2011

A composable approach to design of newer techniques for large-scale denial-of-service attack attribution

Muthuprasanna Muthusrinivasan
Iowa State University

A composable approach to design of newer techniques for large-scale denial-of-service attack attribution

by

Muthuprasanna Muthusrinivasan

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Program of Study Committee:
Manimaran Govindarasu, Major Professor
Arun Somani
Suraj Kothari
Yong Guan
Zhengdao Wang

Iowa State University

Ames, Iowa

2011

In loving memory of my maternal grand-mother, ***Late Mrs. Navaneetham Rajaram****, whose love and support was a constant motivation through the long arduous journey this has been.*

TABLE OF CONTENTS

LIST OF FIGURES vi

ACKNOWLEDGEMENTS viii

ABSTRACT x

CHAPTER 1. DENIAL OF SERVICE ATTACKS 1

1.1 DDoS Attack Characterization 2

1.1.1 Attack Instances 2

1.1.2 Attack Tools 4

1.1.3 Attack Mechanism 6

1.1.4 Attack Enablers 9

1.2 DDoS Attack Classification 11

1.2.1 Target - Who do you attack? 12

1.2.2 Motive - Why do you attack? 14

1.2.3 Mechanism - How do you attack? 16

CHAPTER 2. DENIAL OF SERVICE DEFENSES 22

2.1 DDoS Defense Characterization 22

2.1.1 Simple Approaches 22

2.1.2 Basic Design Challenges 24

2.1.3 Practical Design Considerations 29

2.2 DDoS Defense Classification 32

2.2.1 Attack Prevention 32

2.2.2 Attack Mitigation 35

2.2.3 Attack Traceback 38

CHAPTER 3. COMPOSABLE MODEL 44

3.1 Motivation 45

3.1.1 Traceback-enabled Mitigation 46

3.1.2 Subscription-based Services 47

3.2 Data Cube Model 48

3.2.1 Multi-Axis Data Representation 49

3.2.2 Data Transmission Schedule 53

3.3 Theoretical Analysis 58

3.3.1 Data Utility 58

3.3.2 Packet Overhead 62

3.4 Experimental Evaluation 63

CHAPTER 4. ROUTER LABELING: A Graph Coloring Approach 66

4.1 Proposed Concepts 66

4.1.1 Localized Labels 67

4.1.2 Logical Routes 73

4.2 Practical Considerations 76

4.2.1 Scalability 76

4.2.2 Incremental Deployment 79

4.3 Experimental Evaluation 81

4.3.1 Fingerprint Size 81

4.3.2 Coloring Efficiency 83

4.3.3 Convergence Speed 84

CHAPTER 5. PATH LABELING: A Divide-and-Conquer Approach 95

5.1 Proposed Concepts 95

5.1.1 Attack Tree Construction 98

5.1.2 Attack Path Frequency Detection 102

5.1.3 Packet to Path Association 106

5.2 Experimental Evaluation 108
5.2.1 Out-of-band Traffic 109
5.2.2 In-band Traffic 110
5.2.3 Router Storage 111
CHAPTER 6. SOCIAL ACCEPTANCE OF ATTACK ATTRIBUTION . . 118
6.1 Anonymity 119
6.2 Anonymity in the Traceback Context 121
6.3 Theoretical Analysis 125
CHAPTER 7. CONCLUSIONS 127
BIBLIOGRAPHY 131

LIST OF FIGURES

Figure 1.1 DDoS Attack Classification 21

Figure 2.1 DDoS Defense Classification 43

Figure 3.1 Composable Data Cube Model 50
Figure 3.2 Composable Model: Router-Axis Optimization 51
Figure 3.3 Composable Model: Fragment-Axis Optimization 52
Figure 3.4 Traffic-weighted Attack Tree 54
Figure 3.5 Composable Model: Data Transmission Schedule A 55
Figure 3.6 Composable Model: Data Transmission Schedule B 56
Figure 3.7 Composable Model: Data Transmission Schedule C 57
Figure 3.8 Theoretical Bounds for Utility Rate (#packets vs #utility) 61
Figure 3.9 Utility Rate: Top-k Percentile Filtering (#packets vs #utility) 64
Figure 3.10 Utility Rate: Number of Packets 64
Figure 3.11 Thesis Contributions 65

Figure 4.1 Localized Labels 69
Figure 4.2 Ambiguous Traceback Decoding - Need for Star Coloring 70
Figure 4.3 Star Coloring - $\Theta(\Delta^2)$ Colors 71
Figure 4.4 Logical Routes 74
Figure 4.5 Graph Resizing 75
Figure 4.6 Scalable Router Labeling 77
Figure 4.7 Incrementally Deployable Router Labeling 80
Figure 4.8 Degree Distribution (#neighbors vs #nodes) 86

Figure 4.9 Star Degree Distribution (#star-neighbors vs #nodes) 87
Figure 4.10 Path Length Distribution (#hops vs #nodes) 88
Figure 4.11 Coloring Efficiency (Centralized) (#colors vs #nodes) 89
Figure 4.12 Coloring Efficiency (One Palette) (#colors vs #nodes) 90
Figure 4.13 Coloring Efficiency (Four Palettes) (#colors vs #nodes) 91
Figure 4.14 Convergence Speed (One Palette) (#iterations vs #nodes) 92
Figure 4.15 Convergence Speed (Two Palettes) (#iterations vs #nodes) 93
Figure 4.16 Convergence Speed (Four Palettes) (#iterations vs #nodes) 94

Figure 5.1 Modular Path Tree 99
Figure 5.2 Modular Path Frequency Tree 103
Figure 5.3 Illustration: Path Tree & Path Frequency Tree 107
Figure 5.4 Traffic-Annotated Attack Tree 108
Figure 5.5 Illustration: In-Band Path Identifiers 109
Figure 5.6 Degree Distribution (#hops vs #nodes) 110
Figure 5.7 (Out-of-band) Path Tree Size (#hops vs #KBs) 113
Figure 5.8 (Out-of-band) Path Frequency Tree Size (#hops vs #KBs) 114
Figure 5.9 (In-band) Packet Marking Size (#hops vs #bits) 115
Figure 5.10 (Router Storage) Lookup Table Size (#hops vs #bytes) 116
Figure 5.11 Comparison: Network Traffic and Storage Overhead 117

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere thanks to all those who helped me during the various stages of my life - to realize the true potential of scientific thinking and critical analysis, and thereby embark on this sacred journey of conducting advanced research.

First and foremost, my deepest gratitude to my advisor, Manimaran Govindarasu, for his guidance, patience and support throughout this research. Whether I was exploring nascent research ideas, or learning the art of authoring scientific articles, or navigating through tricky situations on the academic (or even personal) front, he was always around to lend a helping hand. His ability to motivate, challenge, and bring out the best in a person is simply unparalleled. Finally, a big salute to him for the enormous trust he placed in me and my abilities.

I also extend my sincere thanks to the faculty at Iowa State University - Arun Somani, Suraj Kothari, Zhengdao Wang, Srinivas Aluru, Srikanta Tirthapura, Yong Guan - whose words of encouragement and key insights into various aspects of my work, not only helped me gain a good breadth of knowledge in other allied areas of research in computer engineering, but also achieve a good measure of finesse and perfection in my scientific pursuits.

I would also like to thank my mentors at various scientific organizations, for providing me opportunities to experience first-hand the miracles of modern-day research and development, and inspiring me to aim for the skies. My special thanks to Vijay Kumar, Vishy Poosala, Mansoor Alicherry, Anurag Srivastava, Pankaj Risbood at Alcatel-Lucent Bell Laboratories for giving me a chance to walk the hallowed hallways of scientific achievements. My sincere

thanks to Paul Osterhus, Panos Koutsoyannis, Matt Cutts at Google Inc for letting me take time off as needed to work on my research, and also for continued financial support for my work.

I would also like to acknowledge the contributions of various other people during the early days of my education, whose passion, dedication and vision, had been great contributing factors in my development as a young researcher then. My sincere thanks to Thambipillai Srikantan, Lam Siew Kei at Nanyang Technological University, Singapore for providing me the first opportunity to conduct any meaningful research. My special thanks to Devendra Jalihal, K. Sridharan, Hari Ramachandran, Anil Prabhakar, C. S. Ramalingam, John Bosco Lourdusamy at Indian Institute of Technology Madras, India for making me believe that I have the ability to pursue a career in advanced research and development, for the very first time.

Finally, I would also like to thank the many people who provided me a conducive environment to pursue my research goals in great earnest. My heart-felt thanks to the administrative staff at Iowa State University - Pam Myers, Vicky Thorland Oster, Virginia McCallum - who helped me navigate the myriad immigration issues involved with out-of-state residency and part-time academic research while employed in the industry. A big salute to all my lab-mates - Sudha Anil Kumar, Kavitha Balasubramanian, Basheer Al-Duwairi, Mohammed Fraiwan, Durga Kocherlakota, and also Aaron Striegel, Anirban Chakrabarti - for their lively discussions and constructive criticism of my research on a regular basis. A big salute to my friends - Samarth Shetty, Girish Lingappa, Srivatsan Balasubramanian, Srinivas Neginhal, Ashutosh Tiwari, Navneet Malani, Ashwin Natarajan, Mahadevan Gomatishankar, Ganesh Subramanian, Satya Lakmi Kanth, Souvik Ray, Ravi Cherkuri, Veerendra Allada, and others - without whom life in Ames, Iowa might not have been such a pleasant experience.

Last but not the least, I reserve my deepest gratitude for my parents, my sister, and more importantly my maternal grand-parents, who always believed in me and stood by me through all times, good and bad. But for them, I might not be half the person I am today!

ABSTRACT

Since its early days, the Internet has witnessed not only a phenomenal growth, but also a large number of security attacks, and in recent years, denial-of-service (DoS) attacks have emerged as one of the top threats. The stateless and destination-oriented Internet routing combined with the ability to harness a large number of compromised machines and the relative ease and low costs of launching such attacks has made this a hard problem to address. Additionally, the myriad requirements of scalability, incremental deployment, adequate user privacy protections, and appropriate economic incentives has further complicated the design of DDoS defense mechanisms. While the many research proposals to date have focussed differently on prevention, mitigation, or traceback of DDoS attacks, the lack of a comprehensive approach satisfying the different design criteria for successful attack attribution is indeed disturbing.

Our first contribution here has been the design of a *composable data model* that has helped us represent the various dimensions of the attack attribution problem, particularly the performance attributes of accuracy, effectiveness, speed and overhead, as orthogonal and mutually independent design considerations. We have then designed custom optimizations along each of these dimensions, and have further integrated them into a single composite model, to provide strong performance guarantees. Thus, the proposed model has given us a single framework that can not only address the individual shortcomings of the various known attack attribution techniques, but also provide a more wholesome counter-measure against DDoS attacks.

Our second contribution here has been a concrete implementation based on the proposed composable data model, having adopted a *graph-theoretic* approach to identify and subse-

quently stitch together individual *edge fragments* in the Internet graph to reveal the true routing path of any network data packet. The proposed approach has been analyzed through theoretical and experimental evaluation across multiple metrics, including scalability, incremental deployment, speed and efficiency of the distributed algorithm, and finally the total overhead associated with its deployment. We have thereby shown that it is realistically feasible to provide strong performance and scalability guarantees for Internet-wide attack attribution.

Our third contribution here has further advanced the state of the art by directly identifying individual *path fragments* in the Internet graph, having adopted a *distributed divide-and-conquer* approach employing simple *recurrence relations* as individual building blocks. A detailed analysis of the proposed approach on real-life Internet topologies with respect to network storage and traffic overhead, has provided a more realistic characterization. Thus, not only does the proposed approach lend well for simplified operations at scale but can also provide robust network-wide performance and security guarantees for Internet-wide attack attribution.

Our final contribution here has introduced the notion of *anonymity* in the overall attack attribution process to significantly broaden its scope. The highly invasive nature of widespread data gathering for network traceback continues to violate one of the key principles of Internet use today - the ability to stay anonymous and operate freely without retribution. In this regard, we have successfully reconciled these mutually divergent requirements to make it not only economically feasible and politically viable but also socially acceptable.

This work opens up several directions for future research - analysis of existing attack attribution techniques to identify further scope for improvements, incorporation of newer attributes into the design framework of the composable data model abstraction, and finally design of newer attack attribution techniques that comprehensively integrate the various attack prevention, mitigation and traceback techniques in an efficient manner.

CHAPTER 1. DENIAL OF SERVICE ATTACKS

The Internet has witnessed a phenomenal growth since its early days at ARPA [1] - what began as an experimental research project in the early 1970's has today grown exponentially both in size and complexity to become an essential underpinning of our global society. While the Internet in the early days was restricted to a few universities and used primarily for research, the advent of the World Wide Web [2] in the early 1990's gave it a public face - it is widely used today not just for communications and entertainment, but also for electronic commerce and managing critical infrastructure. Thus the environment in which the Internet operates is no more benign, and has turned increasingly hostile having great implications not only for individual users but also for corporations and governments alike.

In recent years, we have witnessed a wide range of attacks exploiting the many vulnerabilities in the various Internet protocols and their software implementations at each layer of the OSI stack [3]. These attacks can be broadly classified as *infrastructure attacks* and *application attacks*. While the infrastructure attacks range from ARP cache poisoning to DNS hijacking to IP spoofing to large-scale distributed denial-of-service attacks, the application attacks arise due to many security holes at the operating system layer or the installed software packages including mail clients, web browsers, etc. The wide variety of attacks and the need for successful defenses against each of them has led to the emergence of multiple areas of research, some distinct while others inter-dependent, broadly recognized as *network security*, *software security*, *web security*, etc. Given the wide spectrum of attacks, we hereby restrict the scope of this dissertation to addressing the issue of denial-of-service attacks on the Internet alone.

The denial-of-service (DoS) attacks consist of an overwhelming quantity of packets being sent to a victim - these packets arrive in such high quantities that some key resource at the victim (network bandwidth, memory or I/O buffers, CPU time to compute responses) is quickly exhausted. The victim subsequently either crashes or spends so much time handling the attack traffic that it cannot attend to its real work, thereby depriving legitimate clients access to the victim. The distributed denial-of-service (DDoS) attacks are more sophisticated attacks where multiple sources of traffic try to overwhelm the victim, often leading to a catastrophic failure of the service provided by the victim. These DDoS attacks pose a significant threat to the Internet today as they can not only incapacitate individual businesses at will, but also potentially cost millions of dollars in lost revenue and productivity. The increasing sophistication of these attacks makes them one of the top threats to the Internet community today [4].

1.1 DDoS Attack Characterization

The problem of denial-of-service first became evident in 1986 when the Internet suffered a series of congestion collapses [5]. This led to a series of steps including the design and deployment of several TCP congestion control mechanisms [6] and end-to-end flow management techniques for fair allocation of resources to the different packet streams [7]. While this resource exhaustion attack was purely unintentional, it did lay the preliminary ground-work for a greater understanding of the denial-of-service problem and eventually led to its malicious use by abusive attackers on the Internet in the years to come [8].

1.1.1 Attack Instances

The goal of a DDoS attack is to simply inflict damage on the victim. Frequently, the ulterior motives are personal reasons - revenge in the case of a large number of DDoS attacks against home computers, or even to gain the respect of the hacker community by conducting successful attacks on popular web servers. However, it is not unlikely that some DDoS attacks

are performed for material gain or even for political or economic reasons.

The first appearance of large-scale DDoS attacks occurred in 1999 affecting major corporations [9] like Yahoo, EBay, CNN, Amazon, etc. The closely coordinated attacks sent more than 800 Mbps of malicious traffic to these high profile websites, thereby degrading their services for a considerable duration of time. The first major reflector-based DDoS attack occurred in 2001 wherein forged requests for tens of thousands of DNS records were employed for about a week [10]. In October 2002, a massive DDoS attack was launched against nine (out of thirteen) root DNS servers, significantly impacting their services for the greater part of an hour. It was the first attack of its kind against the Internet backbone, directed at disabling the Internet itself instead of a few specific sites [11].

