 0} p^{k}\end{aligned}$$

with the latter converging to $A = \frac{1}{1-p}$ for $-1 < p < 1$ [23].

Given this estimate of the average number of times a packet is retransmitted, A, and the number of packets that would need to be transmitted without packet loss, d, the average number of packets sent by a flow, d_p, is found to be:

$$d_p = d \times A = \frac{\lceil \frac{b}{S} \rceil}{1-p} \tag{3.4}$$

This result for d_p provides the total number of packets sent during the data transfer phase of a finite TCP flow that needs to retransmit b bytes in packets of maximum payload size S bytes, with average packet loss probability p.

3.2.2.2 Packets sent in Slow Start

As described in the previous sections, the data transfer phase of a finite TCP connection is divided into two distinct phases from a flow control point of view. During the initial Slow Start phase in which the TCP congestion window grows exponentially, the estimated number of packets sent is given in [46] as:

$$d_{ss} = \begin{cases} 1 + \frac{(1-(1-p)^d)(1-p)}{p} & \text{if } p > 0 \\ d & \text{if } p = 0 \end{cases}$$

where d is the number of packets needed to deliver the connection data without including retransmits and p is the random loss probability suffered by the flow. Considering a TCP connection that wishes to transfer b bytes with a maximum segment size of S bytes, the final equation for the expected number of packets sent in Slow Start is:

$$d_{ss} = \begin{cases} 1 + \frac{(1-(1-p)^{\lceil \frac{b}{S} \rceil})(1-p)}{p} & \text{if } p > 0 \\ \lceil \frac{b}{S} \rceil & \text{if } p = 0 \end{cases} \tag{3.5}$$

3.2.2.3 Packets sent in Congestion Avoidance

The expected total number of packets sent during the transfer period of a finite TCP flow has been given in equation 3.4. This total number is split between the two phases that the transfer can be divided upon, the Slow Start phase and the Congestion Avoidance phase. The expected number of packets in Slow Start was given in the previous section in equation 3.5.

These results allow the estimation of the amount of packets sent during the Congestion Avoidance phase. The total number of packets sent, d_p, can be expressed as $d_p = d_{ss} + d_{ca}$, where d_{ss} is the number of packets sent in Slow Start and d_{ca} is the number of packets sent in Congestion Avoidance. Using the existing equations for d_p (3.4)and d_{ss} (3.5) an estimate for the amount of packets sent in Congestion Avoidance can be given as:

$$d_{ca} = d_p - d_{ss} = \begin{cases} \frac{\lceil \frac{b}{S} \rceil}{1-p} - 1 - \frac{(1-(1-p)^{\lceil \frac{b}{S} \rceil})(1-p)}{p} & \text{if } p > 0 \\ 0 & \text{if } p = 0 \end{cases} \tag{3.6}$$

where p is the average loss probability, b is the number of information bytes that the destination must receive from the sender and S is the maximum packet size.

3.2.3 Latency

This section will provide a method to estimate the latency of the data transfer period of a finite TCP connection. This latency estimation together with the expression for the number of packets sent, d_p, given in the previous section will allow for the calculation of the expected average packet rate for a given TCP data transfer using equation 3.3.

The approach developed in this section to estimate the total transfer latency L in round-trip time (RTT) units combines elements of both the transient model proposed in [46] and the study of TCP latency in [25]. The structure of a finite TCP connection's congestion control mechanism during its data transfer phase is composed of primary phases: Slow Start and Congestion Avoidance. The event that causes the flow to transition from the slow-start to the congestion avoidance window regime is the detection of the first packet drop.

Three different components can be considered as contributing to the TCP connection data transfer latency:

Slow Start This is the latency incurred for all the packets that were delivered during the Slow Start phase of the flow in which the congestion window grows exponentially.

First Loss This is the latency impact of the recovery from the first packet loss experienced. This recovery can happen either by retransmission timeout or by fast recovery TCP mechanisms.

Congestion Avoidance This is the latency incurred on transferring all the remaining information after the flow has suffered the initial packet loss event.

A complete continuous congestion window evolution model is derived based on these three phases. This complete congestion window model can then be used to predict TCP connection data transfer latency, and with it the average congestion window W_{avg} of the flow.

More formally, let L be the data transfer latency, measured in RTTs, of a finite length TCP connection. The latency, L, can be defined as:

$$L = L_{ss} + L_{t_0} + L_{ca} \tag{3.7}$$

where L_{ss} is the latency of transferring the packets sent during the Slow Start phase, L_{t_0} is the expected latency cost of the first packet loss event and L_{ca} is the latency to transfer any packet left after the first loss in the Congestion Avoidance regime.

3.2.3.1 Slow Start latency

This section will provide a method to estimate the Slow Start latency L_{ss}, which is the time in RTTs that the TCP flow spends transmitting data during the initial phase of the connection, when the congestion window is increased exponentially. This phase will finish when either all the data to transfer has been delivered to the receiver or the first packet loss event occurs.

The TCP congestion window evolution in this phase can be expressed as:

$$W_{ss}(t) = W_i \times r^t \tag{3.8}$$

where W_i is the initial value of $cwnd$ used by the flow (typically $1 \leq cwnd \leq 3$); $r = 1 + \frac{1}{b}$, with b indicating the number of packets received before an ACK is sent ($b > 1$ when delayed ACKs are being used), and t is the time elapsed since the beginning of the transfer.

The number of packets already sent by the flow in slow-start at time t_n can be calculated by integrating W_{ss} from 0 to t_n:

$$D_{ss}(t) = \int_0^{t_n} W_{ss} = \frac{(r^{t_n} - 1)W_i}{\ln r} \tag{3.9}$$

From this we can estimate the number of RTTs needed to send s packets in slow-start as:

$$R(s) = log_r \left(1 + \frac{s \ln r}{W_i}\right) \tag{3.10}$$

$R(s)$ provides a prediction of the latency of the initial slow start phase of a TCP connection data transfer that is dependant on the number of packets sent

during slow start, s. For TCP connections this represents either all the packets of the flow or the number of packets sent before a loss event occurs, whichever is smaller. Equation 3.10 for $R(s)$ also depends on the parameters r and W_i, which can be set globally to values used by standard TCP implementations.

Let t_0 be defined as the time instant relative to the beginning of the data transfer period in which the first packet loss event occurs. This value can also be understood as the time required from the start of the connection to transfer d_{ss} packets, which is the number of packets sent in slow start given in equation 3.5. From equation 3.5 and equation 3.10 it can be seen that the time t_0 in round-trip times at which d_{ss} packets have been sent in slow start can be calculated as follows:

$$t_0 = R(d_{ss}) = \begin{cases} log_r \left(1 + \frac{(1-(1-p)^{\lceil \frac{b}{S} \rceil})(1-p) \ln r}{p \times W_i} \right) & \text{if } p > 0 \\ log_r \left(1 + \frac{\lceil \frac{b}{S} \rceil \ln r}{W_i} \right) & \text{if } p = 0 \end{cases} \tag{3.11}$$

Additionally, the value of the TCP congestion window *cwnd* at the end of slow start can be obtained using equation 3.8 as

$$W_{to} = W_{ss}(t_0) = W_i \times r^{t_0} \tag{3.12}$$

The t_0 of the end of the Slow Start phase is effectively the latency incurred by the TCP sender in Slow Start, L_{ss}. Thus, from equation 3.11:

$$L_{ss} = t_0 = \begin{cases} log_r \left(1 + \frac{(1-(1-p)^{\lceil \frac{b}{S} \rceil})(1-p) \ln r}{p \times W_i} \right) & \text{if } p > 0 \\ log_r \left(1 + \frac{\lceil \frac{b}{S} \rceil \ln r}{W_i} \right) & \text{if } p = 0 \end{cases} \tag{3.13}$$

This latency L_{ss} denotes either the time to send all packets in slow-start (if there is no loss) or the time at which the first packet loss occurs (when the flow moves from the slow-start to the congestion avoidance phase).

3.2.3.2 First packet loss latency

The next component of the total TCP data transfer period L (as introduced in equation 3.7) is the latency impact of the recovery from the first packet loss,

L_{t_0}. This initial loss latency impact can be estimated using the approach in [25] as:

$$L_{t_0} = \frac{l_{ss} \times [(Q(p, W_{ss}) \times Z^{TO} + (1 - Q(p, W_{ss}) \times RTT))]}{RTT} \tag{3.14}$$

where $Q(p, w)$ is the probability that the initial packet loss will be detected with a retransmission timeout and $Z^{TO}$is the average cost of a timeout loss in seconds. From [25] both $Q(p, w)$ and Z^{TO}may be estimated as:

$$Q(p, w) = min(1, \frac{(1 + (1 - p)^3(1 - (1 - (1 - p)^{w-3}))}{(1 - (1 - p)^w)/(1 - (1 - p)^3)}) \tag{3.15}$$

$$Z^{TO} = \frac{T_0 \times (1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6)}{1 - p} \tag{3.16}$$

where T_0 is the average duration of the initial timeout in a sequence of one or more timeouts.

3.2.3.3 Congestion Avoidance latency

Of the three components of the total TCP data transfer latency L, discussed above, estimation equations for the Slow Start and first packet loss latencies, L_{ss} and L_{t_0}, have already been provided. The third component needed to be able to fully estimate L is the congestion avoidance latency, L_{ca}; the time needed to send those packets not sent during the slow-start phase (if any).

Congestion window evolution Once the first loss occurs, the TCP sender initiates a phase in which *cwnd* converges to a steady-state value W_{st}. The actual congestion window used by TCP does not instantaneously become W_{st} after the packet drop, rather there is a transient period in which the congestion window converges towards W_{st}[46] from its value at the time of the first packet loss (W_{to} given in equation 3.12).

The effect of this transient phase can go unnoticed for very small or very large transfer sizes, but it is important for small to mid-size transfers that are long enough to suffer packet losses but finish before the window has converged to W_{st}. The evolution of the congestion window after the first loss can thus be

expressed as

$$W_{ca}(t) = W_{st} + (W_{t_0} - W_{st})|\Delta_2|^{t-t_0} \tag{3.17}$$

where $W_{ca}(t)$ represents the value taken by the average congestion window at time t, in RTTs, from the beginning of the transfer, t_0 is the time of the first loss event in RTTs, W_{t_0} is the congestion window at the time of the first loss and Δ_2 is the convergence rate of the system to the steady state window W_{st}.

The steady-state congestion window model used herein is derived from the one proposed in [106]. This model provides a prediction of the TCP transfer rate as

$$B_{st} = \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 min(1, 3\sqrt{\frac{3bp}{8}})p(1 + 32p^2)} \tag{3.18}$$

where B_{st} is the estimated steady-state packet rate generated by the flow, RTT is the round-trip time, p is the loss probability and T_0 is the initial value of the retransmit timeout. This model takes into account both the triple-duplicate ACK losses as well as losses detected with timeouts.

The consideration of both kinds of loss detection mechanism is an important characteristic of this model. Relatively short and low-bandwidth TCP flows that suffer correlated losses will typically not be able to provide three duplicate ACKs for the loss detection due to the low number of packets that actually reach the receiver. Thus, the sender will frequently detect packet losses using the timeout mechanism built in the TCP implementation, which considers that a packet loss has occurred after a given segment or packet has not been ACKed for longer than a given T_0 retransmission timeout period.

Given the constraints introduced due to the fact that we are operating in an environment with limited information and resources available for the model, it would be desirable to have an equation for W_{st} that depends only on the loss probability p. With this goal in mind, the equation can be further simplified to remove the T_0 and RTT parameters.

An approximation for the initial retransmission timeout value of $T_0 = 0.1$ may be used. The RTT parameter for the special case of calculating W_{st} may also be fixed at $RTT = 0.2$. This is a reasonable value given that Internet

round-trip times are typically between 50 and 500 ms. These approximations allow us to obtain an expression for the steady-state window that only depends on the loss probability p. It is noted that even with these modifications, this equation provides a more accurate model of the system than that obtained by excluding the timeout loss component.

Given the above simplifications, the final equation for the steady-state Congestion Avoidance value for the congestion window *cwnd* may be calculated as:

$$W_{st} = \frac{1}{\sqrt{\frac{2bp}{3}} + \frac{min(1,3\sqrt{\frac{3bp}{8}})p(1+32p^2)}{2}} \tag{3.19}$$

This value for W_{st} can be substituted into equation 3.17 for W_{ca} to obtain the predicted value of the congestion window at time t. The rate of convergence of W_{ca} towards W_{st} is obtained from a transition probability matrix, P, populated with the congestion window state change probabilities for all possible window values. The system described by P converges to steady state with rate $|\Delta_2|$, where Δ_2 is the second largest absolute eigenvalue of P [46].

A complete equation for the evolution of the congestion window during the transfer period of a TCP flow is given below. We can estimate the values taken by the congestion window of a TCP flow during its data transfer phase by 3.8 during the slow-start phase and 3.17 after the first packet loss [46] has occurred. Thus, the combined model for the evolution of the congestion window of a TCP flow is as follows:

$$W(t) = \begin{cases} W_i \times r^t & \text{if } t \leq t_0 \\ W_{st} + (W_{t_0} - W_{st})|\Delta_2|^{t-t_0} & \text{if } t > t_0 \end{cases} \tag{3.20}$$

This equation is not of direct relevance to the work described herein but is included here for completeness.

Latency The number of packets sent in the congestion avoidance phase is $d_{ca} = d_p - d_{ss}$, where d_p is the total number of packets sent, including retransmissions, as given in equation 3.4, and d_{ss} is number of packets sent in the Slow Start phase, given in equation 3.5 above.

To calculate the latency in the Congestion Avoidance phase, L_{ca}, we first cal-

culate the number packets sent in Congestion Avoidance up to time t in RTTs from the beginning of the data transfer period. This is obtained from integrating the congestion avoidance window value W_{ca} (see equation 3.17 above) over the appropriate time interval. The following equation estimates the number of packets sent in congestion avoidance at time t:

$$D_{ca}(t) = \int_0^{t-t_0} W_{ca}(t)dt = tW_{st} + \frac{(W_{t_0} - W_{st})(|\Delta_2|^{t-t_0} - 1)}{\log(|\Delta_2|)} \tag{3.21}$$

The value for the amount of packets sent in congestion avoidance d_{ca} given in equation 3.6 can be substituted for D_{ca} in the previous equation. Solving for t then gives an estimate for the congestion avoidance latency:

$$L_{ca} = \begin{cases} \frac{(W_{t_0} - W_{st}) + d_{ca}\log(|\Delta_2|) - W_{st}W(\frac{(W_{t_0} - W_{st})|\Delta_2|^{\frac{(W_{t_0} - W_{st}) + d_{ca}log(|\Delta_2|)}{W_{st}log(|\Delta_2|)}}}{W_{st}})}{W_{st}\log(|\Delta_2|)} & \text{if } p > 0 \\ 0 & \text{if } p = 0 \end{cases} \tag{3.22}$$

where W is the Lambert W function [24].

From equation 3.22 we can obtain the Congestion Avoidance latency L_{ca}, and so we can now estimate all three components of the TCP latency in equation 3.7: $L = L_{ss} + L_{t_0} + L_{ca}$. Thus, a full characterisation of the TCP transfer latency has been provided. Estimation equations have been derived for the Slow Start latency L_{ss} (equation 3.13), first packet loss event latency L_{t_0} (equation 3.14) and Congestion Avoidance latency L_{ca} (equation 3.22). L provides an accurate estimator for the latency incurred by finite length TCP connections during their data transfer period.

3.3 TCP aggregate behaviour

The previous section develops a full characterisation of the data transfer behaviour of a single TCP flow. An estimator has been provided for the average packet rate of a flow that is expected to transfer a fixed amount of bytes, as well as related equations for the expected number of packets generated and the

latency of transmission.

This characterisation of a single flow provides an accurate view of the behaviour of a TCP connection during the data transfer period. This is the core component of our system to characterise traffic aggregates that are composed of multiple TCP flows.

The average generated packet rate expected from a finite size TCP connection, B, was introduced in equation 3.3. This result, together with basic metrics about specific traffic aggregates, allows for the study of those aggregates from the point of view of their *aggressiveness* compared to equivalent TCP flows with the expected characteristics.

Let I be the total incoming packet rate generated by a traffic aggregate, and F be the estimated number of flows that are actively transferring data in the aggregate. Using predefined parameters for the average number of bytes transferred by each flow, b, and the average round-trip time experienced by the sources, RTT, the traffic aggregate can readily be compared to the expected behaviour of similar aggregates that are composed of finite TCP flows.

The expected upper bound for the incoming rate for the aggregate using the reference TCP characterisation would be:

$$E[I] = F \times B$$

where F is the measured number of flows in the traffic aggregate and B is the predicted average packet rate for individual flows in the aggregate.

The expected upper bound for the number of flows that compose the traffic aggregate would be:

$$E[F] = \frac{I}{B}$$

where I is the measured generated packet rate by the aggregate and B is, as before, the predicted per-flow average packet rate.

The calculation of $E[I]$ and $E[F]$ provides the ability to reason about a traffic aggregate's *aggressiveness* when compared to TCP traffic. Those two metrics provide upper bound definitions for the expected generated packet rate and created TCP flows respectively. This provides the ability of characterising specific traffic aggregates as showing an aggressive behaviour when either $I >$

$E[I]$ or $F > E[F]$.

This characterisation of the *aggressiveness* of traffic aggregates will form the basis of the malicious DoS attack detection and defence queueing scheme introduced in the next chapter.

3.4 Evaluation

This section will provide a simulation based evaluation of the TCP flow and aggregate characterisation developed in the previous two sections. The main aspects evaluated are the TCP latency prediction, the estimate of the number of packets generated and the effectiveness of the method to provide an upper bound packet rate for traffic aggregates composed of finite length TCP flows.

It is important to stress here that the ultimate goal for this TCP characterisation method is to be able to provide an effective mechanism by which a given traffic aggregate composed of multiple flows can be compared to reference aggregates composed of well-behaved congestion responsive TCP flows.

The evaluations carried out in this section have all been executed using an Evaluation Framework [15, 16] originally designed for the evaluation of Active Queue Management (AQM) schemes. This framework is built on top of the NS-2 Network Simulator and provides several facilities that improve the productivity and accuracy of simulation-based evaluation studies. The framework is accessed through a very high level Python [134] API that allows easy configuration of network topology scenarios, traffic mixes and the calculation of metrics. It also provides the ability to coarsely parallelise multiple simulation executions on a cluster of networked computers.

The simulations were run on a cluster of Intel processor based Linux PCs running the Linux kernel version 2.4. The core simulator used was NS-2 Network Simulator [20] version 2.29. The simulated TCP model used in NS-2 is the built-in FullTCP implementation. The TCP variant used included Selective Acknowledgements (TCP SACK [95]) in the TCP implementation within NS-2.

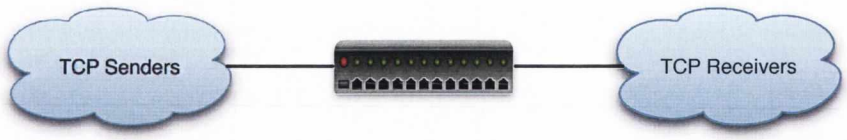


Figure 3.1: Evaluation network topology

3.4.1 Evaluation scenarios

This section describes the core scenarios used in this evaluation study of the TCP predictive characterisation framework. Common aspects of the different evaluation experiments will be discussed below, with configuration details specific to given simulation runs being specified together with their results in subsequent sections.

A single-bottleneck "dumbbell" topology was used for the simulations, as shown in figure 3.1. This topology consists of a set of 100 traffic source nodes paired with another set of 100 traffic receivers. The senders transfer data to the receivers using TCP through the bottleneck router. This node is the core router where measurements will be taken and actual flow behaviour will be compared to model predictions. The link propagation delay that determines the end-to-end round-trip time (RTT) experienced by the TCP senders and receivers is configured on a per-experiment basis.

The traffic mix used for all simulation experiments consists of 100 closed-loop On-Off TCP sources. Each one of these sources is configured to alternate active (ON) periods with inactive (OFF) periods. When entering one active period, sources create a new TCP connection to send ON_b bytes, with the active period finishing as soon as the data transfer has completed. Each active period is followed by an inactive OFF period in which the sources wait OFF_t seconds without generating any traffic. Once these inactive time finishes sources start a new active ON period. By specifying the active period length ON_b in bytes instead of seconds, the desired closed-loop behaviour of the scenario is achieved; as the rate of new connection establishment by the sources will depend on the packet rates achieved by them during the ON periods. The default source configuration defines $ON_b = 100000$ bytes and an OFF_t value that is constant

but dependant on the specific needs of the experiment being carried out.

3.4.2 Latency

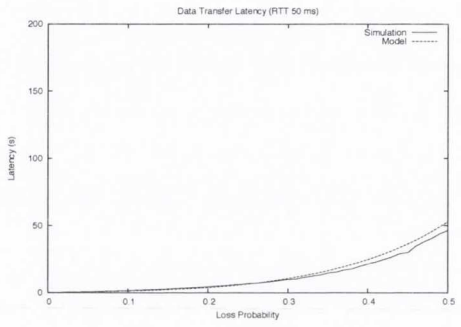
In this section the expected latency of finite length TCP flows derived in equation 3.7 is evaluated. Predictions obtained are compared to simulated scenarios with 100 On-Off TCP sources with a wide range of $RTTs$ and loss probabilities. Figures 1 to 6 compare the predictions with the average data transfer latency experienced by the simulated flows. An inter-connection time of $OFF_t = 10$ seconds is used for these simulation scenarios with the rest of the parameters set as specified above.

Each scenario is configured with links that provide a fixed RTT between sources and destinations. This RTT value is the main difference between the result graphs. Within each graph, the TCP transfer latency results are compared for the TCP connections and the predictive model. The latency values in each graph are the result of multiple simulation executions with different constant bottleneck queue drop probabilities between $p = 0$ and $p = 0.5$.

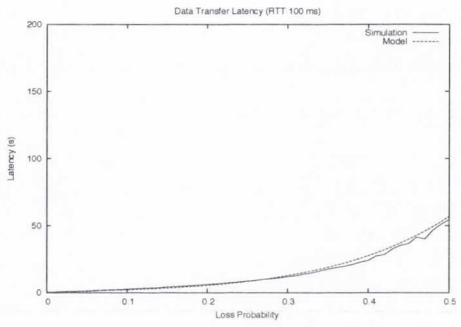
From the graphs presented in Figure 3.2 it is clear that the TCP characterisation proposed is a good predictor of the transfer latency experienced by the TCP connections. The prediction slightly under-estimates the latency for lower loss probabilities and for most RTT values. Also, it over-estimates the latency experienced by flows under higher loss probabilities, especially in the experiments that have $RTT = 400$ and $RTT = 500$ milliseconds. These scenarios with loss probability approaching $p = 0.5$ and RTT close to half a second are extreme situations that are not typical in any well operated network.