The year 2004 marked the emergence of DDoS attacks for political and economic gains - the politically motivated attacks against SCO and RIAA [12] and subsequent attacks against UK betting sites like 2Checkout [13] driven by economic considerations established a newer trend. The use of fast spreading viruses and worms to launch truly distributed attacks was also evident in the attacks against Delta Airlines and the popular French News Agency AFP [14]. Persistent high volume attacks against the anti-spam company, Blue-Security, [15] eventually led to the company shutting shop in 2006 as it was evident that the attackers would sustain the attack as long as their services were accessible to any of their clients.

The DDoS attacks against the government sites of Estonia [16] in 2007 and Georgia [17] in 2008 marked the first instance of an attack against nation-states themselves, and led to increased chatter about cyber-warfare and national security concerns. The year 2009 saw multiple waves of attacks against various organizations including The Pirate Bay, Twitter, Facebook, LiveJournal, Google, and many popular sites in South Korea [18]. The year 2010 then saw the first instance of retaliatory DDoS attacks - the first being launched against WikiLeaks servers, and subsequent retaliation against MasterCard, PayPal, Visa, etc. as part of

the *Operation Payback* campaign by a group calling themselves *Anonymous* [19].

A security analysis report by Arbor Networks [20] for the year 2010 states that DDoS attacks now happen against some Internet entity on a hourly basis. More disturbingly, they note that DDoS attacks with traffic volumes in excess of 40Gbps are fairly common place now, and few attacks even managed to scale 100 Gbps of total attack traffic.

1.1.2 Attack Tools

During the early days, the ability to launch DDoS attacks was limited to just a few sophisticated attackers who had the technical know-how and resources to launch such attacks. However, the emergence of massive botnets (like Conficker, Stuxnet) and the easy availability of automated attack tools now allow just about anyone to launch crippling attacks with just a few key-strokes [21]. While there exist numerous scripts for scanning and infecting vulnerable machines, we discuss only a few popular user-friendly tools here. The reader should bear in mind that the features described here are those that have been observed in instances of attack code detected on some of the infected machines. Many variations may (and will) exist that have not yet been discovered and analyzed.

Trinoo: It deploys a master/slave architecture where an attacker sends commands to the master via TCP, and the master-slave communication happens via UDP. Both the master and the slaves are password protected to prevent them from being taken over by another attacker. This attack tool generates UDP packets of a given size to random ports on one or more target addresses during a specified attack interval [22].

TFN: It also deploys a master/slave architecture, and can initiate a UDP flood, TCP SYN flood, ICMP ECHO flood and Smurf attacks at specified or random victim ports. The attacker communicates with masters using any of a number of connection methods - remote shell bound

to a TCP port, UDP based client/server remote shells, ICMP-based client/server shells, SSH terminal sessions, normal telnet TCP terminal sessions, etc. All commands sent from the master to the slaves through ICMP packets are specially coded which hinders detection [23].

Stacheldraht: It combines the features of Trinoo and TFN tools and adds encrypted communication for greater stealth. Stacheldraht uses TCP for encrypted communication between the attacker and the masters, and TCP or ICMP for communication between master and slaves. Another added feature is the ability to perform automatic updates of the attack code [24].

Shaft: It is a DDoS tool similar to Trinoo, TFN and Stacheldraht. It additionally supports the ability to switch master servers and master ports on the fly, thereby hindering detection by most intrusion detection systems in use today. It also uses a *ticket* mechanism for keeping track of its individual agents, and employs a simple letter-shifting (Caesar cipher) to obscure passwords in the attack directives. Masters can also issue a special command to the slaves to obtain statistics on malicious traffic generated by each slave. It is suspected that this is used to calculate the yield of a DDoS network [25].

TFN2K: It is an improved version of the TFN attack tool. It includes several features designed specifically to make TFN2K traffic difficult to recognize and filter, to remotely execute commands, to obfuscate the true source of the traffic, to transport TFN2K traffic over multiple transport protocols including UDP, TCP, and ICMP, and to foil attempts to locate other nodes in a TFN2K network by sending *decoy* packets. It also obfuscates the true traffic source by spoofing source addresses. In addition to flooding, it can also perform some vulnerability attacks by sending malformed or invalid packets [26].

Trinity: It is the first DDoS tool that is controlled via IRC or ICQ. Upon compromise and infection by Trinity, each machine joins a specified IRC channel and waits for commands. Use of a legitimate (IRC or ICQ) service for communication between the attacker and the slaves

eliminates the need for a master machine and elevates the level of the threat significantly. Trinity is capable of launching several types of flooding attacks on a victim site, including UDP, IP fragment, TCP SYN, TCP RST, TCP ACK, and other floods.

These automated attack tools are becoming more sophisticated by the day, and not only provide a means for precise targeting of different victims, but also provide a whole range of attack options. A detailed report by Symantec recently revealed that a network of 5500 zombies could be rented for as low as $350 weekly [27], and launching DDoS attacks is thus no more constrained due to technical know-how or steep costs.

1.1.3 Attack Mechanism

A DDoS attack is usually carried out in several phases [8] - the attacker recruits a few compromised (master) machines which are in turn used to recruit multiple *agent* (slave) machines. This process is usually automated - the attacker deploys a network scanning tool which scans remote machines probing for security holes which can then be exploited. These machines are subsequently infected with the attack code, and further used for recruitment of newer agents. Note that we use the terms agent, slave, and zombie inter-changeably to essentially represent the many compromised machines under the control of the attacker.

Attackers typically attempt to cover the fact that agent machines have been compromised. They erase all system logs showing malicious activity to destroy evidence that could incriminate them. They also hide attack scripts under system directories and give them obscure, non-suspicious names so that they remain inconspicuous to the users. Sometimes they patch the vulnerability used for the exploit to prevent other attackers from taking over the machine. The current exploit/infection scripts contain many automated tools for covering tracks, so even the most inexperienced attackers do not leave much evidence of the subversion process. Once this initial setup phase is complete, the slave/zombie/agent machines are now ready to

participate in a DDoS attack, awaiting the command of the attacker.

During a DDoS attack, agent machines are engaged to send attack packets to the victim. The attacker orchestrates the onset of the attack, and customizes various details such as the desired type and duration of the attack, the rate of sending attack packets, and the target of the attack. While there are many ways to create the DoS effect, there are a handful of attacks that have been commonly observed in the majority of DDoS incidents to date.

UDP flood: During this attack, the victim is flooded by numerous UDP packets that overwhelm its network bandwidth. To fully exploit bandwidth resources, the attack packets usually have a very large size. This attack is very simple to perpetrate as the attacker need not exploit any vulnerability at the victim - by simply deploying a large number of agents, he can ensure the attack's success. On the other hand, many victim sites do not regularly receive incoming UDP traffic and can discard attack packets by dropping all such packets, paying the penalty of dropping a few legitimate UDP requests in the process.

ICMP flood: The attacker here generates a flood of ICMP ECHO packets directed at the victim. The victim replies to each ICMP request, consuming precious CPU and network resources for generation a reply. This attack is as simple to perpetrate as a UDP flooding attack. As machines usually receive a very low volume of incoming ICMP packets, they can substantially defend against ICMP flooding attacks by dropping all such packets, paying the penalty of dropping a few legitimate ICMP requests in the process.

TCP flood: An attacker takes advantage of a vulnerability in the TCP protocol design to perpetrate this attack. A TCP session starts with negotiation of session parameters between the client and the server, wherein the client sends a TCP SYN packet requesting some service. Upon receipt of the SYN packet, the server allocates a connection buffer record storing information about the client, and then replies with a SYN-ACK packet informing the client

that its service request will be granted. The client, upon receipt of the SYN-ACK packet, allocates a connection buffer record storing information about the server, and then replies with a ACK packet to the server which completes the three-way handshake for establishing TCP connections. When the server has sent a SYN-ACK packet, the connection is said to be half-open. The allocated resources at the server are now tied up until the client sends a ACK packet, closes the connection by sending a RST packet or until a timeout occurs and the server automatically closes the connection, releasing the buffer space. During a TCP SYN flooding attack, the attacker generates a large number of half-open connections that eventually time-out, thereby quickly exhausting the connection buffer space at the server, such that it can no longer accept any incoming connection requests. In some cases, the server machine crashes, exhausts its memory or is otherwise rendered inoperative. Similarly, TCP RST and TCP ACK attacks are feasible and are easy to perpetrate as a UDP flooding attack.

HTTP flood: During this attack, the attacker deploys a simple script that fetches multiple web-pages from the targeted server over and over again, and does so at each of the hundreds or even thousands of agents under his control. This attack is designed to emulate a *flash-crowd* wherein a large number of legitimate users flock to a single site due to some popular content therein. This is a particularly vicious attack as the victim cannot distinguish between legitimate and attack traffic, and is thereby forced to degrade the service for all users. This attack can also be viewed as an application-layer attack and can work equally well against any of the other application-layer protocols in the OSI stack.

Thus flooding-based denial-of-service attacks can be deployed at any layer of the OSI stack; while it has traditionally been deployed at the transport layer, newer attack vectors do seem to operate at the application layer for greater stealth and to hinder advanced traffic filtering techniques. We now provide another broad classification of DDoS attacks based on whether the victim directly receives traffic from the multiple agents or not.

Direct vs Reflector Attacks: A direct DDoS attack represents the case wherein the attacker instructs the many agents to flood the victim directly using any of the above mentioned techniques. Alternately, the attacker might choose to spoof the source address on any attack packet such that the subsequent responses from the primary victim are directed to some other secondary victims. If the attacker chooses a large number of primary victims to serve as intermediaries, and thereby indirectly attack a single secondary victim, it is called a reflector DDoS attack. Note that this is a two-legged attack wherein the attack traffic is generated by the intermediaries and not by the actual agents under the control of the attacker. Reflector attacks are known to be more potent due to greater potential in achieving attack amplification and also their ability to stay hidden (an additional hop away). Note that the flooding techniques that employ semi-open connections such as UDP flood, ICMP flood, and TCP flood can be used to propagate reflector attacks - note that HTTP floods are not feasible here.

1.1.4 Attack Enablers

Since the early days of the Internet, its design has primarily been governed by concerns of scalability, stability and performance. Security has largely been deemed optional, thereby resulting in several vulnerabilities that can easily be exploited. In the past few years, various efforts have been made to retrofit security primitives onto the current Internet design in multiple small steps. However, such efforts have proven both ineffective and insufficient, further contributing to the woeful state of the Internet today. Many security professionals/researchers have consequently blamed the fundamental nature of the Internet as a key enabler of DDoS attacks. The following list summarizes several features of Internet design that are commonly believed to have created opportunities for launching DoS attacks.

Internet security is highly interdependent. DDoS attacks are commonly launched from systems that are subverted through security-related compromises. Regardless of how well secured the victim system may be, its susceptibility to DDoS attacks depends on the state of

security in the rest of the global Internet.

Internet control is distributed. Internet management is distributed, and each network is run according to local policies defined by its owners. Hence there is no way to enforce global deployment of a particular security mechanism or security policy, and due to privacy concerns, it is often impossible to investigate cross-network traffic behavior.

Internet resources are limited. Each Internet entity (host, network, service) has limited resources that can be consumed by too many users. This means that every DDoS attempt will be successful (in absence of defenses) if it acquires a sufficiently large pool of agent machines.

The power of many is greater than the power of few. Coordinated and simultaneous malicious actions by some participants will always be detrimental to others if the resources of the attackers are greater than the resources of the victims.

Intelligence and resources are not collocated. An end-to-end communication paradigm led to storing most of the intelligence needed for service guarantees with end hosts, limiting the amount of processing in the intermediate network so that packets could be forwarded quickly and at minimal cost. At the same time, a desire for large throughput led to the design of high bandwidth pathways in the intermediate network, while the end networks invested in only as much bandwidth as they thought they might need. Thus, malicious clients can misuse the abundant resources of the unwitting intermediate network for delivery of numerous messages to a less provisioned victim.

Accountability is not enforced. The source address field in an IP packet is assumed to carry the IP address of the machine that originates the packet. This assumption is not generally validated or enforced at any point along the route from the source to the destination. This creates the opportunity for source address spoofing. Along with the *destination-oriented*

routing and *stateless nature* of the Internet, source address spoofing gives attackers a powerful mechanism to escape accountability for their actions, and sometimes even the means to perpetrate sophisticated DDoS attacks, such as reflector attacks.

While many of the issues raised above are not necessarily problematic, and some have even contributed to the very success of the Internet today, they do lend well for easily launching DDoS attacks without any immediate fear of attribution. There is thus a fine line between a good design principle from an adoption perspective, and a bad design principle from a security perspective - achieving the best of both worlds is not only desirable but also imperative.

1.2 DDoS Attack Classification

We have thus far provided only an ad-hoc description of the denial-of-service problem, the high-level workings of the attack mechanism, and the resulting impact of these attacks. Given the wide variety of DDoS attacks noticed in the past few years and the emergence of newer techniques and tools, it is incumbent upon us to provide a more robust classification to not only bring in more structure but also provide a more deeper understanding of the problem at hand. Such a classification of DDoS attacks would also help us identify the different parameters that underlie any successful attack process, thereby enabling us to design and evaluate different denial-of-service defense mechanisms.

As a detailed classification would be too elaborate and confusing for the reader, we limit our scope here to broad categories that are both immediately perceptible and required for a better understanding of our proposed work. In this regard, we propose three broad categories discussed in greater detail below - the who (target), the how (mechanism), and the why (motive). The reader should remember that it is possible for a single attack to belong to multiple different sub-categories based on its different attack characteristics (Fig. 1.1).

1.2.1 Target - Who do you attack?

In order to understand the true intentions of the attacker with respect to who they wish to attack and the level of disruption they intend to cause, we introduce the target-based classification of distributed denial-of-service attacks. In generic terms, it clusters different attack techniques based on *who* is the target of the attack.

Type of Victim: DDoS attacks need not always be perpetrated against a single host machine, and depending on the type of victim, we differentiate between application, protocol, operating system, host, network and infrastructure attacks.

Application: Application attacks exploit some feature of a specific application on the victim host, thereby disabling legitimate client use of that application and possibly tying up resources of the host machine. For example, a bogus signature attack on an authentication server can tie up resources of the signature verification application on the target machine.

Protocol: Protocol attacks misuse a vulnerability in a specific version of a protocol on the target machine to consume some critical resource. An example of this attack is a TCP SYN flooding attack. As with application attacks, other applications/protocols on the same host that do not use the overloaded resources continue to operate unhindered, thereby delaying attack detection.

Operating System: Operating system attacks misuse a vulnerability in a specific version of an operating system installed at the target machine. One example of such an attack would be a flood of incoming packets that causes fast process forking until the process table is overloaded. Such attacks can completely freeze or slow down the operation of the target machine.

Host: Host attacks disable access to the target machine completely by overloading or disabling its communication mechanism. Examples of this attack are ones that overload the network link of the target machine. Since its network resources are consumed, the host cannot defend

against these attacks alone, but can usually request help from some upstream machine.

Network: Network attacks consume the incoming bandwidth of a target network with attack packets whose destination address can be chosen from the address space of the target network. These attacks can deploy various packets, since it is volume and not content that matters, and are easily detected due to their high volume. The victim network must request help from upstream networks for defense since it cannot handle the attack volume itself.

Infrastructure: Infrastructure attacks target some distributed service that is crucial for global Internet operation or operation of a sub-network. Examples include the attacks on domain name servers, large core routers, routing protocols, certificate servers, etc. They can only be countered through the co-ordinated action of multiple Internet entities.

Impact on Victim: Depending on the impact of a DDoS attack on the victim, we can also differentiate between disruptive and degrading attacks.

Disruptive: The goal of disruptive attacks is to deny the clients access to the victim's service. Almost every known attack today belongs to this category. Depending on the possibility of dynamic recovery during or after the attack, we can further differentiate them as recoverable and non-recoverable attacks. In the case of recoverable attacks, the victim recovers as soon as the influx of attack packets is stopped, or shortly afterwards. For example, if the attack is a TCP SYN flooding attack, the connection table space is freed shortly after the attack has stopped, when the half-open connection records expire. A victim of a non-recoverable attack cannot automatically recover after the attack is stopped, but requires some human intervention. For example, an attack that causes the victim machine to crash, freeze or re-boot would be classified as a non-recoverable attack, as manual initialization might be required.

Degrading: The goal of degrading attacks is to consume some portion of a victim's resources.

Since these attacks do not lead to total service disruption, they could remain undetected for significant periods of time, leading to increasingly dis-satisfied customers due to reduced quality of service. Various targeted attacks against Internet protocols that do not exhaust the victim's resources but make them inaccessible also fall under this category. For example, low-rate TCP attacks targeting the congestion control window characteristics can ensure that all TCP flows back-off simultaneously assuming a network congestion when there is none.

1.2.2 Motive - Why do you attack?

In order to understand the true mindset of the attacker, we introduce the motive-based classification of DDoS attacks. In generic terms, it tries to answer the abstract question, *why* is the attacker behaving in such a manner. Such considerations are vital in recognizing and sometimes even predicting various high-level attack trends such as the duration of the attack, the persistence of the attacker, replay probability of the attack, etc.

Retribution: It is evident in almost all DDoS attack instances that the ulterior motive for launching such attacks is the firm belief of the attacker that they have been deprived on something that they believe is rightly theirs. We broadly differentiate the retributive desires of an individual to be determined by prideful, vengeful, economic, or political considerations.

Pride: This motive has been increasingly noticed among young hackers for whom peer recognition is a primary directive for most actions - an attack on some high-profile target or exploiting a security vulnerability previously believed to be difficult if not impossible to tackle, usually brings more credibility and recognition for the attacker. In a few cases, it has been noticed that script kiddies have indulged in such activities just because they are capable of doing so and consider it a purely playful activity. It is interesting to note that most of the attack in this category as targeted sophisticated attacks as opposed to generic brute-force attacks.

Vengeance: Almost all attacks against home computers today are driven by the human desire to punish an individual for their improper actions as perceived by the attacker. Data corruption or password stealing or webpage defacing are common place examples that belong to this category. While not strictly denial-of-service in many cases, it does cause sufficient discomfort and inaccessibility to the victim of the attack.

Economic Reasons: The recent emergence of denial-of-service attacks demanding appropriate ransom or as payback for morally despicable means of generating revenue, has established a disturbing trend. While these instances have been few and far apart, the targeting of individual users by encrypting their data for ransom, and of major betting agencies and financial institutions to disrupt their services has been well documented.

Political Reasons: Strong feelings against certain organizations for their politically incorrect or legally suspect actions as perceived by some is again a strong motivator to launching DoS attacks. Attacks against repressive policies of SCO and RIAA, and the actions of the Anonymous group in support of Wikileaks is a clear indication of the attacker's political stance.

Attribution: The ability to detect malicious actions and enforce punitive measures has long been recognized as a single primary reason for people behaving in accordance with the established social order, both in the physical world and the virtual environment of the Internet. The lack of attribution thereby provides greater incentives to behave in a morally culpable manner inflicting maximum harm on one's adversaries. In this regard, source address spoofing plays a crucial role in denial-of-service, since malicious packets cannot be traced back to the source, and responsibility for actions cannot be directly assigned. We differentiate the different attack techniques here based on the spoofing technique employed - how the attacker chooses the spoofed source address in the attack packets. Based on the spoofing technique, we divide spoofing attacks into random, subnet and en route spoofed source address attacks.

Random Spoofed Source Address: Many attacks spoof random source addresses in the attack packets, since this can simply be achieved by generating random 32-bit numbers and stamping packets with them. Recent attempts to prevent spoofing using ingress filtering and route-based filtering have forced attackers to devise more sophisticated techniques to avoid current defense mechanisms. This technique has however been widely used in reflector attacks and also in falsely implicating other network entities as true perpetrators of the ongoing attack.

Subnet Spoofed Source Address: In subnet spoofing, the attacker spoofs a random address from the address space assigned to the subnet of the agent machine. Since machines at a subnet share the same physical layer (ethernet) to reach the default gateway router, it is impossible to detect it anywhere in the network other than the gateway router itself.

En Route Spoofed Source Address: An en route spoofed source address attack would spoof the address of a machine or subnet that is en route from the agent machine to the victim. There are no known instances of attacks that use en route spoofing, but this potential spoofing technique could affect route-based filtering and is thus discussed here.

Valid Source Address: Attackers benefit from source address spoofing and are likely to deploy it whenever possible. Valid source address attacks frequently originate from agent machines running Windows, since all Windows versions prior to XP do not export user-level functions for packet header modification. Those attacks that target specific applications or protocol features must use valid source addresses if the attack strategy requires several request/reply exchanges between an agent and the victim machine.

1.2.3 Mechanism - How do you attack?

In analyzing the various DDoS attacks launched to date, we notice a significant evolutionary trend towards greater automation, increased stealth, and higher impact. In order to devise

efficient DDoS defense techniques, we need to understand the different dimensions of the attack process, and subsequently address each of them independently to mount any successful defense. We now introduce the mechanism-based classification of DDoS attack, loosely addressing the *how* to attack question of technical know-how and strategy.

Automation: As discussed previously, easily accessible automated attack tools have not only made it easy to launch DDoS attacks today, but have also provided attackers a wide variety of options to remain hidden and achieve maximal impact. While the early DDoS attacks were purely manual, the automation of the exploit/infect phases enabled attackers to quickly grow their agent network. Finally the pre-programming of the attack directives into the attack code deployed on the different agents offered minimal exposure to the attacker, hereby limited to just initiating the attack script in the first place. While sophisticated automated scripts are flexible enough to even allow periodic updates by the agents, the potential reverse-engineering of the attack code at any agent would enable the entire network to be shut-down quite easily. We now seek to differentiate between the different attacks based on the degree of automation at each stage, namely communication, scanning, and propagation techniques.

Communication: During attacks with direct communication, the agents and the master machines need to know each others identity in order to communicate. This is usually achieved by hard-coding the IP address of the master machines in the attack code that is later installed at the agent machine. Each agent then reports its readiness to the masters thereby establishing a clear channel for future communication. The obvious drawback of this approach is that discovery of one compromised machine can expose the whole DDoS network, and since they also listen for network connections they are potentially identifiable by network scanners. Attacks with indirect communication deploy an additional level of indirection to increase the survivability of a DDoS network, such as the use of an IRC channel. The use of IRC services replaces the function of a master and additionally offers sufficient anonymity to the attacker. Since DDoS agents establish outbound connections to a standard service port used by the legitimate

network service, they are indistinguishable from legitimate network traffic. While discovery of a single agent may lead to the identification of one or more IRC servers and channel names used by the DDoS network, frequent IRC channel-hopping by the attackers provides greater resilience to the entire agent network.

Scanning: The goal of the scanning strategy is to locate as many vulnerable machines as possible while creating a low traffic volume to escape detection. During random scanning, each compromised host probes random addresses in the IP address space, using a different seed. In addition to creating a high traffic volume, the probability for collision increases significantly as a larger portion of the total address space gets infected and is hence not scalable. During hit-list scanning, each machine probes all addresses from an externally supplied list - when it detects a vulnerable machine it sends a portion of the initial hit-list to the recipient and keeps the rest. While it allows for great propagation speed and eliminates address space collisions, computing a large enough hit-list in advance is not always feasible. Topological scanning uses information from the compromised host to select newer targets, say from address books or even from a web server to its clients, and in many cases shows distinct characteristics of a social graph as it grows in size. Various other scanning techniques have also been explored by virus/worm designers, such as localized scanning to achieve a higher node degree in the agent network, or even remote scanning to achieve a larger diameter in the agent network.

Propagation: The yield of a DDoS attack is primarily governed by the size of the agent network under the command of an attacker, and the propagation technique used for distributing the attack code plays a vital role here. During central-source propagation, the attack code resides on a central server or a set of servers, and is downloaded through a simple file transfer mechanism. This design is greatly sub-optimal as it not only provides a single point of failure, but also prone to discovery due to high traffic volumes at the central servers. During back-chaining propagation, the attack code is downloaded from the machine that was used to exploit the system, the infected machine then becomes the source for the next propaga-

tion step. Back-chaining propagation is more resilient than central-source propagation since it avoids a single point of failure, but does not easily allow for updates to the attack code. Finally, autonomous propagation avoids the file retrieval step by injecting attack instructions directly into the target host during the exploit phase. It reduces the frequency of network traffic needed for agent mobilization, thereby making them virtually impossible to detect.

Persistence: Attacks have been known to vary different features, so as not to stand out and reveal the identity of the agent machines, including the type of traffic, attack packet headers and contents, use of interleaved decoy packets, dynamic attack rate modifications, use of independent sets of agents during a single attack, etc. We consider these variants to be extremely important since they invalidate initial assumptions underlying many defense mechanisms about always-active agents following a pre-determined characterizable behavior. While many different groupings are possible here, we restrict ourselves to differentiate attacks based on the size of the agent network and the attack rate of the individual agents, for ease of analysis.

Size of Agent Network: During attacks with a constant agent set, all agent machines act in a similar manner, taking into account resource constraints. They receive the same set of commands and are engaged simultaneously during the attack. During attacks with a variable agent set, the attacker divides all available agents into several groups and engages only one group of agents at any one time - like the army general who deploys his battalions at different times and places. A machine could belong to more than one group, and groups could be engaged again after a period of inactivity. One example would be a pulsing attack where different agent sets take turns such that their cumulative effect is a steady pulse of attack traffic to the victim.

Attack Rate Dynamics: The many agent machines participating in an attack have been known to send malicious traffic either as a steady burst or sometimes even at variable rate to delay or avoid detection and subsequent response from the victim. The majority of known attacks deploy a constant rate mechanism - the agent machines generate attack packets at a steady

rate, usually as many as their resources permit. This approach while being cost-effective in requiring a minimal number of agents, is also highly prone to discovery due to the highly anomalous nature of the attack traffic generated. On the other hand, variable rate attacks gradually increase the attack rate to achieve quality of service degradation long before they lead to a slow exhaustion of the victims resources. Some variable rate attacks also dynamically fluctuate the attack rate based on the victim's response or on the fly directives from the attacker. A pulsing attack provides a classic example of a fluctuating rate attack inducing intermittent service disruptions and continual service degradation.

In conclusion, distributed denial-of-service attacks occur in many forms and are constantly evolving by the day, to stay ahead of the defense mechanisms being deployed to counter them. Having achieved a detailed understanding of DDoS attacks in general, we now proceed to analyze the different approaches to defending against such attacks, the fundamental requirements, and their current limitations that prohibit a Internet-wide deployment.

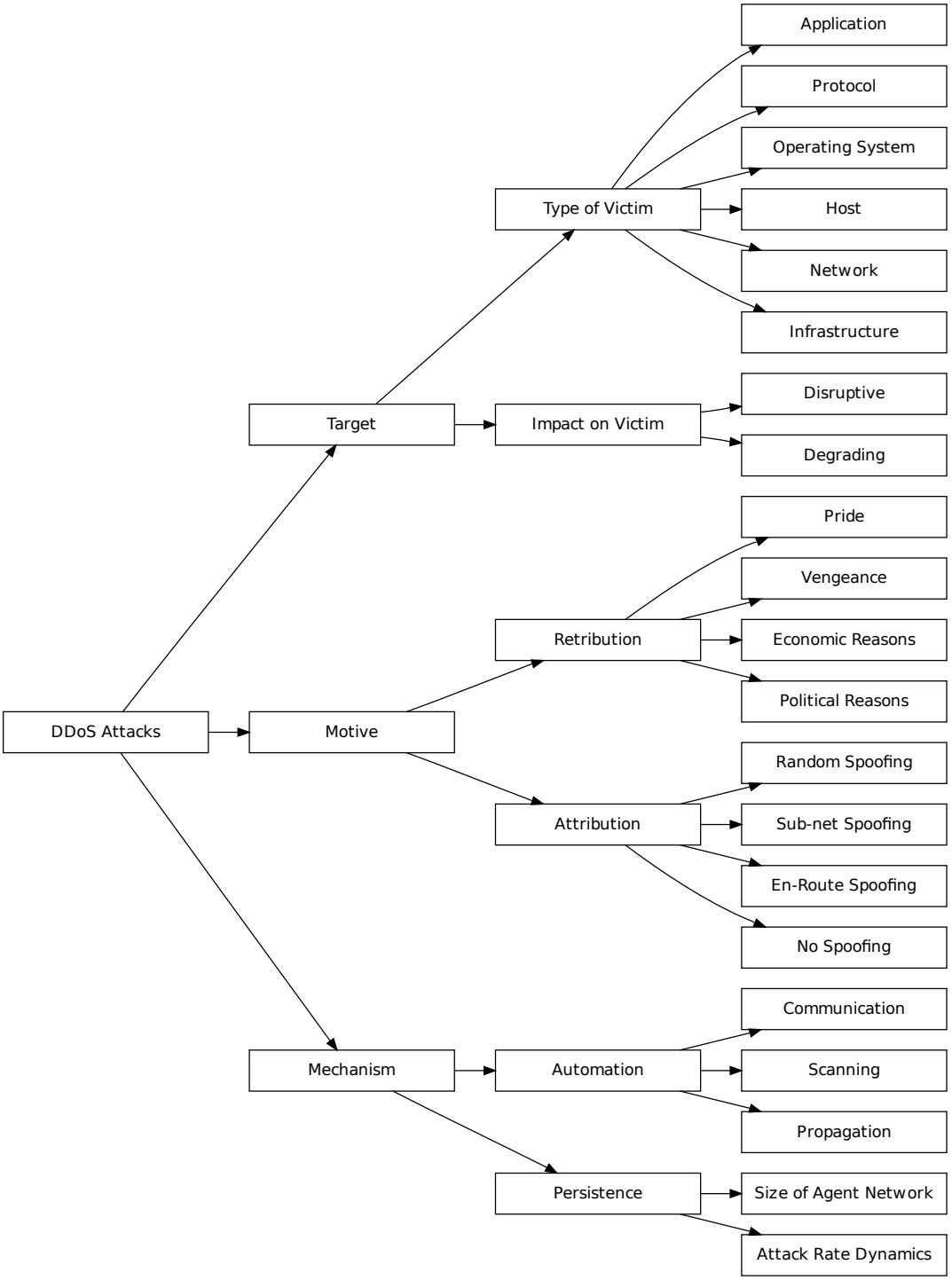


Figure 1.1 DDoS Attack Classification

CHAPTER 2. DENIAL OF SERVICE DEFENSES

The rapid evolution of DDoS attacks in recent years, combined with their ever increasing sophistication and size/volume, has made them a potent force on the Internet today. The alarming increase in the ease of launching such attacks has led to a growing fear of powerful coordinated attacks at any time against various soft targets on the Internet. Additionally, as discussed in Chapter 1, the attacker can employ a wide variety of tools and techniques at his disposal, thereby making them a formidable challenge to defend against. The state of the art in DDoS defense today is very limited to providing only partial solutions to the problem, or in some cases, solutions for specific instances of DoS attacks only, rather than providing a comprehensive solution capable of handling all variants of DDoS attacks.

2.1 DDoS Defense Characterization

We now discuss the various DDoS defense mechanisms commonly used today and their various limitations against the continuously evolving nature of the threat today. We then discuss the fundamental challenges in defending against DDoS attacks, and some broad initiatives for a more comprehensive solution and the much desired goals of any such design.

2.1.1 Simple Approaches

The commonly deployed approach to DDoS defense today is to simply secure the potential victim so as to reduce the probability of denial-of-service effects, for all but the very high-volume or stealthy DDoS attacks. As commoditized solutions are not readily available today,

most organizations turn to their internal IT personnel or external security vendors for such protections that are custom tuned to their network setup. This is usually done by following some of the basic security guidelines listed below.