3.4.3 Packets sent

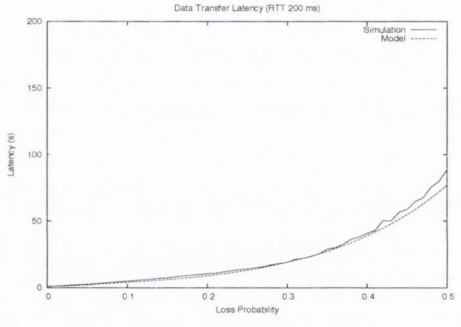
This section presents an evaluation of the prediction of the average number of packets sent by finite size TCP connections using a variety of loss probabilities. The prediction of the number of packets sent by TCP flows under specific loss conditions is given by equation 3.4 above. For the simulation scenarios presented in this section $\mathrm{OFF_t}$ was set to 150 seconds. This was done to maximise the separation between consecutive flows from the same source and thus avoid measurement errors due to long TCP retransmission timeout periods. 100 On-



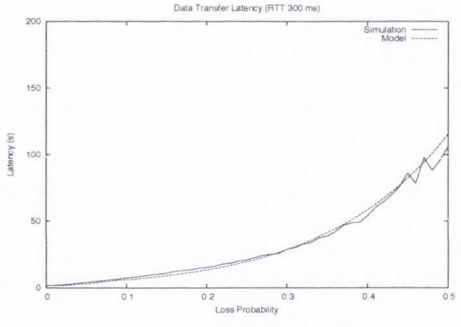
(a) Latency with RTT = 50 ms.



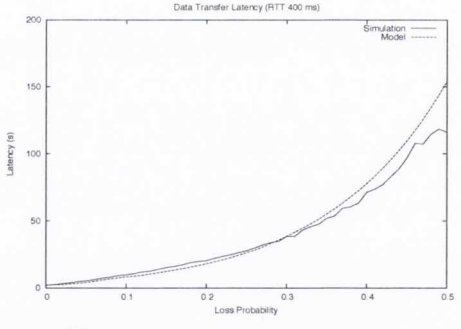
(b) Latency with RTT = 100 ms.



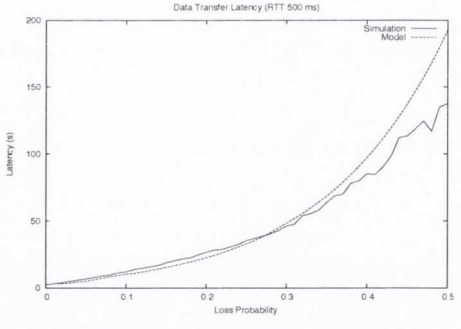
(c) Latency with RTT = 200 ms.



(d) Latency with RTT = 300 ms.



(e) Latency with RTT = 400 ms.



(f) Latency with RTT = 500 ms.

Figure 3.2: Evaluation of transfer latency prediction.

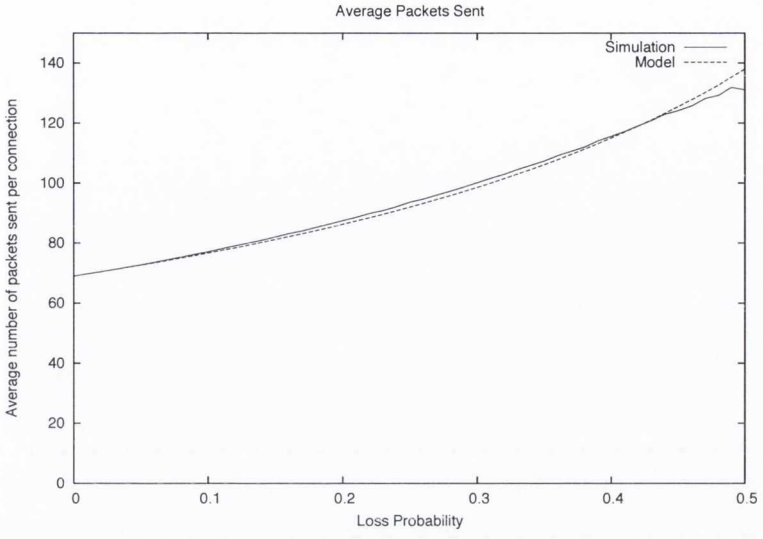


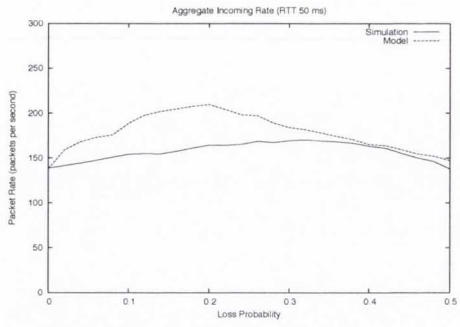
Figure 3.3: Average number of data packets sent by TCP connections.

Off TCP sources with RTT values that were uniformly distributed between 50 and 500 milliseconds were used for these simulations.

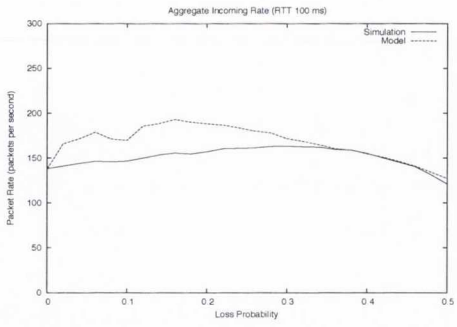
Figure 3.3 shows how the average number of packets sent by the TCP flows evolves as the loss probability experienced varies. The results show that equation 3.4 provides a very good fit for the number of packets sent. The error is very small apart from when the loss probability reaches close to $p = 0.5$, an extreme situation where TCP congestion control becomes less predictable.

3.4.4 Aggregate TCP rate

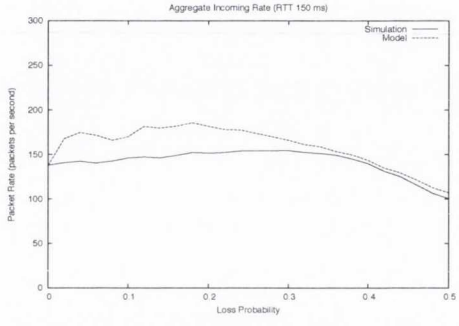
Finally the predictive performance of the offered packet rate of TCP aggregates is evaluated. The inter-connection time was set to $OFF_t = 50$ seconds. Figures 3.4 and 3.5 compare the total aggregate offered load of the traffic mix in the simulations with the predictions. The TCP prediction inputs were set as follows: the mean connection size $ON_b = 100000$ bytes, the loss probability p was in the range of $p = 0$ and $p = 0.5$. The measurement period was configured to a duration $\Delta = 0.5$ seconds. The simulations in Figure 3.4 use a dumbbell topology with the same RTT for all of the sources, while those in Figure 3.5 uniformly distribute the source node RTT in the ranges 50ms to 150ms and 50ms to 250ms.



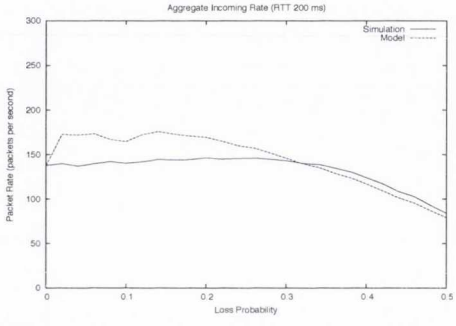
(a) Aggregate TCP rate with RTT = 50 ms



(b) Aggregate TCP rate with RTT = 100 ms

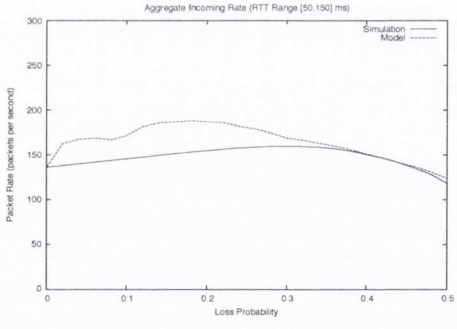


(c) Aggregate TCP rate with RTT = 150 ms

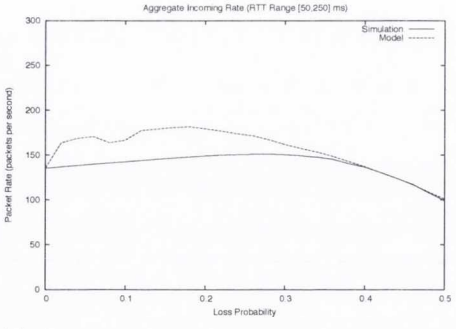


(d) Aggregate TCP rate with RTT = 200 ms

Figure 3.4: Evaluation of aggregate TCP rate prediction with single RTT.



(a) Aggregate TCP rate with RTT between 50 and 150 ms.



(b) Aggregate TCP rate with RTT between 50 and 250 ms.

Figure 3.5: Evaluation of aggregate TCP rate prediction with different RTTs.

The graphs show that the model provides a close approximation to the aggregate rate. There is an overestimation of the rate, particularly for non-zero, but low, loss probabilities. In these cases, the simulated TCP flows did not achieve the predicted performance, providing a lower offered rate.

Given these results, it is recommended that the model predictions be used as an upper bound on the expected performance of TCP-compliant traffic aggregates. This overestimation does not impact significantly on the effectiveness of the model for its intended usage scenarios, which is to differentiate aggressive non-congestion responsive traffic to well-behaved TCP traffic. Thus, when comparing actual rates with those obtained from the model it is acceptable that a slight over-estimate is obtained.

3.5 Conclusion

This chapter has addressed the need for a methodology to use at network routers that will characterise incoming traffic aggregates in respect of their compliance with congestion responsive TCP-like behaviour. Such a method is required at the core of any network mechanism that intends to be able to detect and penalise malicious non-congestion responsive bandwidth flooding DoS attack traffic.

The work in this chapter builds on recent TCP modelling efforts to provide accurate practical methods that are able to estimate the latency, number of packets generated and finally the average packet rate of finite TCP flows. Concrete equations are provided that minimise the required number of input parameters and allow for simple implementation of the TCP characterisation on network intermediaries.

Based on the characterisation of single TCP flow behaviour, a flexible approach for comparing traffic aggregates to reference, well-behaved aggregates of TCP flows is proposed. This method allows for the use of a minimal set of measurement metrics to evaluate the level of deviation a given aggregate displays from expected TCP behaviour. A novel concept of session creation aggressiveness is introduced, which allows for the dual characterisation and comparison to reference TCP traffic aggregates based on number of simultaneously active connections as well as generated aggregate packet rate.

Finally, a detailed simulation-based evaluation is carried out that compares actual TCP flow behaviour to the predictions of the TCP characterisation method. Both single flow and aggregate level results are evaluated. The latency and number of packets generated are compared in the case of individual TCP flows. For TCP traffic aggregates, the suitability of the characterisation as an upper bound on aggressiveness for traffic aggregates is considered.

To summarise, this chapter provides a practical and accurate characterisation of TCP traffic at both the single flow and traffic aggregate level. This method will be used in the following chapter as the core mechanism by which the DoS defence queueing scheme proposed implements its ability to evaluate the aggressiveness of a given traffic aggregate when compared to expected behaviour of well-behaved congestion responsive flows.

Chapter 4

Responsive Aggregate Defence (RAD)

Active Queue Management (AQM) schemes were introduced as an alternative network router queueing approach to the basic drop-tail queueing that was historically deployed in Internet routers. RED [54] was the first such AQM scheme and it spawned a large body of research on how to improve basic Internet router queueing mechanisms. AQM schemes strive to optimise global performance for congestion responsive TCP traffic, improving link utilisation and lowering packet loss rates caused by the use of drop-tail queueing and its impact on TCP congestion control mechanisms. Another core aspect of many AQM algorithms is that they implement a light-weight, scalable queueing mechanism that does not make use of per-flow information and should better be seen as an evolutionary step from drop-tail queueing rather than a completely new approach.

Traditional AQM schemes, as represented by RED and related proposals, do not consider the asymmetrical interactions of different types of traffic in the network router queue. Instead they assume that most traffic follows TCP congestion control rules and is not aggressive or malicious in any way. However, flow fairness is a real concern for users and network operators. Given an environment in which part of the traffic is composed of congestion responsive TCP flows and part is composed of non-congestion responsive traffic, the non-congestion responsive traffic will obtain an unfair share of the network bandwidth when

congestion occurs, as TCP sources will lower their offered load but the non-TCP traffic does not.

In an environment with either drop-tail or RED-like router queueing mechanisms, TCP traffic will obtain a comparably lower throughput to that of other non-congestion responsive traffic. This does not necessarily imply a malicious intent from the non-congestion responsive traffic. There are many types of traffic for which TCP-like congestion responsiveness does not provide the desired properties, such as streaming multimedia protocols like RTP [121].

This issue of flow fairness has motivated multiple proposals that extend AQM mechanisms to minimise the lack of fairness resulting from the presence of non-congestion responsive flows. These fairness-oriented AQM schemes were discussed in Chapter 2 and include CHOKe [108] and RED-PD [93]. These queueing schemes intend to approximate fair bandwidth allocation with a scalable AQM-inspired design that does not make use of extensive per-flow state.

AQM schemes that provide approximate fair bandwidth allocation can be effective in the presence of certain types of non-congestion controlled traffic. However, these schemes are not designed to counteract malicious sources of aggressive traffic, such as a bandwidth flooding DoS attacks. This is especially important when considering the common reliance on packet header information to classify and preferentially drop traffic that obtains an unfair share of the bandwidth. This approach is unsuitable when considering packet header spoofing mechanisms that are typically used by DoS attack traffic generation tools.

There have been some efforts to evolve AQM-based schemes that would provide some level of protection from malicious DoS schemes. ACC [92] implements a design based on local detection of aggressive traffic from the RED drop history together with a pushback mechanism, while Xu et al. explored in [144] a dual horizon approach to penalise both pulsing and persistent DoS attacks.

There is a need for a complete queueing DoS defence scheme that incorporates recent advances in TCP congestion control modelling, uses an architecture that extends scalable AQM approaches and uses traffic information data structures without per-flow information. Most importantly, this should be effective against a wide variety of malicious DoS traffic, independent of the time-scale at

which it impacts on TCP flows and its use of packet header spoofing to achieve stealthiness. This chapter proposes Responsive Aggregate Defence (RAD), a router queueing mechanism that implements such a DoS defence design.

The chapter is organised as follows: first the ideas behind the RAD DoS defence queueing scheme are introduced, with the RAD architecture being described next. The following section addresses how RAD deals with the problem of maintaining detailed traffic measurement metrics without requiring per-flow information. The RAD DoS penalisation algorithms are described next, including a discussion of their potential for false positives and negatives. Finally the chapter concludes with a summary of RAD's contributions.

4.1 Overview

Responsive Aggregate Defence (RAD) is a router queueing scheme that works in conjunction with existing Active Queue Management approaches to protect responsive TCP traffic from bandwidth-flooding DoS attacks. The main goal of RAD is to minimise the effect of aggressive traffic generated by bandwidth hogging DoS attacks on responsive TCP traffic at the router queue. RAD intends to achieve this goal by relying only on information available at the queue, without requiring coordination between different routers or other network equipment.

RAD is expected to protect congestion responsive TCP traffic at the router queue while being as transparent as possible to the TCP traffic itself. In a router queue that is receiving flooding DoS attack traffic, RAD should maximise the portion of TCP traffic that does not notice any impact from the ongoing attack.

RAD should also be effective against flooding DoS attacks that operate at both large and small time scales. It implements a dual scale system in which extreme aggressive traffic behaviour is penalised independently of whether its impact happens on high frequency queue dynamics or over a longer term.

RAD is designed to work at the granularity level of traffic *aggregates*. An aggregate can be defined as a set of flows traversing the link that share a common attribute. What constitutes a *flow* and *aggregate* can vary depending on what the most relevant aspects of the incoming traffic are for the system being implemented. In the case of RAD, a flow will be defined as the set of packets

that share the same source and destination IP addresses, source and destination port numbers, and protocol header field. Note that this maps directly to how TCP connections are identified.

An *aggregate* will be defined in RAD as being composed by all the *flows* that share the same destination IP or subnet address. Combining flows into aggregates provides a flexible model in which fine-grained per-flow information is not maintained at the queue and decisions are taken at the aggregate level. Having a relatively low number of aggregates has the downside of increasing the likelihood of penalising "unlucky" traffic, flows that map to the same aggregate as the malicious traffic and thus suffer preferential drops.

However, this coarse granularity of operation enables RAD to be implemented on high-bandwidth links with bounded memory usage and to be effective against source address spoofing attack traffic. Using aggregate level measurements, and other implementation techniques described below, RAD can be implemented in a very scalable way. The small and constant memory usage of RAD is independent of incoming traffic rates or characteristics, making RAD especially suited for flooding bandwidth attacks.

An additional consequence of using destination address based aggregate measurements in RAD is the protection provided from packet header spoofing attack traffic. DoS traffic can spoof any packet header, thus being able to potentially manipulate how packets will map to aggregates. However, even DoS traffic must provide a correct value for the destination IP address, as this field will be used to route the attack traffic towards the victim. Thus, using the destination IP address as the packet header field that maps packets to specific aggregates provides the highest likelihood of being able to identify the set of packets that form part of an attack stream. One variation of this approach is to only use the first N bits of the destination IP address as the aggregate key, thus allowing all traffic going to the same network or subnetwork to map to the same aggregate. This can be useful in the case were DoS attacks target more than one host on a given network.

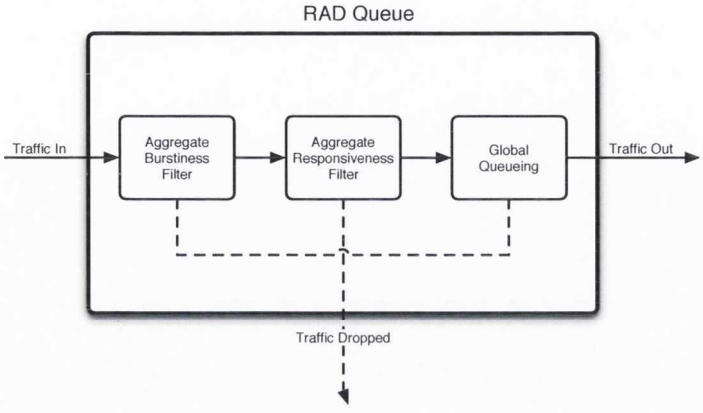


Figure 4.1: RAD Architecture

4.2 Architecture

RAD follows a pipeline architecture, in which incoming traffic must pass through multiple queue management modules to be accepted at the queue. Figure 4.1 shows a diagram of the high-level architecture of RAD. It is composed by the Aggregate Burstiness Filter, the Aggregate Responsiveness Filter and the Global Queueing module.

Each of the queue management modules that are part of the RAD pipeline can decide either to drop or to accept a given packet, and packets will only be accepted once they have been accepted by all of the modules. The incoming packet will finally be added to the queue buffer once the last module has accepted it.

The following pseudocode illustrates the high-level operations taken by a RAD-based queue which uses RED as its global queeuing algorithm to decide whether to accept an incoming packet:

```
bool RADAccept(packet) {
    return BurstinessAccept(packet) &&
        ResponsivenessAccept(packet) &&
        GlobalQueueingAccept(packet);
}
bool BurstinessAccept(packet) {
    return PeriodIncomingRate(packet) < incoming_rate_threshold;
}
bool ResponsivenessAccept(packet) {
    return RandomIn(0, 1) > ResponsivenessDropProbability(packet);
}
bool GlobalQueueingAccept(packet) {
    return GlobalQueue::Accept(packet);
}
```

In the pseudocode above, the PeriodIncomingRate() function returns the packet count for the current measurement period, the RandomIn() function returns a pseudorandom number in the given range, the ResponsivenessDropProbability() calculates the RAD drop probability based on the TCP characterisation model, and GlobalQueue::Accept() returns whether the global queueing scheme (for example RED or DropTail) would accept the incoming packet.

4.2.1 Aggregate Burstiness Filter

The Aggregate Burstiness Filter is the initial module packets must go through when arriving at the queue. Its goal is to penalise extreme short-term burstiness, such as that seen in low-rate TCP targeted DoS attacks. These attacks impact congestion responsive TCP traffic by causing extreme congestion at the link on very short time scales, while not appearing aggressive when looked at on longer time scales.

Given the nature of these attacks, the Aggregate Burstiness Filter does not collect any measurements over multiple measurement periods, as this averaging over time would mask the short-term impact of the attack traffic. The approach taken by this RAD module is to prevent any traffic aggregate from monopolising the link on the time scale of a single measurement period. This is achieved

by having an absolute maximum share of the available bandwidth during a measurement period that each aggregate is constrained to. Any aggregate that exceeds the maximum share of incoming capacity will be penalised by having any subsequently received packets dropped during a single measurement period.

While quite aggressive against the penalised traffic, this approach is required as, due to the nature of the attack traffic, both detection and penalisation must be performed at the time scale of a single sub-second measurement period. It can be made quite safe for bursty but overall congestion responsive TCP traffic by configuring its maximum share threshold conservatively. Given the extreme congestion caused by these attacks on short time-scales, false negatives are not a big concern even with thresholds are configured well above the expected behaviour of TCP traffic.

4.2.2 Aggregate Responsiveness Filter

This RAD module analyses the long-term behaviour of aggregates, and penalises those which deviate most from expected TCP-like congestion-responsiveness. This module maintains per-aggregate measurements that span long-term behaviour of the incoming traffic during multiple measurement periods.

The analytical characterisation of congestion responsive TCP traffic described previously is used to evaluate each aggregate's behaviour. Traffic aggregates are considered non-congestion responsive, and thus likely to be part of a bandwidth flooding DoS attack, as they deviate from the traffic rates predicted by the model.