- Keep all protocols and applications at the potential target up to date - apply security updates and patches regularly.
- Install up-to-date anti-malware and anti-spyware tools in the target network.
- Close all unused network I/O ports.
- Rate-limit or filter at any upstream router all incoming UDP and ICMP packets, that are not responses to requests issued from the target network.
- Implement a network firewall that can inspect, and filter if necessary, all incoming and outgoing network traffic.
- Deploy intrusion detection systems to flag any anomalous behavior in the network.
- In the case of DDoS attack, quickly devise the attack signature if possible, and then ask the administrator of the upstream network to deploy appropriate packet filters.
- If the target network provides any Internet service, create a server pool and place all servers behind a load balancer.
- Purchase sufficiently large amount of network resources - leave enough unused capacity to handle large traffic spikes.

In addition to adhering to these simple guidelines, some networks also employ commercial solutions such as Mananet [28] sold by CS3 Inc, PowerSecure [29] sold by Mazu Networks, PeakFlow [30] sold by Arbor Networks, Vantage System [31] sold by Asta Networks, etc. to protect themselves from DDoS attacks. These solutions deployed across the target network can usually identify (using proprietary techniques) the ingress points that deliver a large portion

of the attack traffic, such that filtering or rate-limiting rules can then be placed only at the identified ingress points. In this regard, various organizations like CERT, SANS, Microsoft, Arbor Networks, Symantec, etc. have also helped users to stay continually informed of the newer attack trends and adapt their security strategies accordingly [20].

A more aggressive strategy of pursuing the actual attack perpetrators to hold them accountable both legally and financially, has also led various governmental agencies to employ sophisticated measures [32] such as content filtering, wiretapping, etc. to enforce various policies and to track illegal activities online. In recent years, this issue of tracing all communication links and data packets back to their original source has come into greater focus, as was evident during the G-8 talks on traceback capabilities for the Internet [33]. Subsequent initiatives by Japan, China, USA, etc. at various forums like ITU-T Q6/17 [34] have propelled this idea further. The recent debate in Australia, EU, etc. about legislative measures for mandatory *data retention* to provide "attribution" and "deterrence" for cyber-attacks [35] [36] [37] have significantly raised the awareness of the general public in this regard.

Finally, a few bold initiatives, including GENI [38] and FIND [39], are also exploring the feasibility of fundamentally overhauling the very design of the Internet tackling security as a fundamental design parameter. Such an effort would alleviate the pain of individual networks by directly shifting the onus of security back to the network itself - while these might seem impractical in the short run, they are expected to be far more applicable in the long run.

2.1.2 Basic Design Challenges

Defending against distributed denial-of-service attacks has proven extremely hard, not for want for greater efforts on the part of research community, but for a wide variety of design issues they face in any of their attempts. The distributed nature of these attacks with a large number of agent machines, the use of legitimate traffic models to mask the attack traffic, the

use of IP spoofing, etc. greatly limit the scope of any viable defense mechanism, while also providing the attackers a plethora of options. In addition to that, the advance of DDoS defense research is hindered by the lack of attack information and the absence of standardized evaluation and testing approaches. The following list summarizes and discusses in detail some of the critical challenges we face today.

Internet Design: As discussed in Chapter 1, the very design of the Internet, with no support for basic security primitives, is the single largest challenge for the design of any DDoS defense mechanism. The Internet has been designed as an open public infrastructure to share information resources. This has two main consequences. First, the potential victims, such as web servers, must connect to the Internet and be visible to the public via a globally routable IP address in order to provide public service. Second, the Internet is based on a packet-switching paradigm, unlike its counterpart, the public telecommunication networks, which are based on the circuit-switching concept. For circuit-switched networks, each service (e.g. a phone call) will be allocated a separate channel until the end of the service, and a user's service will not be affected by other users' behavior. In contrast, for packet-switched networks, users share all the resources and one user's service can easily be disturbed by other users' behavior.

Such an open Internet design permits an attacker to exploit it in any of the following ways:

- An attacker can transmit any data in a packet.
- An attacker may generate any number of packets.
- Packets may be addressed to more than one host.
- Packets may be lost or reordered.
- Duplicate packets may exist in the network.
- The cost of transmitting a packet is negligible.

- The routing behavior of the network may be unstable.

These bandwidth exhausting DDoS attacks can now operate with certainty, that the Internet will behave in a deterministic fashion to ensure that attack packets will always be delivered to the victim irrespective of whether they are malicious or not, and by occupying most of the shared resources they can easily disrupt any service for all legitimate users.

IP Spoofing: Another fundamental consequence of the current Internet design is that it supports no authentication or identity verification whatsoever. IP spoofing refers to generating an IP packet containing a fake source address other than the one assigned by the computer system. Without an integrity check for each IP packet, attackers can spoof any field of an IP packet and inject it into the Internet. Moreover, the routers generally do not have any packet tracing functions, for example, keeping all previous connection records. In practice, this cannot be done due to the huge amount of traffic that they routinely forward. Therefore, once an IP packet is received by the victim, there is no way to authenticate whether it actually comes from where it claims, and is forced to operate on an implicit trust basis.

A significant benefit that attackers receive from IP spoofing is that it is extremely difficult to trace the agent machines. This, in turn, brings several other dire consequences. Since agent machines run a very low risk of being traced, information stored on them (i.e., access logs) cannot help to locate the attacker himself, thereby greatly encouraging further DDoS incidents. Furthermore, hiding the address of agent machines enables the attacker to reuse them for future attacks. Finally, as attack packets carry a wide variety of addresses, they appear as if they come from many disparate sources, hence defeating any fair-sharing mechanisms that might be employed to address the resource overloading problems. The other advantage that IP spoofing offers is the ability to perform reflector attacks, discussed in detail in Chapter 1.

Distributed Attacks: Almost all DDoS attacks are bandwidth exhausting brute-force at-

tacks, wherein an overwhelming number of packets overload the victim. On the other hand, DDoS defense mechanisms need to process each individual packet to validate and then possibly filter them out. Thus performing this task for a large number of packets at very high Internet speeds is a major challenge. Additionally, such large-scale packet processing requires a significant amount of resources, thereby placing an implicit limit on the attack volume that can be handled. The attacker can always circumvent this limitation by deploying more agent machines. Additionally, distributed DoS attacks can also cause significant collateral damage on other users/services by saturating network pipes at multiple different locations on the Internet.

Attack Traffic Detection: Malicious packets differ from legitimate packets in intent but not in content. This is due to the fact that attackers use the same Internet protocols used by legitimate users and generate packets that appears to be genuine. Therefore, attack traffic characterization is a very challenging problem. Often it is impossible to separate legitimate traffic from attack traffic based purely on examination of individual packets, and would require sophisticated traffic characterization techniques based on various flow metrics. While this is not only an inaccurate process thereby causing collateral damage to legitimate users, but also a prohibitively expensive proposition.

Insufficient Knowledge: It is widely believed that reporting occurrences of security attacks damages the business reputation of the victim network. Therefore, very limited information exists about various attacks, and incidents are reported only to government organizations under obligation to keep them secret. It is therefore difficult to design imaginative solutions to the problem if one is not aware of the true nature of the threat as it exists. Note that the attack information should not be confused with attack tool information, which is publicly available at many Internet sites. Attack information would include the attack type, time and duration of the attack, number of agents involved (if this information is known), attempted response and its effectiveness, the overall damages suffered, etc.

Deployability: Designing DDoS countermeasures that are transparent to existing Internet protocols represents a major challenge. Most of the known DoS countermeasures require either a network support, an end-system support, or a protocol modification. As it is very difficult to enforce deployment of DoS countermeasures in an uncontrolled domain such as the Internet, it is unclear whether user adoption of even the most advanced DDoS defense mechanisms is likely to happen. The long-pending transition from IPv4 to IPv6 and the current transition to a secure DNS model clearly indicate that Internet-wide changes may take a few years or even a few decades to happen. Thus, in addition to discouraging many researchers from even designing distributed solutions, it has also forced them to consider techniques that would allow for incremental deployment rather than assume any widespread network support.

Evaluation: Many commercial vendors make bold claims that their proprietary solutions completely handle the DDoS problem. However, there is currently no standardized approach for testing DDoS defense systems that would enable their comparison and characterization. This has two detrimental influences on DDoS research - (1) since there is no attack benchmark, defense designers are allowed to present those tests that are most advantageous to their system, and (2) researchers cannot compare actual performances of their solutions to the existing defenses; instead they can only comment on design issues.

In addition to the lack of system benchmarks for evaluation, there are no provisions to test DDoS defenses in a realistic environment. This is currently impossible due to the lack of large-scale testbeds, safe ways to perform live distributed experiments across the Internet, or detailed and realistic simulation tools that can support several thousands of nodes. Thus claims about defense system performance are made based on small-scale experiments and simulations handling very small volumes of attack traffic, and are hence not fully credible.

2.1.3 Practical Design Considerations

Having discussed the various forms of DDoS attacks and the many challenges in defending against them, we realize that any DDoS defense mechanism must support a large number of features for it to be both technically feasible and practically viable. Hence we need to explicitly lay out a set of fundamental requirements for the design of an ideal denial-of-service defense mechanism, that would not only help benchmark individual designs but also provide us a qualitative evaluation of the different approaches. The focus here is not on an exhaustive enumeration of all mandatory and optional features that need to be supported, but on identifying a few critical design considerations that can help propel any research idea into the realm of practical deployment on an Internet scale.

Effectiveness: The effectiveness of any DDoS defense can be fully evaluated by considering a whole array of performance metrics, namely the false positive rate, false negative rate, processing overhead, storage overhead, network communication overhead, etc. While an ideal solution would have no false positives or false negatives, it is likely that no sophisticated traffic characterization technique would realistically meet these standards, and hence a more pragmatic approach would be to try to minimize their impact as much as possible. The framework should additionally be secure in its own right and not be vulnerable to various exploits or denial-of-service in itself - making reliability and fault tolerance operational policy constraints.

A DDoS defense solution can be either pro-active or re-active by design. While the pro-active techniques are operational all the time by design, the re-active techniques are activated only on the onset of an attack, and are further limited to the attack duration only. The pro-active schemes can usually initiate any defense strategy during the very early stage of an attack, but it often incurs a significantly higher cost for round the clock operation even in the absence of an attack. Thus measuring the true effectiveness usually involves a trade-off between the speed of operation (reaction time) and the resulting operating costs.

Adaptability: As discussed in Chapter 1, a DDoS attack may occur in many different forms, and can also vary from time to time. Therefore, a good defense solution should not only protect all the different entities in the network against any external threat, but also dynamically detect and adapt to any change in the attack behavior. This requirement is highly desirable as the attackers today seek to analyze and exploit specific vulnerabilities in the DDoS defense tools deployed in the victim network. While implementing such generality in any single monolithic technique is difficult and also expensive, a more prudent approach would be to deploy an umbrella of solutions, mutually exclusive but also compatible for an integrated operation.

Scalability: The Internet continues to grow exponentially on an annual basis, with more users, more machines, and finally more traffic in its midst. Hence, any DDoS defense solution would need to handle more than one victim, significantly more attackers, and multiple orders of magnitude more traffic simultaneously. Any centralized approach or minor bottlenecks in the design that can be easily accommodated today may not scale in the long run. Rather than re-invent the wheel at a later date, it is more practical to design a system that can naturally scale to support the current growth of the Internet for multiple decades into the future.

Incremental Deployment: The Internet is extremely large and is managed in a distributed manner. No solution, no matter how effective, can be deployed simultaneously in hundreds of millions of disparate places. Hence, any DDoS defense mechanism that requires network-wide support for its operation would only be of academic interest and of little practical value. Any solution that operates in some limited fashion even if only a few network entities support its operation is far more likely to succeed, even if it is generally considered to be less effective. An additional desirable feature would be incremental performance, providing customers increased benefits as the degree of deployment increases. This feature can not only help with the bootstrapping of any deployment, but also in accelerating its growth.

Socio-Economic Incentives: To achieve a quick widespread deployment on the Internet, any

solution must provide appropriate incentives for the different entities involved. While security guarantees against the adverse effects of a DDoS attack probably suffices for most end-user services to want to modify their network setup, the incentives for the network providers of the Internet back-bone need to be fundamentally different. Certain assurances such as limited collateral damage to legitimate customers and reduced peering traffic due to attacks originating in other peer networks are probably sufficiently strong quality-of-service improvement guarantees that can entice network providers to deploy any massive infrastructure changes.

However, providing direct financial incentives has usually been known to work much more effectively, and hence it is preferred that any proposed solution also have a strong economic model to support the different participants. A customer must gain direct economic benefit, or at least reduce the risk of economic loss, or alternately must be able to charge others for improved services resulting from deploying any such product. For example, a subscription-based model that provides DDoS protection with a stringent service level agreement would benefit both the network provider and the service owner, and thereby ensure a faster deployment globally. Finally, any large-scale changes to the Internet architecture would require widespread support of both the Internet users and their respective governments. Hence, it would be greatly beneficial if the proposed solution did not significantly infringe upon free speech, user privacy and anonymity, network neutrality, and other fundamental rights in any way.

It is important to mention that the above criteria should be considered collectively when evaluating DDoS counter-measures. In practice, it is rather likely that most solutions meet some of these criteria rather than satisfying all of them, and hence we would need to delve deeper to measure the degree of conformance with each criterion for a more robust comparison.

2.2 DDoS Defense Classification

Having discussed the different DDoS attack mechanisms and the resulting difficulties in defending against them, we now study the various research proposals put forth to mount a meaningful DDoS defense. Broadly speaking, DDoS defenses can be classified as belonging to one of three categories: attack prevention, attack mitigation, and attack traceback (Fig. 2.1). While prevention techniques strive to avoid the occurrence of DoS attacks in the first place, mitigation techniques focus purely on surviving ongoing attacks. Finally, traceback techniques represent the first step of an online/automated forensic analysis to locate attack sources to hold them accountable, both financially and legally. We now discuss each of these different approaches in more detail, to paint a clear picture of the state of the art in DDoS defense today.

2.2.1 Attack Prevention

It is simply a mechanism which tries to stop an attack before it can actually cause any damage. On a basic level, attack prevention techniques seek to introduce changes to the Internet protocols, applications, and hosts in order to patch existing vulnerabilities and reduce the incidence of network intrusions and exploits. Their goal is to prevent vulnerability attacks, and to impede the attacker's attempts to gain a large agent army for mounting large-scale DDoS attacks. The prevention schemes are usually proactive in nature [40] [41], and generally require wide-spread deployment in order to be effective. While researchers consider them to be limited in scope given the wide range of attacks feasible today, they also represent the most preferred approach in any defense framework since they can significantly reduce the probability of an attack. We now classify DDoS prevention techniques based on the deployment patterns, namely source-based, network-based, and victim-based approaches.

Source-based: In the source-based DDoS prevention approach, the verification of the legitimacy of IP packets is performed at the periphery of the Internet, typically at the link between a customer network (the source of the packets to be validated) and an ISP network. This

approach has been adopted by various schemes such as ingress filtering [42], DWARD [43], IP EasyPass [44], etc. Ingress filtering is designed to filter IP packets with spoofed source addresses, and is done by configuring the edge router that connects the customer network to the ISP network so as to block packets that arrive with source addresses having prefixes that do not match the customer's network prefix. DWARD performs a pro-active identification and filtering of suspicious flows originating from a customer network - this is achieved by monitoring the nominal per destination traffic arrival and departure rates of TCP, UDP, and ICMP packets, as well as any abnormal asymmetrical behavior of the two-way traffic at the edge router. IP EasyPass, on the other hand, adopts a lightweight authentication scheme in order to prevent DoS attacks that target real-time applications within a DiffServ architecture [45].

Typically, preventing DoS attacks at their sources is the best solution because it protects the Internet from unwanted traffic at the earliest possible location, thereby preventing malicious flows from consuming network and end system resources. However, the major problem of this approach is the pro-active detection required and the extra overhead that it induces. This significantly discourages network administrators from adopting such approaches, especially given that the benefits of source-based prevention cannot be felt directly by the deploying network. Additionally, it has been known to not function well with some existing protocols that directly contradict ingress filtering rules, such as Mobile IP, multi-homing, etc. Finally, each of these scheme has its own limitations. For example, the scope of IP EasyPass is limited only to environments similar to the DiffServ domain, and DWARD cannot prevent colluding attacks in which attackers generate traffic in both inbound/outbound directions.