Those aggregates that deviate most from the model's predicted behaviour will experience preferential randomised dropping. The aggregates whose behaviour is closer to, or more conservative than, the model prediction will not experience any drops caused by the Aggregate Responsiveness Filter and will be forwarded to the Global Queueing module for consideration.

4.2.3 Global Queueing

This final module implements a queueing algorithm that optimises performance of the queue at the global level, focusing on low queueing delay and high link

utilisation. This module should not concern itself with protection of TCP traffic from DoS attacks.

Many AQM schemes could potentially be used as an implementation approach for this module, in this work RED [54] has been used as the Global Queueing implementation algorithm. RED is the best understood and deployed AQM scheme and thus it is a good choice for this module. The Aggregate Burstiness and Aggregate Responsiveness modules keep some state information and operate at the granularity level of destination address-based traffic aggregates. However, this module should not maintain any non-global state and should focus on global optimisation of traffic in the queue. It should assume that the relative share of traffic and behaviour within the incoming stream has already been studied and acted upon by its two upstream modules.

If the Global Queueing module accepts an incoming packet this will be the final step in the RAD queueing algorithm. Accepted packets will be enqueued in the outgoing link queue buffer and eventually sent by the router.

4.3 Measurement metrics

In addition to any global measurement information maintained by the Global Queueing RAD module, the Aggregate Burstiness and Aggregate Responsiveness filters require the maintenance of a set of metrics based on incoming traffic to be able to implement their logic.

4.3.1 Metrics overview

The primary characteristic of RAD state metric maintenance is that it operates at the granularity of aggregates. RAD defines a fixed number of aggregate "buckets" that all incoming traffic maps to. This mapping is performed by applying a hashing function to the destination IP address of each incoming packet. This guarantees that all traffic that is directed towards a given destination maps to the same aggregate. One possible variant of this scheme would be to only use a subset of the bits on the destination IP address header, so that all traffic towards the same network, instead of host, would map to a given aggregate bucket.

Another design characteristic of RAD's metric calculation mechanism is that it uses both instantaneous, as well as averaged, metrics. RAD operates using measurement periods of duration Δ seconds. RAD maintains both instantaneous counters that are reset at the beginning of each measurement period and averaged metrics that are updated with the instantaneous counter data at the end of each measurement period.

In addition to being space efficient because of its use of lower granularity metrics that operate at the level of aggregates, RAD also implements all of its metrics in a way that guarantees constant space requirements independently of the characteristics of the traffic it is collecting metrics about. Given the malicious nature of DoS attack traffic, this aspect is important to ensure that attackers will not be able to generate traffic with spoofed packet headers that will cause the RAD implementation to run out of memory.

The implementation approach described below consists of a bounded-memory data structure used for each aggregate bucket. These efficient probabilistic data structures provide bounded memory use and controllable estimation error probabilities that are a function of the memory available to them. This allows for a tunable design in which RAD can easily scale up or down to handle different links, traffic mixes and router hardware capabilities.

4.3.2 Metrics implementation

The primary data structure used by RAD for its metrics is named *AggregateState*, and contains the following state data for a given aggregate bucket i:

in_i Counter of incoming packets that are part of the aggregate received by the queue from the beginning of the measurement period.

r_i Incoming generated traffic load by the aggregate in packets per second. It is a weighted moving average of the aggregate packet rate, calculated as follows at the end of each measurement period:

$$r_i = (1 - \alpha) \times r_i + \alpha \times \frac{in_i}{\Delta} \tag{4.1}$$

$drop_i$ Number of packet drops experienced by an aggregate during the measure-

ment period. It includes drops caused by any of the three RAD modules.

$loss_i$ The average loss probability experienced by packets in the aggregate, taking into account all the packet drops caused at the queue. This metric is a weighted moving average updated with the following equation at the end of each measurement period:

$$loss_i = (1 - \alpha) \times loss_i + \alpha \times \frac{drop_i}{in_i} \tag{4.2}$$

A_i This measurement is a Bloom filter that represents the set of flow identifiers experienced by the queue for the aggregate since the beginning of the measurement period.

f_i Average number of distinct active flows entering the queue during each measurement period for the aggregate. It is implemented by calculating a moving average of the number of elements in the instantaneous measurement A_i (named as the *size* function below) at the end of each measurement period:

$$f_i = (1 - \alpha) \times f_i + \alpha \times size(A_i) \tag{4.3}$$

The averaged per-aggregate metrics maintained as part of the *AggregateState* data structure are all implemented using an exponential moving average that tracks the instantaneous measurements while smoothing the impact of short term changes of traffic behaviour. A weight parameter α is used that defines how closely the averaged metrics track the instantaneous measurements. The larger the value for α, the quicker the metrics will reflect changes in the measurements, with the downside of also being exposed to large potential variability of the measurements due to their short-term nature.

Additionally, RAD employs a global incoming rate metric, r, that aggregates all the incoming traffic experienced by the queue. This averaged metric is also implemented as an exponential moving average and updated at the end of each measurement period as follows:

$$r = (1 - \alpha) \times r + \alpha \times \frac{\sum_{i=1}^{i=N} in_i}{\Delta} \tag{4.4}$$

4.3.3 Flow metrics implementation

Most RAD metrics are based on relatively simple integer or floating point counters. The exception is the estimate of the number of flows seen for each aggregate. This metric, as defined in equation 4.3, is an exponential moving average of the instantaneous value of the number of experienced flows, $size(A_i)$, for each aggregate $i \in [1, N]$, where N is the total number of aggregate buckets configured in RAD.

This implementation of the average number of flows metric requires a data structure that will provide the implementation of A_i. A naive implementation of this data structure could be provided by a standard Set implementation that is provided by most programming languages, such as the C++ set class or the Java Set interface implementations HashSet and TreeSet.

While providing a rich set of functionality and perfect accuracy, these implementations would typically have a memory usage that is dependant on the number of elements stored in the Set. In the context of RAD, the elements that would be stored in this Set would be flow identifiers. Given that RAD must keep track of every flow seen during a measurement period, use of these standard implementations would make RAD's memory usage dependent on the characteristics of the incoming traffic. Given that DoS attack traffic can be expected to use packet header spoofing, this effectively means that attackers would be able to cause increased memory usage within the RAD router. This could easily allow the attackers to render the router unavailable due to a memory overflow.

Given that standard Set data structures would not be effective for implementing A_i, an alternative must be found that provides the required functionality while being resilient to packet header spoofing by DoS attack traffic. One thing to note is that RAD does not require the rich interfaces provided by full Set implementations. The operations that RAD requires to be available for a given flow counter A_i and a flow identifier $flowID$ are:

$insert(A_i, flowID)$ Adds the $flowID$ flow identifier to the set of seen unique flow identifiers in the counter.

$size(A_i)$ Returns the number of distinct flow identifiers added to the counter.

$reset(A_i)$ Initialises the flow counter so that $size(A_i) = 0$.

These operations will allow for RAD to use an alternative A_i implementation that is based on Bloom filters [17, 21].

4.3.4 Bloom filters

Bloom filters are space-efficient randomised data structures that can be used to provide set membership queries. Bloom filters provide very high space efficiency at the cost of a controllable probability of false positives (membership queries that return true even though the entry has not been added to the filter).

Burton Bloom's seminal 1970 article [17] introduced Bloom filters, they remained relatively little used outside of database applications until recently. Within the past few years they have begun to be used in addressing a wide variety of network related problems [21]. The space saving aspect of Bloom filters makes them especially suited to applications in which membership queries on a set or list need to be supported in an environment where the size of data stored or transmitted is a primary concern.

A Bloom filter can be used to represent a set $S = \{x_1, x_2, ..., x_n\}$ of n elements. The filter will use an array A of m bits all set to 0 and k independent hash functions $h_1, h_2, ..., h_k$. These k hash functions map each input item to a random uniform number in the range $\{1, ..., m\}$. Given a new element x, this is inserted in the filter by setting to 1 the bits $h_i(x)$ for $1 \leq i \leq k$. If any bit location is already set to 1 then new insertions do not affect it, it is only the first insertion that changes it from 0 to 1.

The probability that a specific bit is set to 0 after all n elements of S are inserted into the filter is approximated by [21]:

$$p = e^{-kn/m} \tag{4.5}$$

This p can also be used to approximate the ratio of 0 bits after all the n elements have been inserted. Thus, the probability of a false positive, an element that the filter will return true when asked for membership when it has never been inserted before, is $f = (1 - p)^k$ [21].

Bloom filters allow for a configurable expected false positive rate. Assuming fixed values for the Bloom filter bit array size m and the number of items it

has to support n then the optimal number of hash functions, k, to minimise the false positive rate is $k_m = \ln 2 \cdot (\frac{m}{n})$ [21]. For fixed m and n, the false positive rate occurs when $p = 1/2$, that is when half of the bits in the filter are set to 1.

Bloom filters are thus a very good choice for implementation of the A_i flow counter data structure. Their constant and efficient space use and controllable accuracy degradation under high numbers of elements make them well suited for the requirements of the RAD A_i flow counter data structure.

4.3.5 Implementing the flow measurement metric with Bloom filters

We can now consider how A_i can be implemented using Bloom filters. Given A_i, the flow counter data structure for an aggregate i, a Bloom filter backed by an array of m bits can be used for its implementation. This filter requires k independent hash functions $h_i^1, h_i^2, ..., h_i^k$ that map the flow identifier; composed of protocol header field, source IP address, source port, destination IP address and destination port; to one of the m bits in A_i.

Using a Bloom filter for A_i means that the operations that RAD requires to be available for use by the A_i data structure are implemented as follows:

insert(A_i, *flowID*) This is implemented by inserting the flow identifier into the Bloom filter that A_i backs. This is achieved by setting to one each one of the bits in A_i that the flow identifier maps to once hashed with the k different hashes used for the Bloom filter. The operation is implemented in pseudo-code below:

```
for i = 0 to k:
  hash = hashes[i]
  bin = hash(flow_id)
  A_i[bin] = 1
```

size(A_i) Given the probabilistic nature of Bloom filters it is not possible to obtain the exact number of items stored in A_i. However, an estimate can be calculated by using equation 4.5 above for the probability p that a bit in A_i is set to 1 after n elements have been inserted. Given a Bloom filter

A_i in which j out of the m bits are set to 1, the estimate for the number of elements inserted in the filter, $size(A_i)$, can be calculated as:

$$size(A_i) = \frac{-m \cdot \ln(\frac{m-j}{m})}{k}$$

$reset(A_i)$ Initialises the flow counter so that $size(A_i) = 0$. This is a very simple operation that just sets all m bits in A_i to zero:

```
for i = 0 to m:
  A_i[i] = 0
```

An important consideration when using Bloom filter implementations for A_i is the performance of Bloom filters with actual hashing functions, as opposed to theoretical perfect hashing functions. This is an area that has received limited attention in the past [113]. The MD5 [116] hashing algorithm is cited as a practical choice in [21], while a stronger hashing algorithm, and thus a better choice than MD5, would the US Secure Hash Algorithm 1 (SHA-1) [44]. SHA-1 is considered the successor of MD5 and is a robust, and widely implemented, secure hashing algorithm. The use of SHA-1 provides the additional advantage that it has been implemented on existing network processors or platforms, as it is extensively used in network security technologies such as TLS [40] and IPSec [74].

RAD utilises SHA-1 for all its hashing needs. When using SHA-1 as the hashing function to implement the k different hashing functions that RAD needs for each aggregate, there is the issue of requiring the availability of k independent hashing functions. One effective way of obtaining these functions is by appending a random *salt* input before the flow identifiers are hashed using SHA-1. Each hashing function in the Bloom filter can have its own *salt* and share the same SHA-1 implementation. These *salts* can be generated at initialisation time using either a pseudo-random number generator or any other source of random information available in the router.

4.4 RAD DoS penalisation

This section will discuss the implementation of the two central RAD modules: the Aggregate Burstiness and the Aggregate Responsiveness filters. Together with the AQM-based Global Queueing module and the RAD measurement metrics they form the core of the RAD architecture. These two modules collaborate to detect and penalise a wide variety of bandwidth flooding malicious DoS traffic. Their ultimate goal is to ensure that during a DoS attack as much as possible of the DoS traffic is dropped before it reaches the Global Queueing module while being highly transparent to non-malicious congestion responsive TCP traffic.

4.4.1 Aggregate burstiness filter

The first stage of a RAD queue is the module in which extreme aggregate-level burstiness is penalised. Certain levels of burstiness are expected from normal TCP operation, caused mainly by the arrival and completion of TCP connections, and the TCP window control algorithms.

However, extreme burstiness can be used by low-rate DoS attacks to maliciously impact TCP traffic. These attacks exploit the sensitivity of TCP congestion-control algorithms to transient high-loss periods, without generating large amounts of long-term attack traffic. Thus, it is necessary for RAD to be able to respond to extreme short-term burstiness in real-time by penalising the aggregates that generate this type of traffic.

The aggregate burstiness filtering component of RAD works within the timescale of a single measurement period. It penalises traffic aggregates as soon as the amount of incoming traffic generated by them during a measurement period reaches a maximum threshold. This penalisation is implemented by applying an aggregate burstiness drop probability p_b to each arriving packet, where p_b is given by

$$p_b = \begin{cases} 0 & \text{if } in_i \leq in_{max} \\ 1 & \text{if } in_i > in_{max} \end{cases} \tag{4.6}$$

where in_{max} denotes the maximum number of packets from the aggregate that will be accepted at the queue during the measurement period and in_i is the value of the instantaneous incoming packets counter for aggregate bucket i.

The in_{max} threshold is calculated using:

$$in_{max} = in_{th} \times \Delta \times B_l \tag{4.7}$$

where in_{th} is a threshold parameter that determines the maximum acceptable packet rate for the aggregate during the measurement period as a ratio of the link packet rate, Δ is the duration of the measurement period in seconds, and B_l is the maximum bandwidth of the link in packets per second.

The main parameter for in_{max} is the in_{th} threshold that specifies how many packets an aggregate will be allowed to generate during a measurement period before having its traffic dropped. This parameter should be set to a value that is conservative enough not to affect TCP traffic while dropping transient attacks. In practise, a conservative value of $in_{th} = 1$ has been shown to be effective against these attacks, as they require very large numbers of packets to be accepted during a very short time, which quickly increases in_i for the aggregate to above in_{max}.

This phase of RAD provides the ability to quickly respond to very aggressive transient behaviour that is only noticeable on short timescales (within a single measurement period). The measurement period Δ can typically be smaller than the average RTT. This prevents the applicability of expected long-term TCP behaviour, as even TCP flows will show short-term behaviour that diverges from their long-term congestion responsiveness.

4.4.2 Aggregate responsiveness filter

The aggregate drop probability p_i is calculated so that behaviour that deviates from expected congestion-responsive behaviour is penalised. The reference model for non-malicious traffic used is given by the finite TCP predictive model provided in the previous chapter. This prediction is compared to the actual traffic that composes the aggregate, and deviations from the model both in offered load and number of flows created are used as indicators of congestion unresponsiveness.

Let $B(p)$ defined in equation 3.1 be the estimated average packet rate for a finite size TCP connection with average connection size b_t, round-trip time RTT

and average loss probability p. The expected packet rate for the aggregate, as predicted by TCP characterisation, can thus be obtained from:

$$r_i^t = f_i \times B(loss_i) \tag{4.8}$$

where f_i is the value of the averaged number of flows for aggregate i and $loss_i$ is the average loss probability metric, also for aggregate bucket i.

The model can also be used to obtain an expectation of the number of flows that would be needed to generate the incoming traffic received from the aggregate. This is given by:

$$f_i^t = \frac{B(loss_i)}{r_i} \tag{4.9}$$

where r_i is the average packet rate metric for the aggregate bucket i.

Considered together, the expected packet rate r_i^t and the expected number of flows f_i^t provide a "profile" of the expected behaviour of the traffic of an aggregate composed of finite-length congestion responsive TCP connections. An aggregate that strictly conforms to the expected behaviour should limit the rate and number of flows to those of the prediction, thus having $r_i \leq r_i^t$ and $f_i \leq f_i^t$, where r_i is the averaged packet rate metric for the aggregate and f_i is the measured average number of flows that compose the aggregate.

We now have available a dual characterisation of the expected behaviour from the traffic that maps to an aggregate, which includes its packet rate as well as its composition in terms of the number flows or TCP connections. These two aspects will allow us to limit the range of acceptable behaviour for an aggregate. Both anomalous behaviour in terms of generated packet rates or flow creation can now be detected.

This aspect of RAD is crucial to its effectiveness against DoS bandwidth flooding attacks. Given that these attacks have the ability of using packet header spoofing, they can disguise themselves as being formed by as many or as few flows as desired. Without being able to evaluate the number of flows that are part of the attack aggregate, RAD would not be able to prevent attacks that look like they are composed of many small flows.

Given a attack rate $r_i = r_i^t$ and average loss rate $loss_i$, an aggregate with

malicious traffic would be theoretically able to appear to RAD as congestion responsive by controlling its packet header spoofing so that it appears as being composed of $f_i = f_i^t$ flows. However, this scenario is extremely difficult for attackers to cause in practice. The attacker would need to know what the loss probability $loss_i$ is at the RAD queue. This would require the attack flows to be able to receive packet loss notification, which would force the attacker to avoid using source address spoofing. Source spoofing is at the core of how DoS attacks hide the actual location of the hosts generating the attack traffic. Thus, RAD can force the attacker to limit its use of packet header spoofing and to be able to receive feedback from the network, which makes the attack traffic much easier to detect and police by existing network diagnostics tools.

RAD maintains an aggregate responsiveness drop probability p_i^r for each aggregate bucket i . This drop probability determines whether incoming traffic will be accepted or dropped by the RAD Aggregate Responsiveness filter. This probability p_i^r is updated for each one of the aggregate buckets at the end of each measurement period, so it is constant for the duration of any given measurement period. Given that p_i^r is calculated from averaged metrics that are only updated at the end of a measurement period, it will be updated just after the rest of the averaged metrics have been refreshed with new data from the instantaneous counters.

For each aggregate bucket i and given packet rate ratio $R_i = \frac{r_i}{r_i^t}$, the flow number ratio $F_i = \frac{f_i}{f_i^t}$, and minimum and maximum thresholds for R_i (R_{min} and R_{max}) and F_i (F_{min} and F_{max}), the aggregate responsiveness drop probability p_i^r will be calculated as follows:

$$p_i^r = \max\left(\min\left(\max\left(\frac{R_i - R_{min}}{R_{max} - R_{min}}, 0\right), 1\right), \min\left(\max\left(\frac{F_i - F_{min}}{F_{max} - F_{min}}, 0\right), 1\right)\right)$$

Thus, p_i^r will increase from 0 to 1 as either the ratio of actual and expected packet rates or actual and expected number of flows of an aggregate increase from the minimum to the maximum threshold for the metric.

4.4.3 Summary of the parameters of RAD

The different components that are included in RAD have been introduced above. These components make use of a series of configuration parameters that have the potential to impact on the performance of RAD deployments in real networks. This impact and suggested settings will be discussed in detail in the RAD Performance Evaluation chapter. In this section, a summary of these parameters is provided for reference:

Round-trip time (RTT) This is the round-trip time configuration parameter used in the TCP characterisation model at the core of the Aggregate Responsiveness filter.

Average connection size (b_t) This parameter specifies the average size of TCP transfers in bytes and is used in the TCP characterisation model.

Measurement period (Δ) It specifies the duration of the measurement periods that are used by the Aggregate Responsiveness module to obtain average metrics for the incoming traffic.

Incoming rate thresholds ($R_{\mathbf{min}}$ and $R_{\mathbf{max}}$) These are used to limit the range of the average incoming rate for an aggregate that is deemed acceptable given the TCP characterisation predictions.

Flow number thresholds ($F_{\mathbf{min}}$ and $F_{\mathbf{max}}$) These two thresholds bound the acceptable number of flows that can be observed in a given aggregate, relative to the expected quantity given by the TCP characterisation model.

Incoming rate threshold ($in_{\mathbf{th}}$) It specifies the maximum threshold applied to the incoming number of packets in an aggregate during a single measurement period. It is used in the Aggregate Burstiness module to penalise highly aggressive traffic.

4.4.4 False positives and negatives

The likelihood of false positives, in which RAD would penalise non-malicious traffic, and false negatives in which RAD would not penalise DoS traffic must be considered when discussing RAD DoS penalisation. The overall design goals of

RAD include DoS penalisation and TCP transparency. However, both features do not have the same priority. RAD must primordially ensure it minimises the rate of false positives. It should only penalise a given traffic aggregate when the likelihood of suffering false negatives by not doing so far surpasses the likelihood of false positives when acting against an aggregate.

4.4.4.1 False positives

The probability of false positives, or the likelihood that either the Aggregate Responsiveness or Aggregate Burstiness modules will penalise non-malicious traffic, is an important aspect of RAD that must be considered. There are three main scenarios in which RAD could cause false positives to occur:

- DoS unrelated traffic is penalised due to its mapping into the same aggregate bucket as other malicious flows.
- The Aggregate Responsiveness module's drop probability, p_i^r, causes the over-penalisation of non-malicious but aggressive TCP traffic.
- The Aggregate Burstiness module penalises extremely bursty, but otherwise congestion responsive, TCP flows.

RAD provides a configurable measurement and penalisation granularity through its use of traffic aggregates. The impact of having non-malicious traffic unrelated to the victim penalised alongside malicious DoS traffic on the same aggregate can be minimised by different means. The number of aggregates used by RAD is a configurable aspect of the scheme, which can be incremented to lower the amount of unrelated flows mapped together to the same aggregate bucket. The larger the link capacity of the queue, the larger the number of aggregates that should be configured in RAD, assuming availability of CPU and memory resources.

However, increasing the number of aggregate buckets requires more resources and is not a choice that some deployments will be able to make. Another approach to minimise the impact of mixing malicious and non-malicious flows in the same aggregate bucket is to change the seed used by the hashing algorithm that maps flows to aggregates. Periodically changing this seed lowers the likelihood that the same "unlucky" flows will consistently be penalised during a long

DoS attack. This change does not lower the number of flows that will potentially be over-penalised, but it will spread this over-penalisation over a wider range of flows, allowing degraded service to TCP traffic without having any individual flow highly penalised.

The location of the RAD queue in the path from the attack source to the destination is also an important factor when considering the issue of penalised non-malicious flows. The RAD aggregate responsiveness drop probability effectively removes malicious non-congestion responsive traffic from the network, and does so at granularity levels of an aggregate. These aggregates will contain a bigger share of the non-malicious traffic that is trying to access the DoS attack victim as the router implementing RAD is closer to the victim. Thus, it is preferable to deploy RAD either at the egress routers of the network(s) producing the attack traffic or at the backbone core routers, where RAD will prevent very high bandwidth DoS attacks from impacting on performance for a large numbers of flows.

Another potential cause of false positives is the penalisation of non-malicious traffic due to miss-classification of aggressive or low round-trip time TCP flows as attack traffic. These TCP aggregates use configurations that cause aggressive updates of their congestion windows or the creation of a larger than normal number of TCP connections, a technique used by some TCP based applications to increase TCP traffic performance. While the impact of RAD on these aggressive TCP flows must be minimised, it is also true that they are indeed behaving in a manner that is more aggressive than the reference closed-loop finite TCP predicts. Thus, it is not necessarily true that these flows should obtain the same transparency as other, less aggressive TCP connections. The desired behaviour from RAD in this case it that these aggressive TCP aggregates should obtain the same throughput as other non-aggressive TCP traffic, which would require some level of preferential dropping from RAD.

RAD provides a series of configuration parameters that can help alleviate the impact of this scenario. The thresholds F_{min}, F_{max}, R_{min} and R_{max} as well as the RTT parameter used in the RAD aggregate responsiveness probability calculation allow operators to tune the RAD aggregate responsiveness filter behaviour to the local characteristics of the traffic traversing their routers.

The impact of these parameter settings on RAD performance will be explored in detail in the following chapter. Furthermore, aggressive but still congestion responsive traffic will only suffer a limited level of RAD penalisation, as this traffic will eventually back off from its offered packet load when RAD preferentially drops it over other less aggressive TCP flows. In comparison, malicious DoS traffic would never lower its offered load, causing an increase in RAD penalisation over time as the same or higher generated packet rates are maintained despite increasing packet loss.

4.4.4.2 False negatives

This section addresses the likelihood of RAD not penalising malicious DoS traffic due to its inability to detect it as such. RAD bases its detection of DoS traffic that is a candidate for penalisation by detecting behaviour that it considers aggressive. This aggressiveness detection operates on short and long time scales from a time horizon point of view and is based on either the packet rate metric or the number of active flows.

The Aggregate Responsiveness module operates on a long-term time horizon and utilises both the incoming packet rate, as well as the number of active flows in an aggregate, to calculate its penalisation level. This module is not able to detect aggressive behaviour at time-scales shorter than a single measurement period, as handled by the Aggregate Burstiness module.

It would be theoretically possible for a malicious DoS traffic aggregate to tune its offered load and packet header spoofing mechanism to that which the Aggregate Responsiveness TCP characterisation predicts, thus causing a false negative. However, this scenario is not easy for an attacker to implement in practice. The DoS traffic sources would need a way of being notified about packet losses to be able to estimate the right combination of packet rate and number of flows that would cause most impact while not being detected. This requires the DoS traffic sources to be able to be reachable and to receive ACK packets, which prevents them from being untraceable due to packet source header spoofing. Thus, any attacker that would intend to use this approach to avoid detection by RAD would need to expose itself to detection by other network management systems due to its use of non-spoofed IP source address headers.

4.5 Conclusion

This chapter has introduced Responsive Aggregate Defence (RAD), a queueing scheme that provides DoS defence to TCP congestion control traffic at the network router queue. RAD is based on a dual time scale design and includes advanced TCP characterisation together with a sophisticated measurement metrics collection system to be able to accurately detect and penalise bandwidth flooding DoS attack traffic.

The RAD Aggregate Burstiness and Aggregate Responsiveness filter modules provide a resilient defence from a wide variety of DoS bandwidth attack types. RAD is able to effectively detect and prevent DoS floods at both short and long time-scales and is also effective against packet header spoofing techniques. This makes RAD a novel and effective algorithm for modern bandwidth flooding DoS attack defence.

RAD is designed to be agnostic of where in the network it is deployed. However, it operates by detecting non-responsive behaviour of aggregates at times of congestion, which affects its effectiveness in different scenarios. RAD focuses on protecting traffic aggregates that show congestion-responsiveness from those that do not. This approach effectively provides a mechanism to remove malicious traffic from the network wherever it impacts on throughput of TCP traffic. Thus RAD is not only a good solution to protect backbone networks from DoS attacks that may be able to saturate them, it also improves availability of the victim's services.

Given its aggregate-level granularity, RAD will penalise all traffic towards a victim in the queue equally. This means that RAD's ability to maximise the throughput of non-malicious traffic towards the victim decreases the closer it is located to the victim. This is easy to understand if the routing paths that lead to a victim are considered a tree with the victim as its root. The closer RAD is implemented to the root the higher the likelihood of penalising non-attack traffic. Thus, it is recommended that RAD be deployed as far from the victim as possible but at routers that are traversed by enough traffic for the DoS packets to impact the throughput of TCP flows. The following chapter presents a thorough performance evaluation of RAD.

Chapter 5

RAD Performance Evaluation

Responsive Aggregate Defence (RAD) was presented in the previous chapter. RAD is a router queueing scheme that allows network routers to detect and penalise malicious traffic generated by bandwidth flooding DoS attacks. RAD combines an efficient measurement metrics collection design and an accurate characterisation of finite TCP connection behaviour within an architecture that is based on Active Queue Management (AQM) principles of simplicity and scalability.

This chapter will present a detailed evaluation study of RAD performed using the NS-2 Network Simulator version 2.29 [20]. An evaluation study, such as this one, must focus on a rigorous evaluation methodology and detailed exploration of any relevant parameters of the base scenarios to provide a deep insight into the performance of a router queueing scheme, such as RAD.

When considering the evaluation methodology two aspects are most relevant: the base network scenario with its network topology and traffic generation aspects, and the core metrics used that will allow for a consistent evaluation of RAD performance, independent of the specific input parameters being explored for a given simulation. Having a consistent base scenario and meaningful metric selection allows different simulation experiments to focus on the aspects under consideration.

There are many potential application scenarios for RAD that could be explored in this evaluation. However, it is necessary to concentrate on exploring

the parameter space that is most relevant to the performance of RAD under the environmental conditions where it was designed to be effective. For this evaluation study the aspects considered have been divided into three main areas: DoS detection and penalisation performance, impact on TCP traffic, and impact on RAD performance of its internal parameter space. These three areas focus on the core requirements of RAD to be a simple and robust queueing scheme, that protects TCP traffic from malicious bandwidth flooding DoS traffic while minimising its impact on existing congestion responsive TCP traffic.

5.1 Methodology

The methodology used in the simulation-based evaluation of the RAD queueing scheme addresses aspects such as the metric choice, network topology, and responsive and malicious traffic generation. When appropriately utilised, these factors provide a default base scenario that is realistic within the intended performance environment of RAD, while allowing for specific aspects of the scenario to be explored to study their impact on RAD performance.

This section will first describe the base metrics used in this RAD evaluation study. The core physical network topology used for the simulations will then be described. Network traffic generation and its impact on the effectiveness of the evaluation will be considered next. The model for congestion responsive traffic, a finite length closed-loop TCP traffic generator, will be introduced. The counterparts to the congestion responsive traffic model are the two different malicious traffic generation models used.

Finally, a base scenario for RAD evaluation simulation experiments will be presented. This scenario aggregates the characteristics of the evaluation topology and traffic generation models to define a core scenario from where all simulation scenarios are derived.

5.1.1 Metrics

The selection of metrics for an evaluation study is one of the most critical factors in ensuring that the performance of the multiple aspects studied in the simulation experiments is measured and evaluated in a consistent and meaningful

fashion. Selection of inappropriate or inconsistently defined metrics can lead to difficulty in evaluating results and the inability to meaningfully compare the different performance results.

Another aspect of the metric selection is the number of metrics chosen for characterising the performance of the system under study. From a consistency and understandability point of view, a single metric is a desirable choice. However, it is common when evaluating systems that have inherent trade-off goals to require more than one metric for their evaluation. An example of this would be the evaluation of AQM schemes, where network utilisation and loss rate are both needed to characterise their performance.

For this evaluation study of RAD performance a single metric, *throughput*, will be used. Throughput can be defined as the rate, in packets per second, of the flow or aggregate that are delivered successfully to the receiver. It is a queue or link level metric that provides an indication of how the different traffic types share the available bandwidth.

In the simulation scenarios in this chapter, throughput will be given as the ratio of the incoming traffic packet rate to the available link bandwidth. By comparing the throughput achieved by DoS and TCP traffic, RAD performance can be effectively evaluated, as RAD intends to maximise TCP throughput and minimise the effect of DoS traffic on the throughput achieved by TCP flows.

While throughput is a single metric, it adequately captures the evaluation needs that stem from the core requirements of RAD. The protection of TCP traffic from malicious DoS traffic being a core RAD goal, the throughput, as a ratio of the bottleneck link achieved by the TCP sources, is an accurate and consistent metric for the performance of TCP traffic. Alternative metrics such as flow fairness or queueing latency experienced have the downside of introducing additional complexity to the results. Throughput is a good proxy metric for other lower level metrics such as loss rate or queueing delay due to the behaviour inherent in TCP sources.

5.1.2 Topology

A single-bottleneck or *dumbbell* topology will be used for the evaluation scenarios. This type of topology allows the evaluation to focus on what the effect of

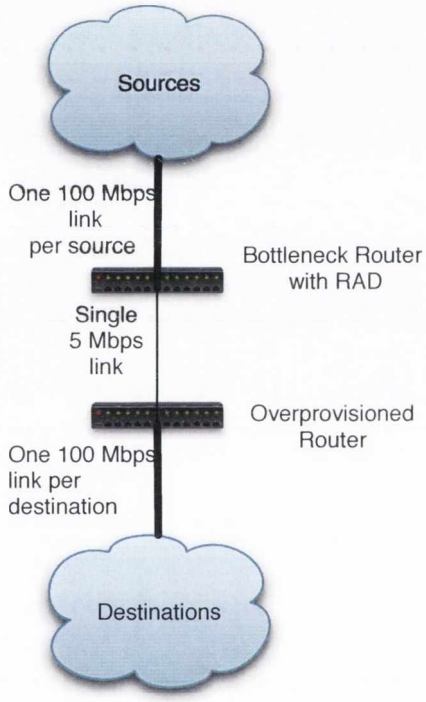


Figure 5.1: Dumbbell scenario topology.

the queueing algorithm is on the traffic at the point where congestion occurs in the path of the traffic. A variety of network configurations can be explored by varying the number of source and destination nodes in the scenario, as well as the distribution of link delays, thus changing the RTT experienced by flows.

Figure 5.1 shows the core structure of the topology used for the evaluation scenarios. The topology consists of two large sets of hosts, the source network and the destination network, plus two routers that are connected via the bottleneck link. The bottleneck routers on the side of the sources will be configured to run RED or RAD depending on the requirements of the simulations. This choice is made given that RED is the seminal Active Queue Management scheme and also the most widely available one in TCP/IP network router implementations. The configurations used compare RED-based router performance to configurations where RAD is implemented over a global queueing module operating using RED queueing. This allows for the evaluation of the impact of RAD's logic on traffic, rather than details of the global queueing scheme chosen.

The link propagation delays will be configured to provide a uniform random delay distribution between each pair of source and destination nodes. While not necessarily representative of a single real world physical network, this distribution of link delays allows for a wide variety of network round-trip times to be included in a single evaluation scenario. The bottleneck link has a fixed propagation delay of 5 ms. All the other links that connect sources to core routers or core routers to destination nodes will have a pseudo-randomly selected link delay so that the round-trip link delays experiencedby packets from the source nodes range between a minimum of 20 and a maximum of 500 milliseconds (excluding any potential queueing delay).

5.1.3 Traffic

Another key parameter in an evaluation study of a queueing scheme is the choice of traffic sources, and the types of traffic that they generate. This is an often overlooked aspect of an evaluation study, for example many authors commonly use unrealistic infinite TCP flows as a base congestion responsive traffic mix.

The models of traffic generation used in a work such as this must support both the definition of a base, "background", traffic mix as well as allowing for

use of traffic generation model parameters as aspects within the evaluation. While a realistic base traffic mix is a basic requirement, the impact of different traffic types on a queueing scheme's performance is considerable and, given the large range of traffic mixes in real networks, the evaluation must explore the parameter space of the traffic generation model.

Two main types of traffic are used in the simulation scenarios: congestion *responsive* traffic and congestion *unresponsive* malicious traffic. Congestion responsive traffic allows fair sharing of the available bandwidth. Congestion unresponsive flows prevent fair usage of bandwidth by flooding the link and not lowering their offered load when losses occur[51]. Thus, we need realistic and flexible models for modelling both the responsive traffic that RAD is designed to protect, as well as the unresponsive traffic that prevails in bandwidth flooding DoS attacks.

5.1.4 Congestion responsive traffic

TCP is a congestion responsive protocol that regulates its sending rate depending on the level of congestion estimated by detecting packet losses. For this work, the *FullTCP* TCP implementation included in the NS-2 version 2.29 simulator [20] will be used.

Apart from the core protocol implementation, a session or connection generation process is also needed for TCP traffic. The most common TCP session generation model used in queueing and congestion control related research is the one of infinite TCP sources that have always data to send and never close their connections. This is a model that approximates the behaviour of long-lived FTP transfers of big files.

However, TCP traffic in real networks will nearly always consist of finite length connections. Furthermore, due to the occurrence of Web traffic within the aggregate traffic on the Internet, most flows will only transfer a relatively small amount of data with only a few large flows that are longer-lived. Thus, the responsive traffic model should use finite-length TCP flows to create more realistic traffic.

Recent work on the congestion responsiveness of Internet traffic has argued that using TCP congestion control for individual flows is not enough to guar-

antee the responsiveness of the aggregate traffic[112]. It is important to also take into account the processes that govern creation of new flows in the system. Given an On-Off traffic model of finite TCP flows, we can say that the traffic can follow either an open-loop or a closed-loop model. In an open-loop traffic model session arrival times (new flow creation events) are independent of the completion times of previous sessions. In a closed-loop model new TCP connections will only be started after previous ones have completed. Even though both use TCP to control their rates, these two models have very different congestion responsiveness behaviour. The closed-loop model results in congestion responsive aggregate traffic, while the open-loop model can lead to persistent congestion and ultimately congestion collapse[112].

A closed-loop On-Off traffic mix of TCP flows will be used as the reference model of responsive traffic in the simulation scenarios. Each responsive traffic source host in the simulation scenario creates a new TCP connection to send ON_b bytes and once the transfer has finished it waits OFF_t seconds before it opens a new connection. By specifying ON_b in bytes instead of seconds, we achieve the closed-loop behaviour for the model, as no new connections will be created until the current one finishes. Both ON_b and OFF_t can be configured to be fixed values or a random sample from a distribution. For example, web-like responsive traffic can be modelled by using a log-normal distribution for ON_b and a constant for OFF_t [43]. The closed-loop On-Off model provides a realistic traffic representation while allowing exploration of a wide range of possible traffic mixes by changing the settings for ON_b and OFF_t.

5.1.5 Malicious traffic

Unresponsive traffic can be generally characterised as the type of network traffic that does not adjust its offered load when packet losses occur. It is considered to be *unresponsive* to packet losses in this regard. In this work, unresponsive traffic models will be used to simulate bandwidth-flooding DoS attack traffic. Malicious bandwidth-hogging DoS attacks affect responsive TCP traffic by not lowering their offered load under congestion. Thus, *unresponsiveness* is a key characteristic of flooding DoS traffic.

Another important feature of DoS bandwidth attack traffic is the ability

to spoof IP packet header fields, notably the IP source address of the packets. Spoofing helps prevent attack traffic from being traced back to its actual source host or hosts. From the point of view of evaluating a DoS defence mechanism, packet header spoofing has the crucial characteristic of allowing the attacker to present the generated DoS traffic as being composed of an arbitrary number of flows. Thus, an attacker is able to manipulate how a router that uses per-flow information perceives its traffic by using packet header spoofing. Being able to simulate the presence of an arbitrary number of flows interferes with the operation of DoS defence mechanisms that focus on the quantity and size of flows to detect malicious traffic mixes.

The DoS source implementation in the simulation scenarios has a *srcReuse* parameter that can be used to model source address spoofing. This parameter specifies how many consecutive DoS traffic packets will reuse the same source address before selecting a new random address. Thus, if $srcReuse = 4$, the DoS source will change the source address used for the generated traffic every four packets.

Two different DoS traffic models will be used in the simulation scenarios: persistent and intermittent flooding traffic.

5.1.5.1 Persistent flooding traffic

The persistent flooding traffic generation model used for this evaluation study emulates the behaviour of highly distributed DDoS attacks, where many compromised hosts generate large amounts of traffic towards a single victim. This type of traffic does not require particularly sophisticated traffic generation methods as it impacts the victim by the sheer amount of traffic generated.

Persistent DoS traffic will be implemented in the simulation experiments using an infinite constant-bit rate (CBR) source that causes extreme overload at the bottleneck link until the responsive traffic is starved of any share of the available bandwidth. This is the simplest type of DoS traffic, a high-bandwidth flow that does not considerably vary its offered load and keeps a constant rate independently of the loss rate in the bottleneck queue.

As noted above, malicious DoS traffic sources can also make use of packet header spoofing. Packet header spoofing does not impact the ability of a DDoS

attack to generate large quantities of traffic towards a victim. However, it can have a significant impact on the performance of any system that analyses the attack traffic, as it allows the attacker to apply stealth techniques to its generated traffic. The persistent flooding traffic generator used in this study uses the *srcReuse* parameter to allow for the exploration of different packet header spoofing strategies.

5.1.5.2 Pulsing attack traffic

Intermittent DoS traffic[143] is a flooding DoS variant that introduces greater sophistication to the attack traffic model. Intermittent DoS works by generating extremely large offered load for very short time periods, and being "silent" the rest of the time. It is an extremely bursty type of traffic that causes traffic generation patterns that are not distinguishable from well behaved traffic at large timescales. By only being active for very short bursts, intermittent DoS can affect existing TCP traffic while escaping detection by filtering devices that may not operate at a time scale that allows its detection.

Intermittent DoS traffic will be modelled in the simulation scenarios by an On-Off CBR traffic model that generates extremely high offered load over very short ON times and has long OFF periods. Adequate parameter settings would be a rate of 5 times the bottleneck bandwidth for an ON period of 50 ms. and OFF period of 950 ms. Traffic generated with these parameters will impact TCP traffic at time scales lower than typical round-trip times in the Internet [143].

5.1.6 Base scenario

All simulated scenarios in this evaluation of the RAD queueing scheme share some configuration properties, with changes introduced as required by the goals of each particular evaluation. The topology used is a single bottleneck dumbbell topology as described in Section 5.1.2, with 5 Mb per second bottleneck link bandwidth and 100 Mb per second links connection source and destination hosts with routers on both sides of the bottleneck link.

The base traffic mix will consist of TCP Sack [95] sources with a closed-loop on-off behaviour. The TCP sources transmit ON_b bytes of data during their ON

RED Parameter	Value
$qWeight$	0.001
$maxProb$	0.1
$minTh$	10
$maxTh$	50

Table 5.1: RED parameter values used in RAD evaluation base scenario.

RAD Parameter	Value
b_t	100000
RTT	250 ms.
Δ	0.02 s.
α	0.01
in_{th}	1
R_{min}	2
R_{max}	3
F_{min}	1
F_{max}	3

Table 5.2: RAD parameter values used in evaluation base scenario.

period and when this transfer is finished they enter their OFF period, in which they wait OFF_t before starting a new ON period. There will be 400 TCP source nodes, each paired with their respective destination nodes. ON_b will be set to 100000 bytes and OFF_t will be set to 10 seconds. The TCP sources use the Sack [95] TCP variant, with a maximum congestion window of 40 packets and with delayed ACKs [4] disabled. The TCP sources will be configured to start at a random time during the initial 10 seconds of each simulation to avoid TCP source synchronisation effects caused by all sources starting at the beginning of the simulation.

RED will be configured with the default settings as shown in table 5.1. The $qWeight$ parameter defines the speed at which the average queue metric tracks the instantaneous queue occupancy. The $minTh$ parameter is the lower bound for the moving average queue occupancy, below which no random dropping probability is applied. The $maxTh$ specifies the value of the average queue at which RED applies a dropping probability of $maxProb$. RAD specific parameters are described in table 5.2. It will be configured with a default number of ten aggregates considered in the traffic measurement metrics. This relatively low number of aggregates is used as it allows the study of the impact low numbers of aggregates can have on traffic that maps to penalised aggregates.

Both bandwidth flooding persistent DoS and pulsing DoS traffic will be evaluated in the simulation scenarios. For persistent flooding traffic, a single constant bit-rate (CBR) flow will be used with an offered load of 10 Mbps (twice the bottleneck link capacity). Pulsing DoS will be implemented by default on the simulation scenarios as an ON-OFF constant bit-rate source that will generate an offered load of 10 Mbps during ON periods of 0.05 seconds, and will not generate any traffic during OFF periods of duration 0.95 seconds, as described in Section 5.1.5.2 above.

All simulations will run for 250 seconds. The base traffic mix of closed-loop TCP sources will be started at random times within the initial 10 seconds and will be active during the entire duration of the simulation. For simulations that have active DoS traffic, this will start 50 seconds after the beginning of the simulation and be active until 50 seconds before the end of the simulation. Whenever throughput results are provided as averages during a simulation run, they will be calculated using only the second half of the period in which the DoS traffic is active, thus between seconds 100 and 200 of the simulation. This is done to avoid influencing the results with both TCP startup effects and fluctuations introduced by DoS traffic startup.

One important aspect of any simulation-based evaluation scenario and its associated results is that the raw output obtained from it are statistical samples [109]. This requires the simulations to be carefully configured and their output results correctly analysed to be able to reach valid conclusions.

In this evaluation, the base scenario described above has been designed so that any potential bias in its results are minimised. The network topology includes randomised round-trip times. The traffic generation processes avoid causing global synchronization effects by randomly selecting the start time of each source. Furthermore, the results presented in this chapter are typically produced by averaging samples obtained from the simulator, with the exception of those sections where short-term queue dynamics are studied. These samples are also taken after event generation agents have had time to achieve steady-state behaviour, thus avoiding the impact of initial startup effects in the results.