Network-based: As a proactive solution to DoS attacks, several network-based filtering techniques have been proposed to prevent spoofed IP packets from reaching their intended victims. In this approach, if the source address of a given packet does not map to an expected router's interface, then the packet is dropped. The main challenge in this approach is to establish a valid mapping between a router's interface and a source address space. This approach has been

adopted by the distributed packet filtering architecture (DPF) [46], in which it has been shown that an edge router of an AS (autonomous system) can perform pro-active packet filtering by utilizing the routing information to determine whether an incoming packet is valid with respect to the packet's inscribed source and destination IP addresses. This approach has also been adopted by the SAVE protocol [47] which provides Internet routers with information to build incoming tables that map source address spaces to the router's network interfaces. Generally, the main problem with this approach is that IP packets with en-route spoofed source address cannot be filtered [48], because an attacker can spoof the address of a node that is located along the path between the attack node and the intended victim. Additionally, this approach requires a large-scale deployment to be effective. Moreover, due to frequent route changes, legitimate traffic may sometimes get filtered even in the absence of an attack. Finally, each technique has its own limitations - for example, the major challenge in the DPF architecture is to make source-based routing information available to routers such that they can perform spoofed packet filtering. The SAVE protocol, on the other hand, is subject to router table poisoning due to the attacker's ability to inject false update messages.

Victim-based: Deploying DoS prevention schemes at potential targets provides far greater incentives, compared to their deployment at the source of an attack or inside the network, as the benefits of DDoS prevention can be experienced directly by the deploying system or network. TCP's time-out randomization [49] and spoofing prevention method (SPM) [50] have adopted this approach. TCP's time-out randomization requires a slight modification to the TCP stack such that TCP clients can avoid the effects of low rate TCP attacks [51]. Generally, TCP's time-out randomization has limited benefit because an attacker can still degrade the performance of the targeted TCP connections. SPM, on the other hand, requires installation of packet filters at the edge of a network to filter IP packets with spoofed source addresses. In order to achieve this functionality, the scheme requires that each IP packet leaving a source network towards a certain destination be tagged by a key that is associated with that (source network, destination network) pair. The dependency of SPM on authentication makes it

similar to IP EasyPass [44]. The main difference is that IP EasyPass performs light-weight authentication at a single domain scale, while SPM performs light-weight authentication at an Internet scale. Although deploying such a scheme may be preferred to ingress filtering, mutual cooperation among large number of ISP networks is still required. Moreover, SPM imposes additional overhead at the source network (due to key tagging) as well as at the destination network (due to key verification), and thereby represents a significant processing overhead.

Thus DDoS prevention schemes primarily address the IP spoofing problem, and are not designed to handle more complex attack patterns at the disposal of an attacker today. Additionally, the emergence of stepping stone botnet attacks and limited IP spoofing [52] has led many researchers to question the limited use of deploying such techniques today.

2.2.2 Attack Mitigation

It is simply a re-active technique that is usually initiated by the victim after detecting an ongoing DDoS attack. The main challenge in DDoS mitigation is the ability to accurately identify attack packets and filter them without causing collateral damage to legitimate traffic destined to the victim. There have been three major approaches to perform DoS mitigation, namely rate-based [53] [54], anomaly-based [55] [56], and path-based [57] [58] approaches. Rate-based techniques deal with DDoS attacks either as a congestion control problem or as a resource management problem. In both cases, the main idea is to characterize attack traffic and rate limit it. This approach is reactive, does not require wide scale deployment in most cases, and is generic in the sense that it applies to flooding attacks of any protocol type. However, it is not very effective since it does not avoid collateral damage. Anomaly-based techniques are based on the fact that attack traffic is more likely to hold a combination of attributes that are rarely seen by the victim. Therefore, incoming packets that hold attributes that do not match the attributes of an up-to-date traffic profile maintained at the victim are filtered. This approach is not always effective as it introduces a high false positive rate. Path-based

techniques perform attack traffic filtering based on the identification of the paths traversed by attack packets [59] [60], the length of the paths followed by attack packets [57], or even by controlling the paths followed by legitimate packets [58].

Rate-based: DDoS attacks are usually characterized by huge packet volumes that lead to a network congestion and overloading of the end-system. Therefore, it is logical to deal with this problem as a congestion control problem as in the Pushback scheme [53] [61], or as a resource management problem as in the max-min fair server-centric router throttles [62]. In both cases, the main objective is to throttle attack flows, such that legitimate flows obtain a fair share of the available resources, and can be achieved by applying rate limiting on incoming flows selectively. Pushback adds a functionality to each router to detect and preferentially drop packets that probably belong to an attack flow. In the max-min fair server-centric router throttles, the routers along the forwarding paths to a server under attack are asked to regulate the contributing packet rates to more moderate levels such that overall packet arrival rate at the server falls within an acceptable range. The deployment scenario of rate-limiting based DoS mitigation varies from one scheme to another. For example, the pushback scheme requires wide-scale deployment in order to be effective. On the other hand, the max-min fair server-centric router throttles require only a local deployment. In terms of effectiveness, rate-limiting approach affects legitimate traffic significantly due to the inability of Internet routers to characterize malicious traffic precisely before being able to throttle it. Moreover, attackers who are aware of the existing defense mechanisms can dynamically change their attack traffic patterns in such a way so as to avoid the filtering rules.

Anomaly-based: It has been observed across a wide variety of DDoS attacks that the attack packets are more likely to hold certain discernible attributes that are rarely seen by the victim, as evident from historic traffic profiling of legitimate packets. By maintaining an up-to-date traffic profile, a system under attack can easily perform per-packet filtering based on these aggregate trends. History-based filtering [55] and Packetscore [56] [63] techniques have

adopted this approach. During an attack, history-based filtering drops packets from sources that rarely communicate with the targeted system in the absence of an attack, while allowing packets with source addresses that regularly visit the network. Packetscore enables each edge router of an ISP network to compute a score for each suspicious packet and ranks the likelihood of the packet being an attacking packet, given the attribute values it carries, using a Bayesian-theoretic approach. Such anomaly or statistical separation of attack packets has several drawbacks including traffic profile poisoning at the victim or at the edge routers prior to an attack, thereby rendering them potentially ineffective. Note that various other threshold-based anomaly detection techniques have also been proposed - the use of spectral analysis [64] for attack flow detection, the Kolmogorov test [65] for detecting self-similarity in attack packets, time-series analysis [66] for traffic correlation at the source and the victim, MULTOPS [67] for measuring incoming/outgoing packet count ratios, etc.

Path-based: While attackers can easily control the attack packet format and contents, they cannot easily control the actual routing paths traversed by the attack packets due to the distributed policy-based Internet routing and the dynamic nature of its updates. Thus, it is possible to filter the attack packets either by identifying the paths followed by them [59] [60], or by inspecting certain attributes of these paths [57], or even by explicitly controlling the paths followed by the legitimate packets [58] [68]. Hop-count filtering [57] is based on the fact that the number of hops traveled by a packet that holds a spoofed source address is more likely to be different than that traveled by a packet emitted from the spoofed source itself. The SOS architecture [58] is basically an overlay network composed of nodes that communicate with one another atop the underlying network substrate. This architecture performs access verification at several nodes distributed throughout the Internet, then authorized traffic is forwarded through the overlay to a secret node. A perimeter that surrounds the protected target allows traffic coming from the secret node only, while filtering any other traffic. While path-based DoS mitigation techniques are protocol independent and can be applied to any type of attack traffic, they do require an extensive network support which is not easy to achieve. In terms of

effectiveness, although the collateral damage is usually low, it still exists, especially when the paths followed by the legitimate packets share large portions with the paths followed by attack packets themselves as both types of packets are more likely to have the same path identifier by the time they reach the victim. Other drawbacks include the potential for poisoning the path-based signatures by loose-source routing employed by the attackers, or in the case of overlay routing the significant increase in the latency along the newer routing path.

Generally speaking, mitigation techniques serve as a more practical solution to counter known/unknown DDoS attacks. However, the absence of any globally distributed infrastructure for attack traffic filtering and the lack of any coordinated efforts to achieve the same, have vastly limited the scope and effectiveness of such techniques.

2.2.3 Attack Traceback

As the Internet is stateless by design and supports a destination-oriented routing mechanism, the process of identifying the true origins of any IP packet is considered vital for security considerations, and is broadly referred to as the attack traceback (or attack attribution) problem. Over the years, it has been primarily used for detecting zombie attack machines, and also for forensic analysis by law enforcement. However, the need for widespread router-level changes to support any traceback technique and the lack of concrete economic incentives to do so, combined with the inability to alter the attack traffic pattern even after tracking a large number of agent machines in a massive botnet, has significantly reduced their scope. More recently, it has been proposed that real-time assistance in traffic filtering and attack mitigation can be provided as an added benefit, by constructing an approximate Internet router-level graph rooted at the victim to subsequently identify suitable network locations to deploy distributed packet filters in the network. While this problem has received considerable research attention since the early days of DDoS attacks, no practical solution applicable on an Internet-scale has emerged thus far. In understanding the current state of the art in traceback design, we now

discuss the three main approaches explored to date, namely, traffic-based, storage-based, and packet-based techniques.

Traffic-based: The traffic-engineering based traceback techniques are designed to change the network traffic in a controlled fashion during the attack period to study their effects on the overall traffic seen at the victim. The earliest approach to network traceback [69] sought to infer the attack paths by flooding all network links with large bursts of traffic and measuring the resulting perturbation in the attack traffic. This link-testing mechanism applied a short burst of traffic, generated using the UDP chargen service [70], to a network link under test to see whether it was along the attack path or not. The iTrace mechanism [71], proposed by Bellovin [72], employed a low volume ICMP-based out-of-band messaging channel for the victim to detect packet audit trails. These approaches however did not operate well in a multi-attacker or even a variable-rate attack scenario, and were extremely dependent on time-sensitive measurements [73]. Additionally, they induced even greater stress on the network, and also led to attack amplification [74]. The CenterTrack framework [75] then automated this traditional input debugging mechanism for route-inference by re-routing attack traffic over a specialized overlay network architecture. However, it incurred a lot of management overhead and was also limited to handling large attack flows only. Proponents of these approaches did argue that DDoS attacks are an exception rather than the norm, and hence re-active measures are better suited than other pro-active techniques proposed to date.

Storage-based: The use of explicit network support or "packet logging" was first proposed in [76], wherein the packet audit trails were stored in a distributed manner in the network itself. It was designed to store a fingerprint of the packet headers of all forwarded packets at each intermediate router, for subsequent retrieval. Once an attack was detected, the victim could query the upstream routers to check if their internal storage contained the fingerprint of the many attack packets - if it was found in a given router's memory, then that router was assumed to be part of an attack path. When done recursively across all upstream routers, it

eventually led to the identification of the true routing path of the different attack packets in the network. However, storing packet fingerprints on a pro-active basis for all packets at high Internet speeds represented a significant bottleneck with respect to storage capacity at the intermediate routers. Consequently, the use of efficient Bloom Filter data structures [77] and a hash-based approach to packet fingerprinting assisted in reducing the storage requirements, and the use of hardware-based implementations [78] made it feasible to run it at very high speeds. Additionally, the use of sampling techniques [79] further improved their performance significantly to support a large number of victims under heavy traffic loads simultaneously. However, the small window of opportunity before which the packet fingerprints were erased/over-written at the different routers, and the ever increasing processing and storage needs have greatly impacted their deployment at the Internet scale.

Packet-based: The packet logging approach described above stored the identity of a packet at the different routers that forwarded it along. Conversely, it is also possible to store the identities of the different forwarding routers as a fingerprint in the packet itself [80] [81], such that a quick inspection of the packet headers reveals its original route in the network - this is popularly known in literature as the "packet marking" technique. As the use of a low volume out-of-band messaging channel [71] led to attack amplification effects, the use of an in-band messaging channel for packet fingerprints has been preferred. However, it is critical to note that its size (data capacity) is also limited, so as not to cause unnecessary packet fragmentation in the network. The size of the in-band packet fingerprint is thus governed by two main factors: the size of each router identifier, and the number of routers along the forwarding path. While there is no consensus on where in the IP packet should the traceback information be stored, the many proposed options include the 16-bit IP fragment identification field, the IP options field, the 6-bit IP type of service field, the 13-bit fragment offset field, the 8-bit time to live field, etc. in the IP headers of any given packet.

The first use of packet marking, proposed by Belenky and Ansari [80], sequentially listed

the IP addresses of all forwarding routers in the packet headers. This deterministic packet marking (DPM) technique was not of much practical value due to the large number of bits needed per packet to represent all the different intermediate routers. Around this time, an interesting observation was made - during a DDoS attack, a large number of packets are sent from each attack source to the victim, each packet traversing along the same routing path and thereby containing the exact same fingerprint of its intermediate routers. Consequently, it was proposed that we make use of this packet fingerprint duplication by probabilistically marking each packet with the identities of some (and not all) intermediate routers, thereby giving a probabilistic guarantee that the fingerprint of all the intermediate routers would eventually reach the victim across multiple packets - also popularly known as the probabilistic packet marking (PPM) technique [81]. While it did open up the research arena to give rise to a plethora of optimizations to improve the probabilistic guarantees of the packet marking algorithm, it did however also require a packet reconstruction algorithm to assemble the different router fingerprint fragments from multiple different attack/legitimate packets, to construct the attack path in case of a single attacker or the attack tree in case of multiple attackers. A popular approach in this regard has been to use the pushback mechanism [53], where we traverse the routing path in the reverse direction from the destination to the source by incremental per-hop discovery, for accurate attack path reconstruction. The use of a distance or hop-length field [82] was proposed in this context to aid in the successful reconstruction of the original fingerprint information for a successful traceback. We now present the different optimizations to the basic PPM scheme discussed above.

Packet marking optimizations have progressed along two main directions, namely reducing the size of each router identifier (node width) and reducing the number of routers in the forwarding list (path length). The "node width" optimizations employed fingerprint fragmentation techniques [83] [84] to reduce the bit-size per packet while spreading them probabilistically across multiple packets [81]. Other variants including the use of an authentication technique for multi-packet fragment integrity [85], the use of a novel algebraic coding scheme for path

embedding [86], the use of a 1-bit distance tracking field [82], and various highly sophisticated enhancements to the base PPM scheme [87] have been proposed. Various other techniques including the use of Huffman codes [88], algebraic-geometric codes [89], the use of off-line map reconstruction techniques [82], advanced use of information-theoretic encoding [90] [91], etc. have also been explored in great detail by researchers seeking to improve the performance of the PPM-based approaches. An exhaustive theoretical evaluation of PPM-based techniques [92] [93] is also available that brings forth its various limitations for further study.

Various node/edge sampling techniques have also been proposed so as to obtain a higher-level abstraction of the network topology, thereby reducing the effective "path length" from the packet source to its destination. Various approaches including the use of hierarchical overlay routing [58] [94], hybrid packet marking and logging [95], optimizations based on local small worlds [96] [59] [97], etc. have also been explored in sufficient detail - they however do result in a lossy attack path reconstruction due to the logical abstraction of the actual routing path. For example, in the case of AS-level traceback [98], while the inter-AS path can be accurately reconstructed, the intra-AS routing path can greatly vary for large AS-networks.

The efficiency of any DoS attack defense mechanism is typically quantified through a set of, often conflicting, security and performance metrics. The security metrics include false positive rates and false negative rates, while the performance metrics include speed of attack detection and filtering, router processing and storage overheads, amongst other practical restrictions such as scalability and incremental deployment. The majority of existing prevention/mitigation/traceback schemes lack in several qualitative and quantitative metrics, and have proven inadequate in defending against the growing number and sophistication of these DDoS attacks. While there exist multiple different optimizations addressing various individual bottlenecks, there is no one single proposal that can address a multitude of these constraints simultaneously. Thus the need of the hour is an unified approach, that integrates each of these individual advancements into a single practical design.