5.2 Evaluated aspects

This section will present and discuss results from multiple simulation experiments with the goal of thoroughly evaluating Responsive Aggregate Defence (RAD) performance. In these experiments the chosen metric, TCP traffic throughput, will be used. The throughput provided by RAD under different scenarios will be compared to a baseline performance obtained using the well known RED [54] AQM algorithm.

The range and possible values of variables and queueing scheme parameters that can impact on the performance of a DoS defence router queueing mechanism, such as RAD, is large. The set of possible aspects can be divided into two groups depending on whether they are internal or external to the mechanism under study:

External aspects These aspects exist in the environment of the queueing algorithm under study, and include aspects such as the network topology and traffic mix experienced. The study of external aspects is required to validate the performance of the scheme under scenarios that are relevant to the operational environment in which it is expected to be deployed.

Internal aspects These aspects are internal to the deployment and configuration of the scheme, including parameters and constraints arising from the deployment configuration in a physical network router. These aspects must be explored to evaluate the robustness of the design and to provide parameterisation recommendations for optimum performance in a variety of scenarios.

When selecting external aspects that should be considered there are two main categories that are evident from the main goal of RAD. These are RAD performance when experiencing malicious DoS traffic, and its ability to be transparent to well-behaved TCP flows in the absence of malicious traffic. These two categories are sufficient to verify RAD performance for the primary traffic mix scenarios it was designed for.

Internal aspects to be evaluated in this section will include the main parameters that RAD uses to characterise congestion responsive traffic as well as

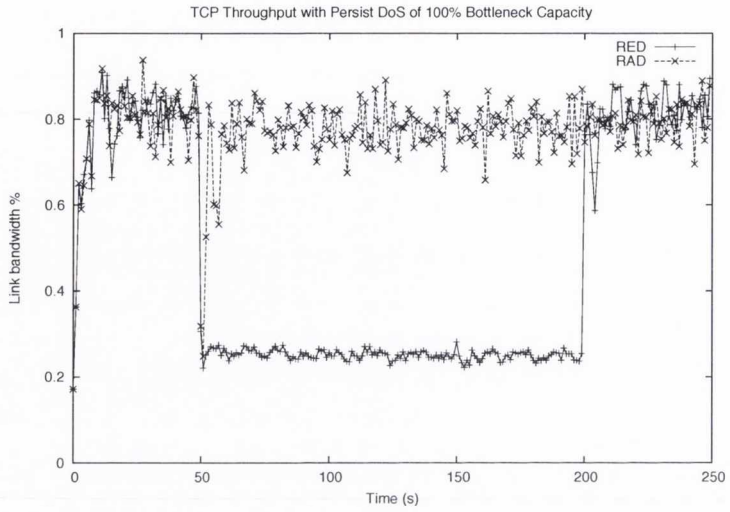


Figure 5.2: Instantaneous TCP throughput before, during and after a persistent DoS attack with RED and RAD.

core parameters that tune the DoS penalisation logic in the RAD architecture. In the following sections the performance of RAD under malicious traffic, TCP traffic and a spectrum of parameter values will be evaluated.

5.2.1 Malicious traffic penalisation

It is the primary goal of RAD to detect, identify and penalise malicious bandwidth flooding DoS attack traffic. This section will explore RAD performance at achieving this goal.

RAD performance will be evaluated under different configurations and types of malicious DoS traffic when mixed with an existing background traffic mix of finite length close-loop TCP flows. The throughput achieved by TCP traffic at the router queue will be the main result metric studied. RAD and RED performance will be compared, with and without DoS traffic, in the scenario traffic mix. This simulation configuration provides an effective view of the impact of a DoS attack, allowing for evaluation of its impact at attack start time, during the attack lifetime, as well as the full recovery of TCP throughput once the attack traffic has stopped.

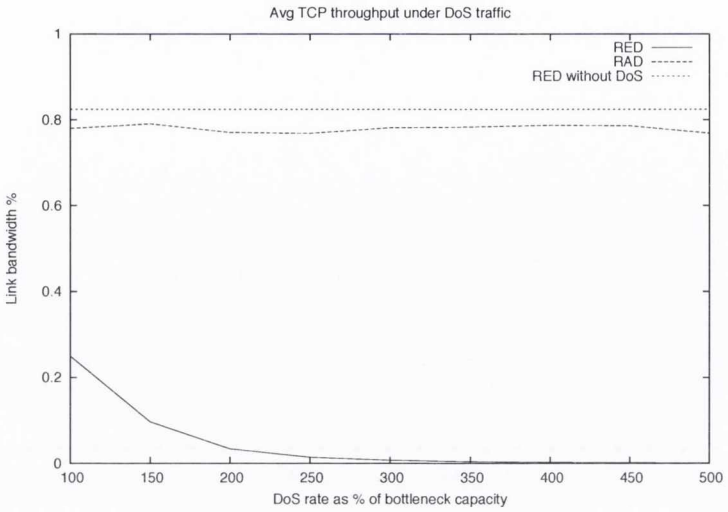


Figure 5.3: Average TCP throughput comparison without DoS traffic and with persistent DoS traffic of a range of offered loads with RED and RAD.

5.2.1.1 Persistent flooding DoS attacks

The most common bandwidth DoS attack traffic will be evaluated here, comparing the impact of a DoS attack consisting of high-rate persistent attack traffic load on a background mix of congestion responsive TCP traffic. Figure 5.2 shows TCP throughput over time with RAD and RED for simulation scenarios in which the DoS attack traffic rate joins the scenario traffic mix 50 seconds after the beginning of the simulation and stops at 200 seconds from the simulation start. The DoS attack traffic is implemented as a constant bit-rate (CBR) UDP source that is configured with a source address spoofing policy that replaces the source address employed every four packets sent.

This graph shows clearly how RED is unable to protect TCP throughput between seconds 50 and 200, when the drop rates caused by persistent DoS traffic at the bottleneck queue force TCP sources to quickly lower their offered load, achieving a TCP throughput of slightly over 20% of the available link capacity. With RAD however, TCP throughput is quickly restored to levels close to those before the onset of DoS traffic.

Figure 5.3 shows the average obtained TCP throughput during persistent DoS attacks of different intensities. These results compare TCP throughput obtained by RAD during a persistent DoS and that achieved by RED both

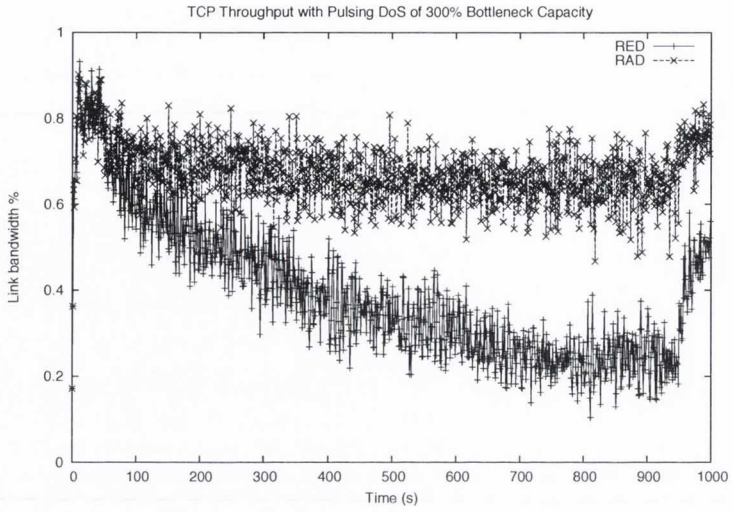


Figure 5.4: Instantaneous TCP throughput before, during and after a pulsing DoS attack with RED and RAD.

during a DoS attack (between seconds 50 and 200) and alongside a situation without attack traffic, as a top-line performance target.

These scenarios show how a RED queue is not able to provide good throughput to TCP traffic during a persistent DoS attack. The achieved TCP throughput goes quickly towards zero for DoS attack loads of more than three times the available bottleneck link capacity. Both figure 5.2 and 5.3 show how RAD is able to quickly and reliably restore most of the TCP throughput that would have been lost with RED. RAD is able to provide TCP throughput that is within 10% of the pre-DoS value for a wide range of the incoming persistent DoS traffic loads.

5.2.1.2 Pulsing DoS attacks

Low-rate pulsing DoS attacks that target TCP flows have already been identified as an evolution of traditional bandwidth flooding DoS attacks. Their behaviour of generating highly bursty traffic at short time scales, without adopting a flooding DoS-like profile at larger time scales. This allows pulsing DoS attacks to escape detection by many DoS defence mechanisms. Even though pulsing attacks are not yet common on the general Internet, this evaluation chapter will study the performance of RAD under such conditions as they may well become

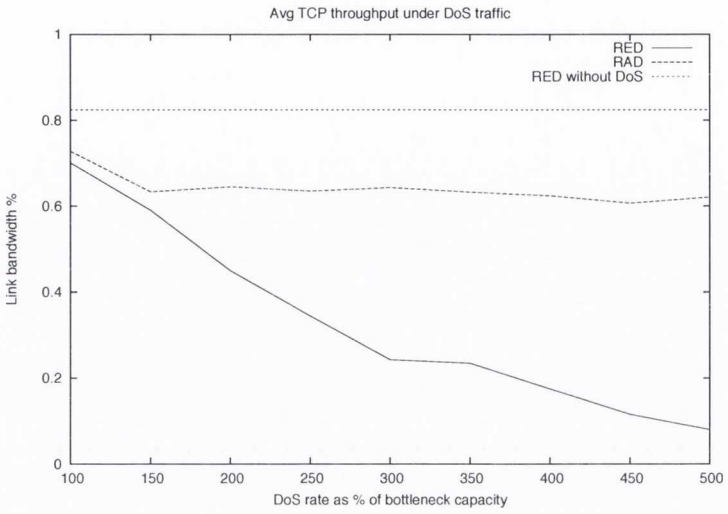


Figure 5.5: Average TCP throughput comparison without DoS traffic and with pulsing DoS traffic of a range of offered loads with RED and RAD.

a practical reality in the near future.

Pulsing attacks impact congestion responsive traffic by causing TCP flows to use retransmission timeouts. This causes a slower decrease of TCP throughput that is more difficult to detect, as well as being caused by a traffic aggregate that does not seem to be aggressive when viewed at larger time-scales.

Figure 5.4 compares TCP throughput obtained by RED and RAD under a pulsing DoS attack traffic that sends three times the available bottleneck link capacity during ON periods. The DoS traffic is only active between seconds 50 and 950 of the simulation. These results show how pulsing DoS impacts TCP performance in a very slow and stealthy way, but the aggregate burstiness module of RAD is able to protect TCP throughput without noticeable degradation in the long term.

A range of pulsing DoS ON period traffic rates are explored in figure 5.5. This graph compares RAD achieved average TCP throughput during a pulsing DoS attack with RED, both in pulsing DoS and no-DoS scenarios. These results show how pulsing attacks are much more difficult to detect and defend against. RAD is able to protect a large part of the TCP throughput from the impact of DoS, but pulsing DoS does cause an impact, even at quite low long-term average DoS incoming rates.

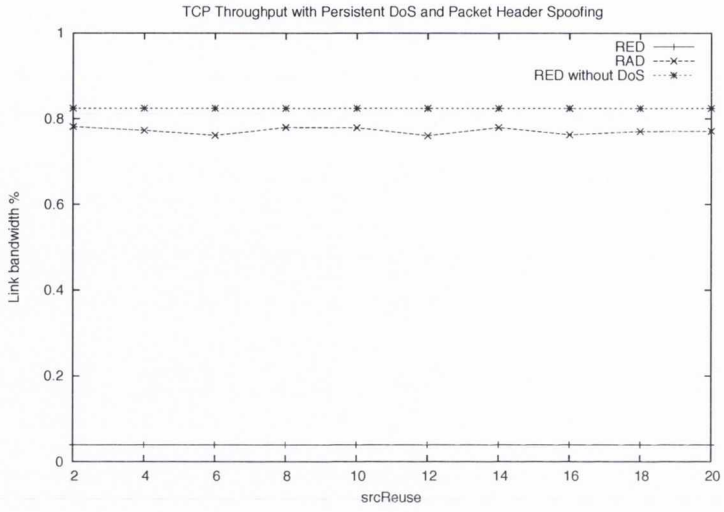


Figure 5.6: Average TCP throughput comparison for RED and RAD without DoS traffic and with persistent DoS traffic under a range of packet header spoofing strategies.

5.2.1.3 Packet header spoofing

Packet header spoofing is a capability easily available to malicious attackers that seek to launch bandwidth flooding DoS attacks. Header spoofing facilitates two aspects of flooding DoS attacks. It:

- Prevents discovery of the actual source of the attack traffic stream when source address spoofing is used.
- Allows the attack traffic to present itself as an arbitrary set of flows by changing packet headers, thus potentially allowing the attack to prevent its detection by DoS defence schemes that rely on using attack traffic packet information for identification.

RAD uses only the destination and source addresses of arriving packets. The destination address is used for aggregate mapping purposes, and the source address is used for aggregate flow count measurements. It is assumed that the destination IP address is not spoofed, as it is the packet header field the attacker relies on to get its traffic to the victim. Thus, the source address is the only header field used that can potentially allow the attacker to circumvent the RAD DoS defence mechanism.

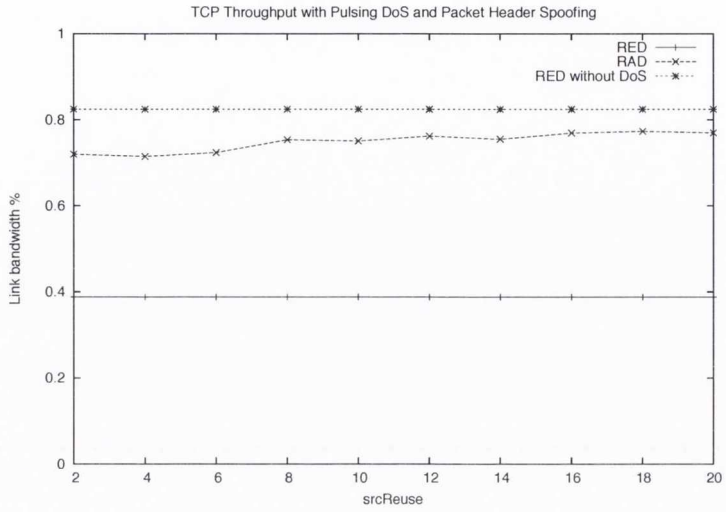


Figure 5.7: Average TCP throughput comparison for RED and RAD without DoS traffic and with pulsing DoS traffic under a range of packet header spoofing strategies.

This section will evaluate RAD performance using different source address spoofing levels. The simulation scenarios implemented for evaluating RAD under source address spoofing use the *srcReuse* parameter, varying its setting to study the impact of different source spoofing levels on RAD performance. Figure 5.6 plots the average TCP throughput achieved during a persistent bandwidth flooding DoS attack of twice the bottleneck link capacity for (RED and RAD) using different values for *srcReuse*. A reference value for TCP with RED when no DoS attack is underway is also shown. Figure 5.7 presents the same simulation scenario results, with a pulsing attack that provides twice the bottleneck link capacity for ON periods of 0.05 seconds and with OFF periods of 0.95 seconds.

These results show how RAD is very robust under different source address spoofing scenarios, and maintains its TCP throughput protection above that of RED for all evaluated values of *srcReuse*. RAD performs better under header spoofing with persistent flooding. With pulsing traffic RAD TCP throughput varies under some spoofing configurations. This behaviour is explained by the fact that (i) most of the TCP protection for pulsing low-rate attack traffic is performed at the Aggregate Burstiness phase of RAD, (ii) the Aggregate Responsiveness phase does not see the long-term aggregate behaviour of the

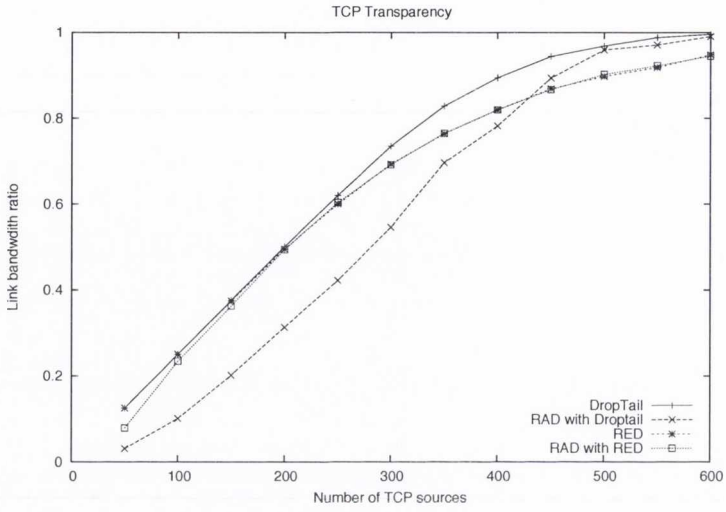


Figure 5.8: Average throughput achieved by TCP traffic comparison between DropTail, RED and RAD for a set of traffic mixes composed of a range of active closed-loop TCP sources from 50 to 600.

attack traffic as deviating a lot from expected TCP behaviour, and (iii) the aggressiveness of the traffic is further masked when source address spoofing is used by the attacker.

5.2.2 TCP traffic transparency

It should be a primary goal of any DoS defence mechanism to be as transparent as possible when no DoS attack is underway and the link is being utilised by congestion-responsive TCP traffic. This is clearly also a goal for RAD at the same time as effectively protecting TCP aggregates at times of attack, it should minimise its impact on throughput of well-behaved TCP traffic when the link is not experiencing a bandwidth DoS attack. RAD should be *transparent* for TCP traffic in these scenarios.

The transparency of RAD is evaluated by comparing TCP throughput of RAD (with RED or DropTail global queueing), Droptail and RED on scenarios that use a traffic mix of closed-loop ON-OFF TCP sources. Figure 5.8 compares average throughput for a set of scenarios with only TCP traffic and different levels of link utilisation, achieved by varying the number of TCP sources at each scenario.

The results show that utilisation levels achieved by RAD depend on both the level of congestion at the link as well as the underlying global queueing scheme in use together with RAD. When used with DropTail, RAD causes lowered link utilisation for uncongested scenarios, while its performance converges to be the same as DropTail under high congestion, when link utilisation is closer to 1. RAD performs better when used with RED, in this case its utilisation is consistently very close to what is obtained by just using RED.

Independent of the global queueing chosen, RAD has a negative impact on link utilisation when incoming traffic rates are low. This suggests that a policy of disabling or limiting the impact of RAD when congestion levels are very low could be successful as a practical deployment strategy. These scenarios are the ones least likely to consist of DoS bandwidth attacks, so RAD could safely be disabled by network operators at times when the link utilisation is below a given minimum threshold. This is a safe choice as high link utilisation and non-zero packet drop probabilities are necessary conditions for DoS attacks to achieve their impact on TCP traffic.

5.2.3 Parameter sensitivity

RAD can be configured with a set of parameters that control either the application of the TCP model or the level of penalisation applied by it to incoming traffic identified as malicious. This section explores how the different RAD parameters interact with the traffic in the queue.

The selection criteria for parameters to focus on is based on the core goals of RAD. RAD must be an accurate and effective queueing algorithm at both differentiating DoS traffic from congestion responsive traffic, and also at penalising DoS traffic while minimising its impact on existing TCP flows in the traffic mix.

Given these requirements, the RAD parameters evaluated here include:

- The *RTT* parameter, influences the assumptions made by the RAD traffic characterisation algorithm regarding the average experienced round-trip time by TCP flows traversing the link.

- The average connection size, configures the RAD traffic characterisation

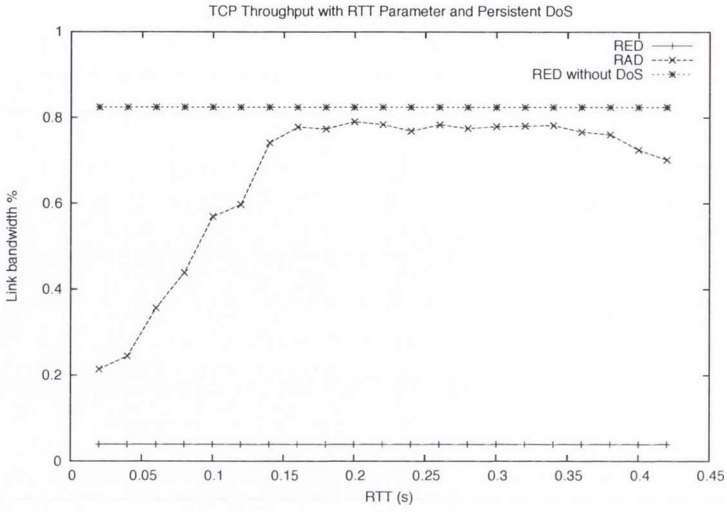


Figure 5.9: Average TCP throughput comparison for RED and RAD without DoS traffic and with persistent DoS traffic under a range of values for the RTT parameter between 0.03 and 0.42 seconds.

algorithm's input regarding the expected average length in bytes of the TCP connections that generate traffic towards the link.

- The Delta (Δ) parameter, for the RAD measurement metrics component, controls how quickly RAD measurements of traffic aggregate behaviour track changes in incoming traffic.

- The thresholds R_{min}, R_{max}, F_{min} and F_{max}, used in the Aggregate Responsiveness module to drive penalisation decisions on aggregate traffic that is suspected of containing malicious DoS traffic (determined by its deviation from expected TCP behaviour).

- The threshold in_{th}, used in the Aggregate Burstiness module to detect extremely bursty traffic aggregates that could be impacting TCP traffic at very short time scales (while not being detected as aggressive by the Aggregate Responsiveness module).

The resultant analysis is a thorough evaluation of the main RAD configurable parameter space and its impact on RAD performance.

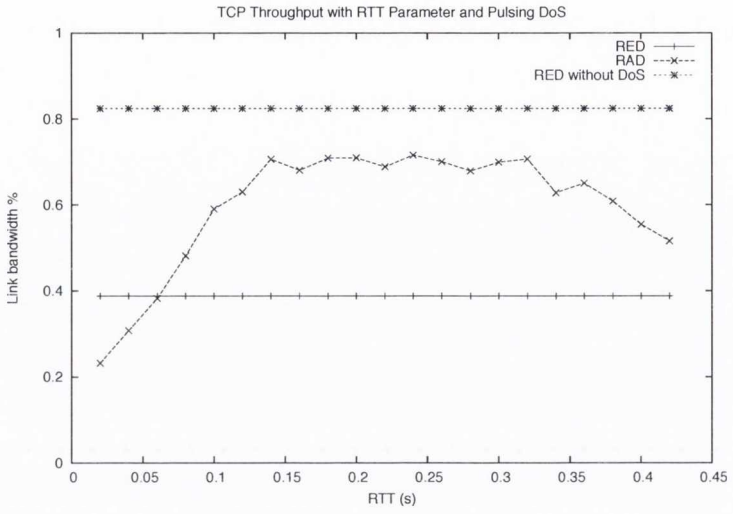


Figure 5.10: Average TCP throughput comparison for RED and RAD without DoS traffic and with pulsing DoS traffic under a range of values for the *RTT* parameter between 0.03 and 0.42 seconds.

5.2.3.1 RTT

RAD uses an *RTT* parameter that denotes the default round-trip time applied in the Aggregate Responsiveness module to obtain the expected average behaviour in the long term of congestion-responsive TCP aggregates. The impact of the *RTT* parameter is evaluated by comparing TCP throughput between RED and RAD on a set of common persistent and pulsing DoS attack scenarios, and with different settings for the *RTT* parameter for RAD.

The actual round-trip values experienced by flows will commonly be anywhere from around 20 ms. for connections within a single Wide Area Network to around 130 ms. for inter-continental connections. The *RTT* parameter values used are not meant to focus only on these common values, but wider ranges are explored to obtain a better understanding of RAD's performance under different configurations.

Figures 5.9 and 5.10 show simulation results denoting the impact of the *RTT* RAD parameter for persistent DoS and pulsing DoS respectively. The plots display the average TCP throughput during a DoS attack with RED and RAD. A third result is also depicted, which is the average TCP throughput with RED for the same simulation scenario but without DoS traffic. This illustrates

what the upper bound for TCP throughput would be if the DoS traffic was not present.

The evaluation results show that there are a range of values for RTT that maximise the performance of RAD. RTT values roughly between 0.15 and 0.35 provide the best TCP throughput when using RAD under DoS attack conditions, both for persistent as well as pulsing attacks. RAD's performance decreases as RTT extends outside of this range. Lower RTT settings adversely affect RAD performance for both DoS traffic types and higher RTT values impact especially performance in the case of pulsing attacks.

These results have quite a clear explanation. The simulation scenarios are configured so that the traffic sources are connected to links from source to destination hosts that have a spread of round-trip times between 20 and 500 ms. Thus, the TCP sources generating traffic traversing the bottleneck link will experience round-trip times in that same range. The RAD model RTT parameter value will be most appropriate the closer it is to the average round-trip time experienced by flows traversing the link. This is why RAD performance is impacted when RTT is set to values farther from the average round-trip time experienced by the TCP flows. When RTT is set too low, it will allow an excess of aggressive traffic traverse the link, and when it is set too high it will cause over-penalisation of non-malicious TCP traffic that is achieving round-trip times much lower than the RTT value.

5.2.3.2 Average Connection size

RAD uses the parameter b_t (which indicates the value used by the underlying TCP model) for the average number of data bytes sent by each connection. In the base evaluation scenario, b_t is set to $100,000$ bytes. However, the distribution of TCP connection data sizes in real networks will vary depending on the network and will evolve with time. It is important that RAD is robust enough to not be strongly affected by differences between the expected average number of bytes on TCP connections traversing the link and the actual distribution of real traffic over time.

Figures 5.11 and 5.12 compare RED and RAD performance using traffic mixes with different average number of bytes during the data transfer periods of

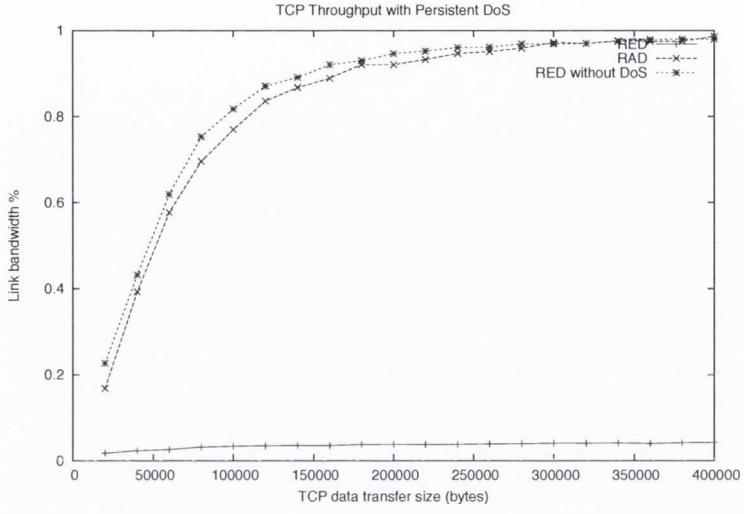


Figure 5.11: Average TCP throughput comparison for RED and RAD without DoS traffic and with persistent DoS traffic under a range of values for the average connection size parameter between 25000 and 400000 bytes.

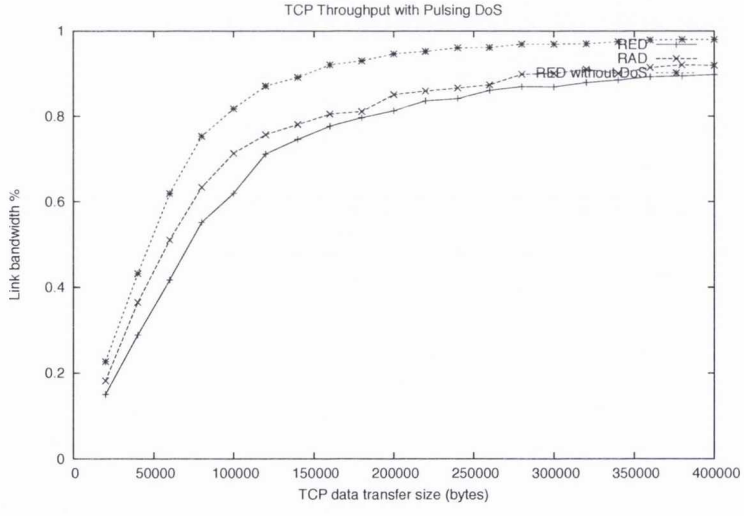


Figure 5.12: Average TCP throughput comparison for RED and RAD without DoS traffic and with pulsing DoS traffic under a range of values for the average connection size parameter between 25000 and 400000 bytes.

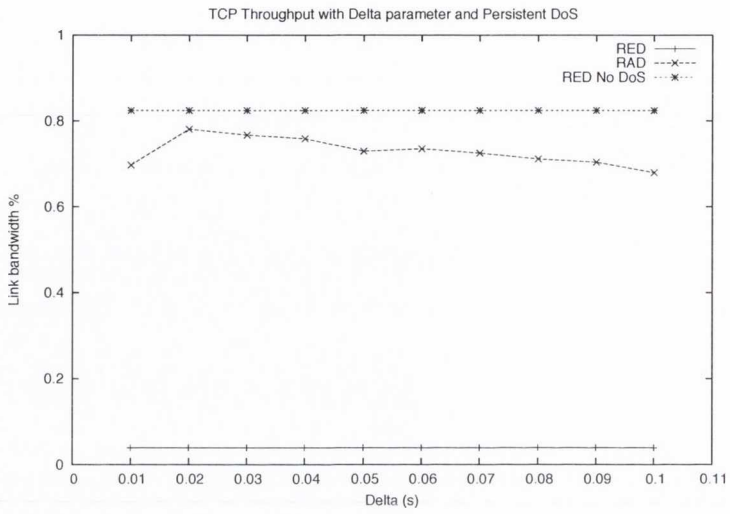


Figure 5.13: Average TCP throughput comparison for RED and RAD without DoS traffic and with persistent DoS traffic under a range of values for the Δ parameter between 0.01 and 0.1 seconds.

TCP connections. The results show that the performance of RAD is stable when under persistent DoS attack with a varying number of connection bytes. In the case of pulsing DoS traffic, RAD performs better when the TCP connections in the traffic mix transfer less data, with decreasing TCP throughput as transfers are larger.

This behaviour can be explained by the "memory" effect of having larger, longer running, TCP connections. For smaller TCP connections, their shorter duration means that any loss caused by pulsing DoS traffic will affect the TCP congestion window for a shorter time than it will with larger connections that will be active longer due to their larger payloads.

5.2.3.3 The measurement period length (Delta)

RAD performs traffic measurements at a fixed time interval of Δ seconds. This Δ parameter indicates the frequency with which RAD counters will be reset, as well as when average aggregate measurements will be updated. A set of different Δ configurations are evaluated in figures 5.13 and 5.14 for persistent and pulsing DoS defence respectively. The results show that the TCP throughput achieved by RAD is not largely sensitive to Δ settings for scenarios with persistent DoS attack traffic. This is expected, as for persistent DoS penalisation RAD relies

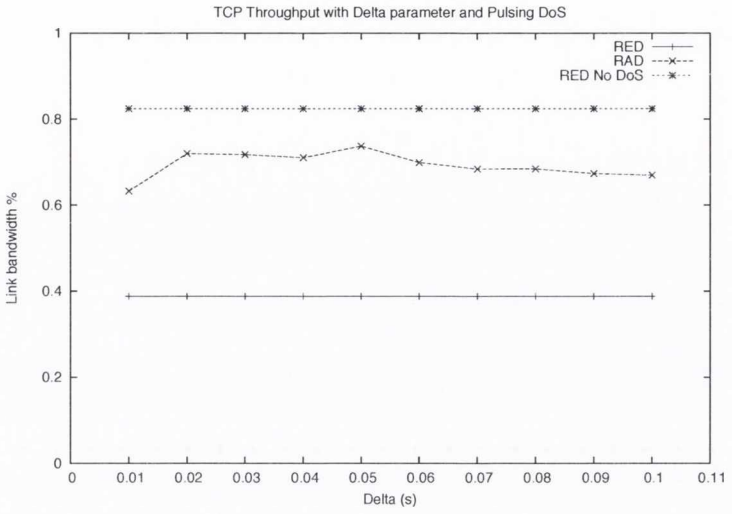


Figure 5.14: Average TCP throughput comparison for RED and RAD without DoS traffic and with pulsing DoS traffic under a range of values for the Δ parameter between 0.01 and 0.1 seconds.

on the Aggregate Responsiveness module, which focuses on larger time scales and is not dependant on being able to detect short-term traffic changes.

Pulsing DoS scenarios cause RAD performance to be more sensitive to Δ configurations. RAD performance suffers when defending against pulsing DoS if Δ is set higher than 0.05 seconds for the scenarios evaluated. This behaviour is explained by the fact that pulsing DoS is mainly penalised by RAD in the Aggregate Burstiness module, and larger Δ values prevent RAD from quickly reacting to very high-rate but short DoS traffic bursts. Thus, it is recommended that Δ should be set in the range of 0.02 and 0.05 to allow RAD to perform well against pulsing DoS attacks.

5.2.3.4 Aggregate Responsiveness ratio thresholds

The RAD Aggregate Responsiveness module utilises a set of threshold parameters applied to the differences between the actual behaviour of a given aggregate and the model prediction. These parameters are R_{min} and F_{min}, the minimum thresholds for the ratio between actual and predicted rate, and number of flows; and R_{max} and F_{max}, the equivalent upper bound thresholds.

Figures 5.15 and 5.16 evaluate how different values between 1 and 3 for R_{min} and F_{min} affect RAD performance for persistent DoS and pulsing DoS

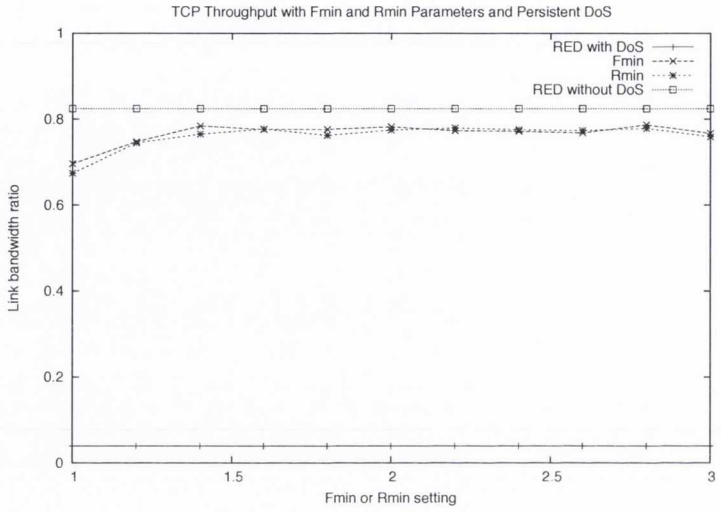


Figure 5.15: Average TCP throughput comparison for RED and RAD without DoS traffic and with persistent DoS traffic under a range of values for the R_{min} and F_{min} ratio threshold parameters between 1 and 3.

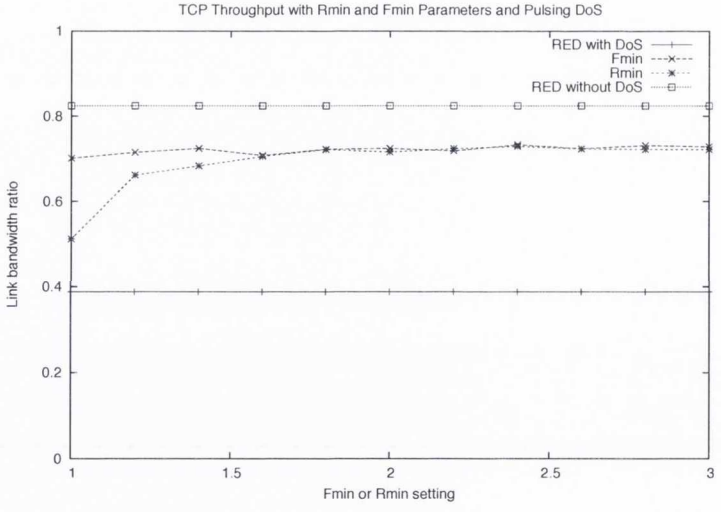


Figure 5.16: Average TCP throughput comparison for RED and RAD without DoS traffic and with pulsing DoS traffic under a range of values for the R_{min} and F_{min} ratio threshold parameters between 1 and 3.

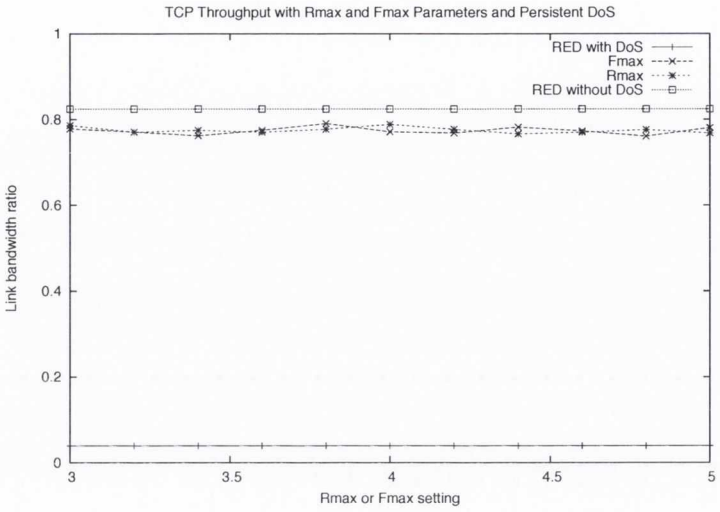


Figure 5.17: Average TCP throughput comparison for RED and RAD without DoS traffic and with persistent DoS traffic under a range of values for the R_{max} and F_{max} ratio threshold parameters between 3 and 5.

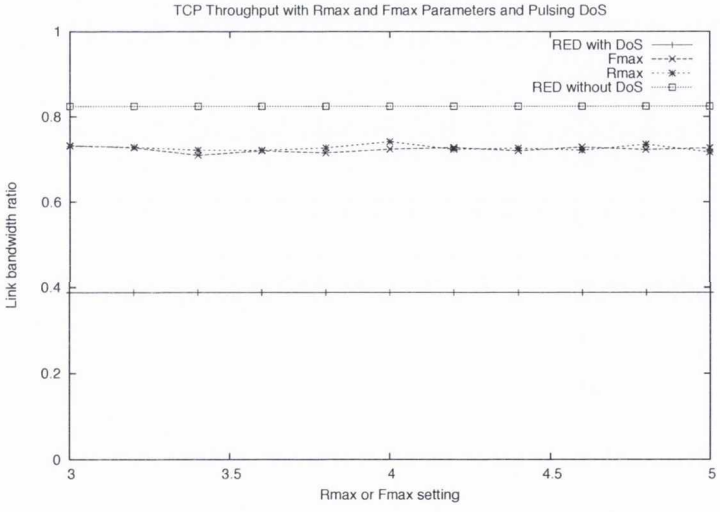


Figure 5.18: Average TCP throughput comparison for RED and RAD without DoS traffic and with pulsing DoS traffic under a range of values for the R_{max} and F_{max} ratio threshold parameters between 3 and 5.

respectively. These results show how, for persistent DoS, there is a slight penalisation for $R_{min} < 1.5$ and $F_{min} < 1.5$, with stable RAD performance for values over 1.5. When considering pulsing DoS traffic, TCP throughput with RAD is severely impacted for low R_{min} settings ($R_{min} < 1.5$), with RAD performance being poorer than RED when $R_{min} = 1$. Larger values of both thresholds do not noticeably affect RAD performance with pulsing DoS. This low performance, with $R_{min} < 1.5$ and pulsing DoS, can be attributed to the higher burstiness of both the attack and TCP traffic during pulsing attacks that introduce extremely short time-scale bursts.

R_{max} and F_{max} are the other two thresholds applied to the ratio of actual and expected rates and flows by the Aggregate Responsiveness modules. Figures 5.17 and 5.18 explore their impact on RAD performance under persistent and pulsing DoS. The TCP throughput achieved with RAD is remarkably stable for all settings evaluated for R_{max} and F_{max} between 3 and 5.

The optimal settings for all four threshold parameters should provide as narrow a range as possible for the divergence between actual and predicted rate and flow metrics, while ensuring protection and transparency for non-malicious TCP traffic traversing the link. Thus, the evaluation results above lead us to recommend the following settings for the RAD thresholds: $R_{min} = 2$, $F_{min} = 1$, $R_{max} = 3$ and $F_{max} = 3$. These settings provide a good trade-off between maximising DoS defence while ensuring high TCP transparency at times of no DoS attack.

5.2.3.5 Aggregate burstiness threshold

The Aggregate Burstiness module is in charge of penalising any short term extreme burstiness that may be caused by low-rate pulsing DoS attack traffic. This module uses one main parameter, in_{th}, which affects the maximum number of packets that will be allowed during one measurement period as a ratio of the available link capacity.

Figures 5.19 and 5.20 evaluate RAD achieved TCP throughput under DoS attacks with persistent and pulsing DoS respectively. RAD is compared with RED and a scenario in which RED is used but there is no ongoing attack. These results show that non-extreme settings for in_{th} do not overly impact

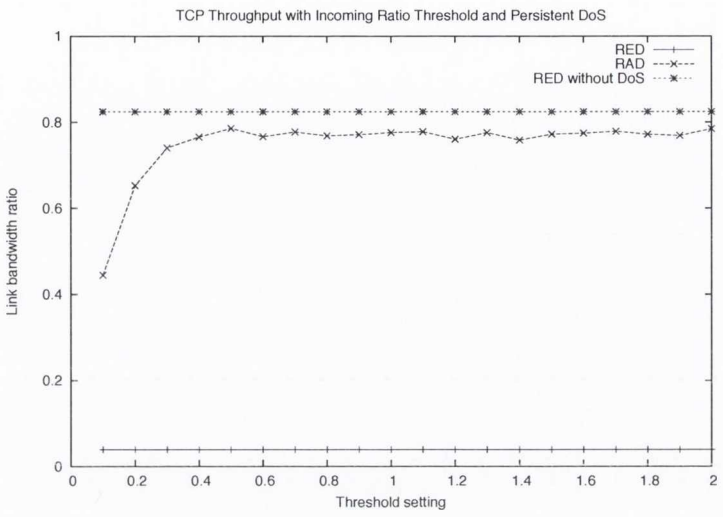


Figure 5.19: Average TCP throughput comparison for RED and RAD without DoS traffic and with persistent DoS traffic under a range of values for the in_{th} ratio threshold parameter between 0.1 and 2.

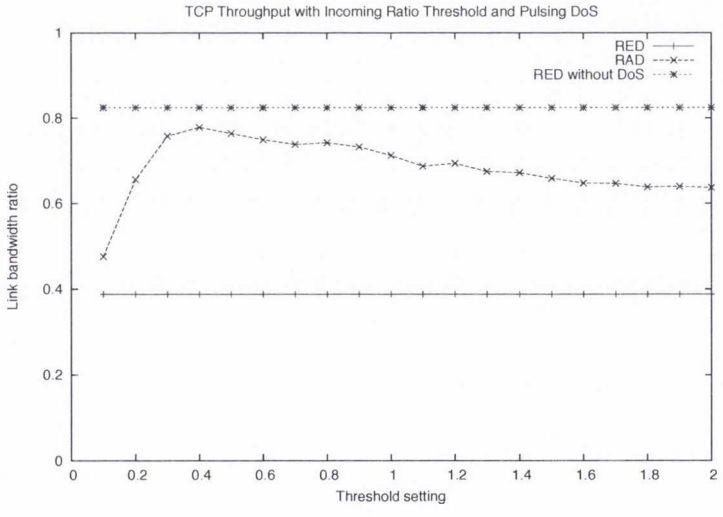


Figure 5.20: Average TCP throughput comparison for RED and RAD without DoS traffic and with pulsing DoS traffic under a range of values for the in_{th} ratio threshold parameter between 0.1 and 2.

RAD performance under persistent DoS.

For low-rate pulsing DoS attacks, in_{th} settings have a much more pronounced impact on the performance of RAD. Excluding extremely low values, the ability of RAD to penalise pulsing DoS attack traffic and defend TCP decreases as the in_{th} setting used is increased, with RAD performance comparable to RED for $in_{th} = 2$.