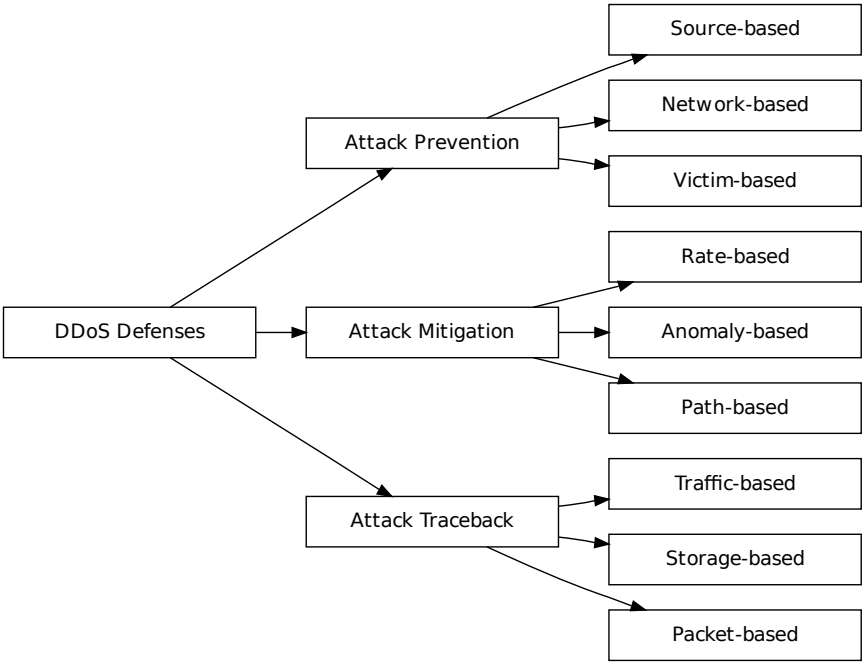


Figure 2.1 DDoS Defense Classification

CHAPTER 3. COMPOSABLE MODEL

The primary focus of Chapter 1 has been to provide the readers an in-depth view of the denial-of-service problems faced by the various Internet services today. The exhaustive enumeration of the different attack instances, the simple yet powerful tools at the attacker's disposal, and the ease of launching such attacks with very little technical or financial resources, as described therein, paint a grim picture of the state of the Internet today from a security perspective. The subsequent focus of Chapter 2 has been on the current approaches in defending against DDoS attacks, and the various challenges in doing so. The packet-switched destination-oriented stateless nature of the Internet, the widespread use of IP spoofing, the inability to deploy any reasonably good defense mechanisms due to secondary considerations like scalability, incremental deployability, socio-economic incentives, etc. have not just made DDoS defense a hard problem, but also a seemingly impossible one to achieve.

In addition to describing the fundamental nature of the DDoS problems we face today, our discussions thus far have also focussed a great deal on the wide variety of attack mechanisms possible. We have also discussed at length the various niche DDoS defense approaches, namely attack prevention, attack mitigation, and attack traceback. However, the inability of the different techniques under each of these categories to address the broad nature of the problem, and their limited scope in tackling specific instances of DDoS attacks, has rendered them both ineffective and insufficient. The focus of this chapter is thus to identify the different shortcomings of the proposed solutions, and subsequently leverage multiple independent optimizations simultaneously to address each of them individually.

3.1 Motivation

We now motivate the need for a single framework to harness the individual benefits of the different DDoS defense techniques proposed to date, thereby not only addressing their individual shortcomings but also providing a practically viable solution for Internet-wide deployment.

Why prevention fails? The different attack prevention techniques in literature can either be described as simple best practices to be followed during routine network setup/maintenance, or at best sophisticated attempts to enforce identity verification on the Internet. Thus not only are they not specific to the denial-of-service problem, but also irrelevant in the face of newer attacks not employing source address spoofing. Additionally, the critical requirement of pre-emptively addressing any and all vulnerabilities in the different protocols and their software implementations is essentially infeasible to achieve.

Why mitigation fails? While ingress filtering in every ISP network would easily solve the DDoS attack problem, the costs associated with changing practically every Internet edge router, combined with the lack of support for incremental deployment, provides no economic incentives for the ISP networks to deploy them. Additionally, the emergence of botnet-based DDoS attacks has rendered individual attack traffic volumes at each network ingress point too small to raise any flags. The current practice of egress filtering at the victim is also not very effective, due to the resulting collateral damage to other co-located services, and the reduced economic value in routing these attack packets across multiple ISP peering points to only be eventually dropped at the victim. Additionally, other network-based approaches that try to pro-actively characterize all Internet traffic as attack or legitimate, fail to do so without any false positives/negatives, and hence need additional assistance in traffic analysis before they can be practically viable for any Internet-wide deployment.

Why traceback fails? Traceback was initially envisioned as a forensic analysis tool, to track down attack agents, to cleanse them and/or to make them accountable, or even for stepping

stone detection of botnets. The many traceback techniques proposed in literature are thus designed to provide precise attack source identification, and have thus served as a critical tool for DoS attack defense during the early days when IP spoofing was predominant. However, the advent of botnet-based DDoS attacks has all but eliminated the IP spoofing problem, thereby significantly reducing their scope. Additionally, the need for network-wide changes to deploy any such mechanism, combined with their limited ability in reducing the impact of any ongoing attack, has made them less desirable economically. Finally, the use of packet logging and/or packet marking techniques entail additional complexities with respect to scalability and accuracy in attack attribution with quite a few false positives/negatives, thereby rendering them of limited use for an Internet-wide deployment today.

3.1.1 Traceback-enabled Mitigation

The classic *filter placement problem* [99] arises in this context - while placing the packet filters closer to the victim is cheaper, it considerable strains the network resources causing collateral damage to other co-located services. On the other hand, the ideal strategy of deploying ingress filters at each subnet is considered impractical given the prohibitively high costs and the limited support that current networks offer. An optimal strategy would thus place the few available filters at appropriate locations in the entire network, thereby not only minimizing the operational costs but also maximizing its impact on the attack traffic. The ideal filter deployment strategy would thus exploit the attack traffic convergence characteristics as witnessed by multiple routers in the network, and hence would require an overview of not only the different attack sources but also the many routing paths taken by their individual attack flows.

On the other hand, the concept of traceback first arose in the early 2000's as a means of fingerprinting all network data packets and persisting their state for later inspection [71]. However, recent interest in intelligent network design has sought to address its many limitations by greatly increasing its scope - to detect infected machines to potentially cleanse them, to

provide valuable insights into Internet packet routing, and to also provide real-time support for online attack mitigation. In this regard, the concept of *traceback-assisted mitigation* has been proposed - while traceback techniques can determine the sub-graph of Internet routing paths used by all attack packets, mitigation techniques can then solve the filter placement problem on this abstract attack graph. This hybrid approach would not only reduce the impact due to collateral damage, but also succeed in flagging reasonably higher traffic volume aggregates somewhere mid-stream between the attack source and the victim. Such collaborative defenses can be greatly useful in handling large-scale DDoS attacks.

We thus envision DDoS attack traceback/mitigation capabilities becoming an integral part of the Internet in the future, providing value-added services like real-time attack mitigation and forensic analysis. Our primary focus in the proposed work is to exploit this synergy, thereby achieving the best of both worlds by following a unified strategy.

3.1.2 Subscription-based Services

Deploying a service on the Internet or any future network for tomorrow would ideally require two main characteristics - simple and efficient techniques that are incrementally deployable and scalable, and those that would function seamlessly with the other elements of the Internet infrastructure. Additionally, we would require a paradigm shift in the mindset of both the consumers and the network providers to deploy such a mechanism and to provide appropriate incentives for the same. We view an intelligent network defense mechanism as a significant revenue generator for the ISP networks, providing enhanced pay per (attack) use services to the enterprise networks and also to the home-users. Providing these services in a paid subscription model with guarantees of attack detection/mitigation, possibly with multiple 9s of network availability (up-time) guarantees, would be useful to the enterprise networks not only in significantly reducing the downtime encountered during an attack, but also in avoiding hassles and prohibitive costs of maintaining redundant and custom defense strategies.

In such a scenario where network defense mechanisms have been commoditized and available as pay per use services, the onus lies on any victim under attack to achieve the maximum bang for the buck. A simple optimization problem would then try to place fewer number of expensive mitigative packet/stream filters at multiple vantage points in the Internet, so as to significantly reduce costs as well as achieve utmost traffic throttling. This is where we see the convergence of traceback and mitigation techniques happening in the future. Traceback techniques not only give a list of traffic sources, but also provide a detailed view of the attack graph, each edge possibly annotated with approximate traffic volumes. The victim now needs to solve the simpler filter placement problem to place a handful of filters at the convergence points of multiple attack flows (paths), so as to achieve the cost-benefit optimality. In short, we believe that a good DDoS defense should not only be able to build and manage a traffic weighted attack graph (attack traceback), but also dynamically reconfigure the deployment of the traffic throttling filters in this constantly changing attack graph view (attack mitigation).

While we now see the benefits of combining the various attributes of attack prevention, mitigation, and traceback, it is still not apparent how the different techniques can be unified. In this regard, we now present an elegant model to represent any DDoS defense mechanism, thereby providing a simple means of unifying multiple different techniques with relative ease, to yield far better performance and utility against DDoS attacks. The thesis contributions are described in greater detail in multiple subsequent chapters, and is shown in Fig. 3.11.

3.2 Data Cube Model

A large body of work has been devoted by researchers in the past few years to continually improve the state of the art in DDoS defense - some proposing incremental changes to already existing techniques, while others proposing a radical shift from then known defense mechanisms. Incidentally, the work on DDoS defense mechanisms has been pursued independently

and along multiple divergent lines. The focus of our proposed work here is to exploit the synergy between attack traceback and attack mitigation, to enable the design of an unified and hence more practical defense mechanism.

We first address the attack traceback problem of optimal representation and construction of the attack graph. Herein we propose a composable data model to succinctly represent the attack graph, and also provide a framework to "combine" multiple independent traceback techniques known in literature to yield higher aggregate benefits. We then focus on the attack mitigation problem of determining an optimal filter placement strategy by annotating the attack graph with edge weights based on their individual traffic volumes. Herein we propose the use of path signatures that not only provide instantaneous traffic volume estimates, but also transmit the entire attack graph to the victim when viewed in aggregate.

3.2.1 Multi-Axis Data Representation

As discussed previously, the attack graph information needed by any DDoS attack victim consists of all the attack sources and their routing paths respectively. When viewed from any individual attack source perspective, the victim needs to not only know its true identity, but also that of the intermediate routers in the network. We now propose a novel composable data model to succinctly represent any attack path. This simple yet elegant model leads us to one remarkable discovery - *every known traceback technique to date can be cast as some instance of this model*. More specifically, we can readily interpret the various theoretical bounds explored by Adler in [92]. We later conclude that it suffices to analyze the proposed model for optimality to obtain realistic bounds for any traceback technique in general, and also for easy comparisons with other popular techniques in use today.

Data Cube: The total data (measured in bits) needed by the victim to completely reconstruct a suspect routing path in the attack graph is represented using the *data cube model* shown in

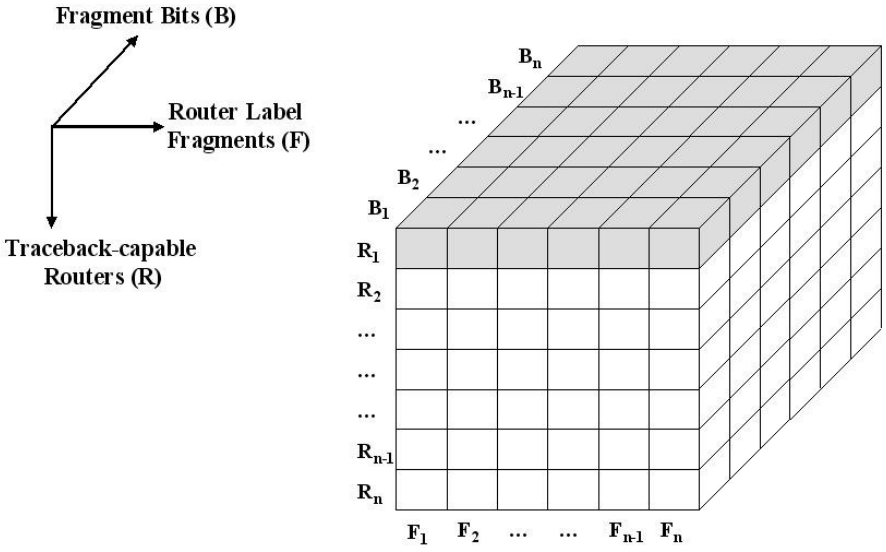


Figure 3.1 Composable Data Cube Model

Fig. 3.1. The Y-axis represents the different routers along the routing path from any attack source to the victim, while the X-axis represents the different fragments for the corresponding router identifiers (say, partial hash of a routers IP address). The Z-axis then represents the different bits in each of the router identifier fragments. Thus the data cube represents the complete information about the suspect routing path along three orthogonal dimensions. The shaded region in Fig. 3.1 shows the router identifier for a particular router, including the different bits in all its different fragments.

Based on the proposed data cube model, we can easily identify three main avenues of improvement for traceback techniques: (1) reduce the size of the data cube along the X-axis, or (2) reduce the size of the data cube along the Y-axis, or (3) reduce the size of the data cube along the Z-axis. A reduction in the size of the data cube along any single dimension leads to a reduction in its total volume, and hence directly results in faster traceback.

Router-axis optimization: The reduction in the number of intermediate routers is usually achieved by employing router sampling and/or some notion of an overlay network. For example, every node on the overlay could correspond to a collection of routers, where at least one

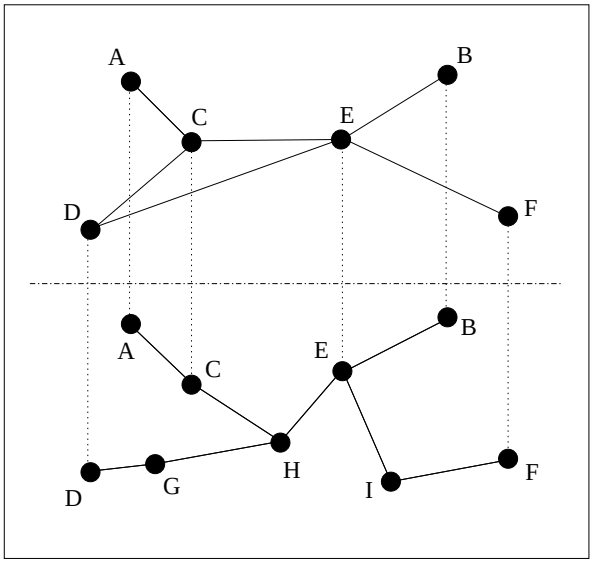


Figure 3.2 Composable Model: Router-Axis Optimization

traceback-capable router exists amongst multiple physical (legacy) routers (Fig. 3.2). Various research ideas have been proposed in the past to support such a scalable and incrementally deployable architecture, including those by authors in [75], [92], [96], etc.

Fragment-axis optimization: Data fragmentation of a router identifier to accommodate various constraints - IP packet size limitations, easy data recovery and/or error correction due to lost or out-of-order fragments, etc. - has been a difficult research problem to address. Various techniques based on robust information-theoretic approaches and probabilistic reasoning have been proposed in the past (Fig. 3.3), including those by authors in [73], [84], [95], etc.

Bit-axis optimization: This problem has also been viewed as the "shorter router labels" problem in literature, and various data encoding techniques with rigorous mathematical analyses have been proposed in this regard - a few examples include ideas in [79], [85], [86], etc.