The optimal setting for in_{th} is a function of the expected distribution of traffic across aggregates at the queue, and the likelihood of individual non-malicious aggregates transmitting large bursts of packets that monopolise the link capacity during individual measurement periods. If the typical traffic mix at the queue is highly-multiplexed and spread between the different aggregates, a low value of $in_{th} = 0.4$ will provide very high pulsing DoS attack protection. However, for links with more sparse or very skewed traffic mixes, it would be recommended for in_{th} to be set closer to 1 to avoid accidentally penalising congestion responsive but bursty traffic.

5.3 Conclusion

This chapter has provided an extensive evaluation of the Responsive Aggregate Defence (RAD) queueing mechanism using network simulation to explore RAD performance under a wide range of conditions.

The core of the RAD performance evaluation consists of a set of base scenario network topology and traffic mix configurations, coupled with a consistent metric used throughout the evaluation study. This allows for different simulation experiment to focus solely on the impact of the aspect under study, and for the results obtained in different experiments to be readily compared to each other.

It is the main goal of RAD to be able to detect and penalise malicious DoS traffic at the router queue. Consequently, a substantial study of RAD performance, under a variety of different flooding DoS attack traffic models, is also carried out in this chapter. The performance of RAD at protecting TCP traffic is evaluated under persistent DoS traffic, pulsing DoS traffic and use of packet header spoofing techniques.

For all of these DoS scenarios, results show that RAD is able to improve upon RED performance in every case, providing TCP traffic with a throughput that is close to that of a scenario without attack traffic. The hybrid approach used in RAD is validated as an effective way of quickly detecting and penalising malicious bandwidth DoS traffic independently of the time scale at which it operates.

For RAD it is as important to be effective against DoS traffic as it is to avoid impact on congestion responsive TCP traffic when no malicious DoS traffic is present. An evaluation study of the transparency of RAD to TCP traffic is carried out. The results from this study show that RAD is largely transparent to TCP flows under most scenarios, and its performance is very close to RED when no DoS attack is ongoing.

Having evaluated the effectiveness and safety of RAD under a variety of traffic scenarios, we focus on the robustness of RAD within its deployment configuration. RAD is a flexible scheme that provides some parameterisation for improved performance in different configurations. Thus, it is important that the robustness of RAD, with respect to parameterisation, is evaluated and good guidelines are provided for the different parameters available in RAD.

The final part of the evaluation section studies the impact of RAD parameters on its performance. The most important RAD parameters are evaluated using a wide range of possible settings and their effect on RAD performance is determined. Overall, the RAD architecture and design proves quite resilient to misconfiguration of parameters. RAD performs well under most parameter configurations, with expected degradation of performance at the extremities of the parameter space.

To conclude, this RAD Performance Evaluation has provided an in-depth investigation of the factors that affect RAD performance under a wide variety of configurations. The results from this study show that RAD is an effective and robust DoS defence scheme that maintains good performance at protecting TCP flows from a variety of malicious bandwidth flooding DoS traffic types.

Chapter 6

Conclusion

In this dissertation the problem of bandwidth flooding Denial of Service attack detection and defence has been addressed. These attacks take advantage of the congestion responsive nature of TCP to prevent legitimate sources from accessing Internet services. This work has studied existing research proposals in this area and developed novel solutions to network router based scalable DoS detection and defence mechanisms.

When considering the architecture of such a DoS defence scheme, network routers and other intermediaries that forward data in the path from the source to the destination are well placed to be augmented with DoS traffic penalisation logic. The addition of lightweight, scalable, DoS defence queueing algorithms at network routers will allow the incremental deployment of such functionality while preserving the end-to-end nature of Internet protocols. These queueing algorithms can take forwarding or dropping decisions based on local information and, thus, not need any level of coordination between different network components. Additionally, focusing on defending the congestion responsive behaviour of existing protocols such as TCP allows for ease of deployment without requiring updating of end host networking stacks. Transparency to TCP flows, whilst effectively penalising DoS traffic allows for incremental deployment while maintaining the existing aspects of the current Internet architecture.

Historically, Internet network routers have used a drop-tail queueing discipline in the packet buffers they use to queue packets in outgoing links. Drop-tail

queueing maintains a first-in first-out (FIFO) queue using the buffer space, accepting all packets while there is available space and dropping any packet that arrives when the buffer is full. This simple queueing mechanism, which is still the most widely deployed Internet router queueing algorithm, provides poor performance for TCP flows, causing global synchronisation of congestion responsive flows and thus poor utilisation and high loss rates.

Active Queue Management (AQM) proposals, starting with the seminal RED scheme [54], provide an alternative to drop-tail queueing. RED seeks to improve upon drop-tail by implementing early notification of congestion using randomised dropping based on averaged queue buffer occupancy. Since the initial RED proposal, many other queueing mechanisms have been proposed in the literature that follow similar principles to achieve better network utilisation with lower loss rates.

One such class of AQM schemes can be considered to have the objective of providing an approximation of flow-fairness [93, 108, 115]. These queueing algorithms intend to approximate the fair allocation of bandwidth at a link across the different flows transmitting at a given time. The fairness-oriented AQM schemes use a variety of techniques to maintain the simplicity and scalability associated with the lack of per-flow state of traditional AQM designs, while minimising the differences between the throughput obtained by each individual flow. The main goal of fairness-oriented AQM is to protect congestion responsive TCP traffic from unresponsive UDP flows such as multimedia streaming traffic. Most of these proposals do not consider malicious traffic generation sources, focusing instead on improving the interaction between congestion responsive and non-congestion responsive traffic at the router queue.

Some recent AQM proposals do target the defence of congestion responsive TCP traffic from potentially malicious bandwidth flooding DoS attacks. Aggregate Congestion Control (ACC) [92] is an AQM based DoS defence scheme that uses both local RED dropping information as well as inter-router communication to detect and rate-limit DoS attack traffic. The double-horizon queueing design in [144] uses a short and long timescale based hybrid approach, together with a simple infinite TCP characterisation, thus being vulnerable to packet header spoofing.

While there are existing DoS defence AQM proposals, these schemes are deficient when considering their effectiveness at protecting modern TCP traffic from current and future bandwidth flooding DoS attacks. A core component of any AQM based DoS defence scheme is the characterisation method used to define what is considered well-behaved congestion responsive traffic and what is considered malicious DoS traffic. The existing schemes base their TCP characterisation on relatively simple steady-state models that do not consider the finite nature of TCP flows as encountered on the Internet.

Existing schemes such as ACC also suffer from the limited view of the traffic provided by their measurement metrics implementation. ACC bases its decisions on a set of high-bandwidth IP addresses obtained from the RED drop history, which does not allow for the implementation of more capable algorithms due to the limited available metrics.

A common issue with most DoS defence schemes based on AQM is the lack of support for defence at arbitrary time scales. Attacks such as pulsing DoS use extremely short duration burstiness to impact TCP traffic. All existing schemes, with the exception of the double-horizon approach in [144], do not provide defence for such attack traffic, focusing instead on long-term averaged behaviour for their analysis.

Finally, a major shortcoming of existing DoS defence mechanisms is the widespread reliance on packet header information for their operations. Bandwidth flooding DoS attacks can easily implement packet header spoofing. This allows the attacker to present the generated traffic as being composed of an arbitrary number of flows, easily circumventing any defence scheme that does not take such packet header spoofing into account.

This dissertation has addressed the challenge of providing novel solutions to the problems associated with the defence of congestion responsive traffic from bandwidth flooding DoS attacks. The remainder of this chapter will summarise the main contributions of this dissertation, discuss future directions this work can be extended into and, finally, conclude with a discussion of the impact of the work described herein on the wider Internet.

6.1 Summary of contributions

The principal focus of this work is that of providing improved bandwidth flooding Denial of Service defence for congestion responsive TCP traffic at the router queue. As part of the effort to solve this problem the following core contributions have been made in this dissertation:

- A comprehensive, state of the art review of the areas of Denial of Service (DoS) attack detection and defence, Active Queue Management (AQM), and the existing AQM based proposals for improved defence of congestion responsive TCP traffic in the face of aggressive traffic, such as bandwidth flooding DoS attacks. This contribution also surveys the state of the art on TCP modelling, a critical component of any networking element that must be able to compare traffic to a reference TCP behaviour. This analysis concludes with a discussion of the shortcomings of state of the art AQM based DoS defence schemes and identifies the requirements for next generation designs.

- The design and efficient implementation of an accurate method for the characterisation of TCP flow aggregate performance at the router queue. This contribution can be seen as being composed of two parts: an efficient measurement metrics collection design and implementation, and the derivation and evaluation of a TCP performance characterisation method based on state of the art finite TCP flow modelling.

  The design of the efficient and flexible subsystem for traffic aggregate level measurement metrics collection at network routers allows for bounded memory usage, high adaptability to available resources and graceful degradation in the presence of malicious traffic. This contribution achieves a better performance in terms of metrics accuracy obtained vs byte state used. No per-flow state is maintained and thus memory requirements can be kept relatively low, an important consideration in the face of potentially malicious traffic. A probabilistic data structure is used to provide for constant memory usage. Metrics are maintained at the level of coarse TCP flow aggregates, allowing for high scalability and the ability to analyse flow creation without detailed per-flow data.

The method for predicting TCP aggregate behaviour is based on state of the art TCP modelling approaches and provides accurate characterisation of finite length TCP flows, while being easy to implement in a network router environment with limited information about the traffic being evaluated. It provides the ability to accurately define expected behaviour from finite TCP flows and aggregates of flows, while allowing for simple implementation in a network router environment. Existing router queueing schemes do not take advantage of recent improvements in finite length TCP flow modelling techniques due to the complexity of adapting these models to be implemented at the router queue. The method contributed here provides for such an implementation of detailed TCP modelling in network routers.

- Design, implementation and evaluation of Responsive Aggregate Defence (RAD), a novel bandwidth flooding DoS defence router queueing scheme based on Active Queue Management (AQM) principles and sophisticated TCP characterisation.

  RAD is a hybrid queueing scheme that builds on Active Queue Management (AQM) design principles to provide a scalable DoS defence mechanism to protect congestion responsive TCP traffic. It is implemented using aggregate-level measurement metrics and accurate TCP characterisation. It protects TCP traffic by detecting and penalising aggressive malicious DoS traffic. The aggressiveness of DoS traffic is determined using both packet generation as well as session or flow creation behaviour. RAD also operates at both instantaneous as well as averaged time scales to be able to detect attack traffic independently of the time scale used.

  The above characteristics make RAD an effective and novel DoS defence proposal. Compared to existing fairness-oriented AQM schemes RAD takes advantage of accurate, but coarse granularity, aggregate-level metrics. RAD uses a definition of aggressiveness for DoS detection that is robust against packet header spoofing by considering flow creation as well as packet generation. Unlike most other DoS and fairness oriented AQM proposals, RAD is also effective against short-timescale attacks as well as attacks that extend over long periods.

The simulation based evaluation of the RAD queueing scheme rigorously explores the performance of RAD under a variety of scenarios. The ability of RAD to provide improved throughput performance for TCP traffic under DoS attack conditions is studied and compared to a RED queueing implementation under the same scenario.

The network traffic mixes used provide a realistic realisation of common TCP background traffic mixes and explore the boundaries of possible bandwidth flooding DoS attack spaces. The TCP traffic mix used implements a closed-loop model in which congestion responsive feedback is included both in the packet generation as well as connection or session creation processes. This allows for more realistic modelling of human created TCP traffic in which session creation is correlated which experienced performance. Pulsing and persistent DoS traffic is evaluated to study RAD performance in a wide variety of current and potential future attack traffic mixes.

- A discussion of the future directions for the extension of the work in this dissertation, and the potential impact of the proposed designs on the robustness of the Internet to bandwidth flooding DoS attacks is provided.

6.2 Future work

In this section potential future work related to the contributions in the dissertation will be assessed. These recommendations arise from the work described herein and are interesting directions in which the core contributions could be further extended.

The work in this dissertation has focused on the detection and penalisation of bandwidth flooding Dos attacks. These attacks form one of the two different categories in which DoS attacks can be divided, the other category being software exploit DoS attacks. It would be an interesting extension of this work to include detection and defence from software exploit DoS attacks. The nature of these attacks is very different to bandwidth flooding DoS in their core approach to victim exploitation. Such a system would require a largely separate approach for software exploit DoS defence and detection. Such a design should not need

any information other than that available at the network router. One possible approach towards solving this problem could monitor the traffic to and from a given destination to detect patterns in traffic generation or header structures that cause receiving hosts to become unavailable. On the other hand, such a scheme would not be effective in asymmetric routing configurations. Thus, this is a highly relevant but complex problem that is out of the scope for this thesis.

Another direction this work could follow is that of embedding the DoS defence logic proposed in RAD on the actual end nodes. Such a scheme would prevent the computers themselves from generating traffic that might be considered part of a DoS attack, or DoS-like, in nature. This idea would need to be implemented at quite a low level in the system to prevent applications with access to privileged roles in the host OS from disabling it. One possible approach would be to embed DoS detection logic in the network card firmware itself. This logic would effectively limit the traffic generation patterns from the host to conform with minimal congestion responsive behaviour.

The RAD proposal implements a design that is composed of independent queueing logic for each one of the links on a network router. This design has the advantage of being highly parallelisable and not requiring any global synchronisation or data sharing at the router level. However, the RAD design could also be extended to have awareness of the state of all links on the system. This would allow RAD to improve both its DoS detection and penalisation logic to take into account information beyond that which is available in a single queue. This could allow for the more effective use of the router's memory and CPU resources to improve accuracy or granularity for specific links that are experiencing attacks.

6.3 Conclusion

Given the current end-to-end architecture of the TCP protocol there is a need for Internet router queueing algorithms that will provide adequate defence against malicious bandwidth flooding DoS traffic, whilst being compatible with existing deployed network hardware and software components. This dissertation has addressed the design of network router based solutions to the problem of detecting

and defending against bandwidth flooding Denial of Service (DoS) attacks that render victims inaccessible through exploitation of the congestion responsive behaviour of TCP traffic.

Responsive Aggregate Defence (RAD) utilises state of the art finite TCP flow aggregate characterisation together with scalable measurement metric implementations to identify suspicious traffic. It is able to detect and penalise a wide range of DoS attacks that exploit TCP congestion responsiveness to achieve an unfair share of the available bandwidth.

RAD is a queueing scheme that does not require any changes to existing protocols and architectures and can easily be deployed incrementally as it operates only on local information at the router queue and does not require any distributed collaboration. RAD uses a hybrid architecture, operating at both instantaneous as well as averaged long-term time scales, to achieve good performance against both low-rate pulsing DoS attacks as well as providing protection from more traditional persistent DoS traffic. It is also effective against the use of packet header spoofing from DoS traffic sources due to its consideration of both packet generation and session creation behaviour.

The contributions made by the RAD DoS defence router queueing design can be used to improve the resilience of the Internet to bandwidth flooding DoS attacks. The existing protocols and architectural principles of the Internet are unable to defend users and services from malicious traffic generation processes. RAD satisfies the need for improved but lightweight and scalable network router based queueing logic that minimises the impact of DoS traffic on TCP achieved performance. RAD allows for its immediate and incremental deployment on Internet routers, making it an effective DoS defence scheme for the current Internet.

Definitions

AIMD Additive Increase Multiplicative Decrease

API Application Programming Interface

AQM Active Queue Management

CBR Constant Bit-rate

CERT The Computer Emergency Response Team

DDoS Distributed Denial of Service

DoS Denial of Service

DRDoS Distributed Reflector Denial of Service

ECN Explicit Congestion Notification

FIFO First-in First-out

FTP File Transfer Protocol

HMAC Hash-based Message Authentication Code

HTTP Hypertext Transfer Protocol

ICMP Internet Control Message Protocol

IETF Internet Engineering Task Force

IP Internet Protocol

ISP Internet Service Provider

LRU Least Recently Used

MSS Maximum Segment Size

PPM Probabilistic Packet Marking

RAD Responsive Aggregate Defence

RED Random Early Detection

RoQ Reduction of Quality

RTO Retransmission Time-out

RTT Round-trip Time

SHA-1 Secure Hash Algorithm 1

TCP Transmission Control Protocol

TTL Time To Live

UDP User Datagram Protocol

Bibliography

[1] VeriSign: DoS Attack Could Shut Down Internet. http://www.zdnetasia.com/news/security/0,39044215,62032653,00.htm .

[2] Micah Adler. Tradeoffs in Probabilistic Packet Marking for IP Traceback. *Journal of the ACM*, 52(2):217–244, March 2005.

[3] Mark Allman and Vern Paxson. On Estimating End-to-End Network Path Properties. In *Proceedings of the Workshop on data communication in Latin America and the Caribbean*, volume 31, pages 124 – 151, 2001.

[4] Mark Allman and W. Richard Stevens. TCP Congestion Control. IETF RFC 2581, April 1999.

[5] Eitan Altman, Konstantin Avrachenkov, and Chadi Barakat. A stochastic model of TCP/IP with stationary random losses. *IEEE/ACM Transactions on Networking (TON)*, 13(2), April 2005.

[6] Martin Arlitt, Balachander Krishnamurthy, and Jeffrey C. Mogul. Predicting Short-transfer Latency from TCP Arcana: A Trace-based Validation. In *IMC '05: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 19–19, Berkeley, CA, USA, 2005. USENIX Association.

[7] Sanjeewa Athuraliya, David E. Lapsley, and Steven H. Low. An Enhanced Random Early Marking Algorithm for Internet Flow Control. In *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000)*, volume 3, pages 1425–1434, Tel Aviv, Israel, March 2000. IEEE.

[8] James Aweya, Michel Oulette, and Delfin Y. Montuno. A Control Theoretic Approach to Active Queue Management. *Computer Networks*, 36:203–235, August 2001.

[9] Deepak Bansal, Hari Balakrishnan, Sally Floyd, and Scott Shenker. Dynamic Behavior of Slowly-Responsive Congestion Control Algorithms. In *Proceedings of the 2001 SIGCOMM conference*, volume 31, pages 263–274, 2001.

[10] Chadi Barakat. TCP/IP Modeling and Validation. *IEEE Network*, 15(3):38–47, May 2001.

[11] Paul Barford, Jeffery Kline, David Plonka, and Amos Ron. A Signal Analysis of Network Traffic Anomalies. In ACM, editor, *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement*, volume 3, pages 71–82, Marseille, France, November 2002. ACM.

[12] D. Basheer, A. Chakrabarti, and G. Manimaran. An Efficient Probabilistic Packet Marking Scheme for IP Traceback. In *Proceedings of the 3rd IFIP-TC6 Networking conference*, Athens, Greece, May 2004.

[13] D. Basheer and G. Manimaran. A Novel Packet Marking Scheme for IP Traceback. In *Proceedings of the 10th IEEE Internation Conference on Parallel and Distributed Systems. ICPADS 2004.*, pages 195–202, Newport Beach (CA), USA, July 2004. IEEE.

[14] Steve Bellovin, Marcus Leech, and Tom Taylor. ICMP Traceback Messages (work in progress). http://tools.ietf.org/html/draft-ietf-itrace-01.txt.

[15] A. Bitorika, M. Robin, and M. Huggard. An Evaluation Framework For Active Queue Management Schemes. In *Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'03*, pages 200–206, Orlando, FL, USA, October 2003. IEEE.

[16] Arkaitz Bitorika, Mathieu Robin, and Meriel Huggard. A Framework For Evaluating Active Queue Management Schemes. Technical Report TCD-

CS-2003-34, Trinity College Dublin, Department of Computer Science, July 2003.

[17] Burton H. Bloom. Space/time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 13(7):422–426, July 1970.

[18] Marjory S. Blumenthal and David D. Clark. Rethinking the Design of the Internet: the End-to-End Arguments vs. the Brave New World. *ACM Transactions on Internet Technology (TOIT)*, 1(1):70–109, August 2001.

[19] B. Braden et al. Recommendations on Queue Management and Congestion Avoidance in the Internet. IETF RFC 2309, April 1998.

[20] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu. Advances in Network Simulation. *IEEE Computer*, 33(5):59–67, May 2000.

[21] Andrei Broder and Michael Mitzenmacher. Network Applications of Bloom Filters: A Survey. *Internet Mathematics*, 1(4):485–509, 2005.

[22] Hal Burch and Bill Cheswick. Tracing Anonymous Packets to Their Approximate Source. In *Proceedings of the 2000 Usenix LISA (New Orleans) conference*, pages 319–327, December 2000.

[23] Mathworld by Wolfram Research Inc. Geometric Series. http://mathworld.wolfram.com/geometricseries.html.

[24] Mathworld by Wolfram Research Inc. Lambert W Function. http://mathworld.wolfram.com/lambertw-function.html.

[25] Neal Cardwell, Stefan Savage, and Thomas Anderson. Modeling TCP Latency. In *Proceedings of INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 3, pages 1742–1751. IEEE, 2000.

[26] CERT Coordination Center (CERT/CC). CERT® Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks.

[27] CERT Coordination Center (CERT/CC). CERT® Advisory CA-1997-28 IP Denial-of-Service Attacks.

[28] CERT Coordination Center (CERT/CC). CERT® Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks.

[29] David Champagne and Ruby B. Lee. Scope of DDoS Countermeasures: Taxonomy of Proposed Solutions and Design Goals for Real-World Deployment. In *Proceedings of the 8th International Symposium on Systems and Information Security (SSI'2006)*, Sao Paulo, Brazil, November 2006.

[30] Ming-Kit Chan and M. Hamdi. An Active Queue Management Scheme Based on a Capture-recapture Model. *IEEE Journal on Selected Areas in Communications*, 21(4):572–583, May 2003.

[31] Wu chang Feng, Dilip D. Kandlur, Debanjan Saha, and Kang G. Shin. BLUE: A New Class of Active Queue Management Algorithms. Technical Report CSE-TR-387-99, University of Michigan, April 1999.

[32] Li-Chiou Chen, Thomas A. Longstaff, and Kathleen M. Carley. Characterization of Defense Mechanisms against Distributed Denial of Service Attacks. *Computers and Security*, 23(8):665–678, 2004.

[33] Yu Chen and Kai Hwang. Collaborative Detection and Filtering of Shrew DDoS Attacks Using Spectral Analysis. *Journal of Parallel and Distributed Computing*, 66(9):1137–1151, 2006.

[34] Chen-Mou Cheng, H.T. Kung, and Koan-Sin Tan. Use of Spectral Analysis in Defense Against DoS Attacks. In *Proceedings of the IEEE Global Telecommunications Conference, 2002. GLOBECOM '02*, volume 3, pages 2143–2148. IEEE, 2002.