All traceback schemes known to date fall into exactly one of these three broad categories. Our main contribution thus lies in identifying the fact that the three optimizations described above are not necessarily mutually exclusive, but complementary to each other. We can thus

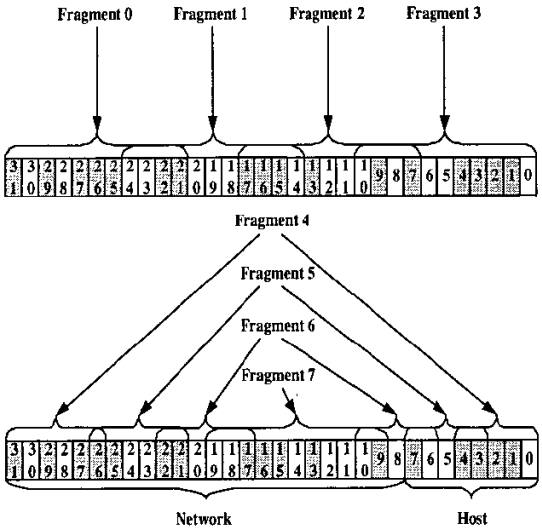


Figure 3.3 Composable Model: Fragment-Axis Optimization

compose these completely unrelated optimizations in any which way we choose, as long as we restrict ourselves to choosing only one optimization from any single dimension, to ensure better aggregate performance. Thus the scope for future research is not restricted to ad-hoc improvements as in the past, but significantly enhanced to systematically encompass newer research into each of the three independent dimensions and also a larger body of work that looks at their combined synergies in real life. This is a significant contribution as it not only provides a clear direction for future research, but also provides a framework for any individual to compare their proposed techniques against a combination of those already known in literature, or against any of the many composable designs feasible, thereby accelerating the development of better attack graph construction techniques.

Illustration: Our intent here is to show that newer research ideas can easily be derived from analyzing the data cube model, and also to reinforce the composable nature of the mutually orthogonal optimizations to yield cumulative benefits. In Chapter 4, we propose a novel multi-axis optimization along the proposed data cube, based on graph coloring primitives that advances the state of the art in data encoding for traceback techniques. In a nutshell, we design a simple packet marking scheme wherein we associate a color to every traceback-enabled router

subject to certain criteria known apriori. The traceback operation then identifies the attack path as a sequence of colors (of the routers) from any attack source to the victim. The use of colors allows for intentional spatial reuse of the conventional bit-space, thus resulting in shorter router labels, and also subsequently requiring far fewer packets to achieve a successful traceback. Additionally, we augment the proposed design with certain graph resizing primitives to allow for greater router sampling along the routing path. Thus, we show that it is possible to design newer DDoS defenses that incorporate individual enhancements along each of the three mutually independent (orthogonal) dimensions of the data cube model.

3.2.2 Data Transmission Schedule

Traditional packet marking traceback techniques capture per-hop behavior, i.e. they transmit information about each intermediate router to the victim so that a global view of the routing path is eventually obtained over multiple packets. Thus traceback data is incrementally gathered and does not function on a per-packet basis. On the other hand, traditional mitigation techniques capture end-to-end behavior, i.e. at some level of abstraction they generate fingerprints that identify the routing paths as opposed to routers themselves; thereby operating on a per-packet granularity. As stated previously, we believe that a good DDoS defense technique should be able to achieve maximal synergy between attack traceback and attack mitigation techniques, by constructing a suitable traffic weighted attack graph, to assist in determining an optimal deployment for the expensive traffic throttling filters throughout the network. Given these myriad considerations, we now primarily focus on the construction of the traffic weighted attack graph by indirect inference rather than the use of any explicit messaging. The subsequent issue of the filter placement problem [99], given the different traffic edge weights and the individual filter costs, is not discussed here, mainly due to the vast amount of research already devoted towards studying this well-known NP-hard problem [100] - usually solved by the use of greedy and/or approximate algorithms.

We have thus far represented all attack paths using the data cube model, and we now

further break it down into a collection of disjoint data cube slices, henceforth referred to as *path signatures*. This novel concept of path signatures serves as a fundamental unit of data transmission, and the receipt of the many path signatures from all attack sources provides the victim a comprehensive understanding of the entire attack graph. The repeated transmission (via in-band packet marking) of the different path signatures in successive packets additionally provides the victim an indirect means of attack traffic characterization from each attack source and thus along each edge of the attack graph - thereby indirectly inferring a traffic weighted attack graph. Fig. 3.4 represents a attack tree constructed at the victim (node 1), and each edge annotated with the relative incoming traffic ratio at each node from its neighbors.

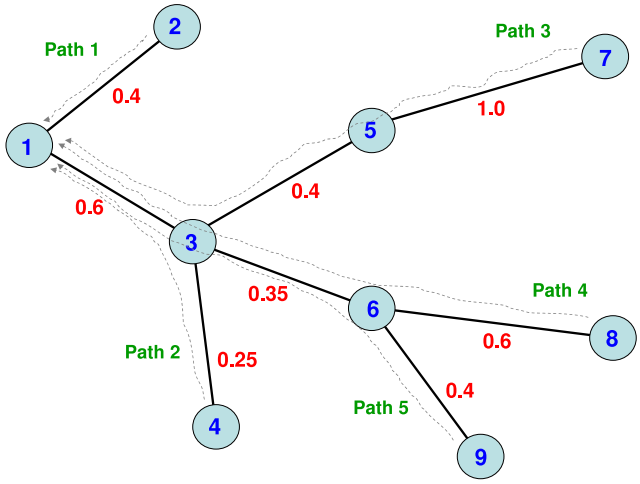


Figure 3.4 Traffic-weighted Attack Tree

Our approach thus entails *splicing* the entire data cube along the lines of path signatures instead of individual router identifiers. Thus we obtain multiple signatures per routing path useful for mitigation at lower thresholds on a per-packet basis, and when viewed as a collection also yields the required traceback data. Thus we can build a homogeneous technique that seamlessly provides us both traceback and mitigation capabilities, operating in parallel.

Path Signatures: The probabilistic packet marking (PPM) scheme [81] has been the most

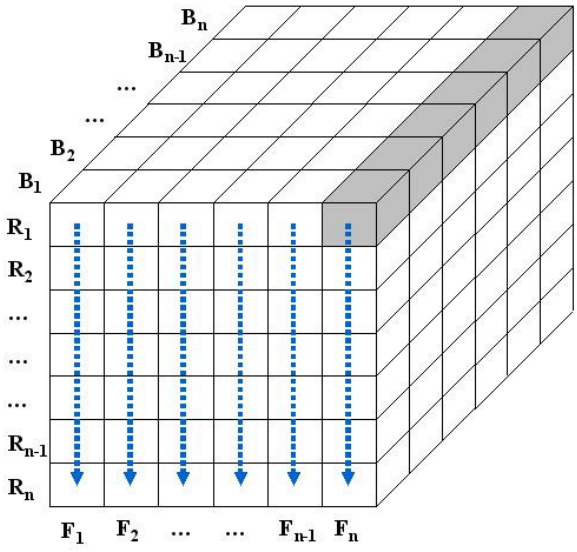


Figure 3.5 Composable Model: Data Transmission Schedule A

widely studied technique in literature - it can be easily argued that PPM and each of its variants represents some random walk on the traceback data cube representing that attack path. Note that we use the term "random walk" loosely here, not necessarily satisfying any strong continuity constraints. If we now assume that the different hops of the random walk each induce a distinct partition in the data cube, then we obtain many disjoint slices (path signatures) of the data cube. Finally, if a random walk is uniquely determined apriori, then the different slices also retain their individual characteristics over multiple walks. Thus the repeated transmission of the path signature slices in successive packets as in-band packet marking information, ensures that the path signatures remain static over time and thus serve as an easy means of clustering the attack traffic at the victim. Thus the path signatures provide both a means of constructing the entire attack graph, and also an indirect means of traffic characterization at the victim.

Slicing the data cube model into disjoint path signatures as described above can be achieved in multiple ways. We now analyze a few random walk strategies to subsequently determine the resulting impact of a partitioning strategy on the overall performance of the attack traceback and mitigation processes. The 3-D traceback data cube in Fig. 3.1 can easily be viewed as a 2-D surface (X-axis and Y-axis) where each unit cell corresponds to a vector (Z-axis) as

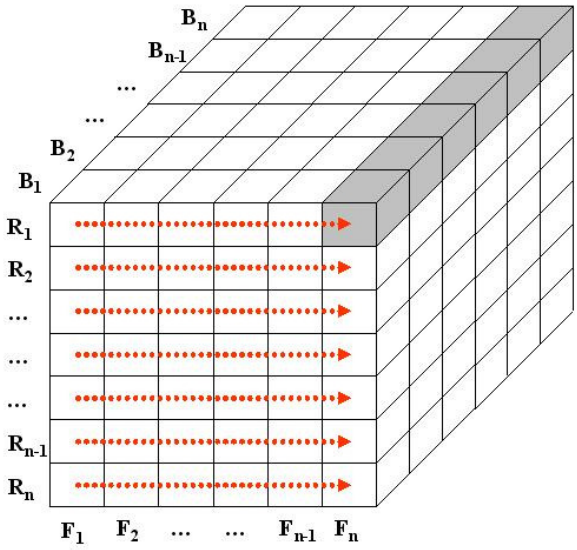


Figure 3.6 Composable Model: Data Transmission Schedule B

indicated by the shaded regions in Fig. 3.5 and Fig. 3.6 respectively. The shaded region thus represents a single path signature, while the different directional arrows represent some random walk on that 2-D surface. The PPM scheme thus reduces to randomly choosing a unit cell (with replacement) from this 2-D surface and sending the corresponding bit-vector to the victim. When all the cells in the 2-D surface have been received by the victim, it can then uniquely identify the attack path.

An alternative random walk strategy is feasible if we view the 3-D data cube as a 2-D surface along the X-axis and the Z-axis, and the unit cell vector along the Y-axis (shaded region in Fig. 3.7). While a large number of such random walks are feasible, we limit our discussion here to the three partitioning techniques discussed here - we later show that these specific walks are essential to bound the overall performance of any DDoS defense mechanism. Hence, the question we answer here is which bits do we prefer to send, as opposed to the problem of how we send them across the network, which has been extensively studied in literature.

Illustration: Our intent here is to show that newer research ideas can easily be derived from analyzing the different transmission schedules on the data cube model. In Chapter 5, we

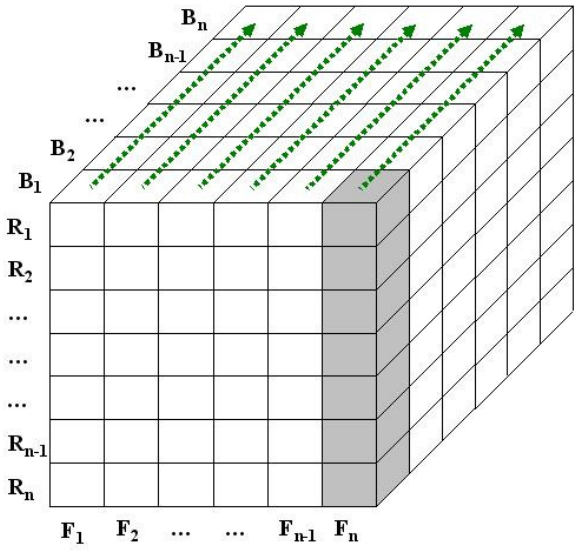


Figure 3.7 Composable Model: Data Transmission Schedule C

propose a novel "random walk" on the proposed data cube, based on distributed divide-and-conquer principles that significantly advances the state of the art in the design of single-packet traceback techniques. In a nutshell, we design a simple packet marking scheme wherein each router marks only a subset of the attack path (data cube), bound by that router on one end and the attack source at the other. The proposed design then entails the bottom-up evolution of these attack sub-path signatures across the different routers from the packet source to the victim, thereby representing a newer approach to transmission of the data cube. In other words, it exploits the fact that each attack path (data cube) representation in the Internet is unique with respect to the victim, and since the number of attackers at any point in time is considerably smaller (sparse) compared to the total number of Internet paths, a much smaller representation can easily be derived - thereby realistically achieving single-packet traceback and mitigation guarantees on the Internet today.

In this manner, we show that it is possible to incorporate the proposed multi-axis representation and transmission schedule optimizations on the data cube model, to design newer practically viable DDoS defenses and thereby further improve the state of the art.

3.3 Theoretical Analysis

While the proposed data cube encompasses all the information needed by the victim to trace any particular attack source, it is not always necessary that the entire data cube be made available before any attack mitigation steps can be initiated. Initially when an attack is launched, the search space of the attack paths for the victim is the entire set of the Internet routing paths. However, as the victim receives more and more data about the attack path by collecting the path signatures from the different attack packets, it can progressively prune down the search space till it has sufficient data to flag a single path in the Internet as the attack path. Thus, traceback can be viewed as an incremental process where the search space is steadily pruned down till a unique element remains in the search space. The use of blacklists, upstream router maps, distributed network monitoring, and other apriori information can enable the victim to approximately estimate an attack source and its routing path, having received only a small fraction of the entire traceback data cube. For example, a victim could be seeded with information that any routing path that traverses routers α and β, also traverses routers γ, δ, and ω in order. Thus an early warning system based on incremental knowledge of the attack graph can assist in early deployment of packet/stream filters, thereby truly supporting traceback-enabled *incremental mitigation* on the Internet.

3.3.1 Data Utility

It is obvious that the speed of the traceback operation is thus determined by the rate of receiving useful incremental information. Hence the transmission schedule used to transmit the entire traceback data to the victim plays a critical role in determining its response time to various DDoS attacks. We now define a fundamental concept called *utility* of a particular bit of information [101] [102], which helps us choose an optimal transmission schedule for the traceback data cube. The basic idea is to associate different weights with each of the bits in the data cube, such that a constraint of maximizing speed leads us to an optimal transmission sequence, that also maximizes the total utility gained in a certain fixed number of packets. Suppose a particular router identifier has a k-bit representation. The search space for this

router identifier for the victim is the range of 2^k different values (or something smaller given some external apriori information). Supplying a single bit of data from the router identifier to the victim thereby reduces the search space to 2^{k-1}, subsequently a second bit of data would reduce the search space to 2^{k-2}, and so on.

We now define *utility* as the ratio of the reduction in the search space due to the knowledge of that particular bit to the total original search space. Thus the utility of the 1^{st}, 2^{nd} and m^{th}-bits are as shown in Eqn. 3.1. As discussed previously, if each of the r router identifiers are split into k fragments, each m bits long, then the utilities of the first, second and t^{th} fragments are as shown in Eqn. 3.2. Note that the terms first, second and so on here do not refer to the digits in the MSB or LSB of the binary number, rather they refer to that bit (any one of the k bits) that was transmitted first, second and so on. Hence, the highest utility rate is achieved when all high utility fragments are scheduled first for transmission.

$$u_1 = \frac{2^k - 2^{k-1}}{2^k} = \frac{1}{2} \quad ; \quad u_2 = \frac{2^{k-1} - 2^{k-2}}{2^k} = \frac{1}{2^2} \tag{3.1}$$

$$u_m = \frac{2^{k-(m-1)} - 2^{k-m}}{2^k} = \frac{1}{2^m} \tag{3.2}$$

The total utility of all bits of a message is as shown in Eqn. 3.3. The $\frac{1}{2^k}$ represents the *self-information* of a k-bit message. In other words, it represents the probability of finding that particular k-bit message in the entire search space, and is thus the implicit information possessed by that k-bit message. We thus see that the total utility of any message (router identifier) is always one, i.e., the shaded portion (slice) of the cube in Fig. 3.1 has a total data utility of one. If the entire attack path consists of r intermediate routers , then the total utility of the traceback data cube is then equal to r.

$$\sum_{i=1}^{k} u_i = \sum_{i=1}^{k} \frac{1}{2^i} = 1 - \frac{1}{2^k} \tag{3.3}$$

We thus see that not every bit of information in the traceback data cube has the same

utility, as has been assumed by researchers to date. *The utility of a bit is governed by how much information has already been transmitted to the victim about a particular slice (router identifier) in the cube.* Thus it is clearly evident that the higher the utility achieved in a certain fixed number of packets, the smaller the search space for the attack path, and hence faster the traceback/mitigation process. We now use the concepts of data cube and utility, to derive higher utility rate transmission schedules and hence faster traceback schemes.

We now define the *utility rate* as the cumulative utility achieved from the traceback data cube for a certain fixed number of packets.

Utility Bounds: We analyze the basic PPM based traceback scheme [81] here, as it has been the most widely studied technique in literature, and also as its analysis holds true for all the different flavors of PPM schemes proposed to date. We now quantify the utility rate of the basic PPM scheme and clearly demarcate its operating region in the utility space.