[35] Roman Chertov, Sonia Fahmy, and Ness B. Shroff. Emulation versus Simulation: A Case Study of TCP-Targeted Denial of Service Attacks. In *Proceedings of the 2nd International IEEE/Create-Net Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom)*. IEEE, 2006.

[36] Cisco Systems. Weighted Random Early Detection on the Cisco 12000 Series Router, March 2002.

[37] Alberto Dainotti, Antonio Pescape, and Giorgio Ventre. Wavelet-based Detection of DoS Attacks. In *Proceedings of the IEEE Global Telecommunications Conference, 2006. GLOBECOM '06.* IEEE, 2006.

[38] Drew Dean, Matt Franklin, and Adam Stubblefield. An algebraic approach to IP traceback. *ACM Transactions on Information and System Security*, 5(2):119–137, 2002.

[39] X. Deng, S. Yi, G. Kesidis, and C.R. Das. Stabilized Virtual Buffer (SVB) - an Active Queue Management Scheme for Internet Quality-of-Service. In *Proceedings of GLOBECOM'02*, volume 2, pages 1628–1632. IEEE, December 2002.

[40] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. *IETF RFC 4346*, 2006.

[41] Eric Douzet. Internet Protocol. IETF RFC 791, September 1981.

[42] Allen B. Downey. An Empirical Model of TCP Performance. In *Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems. MASCOTS 05.*, pages 45–54. IEEE, September 2005.

[43] Allen B. Downey. Lognormal and Pareto Distributions in the Internet. *Computer Communications*, 28(7):790–801, May 2005.

[44] D. Eastlake and P. Jones. US Secure Hash Algorithm 1 (SHA1). IETF RFC 3174, September 2001.

[45] Shirin Ebrahimi-Taghizadeh, Ahmed Helmy, and Sandeep Gupta. TCP vs. TCP: a systematic study of adverse impact of short-lived TCP flows on long-lived TCP flows. In IEEE, editor, *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM 2005.*, volume 2, pages 926–937, March 2005.

[46] Navid Ehsan and Mingyan Liu. Analysis of TCP Transient Behavior and Its Effect on File Transfer Latency. In *Proceedings of the IEEE International Conference on Communications, 2003. ICC '03.*, volume 3, pages 1806–1811, Anchorage, Alaska, USA, May 2003.

[47] W. Feng, Dilip D. Kandlur, Debanjan Saha, and Kang G. Shin. Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness. In *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM 2001*, volume 3, pages 1520–1529. IEEE, April 2001.

[48] Wu-chang Feng, Dilip D. Kandlur, Debanjan Saha, and Kang G. Shin. A Self-Configuring RED Gateway. In *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM 1999.*, volume 3, pages 1320–1328. IEEE, March 1999.

[49] Roy Fielding, Jim Gettys, j. Mogul, H. Frystyk, L. Masinter, P. Leach, and Tim Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. IETF RFC 2616, June 1999.

[50] Sally Floyd. TCP and Explicit Congestion Notification. *ACM Computer Communication Review*, 24(5):10–23, 1994.

[51] Sally Floyd and Kevin Fall. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, August 1999.

[52] Sally Floyd, Ramakrishna Gummadi, and Scott Shenker. Adaptive RED: An Algorithm for Increasing the Robustness of RED. under submission, August 2001.

[53] Sally Floyd, Mark Handley, Jitendra Padhye, and Joerg Widmer. Equation-Based Congestion Control for Unicast Applications. *ACM SIGCOMM Computer Communication Review*, 30(4):43–56, October 2000.

[54] Sally Floyd and Van Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking (TON)*, 1(4):397–413, August 1993.

[55] Sally Floyd and Eddie Kohler. Internet Research Needs Better Models. *ACM Computer Communication Review*, 33(1):29–34, January 2003.

[56] Chuck Fraleigh, Sue Moon, Bryan Lyles, Chase Cotton, Mujahid Khan, Deb Moll, Rob Rockell, Ted Seely, and Christophe Diot. Packet-level Traffic Measurements from the Sprint IP Backbone. *IEEE Network*, 17(6):6–16, December 2003.

[57] Aman Garg and A.L.Narasimha Reddy. Mitigation of DoS attacks through QoS regulation. *Microprocessors and Microsystems*, 28(10):521–530, September 2004.

[58] Steve Gibson. The Strange Tale of the Denial of Service Attacks Against grc.com. http://www.grc.com, March 2002.

[59] Thomer M. Gil and Massimiliano Poletto. MULTOPS: A Data-structure for Bandwidth Attack Detection. In *Proceedings of 10th Usenix Security Symposium*, pages 23–38, Washington, DC, 2001. Usenix.

[60] Michael T. Goodrich. Efficient Packet Marking for Large-scale IP Traceback. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 117–126. ACM Press, 2002.

[61] Mina Guirguis, Azer Bestavros, and Ibrahim Matta. Exploiting the Transients of Adaptation for RoQ Attacks on Internet Resources. In *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP'04)*, pages 184–195, October 2004.

[62] Mina Guirguis, Azer Bestavros, and Ibrahim Matta. On the Impact of Low-Rate Attacks. In IEEE, editor, *Proceedings of theIEEE International Conference on Communications, 2006. ICC '06.*, volume 5, pages 2316–2321, June 2006.

[63] Mina Guirguis, Azer Bestavros, Ibrahim Matta, and Yuting Zhang. Reduction of Quality (RoQ) Attacks on Internet End-Systems. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM 2005.*, volume 2, pages 1362–1372, March 2005.

[64] Qi He, Constantine Dovrolis, and Mostafa Ammar. On the Predictability of Large Transfer TCP Throughput. In *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications. SIGCOMM 05.*, pages 145–156, Philadelphia, Pennsylvania, USA, August 2005. ACM.

[65] C.V. Hollot, V. Misra, D. Towsley, and Weibo Gong. Analysis and Design of Controllers for AQM Routers Supporting TCP Flows. *IEEE Transactions on Automatic Control*, 47(6):945–959, June 2002.

[66] Kevin J. Houle and George M. Weaver. Trends in Denial of Service Attack Technology. CERT Coordination Center, October 2001.

[67] Alefiya Hussain, John Heidemann, and Christos Papadopoulos. A Framework for Classifying Denial of Service Attacks. Technical Report ISI-TR-2003-569, Information Sciences Institute, 2003.

[68] Van Jacobson. Congestion Avoidance and Control. *ACM SIGCOMM Computer Communication Review*, 25(1):157–187, 1995.

[69] Abhishek Jain, A. Karandikar, and R. Verma. An Adaptive Prediction Based Approach for Congestion Estimation in Active Queue Management (APACE). In *Proceedings of the IEEE Global Telecommunications Conference, 2003. GLOBECOM '03.*, volume 7, pages 4153–4157. IEEE, December 2003.

[70] Y. Jiang, M. Hamdi, and J. Liu. Self Adjustable CHOKe: An Active Queue Management Algorithm for Congestion Control and Fair Bandwidth Allocation. In *Proceedings of the 8th International Symposium on Computers and Communications, ISCC 2003.*, pages 1018–1025. IEEE, July 2003.

[71] Cheng Jin, Haining Wang, and Kang G. Shin. Hop-Count Filtering: An Effective Defense Against Spoofed Traffic. In ACM, editor, *Proceedings of the 10th ACM conference on Computer and communications security*, pages 30–41, 2003.

[72] C. Joo, S. Bahk, and S.S. Lumetta. Hybrid Active Queue Management. In *Proceedings of the Eighth IEEE International Symposium on Computers and Communication, 2003. (ISCC 2003).*, volume 2, pages 999–1004. IEEE, July 2003.

[73] Srikanth Kandula, Dina Katabi, Matthias Jacob, and Arthur Berger. Botz-4-Sale: Surviving Organized DDoS Attacks That Mimic Flash Crowds. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI)*, pages 287–300, Boston, MA, May 2005.

[74] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. IETF RFC 2401, 1998.

[75] Inas Khalifa and Ljiljana Trajkovic. An Overview and Comparison of Analytical TCP Models. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, volume 5, pages 469–472, Vancouver, British Columbia, May 2004.

[76] Seong Soo Kim, A.L. Narasimha Reddy, and M. Vanucci. Detecting Traffic Anomalies at the Source Through Aggregate Analysis of Packet Header Data. *Networking*, 3042, 2002.

[77] Eddie Kohler, Mark Handley, and Sally Floyd. Designing DCCP: Congestion Control Without Reliability. In ACM, editor, *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–38, Pisa, Italy, 2006.

[78] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. IETF RFC 2104, February 1997.

[79] S. Kunniyur and R. Srikant. Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management. *IEEE/ACM Transactions on Networking (TON)*, 12(2), 2004.

[80] Aleksandar Kuzmanovic and Edward W. Knightly. Low-Rate TCP-Targeted Denial of Service Attacks. In *SIGCOMM '03: Proceedings of*

the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pages 75–86, Karlsruhe, Germany, 2003. ACM.

[81] Aleksandar Kuzmanovic and Edward W. Knightly. Low-Rate TCP-Targeted Denial of Service Attacks and Counter Strategies. *IEEE/ACM Transactions on Networking (TON)*, 14(4):683–696, 2006.

[82] Minseok Kwon and Sonia Fahmy. A comparison of load-based and queue-based active queue management algorithms. In *Proceedings of SPIE IT-Com*, volume 4866, pages 35–46, August 2002.

[83] Karthik Lakshminarayanan, Daniel Adkins, Adrian Perrig, and Ion Stoica. Taming IP Packet Flooding Attacks. *ACM SIGCOMM Computer Communication Review*, 34(1):45 – 50, January 2004.

[84] Heejo Lee and Kihong Park. On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial Of Service Attack. In *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE INFOCOM 2001.*, volume 1, pages 338–347, April 2001.

[85] Lan Li and Gyungho Lee. DDoS Attack Detection and Wavelets. *Telecommunication Systems*, 28(3-4):435–451, 2005.

[86] Yujian Li and Carey Williamson. A Hysteresis Model for Web/TCP Transfer Latency. In *Proceedings of the IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004. (MASCOTS 2004)*, pages 167–174, 2004.

[87] Kejie Lu, Jieyan Fan, Dapeng Wu, Sinisa Todorovic, and Antonio Nucci. Robust and Efficient Detection of DDoS Attacks for Large-Scale Internet. *Computer Networks*, 51(18):5036–5056, December 2007.

[88] Xiapu Luo and Rocky K. C. Chang. On a New Class of Pulsing Denial-of-Service Attacks and the Defense. In *Proceedings of the Network and Distributed System Security Symposium*, 2005.

[89] Xiapu Luo, Rocky K. C. Chang, and Edmond W. W. Chan. Performance Analysis of TCP/AQM Under Denial-of-Service Attacks. In *Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '05)*, pages 97–104, 2005.

[90] Gordon "Fyodor" Lyon. Ping of Death. http://insecure.org/sploits/ping-o-death.html, October 1996.

[91] m3lt. The LAND attack. http://insecure.org/sploits/land.ip.DOS.html.

[92] Ratul Mahajan, Steven M. Bellovin, and Sally Floyd. Controlling High Bandwidth Aggregates in the Network. *ACM SIGCOMM Computer Communication Review*, 32(3), July 2002.

[93] Ratul Mahajan and Sally Floyd. Controlling High Bandwidth Flows at the Congested Router. Technical Report TR-01-001, AT&T Center for Internet Research at ICSI (ACIRI), April 2001.

[94] Allison Mankin, Dan Massey, Chien-Lung Wu, S. Felix Wu, and Lixia Zhang. On Design and Evaluation of "Intention-Driven" ICMP Traceback. In *Proceedings of the 10th IEEE International Conference on Computer Communication and Networks (ICCCN'01)*, pages 159–165. IEEE, October 2001.

[95] M. Mathis et al. TCP Selective Acknowledgment Options. IETF RFC 2018, 1996.

[96] M. Mathis, J. Semke, Jamshid Mahdavi, and T. Ott. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *ACM Computer Communication Review*, 27(3), 1997.

[97] M. May, J. Bolot, C. Diot, and B. Lyles. Reasons Not to Deploy RED. In *Proceedings of the Seventh International Workshop on Quality of Service, 1999. IWQoS '99*, pages 260–262, June 1999.

[98] Marco Mellia, Ion Stoica, and Hui Zhang. TCP Model for Short Lived Flows. *IEEE Communications Letters*, 6(2):85–87, February 2002.

[99] Jelena Mirkovic, Gregory Prier, and Peter Reiher. Attacking DDoS at the Source. In *Proceedings of the 10th IEEE International Conference on Network Protocols*, pages 312–321. IEEE, November 2002.

[100] Jelena Mirkovic, Gregory Prier, and Peter Reiher. Source-End DDoS Defense. In *Proceedings of the Second IEEE International Symposium on Network Computing and Applications. NCA 2003*, pages 171–178, April 2003.

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

[102] Mariyam Mirza, Joel Sommers, Paul Barford, and Xiaojin Zhu. A Machine Learning approach to TCP Throughput Prediction. In *Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 97–108, San Diego, California, USA, 2007. ACM.

[103] Vishal Misra, Wei-Bo Gong, and Don Towsley. Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED. *ACM SIGCOMM Computer Communication Review*, 30(4):151–160, 2000.

[104] Michael Mitzenmacher and Rajmohan Rajaraman. Towards More Complete Models of TCP Latency and Throughput. *The Journal of Supercomputing,*, (20):137–160, 2001.

[105] David Moore, Colleen Shannon, Douglas J. Brown, Geoffrey M. Voelker, and Stefan Savage. Inferring Internet Denial-of-Service Activity. *ACM Transactions on Computer Systems (TOCS)*, 24(2):115–139, May 2006.

[106] Jitendra Padhye, Victor Firoiu, and Donald F. Towsley. Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation. *IEEE/ACM Transactions On Networking*, 8(2):133–145, April 200.

[107] Rong Pan, Lee Breslau, Balaji Prabhakar, and Scott Shenker. Approximate Fairness through Differential Dropping. *ACM SIGCOMM Computer Communication Review*, 33(2):23–39, April 2003.

[108] Rong Pan, Balaji Prabhakar, and Konstantinos Psounis. CHOKe — A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation. In *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE INFOCOM 2000*, volume 2, pages 942–951, March 2000.

[109] K. Pawlikowski. Steady-state simulation of queueing processes: A survey of problems and solutions. *ACM Computing Surveys*, 22(2):123–169, June 1990.

[110] Vern Paxson. An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks. *ACM Computer Communication Review*, 31(3), July 2001.

[111] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Detecting Distributed Denial of Service Attacks Using Source IP Address Monitoring. In *Proceedings of the International IFIT-TC6 networking conference*, Athens, Greece, May 2004.

[112] Ravi Prasad and Constantinos Dovrolis. Congestion Responsiveness of Internet Traffic (A fresh look at an old problem). In *Proceedings of the joint international conference on Measurement and modeling of computer systems*, pages 385–386, Saint Malo, France, 2006.

[113] M. V. Ramakrishna. Practical Performance of Bloom Filters and Parallel Free-Text Searching. *Communications of the ACM*, 32(10):1237–1239, 1989.

[114] A. Ramanathan, A.L. Narasimha Reddy, and Marina Vanucci. WADeS: A Tool for Distributed Denial of Service Attack Detection. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications. SIGCOMM 02.* ACM, November 2002.

[115] Smitha A. L. Narasimha Reddy. LRU-RED: an Active Queue Management Scheme to Contain High Bandwidth Flows at Congested Routers. In *Proceedings of the IEEE Global Telecommunications Conference, 2001.*

GLOBECOM '01, volume 4, pages 2311–2315, San Antonio, TX, USA, November 2001. IEEE.

[116] Ronald L. Rivest. The MD5 Message-Digest Algorithm. IETF RFC 1321, 1992.

[117] Max Robinson, Jelena Mirkovic, Scott Michel, Matthew Schnaider, and Peter Reiher. DefCOM: Defensive Cooperative Overlay Mesh. In *Proceedings of the 2003 DARPA Information Survivability Conference and Exposition (DISCEX'03)*, volume 2, pages 101–102, April 2003.

[118] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems (TOCS)*, 2(4):277–288, November 1984.

[119] Sandeep Sarat and Andreas Terzis. On the effect of router buffer sizes on low-rate denial of service attacks. In *Proceedings of the 14th International Conference on Computer Communications and Networks, 2005. ICCCN 2005.*, pages 281–286, October 2005.

[120] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Network Support for IP Traceback. *IEEE/ACM Transactions on Networking*, 9(3), June 2001.

[121] H. Schulzrinne, S. Casner, R. Frederick, and Van Jacobson. RTP: A Transport Protocol for Real-Time Applications. IETF RFC 3550, July 2003.

[122] Amey Shevtekar, Karunakar Anantharam, and Nirwan Ansari. Low Rate TCP Denial-of-Service Attack Detection at Edge Routers. *IEEE Communications Letters*, 9(4), 2005.

[123] B. Sikdar, S. Kalyanaraman, and K. S. Vastola. Analytic Models and Comparative Study of the Latency and Steady-State Throughput of TCP Tahoe, Reno and SACK. In *EEE/ACM Transactions on Networking (TON)*, 2003.

[124] Alex C. Snoeren. Hash-based IP traceback. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 3–14. ACM Press, 2001.

[125] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Beverly Schwartz, Stephen T. Kent, and W. Timothy Strayer. Single-Packet IP Traceback. *IEEE/ACM Transactions on Networking*, 10(6), December 2002.

[126] Dawn Xiaodong Song and Adrian Perrig. Advanced and Authenticated Marking Schemes for IP Traceback. In *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE INFOCOM 2001*, volume 2, pages 878–886, Anchorage, AK, USA, April 2001.

[127] Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to 0wn the Internet in Your Spare Time. In *Proceedings of the 11th USENIX Security Symposium*, pages 149–167, August 2002.

[128] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. IETF RFC 2100, January 1997.

[129] Ion Stoica, Scott Shenker, and Hui Zhang. Core-Stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks. *Computer Communication Review*, 28(4), October 1998.

[130] Robert Stone. CenterTrack: An IP Overlay Network for Tracking DoS Floods. In *Proceedings of the 2000 Usenix Security Symposium*, pages 15–15, Denver, Colorado, USA, July 2000.

[131] Jinsheng Sun, Guanrong Chen, King-Tim Ko, S. Chan, and M. Zukerman. PD-controller: a new active queue management scheme. In *Proceedings of GLOBECOM'03*, volume 6, pages 3103–3107. IEEE, December 2003.

[132] Minho Sung and Jun Xu. IP Traceback-Based Intelligent Packet Filtering: A Novel Technique for Defending against Internet DDoS Attacks. *IEEE Transactions on Parallel and Distributed Systems*, 14(9), September 2003.

[133] Rohit Tripathi, Yu-Kwong Kwok, and Kai Hwang. HAWK: Halting Anomaly with Weighted ChoKing to Rescue Well-Behaved TCP Sessions from Shrew DoS Attacks. In *Proceedings of the International Conference on Computer Networks and Mobile Computing (ICCNMC 2005)*, 2005.

[134] Guido van Rossum. Python Reference Manual. Technical Report CS-R9525, CWI, Netherlands, 1995.

[135] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using Hard AI Problems for Security. In *Proceedings of Eurocrypt 2003*, pages 294–311, 2003.

[136] Marcel Waldvogel. GOSSIB vs. IP Traceback Rumors. In *18th Annual Computer Security Applications Conference (ACSAC 2002)*, pages 5–13, December 2002.

[137] Bao-Tung Wang and Henning Schulzrinne. A Denial-of-Service-Resistant IP Traceback Approach. In *Proceedings of the Ninth International Symposium on Computers and Communications, 2004. ISCC 2004*, volume 1, pages 351–356, 2004.

[138] Bao-Tung Wang and Henning Schulzrinne. An IP Traceback Mechanism for Reflective DoS Attacks. In *Proceedings of the 2004 Canadian Conference on Electrical and Computer Engineering*, May 2004.

[139] Haining Wang and Kang G. Shin. Transport-Aware IP Routers: A Built-In Protection Mechanism to Counter DDoS Attacks. *IEEE Transactions on Parallel and Distributed Systems*, 14(9), September 2003.

[140] Haining Wang, Danlu Zhang, and Kang G. Shin. Detecting SYN Flooding Attacks. In *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM 2002*, volume 3, pages 1530–1539, June 2002.

[141] Bartek Wydrowski and Moshe Zukerman. GREEN: An Active Queue Management Algorithm for a Self Managed Internet. In *Proceedings of the IEEE International Conference on Communications, 2002. ICC 2002*, volume 4, pages 2368–2372, May 2002.

[142] Yang Xiang and Wanlei Zhou Zhongwen Li. An Analytical Model for DDoS Attacks and Defense. In *Proceedings of the International Multi-Conference on Computing in the Global Information Technology (ICCGI'06)*, pages 66–66, August 2006.

[143] Ying Xu and Roch Guerin. On the Robustness of Router-based Denial-of-Service (DoS) Defense Systems. *ACM Computer Communication Review*, 35(3), July 2005.

[144] Ying Xu and Roch Guerin. A Double Horizon Defense Design for Robust Regulation of Malicious Traffic. In *Proceedings of SECURECOMM*, pages 1–11, September 2006.

[145] Avi Yaar, Adrian Perrig, and Dawn Song. An Endhost Capability Mechanism to Mitigate DDoS Flooding Attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2004.

[146] Guang Yang, Mario Gerla, and M. Y. Sanadidi. Defense Against Low-rate TCP-targeted Denial-of-Service Attacks. In *Proceedings of the Ninth International Symposium on Computers and Communications, ISCC 2004.*, volume 2, pages 345–350, 2004.

[147] Y. Richard Yang, Min Sik Kim, and Simon S. Lam. Transient behaviors of TCP-friendly congestion control protocols. In *Proceedings of INFOCOM 2001*, volume 3, pages 1716–1725. IEEE, April 2001.

[148] Yin Zhang, Lee Breslau, Vern Paxson, and Scott Shenker. On the Characteristics and Origins of Internet Flow Rates. In ACM, editor, *Proceedings of the 2002 SIGCOMM conference*, pages 309 – 322, 2002.

[149] Ying Zhang, Z. Morley Mao, and Jia Wang. Low-Rate TCP-Targeted DoS Attack Disrupts Internet Routing. In *Proceedings of 14th Annual Network and Distributed System Security Symposium (NDSS)*, 2007.