As described previously, the PPM scheme essentially reduces to a random walk on the traceback data cube as illustrated in Figs. 3.5 and 3.6. If the r router identifiers are each split into k fragments, each m bits long, then the utilities of the first, second and t^{th} fragments for each router identifier are as shown in Eqns. 3.4 and 3.5.

$$u_{F_1} = (1 - \frac{1}{2^m}) \quad ; \quad u_{F_2} = (\frac{1}{2^m}) * (1 - \frac{1}{2^m}) \tag{3.4}$$

$$u_{F_t} = \sum_{i=m(t-1)+1}^{mt} u_i = (\frac{1}{2^{m(t-1)}}) * (1 - \frac{1}{2^m}) \tag{3.5}$$

Hence, highest (lowest) utility rate is achieved when all high (low) utility fragments are scheduled first for transmission. Consider the transmission schedule in Fig. 3.5 where all routers send their first fragment, then their second fragment, and so on. Also consider another transmission schedule in Fig. 3.6 where a particular router sends all its fragments before the

next router is allowed to transmit, as in an implicit token passing system. These represent the highest and lowest utility rate transmission schedules respectively. It is easily seen that any other transmission schedule attains a utility rate in the operating region bounded by these two curves (red & blue respectively) as shown in Fig. 3.8. In this graph, the X-axis represents the total number of packets, while the Y-axis represents the total utility achieved respectively.

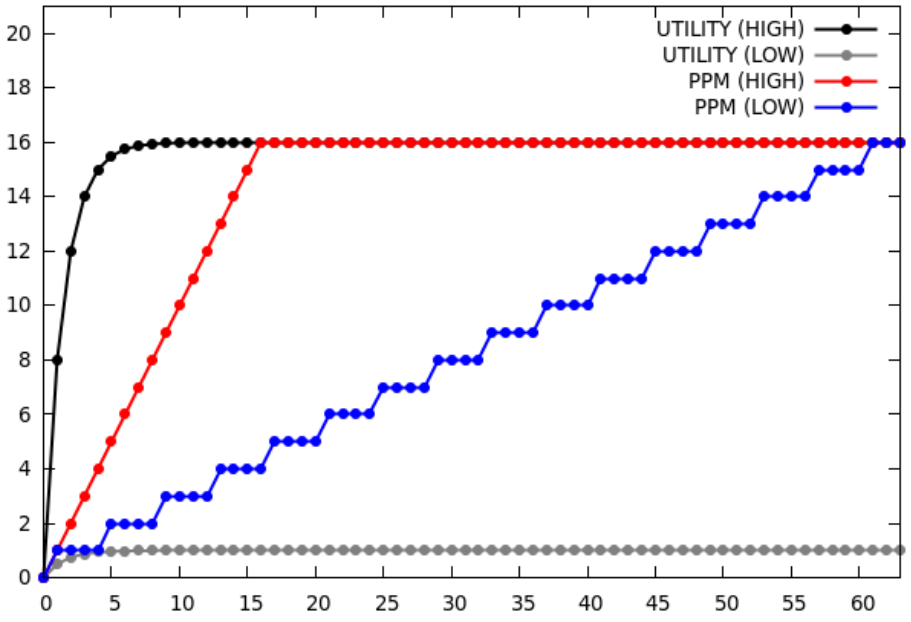


Figure 3.8 Theoretical Bounds for Utility Rate (#packets vs #utility)

Now consider a different transmission schedule as illustrated in Fig. 3.7. This transmission schedule indicates a logical departure from the concept of individual router identifiers to that of path signatures. All the routers send their first bit, then their second bit and so on. Any packet now has only one bit from each router in order, and is hence classified as a path signature. The utilities for the first, second, and t^{th} packets now are as shown in Eqn. 3.6.

$$u_{P_1} = \frac{m}{2} \quad ; \quad u_{P_2} = \frac{m}{2^2} \quad ; \quad u_{P_t} = m * \frac{1}{2^t} = \frac{m}{2^t} \tag{3.6}$$

It can be easily inferred that the proposed transmission schedule forms the upper bound

on the utility that any traceback scheme can achieve. The operating region in utility space of this transmission schedule is shown in Fig. 3.8 (black & grey curves). The lower bound here is caused by the potential loss of the packet fingerprint due to over-writing of the incremental path fingerprint bits by the different intermediate routes. In Fig. 3.8, we assume r=16 (routers), k=4 (fragments), m=16 (bits), for illustrative purposes.

3.3.2 Packet Overhead

We now compare the traditional PPM based schemes and the proposed utility based data transmission technique here, based on the number of packets required for a successful traceback.

Let there be r routers along the routing path, each having a marking probability of p, and whose identifiers each have k m-bit fragments. The probability of receiving a mark from a router i hops away is $P_i(M_m)$ (Eqn. 3.7). If we conservatively assume that marks from all routers appear with the same likelihood as the furthest router, then th probability that a packet delivers a mark from some router is $P(M_m)$ (Eqn. 3.8). From generalized Coupon Collector Problem [103] [104] in Probability Theory, and detailed analysis in [81], the number of packets X_m needed to reconstruct the routing path from all the different fragments for PPM based scheme is given by Eqn. 3.9.

$$P_i(M_m) = p(1-p)^{i-1} \tag{3.7}$$

$$P(M_m) \geq rp(1-p)^{r-1} \tag{3.8}$$

$$E[X_m] < \frac{k * log_e(kr)}{p(1-p)^{r-1}} \tag{3.9}$$

For the proposed utility based scheme, we have km (r-bit) fragments (vectors), and hence the number of packets X_r needed to reconstruct the routing path is given by Eqn. 3.10.

$$E[X_r] < \frac{k * log_e(km)}{p(1-p)^{r-1}} \tag{3.10}$$

If we choose r=m, then our proposed scheme performs no worse than PPM schemes in reconstructing the routing paths from information obtained from all the different fragments.

3.4 Experimental Evaluation

We now compare the traditional PPM based schemes and the proposed utility based data transmission technique here, based on the number of packets needed for a given utility rate.

Data Utility Rate: Often we need to shortlist a few candidate suspect routing paths and not identify all unique attack paths during high volume DDoS attacks, as the immediate need is always to choose optimal (upstream) network locations where mitigation may be deployed. The constraint in such cases is usually to identify the most likely suspect paths in the top 2-5 percentile. Such a constraint not only reduces the effect of the inaccuracies of the traceback process, but also reduces the response time and overall operational complexity.

We performed an extensive evaluation of the basic PPM scheme and the proposed utility based traceback scheme, over a candidate router graph obtained from Rocketfuel [105]. Along the lines of the probabilistic calculations above, Fig. 3.9 (Fig. 3.10) shows the actual number of packets required to filter down to the top $2, 5, 10, 20$ percentile suspect routing paths for the two schemes respectively. In this graph, the (log-scale) X-axis represents the total number of packets, while the Y-axis represents the total utility achieved respectively. We evaluate 2 scenarios namely, *100%* where all intermediate routers are traceback capable, and *50%* where half of them are legacy routers, incapable of traceback. It is to be noted that the performance degradation even in the face of limited deployment is very minimal. We thus notice that for the same given utility rate, the number of packets required by our proposed scheme is roughly an order of magnitude lesser than those needed by the traditional PPM based schemes.

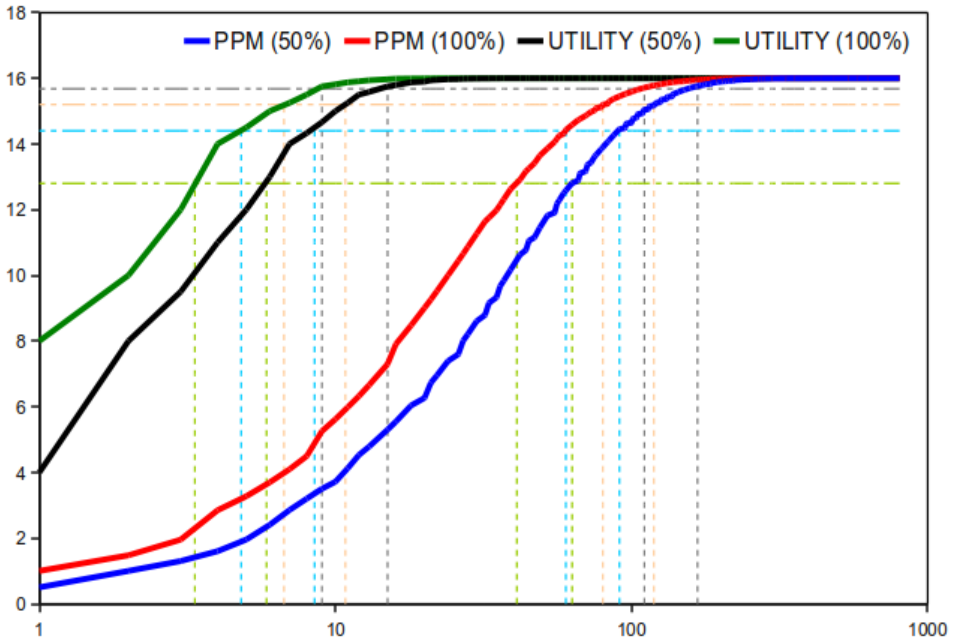


Figure 3.9 Utility Rate: Top-k Percentile Filtering (#packets vs #utility)

Percentile (16-bit packet)	PPM (50% Legacy)	PPM (No Legacy)	Proposed Scheme (50% Legacy)	Proposed Scheme (No Legacy)
2	167	111	15	9
5	119	80	11	7
10	91	60	9	5
20	63	41	6	4

Figure 3.10 Utility Rate: Number of Packets

The proposed composable data cube model, employing multi-axis and transmission schedule optimizations, thus helps us not only analyze any traceback technique proposed to date, but also design and evaluate newer techniques in a fairly structured manner. In the next few chapters, we describe a few novel DDoS defense techniques that not only harness this framework, but also adopt the data utility concept for greater benefits - thereby making the dream of a practically viable DDoS defense a potential reality. The list of all our publications that drew inspiration from these ideas here include those in [106]-[123].

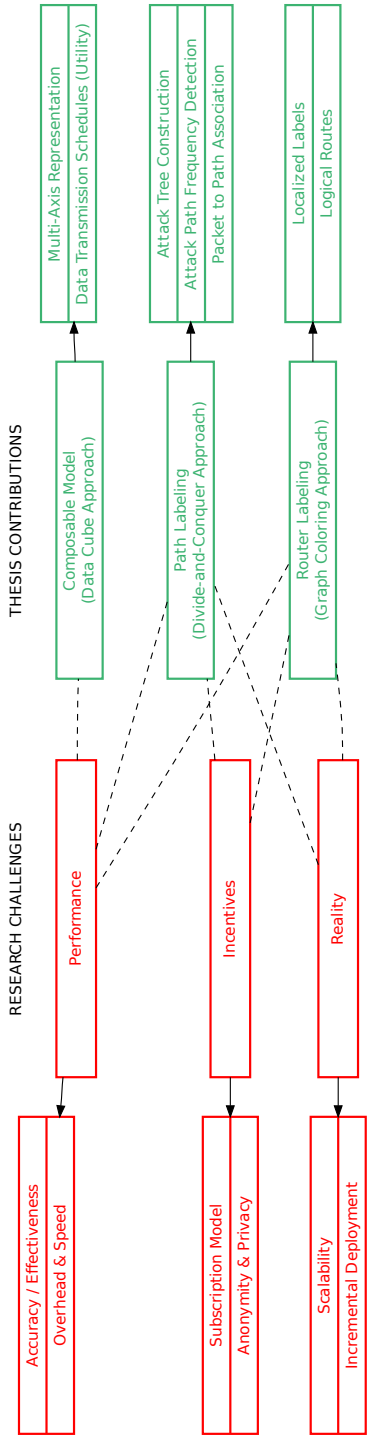


Figure 3.11 Thesis Contributions

CHAPTER 4. ROUTER LABELING: A Graph Coloring Approach

As discussed in Chapter 2, packet marking traceback techniques store a fingerprint of the different network routers that forwarded a certain data packet, that may then be retrieved as needed to reveal the true origins of that packet. Its two primary functions thus include - the translation of the physical identities of the many packet forwarding routers to a concise fingerprint (encoding), and the inverse operation of accurately translating any fingerprint back to its original list of intermediate routers (decoding) for all suspicious packets. The traceback encoding/decoding processes thus deal with the core issues of what to, how to, and where to store the fingerprint. We now seek to improve the state of the art in IP traceback by employing novel graph theory concepts to provide greater security and performance guarantees for a practically viable DDoS defense solution. Given the many different variants proposed to date, we limit our scope here to probabilistic packet marking techniques [81] for traceback encoding and the pushback mechanism [53] for traceback decoding, and use them as fundamental building blocks for our proposed idea here.

4.1 Proposed Concepts

As discussed previously, the size of the in-band packet fingerprint used by packet marking traceback techniques is governed by two main factors: the size of each router identifier, and the number of routers along the forwarding path. In this regard, the explosive growth and increasing connectivity of Internet hosts has led to a steadily increasing network size/diameter, thereby mandating the design of newer approaches to limit the ever-increasing packet fingerprint size to more manageable levels. To address this issue, we propose two different optimizations here,

namely reducing the size of each router identifier (node width) and reducing the number of routers in the forwarding list (path length).

Localized Labels: *(Node Width Optimization)* We propose a simple router labeling scheme that assigns randomly generated labels (identifiers) to each router such that the potential reuse of the same label across multiple routers in the network is highly likely. Additionally, in order to allow for unambiguous fingerprint decoding, we do constraint the randomized router labels to be unique only in their immediate vicinity and not in the global context. A critical observation we make here is that the node width to represent these localized router labels now depends only on the cardinality of the set of routers assigned the same label, and not the total network size. Hence the proposed optimization can not only yield smaller router labels, but also scale well to satisfy the ever-increasing network size/bandwidth.

Logical Routes: *(Path Length Optimization)* We also propose a simple graph resizing technique that provides individual routers significant control over the nature and the extent of logical abstraction of the network topology. It specifically permits the use of pseudo-identities to let a group of routers (probably across multiple ISP boundaries) to masquerade as a single entity, and/or a single router to masquerade as multiple network entities. Thus the proposed optimization can not only suitably modify the effective path length, but also scale well to satisfy the ever-increasing network size/bandwidth.

We now discuss these ideas in greater detail, justifying our specific design choices and their resulting impact on the performance of any DDoS defense mechanism that employs them.

4.1.1 Localized Labels

The traditional probabilistic packet marking techniques have usually assigned the IP address or its hash encoding as the identifier (label) for the different routers in the network, such

that the traceback decoding process can easily reconstruct the identities of the intermediate routers simply by inspecting their individual fingerprints. On the other hand, our proposed router labeling scheme is essentially a simple graph coloring algorithm that assigns randomly generated labels to each router. Hence, the existing traceback decoding algorithms may not work due to the lack of an explicit mapping from a router label to the identity of the router that marked that label, and may lead to false positives/negatives. Hence, in order for the proposed traceback encoding process to be operational, we would need to adapt the traceback decoding process accordingly.

The popular pushback mechanism for traceback decoding works by querying each router about its neighboring routers, in the reverse direction from the destination to the source for each suspicious data packet. Thus, during *incremental per-hop discovery*, given any router label sequence, a router must be able to uniquely distinguish which of its neighbors originally issued the given label. In other words, any two path segments to a common destination need to at least have different router labels at their convergence point to ensure correctness during the traceback decoding process. We now formally analyze this requirement and the resulting constraints on the random label assignment technique.

Lemma 4.1.1. *No two path segments to any destination can be assigned the same sequence of labels for correctness in traceback decoding.*

Proof: Consider two network paths l_1 and l_2, converging at some node m. Let nodes s_1 and s_2 be their respective sources, and node d their common destination, such that $(s_1 \rightsquigarrow m \rightsquigarrow d)$ and $(s_2 \rightsquigarrow m \rightsquigarrow d)$. If all nodes are now assigned the same color t, then m cannot uniquely identify whether l_1 or l_2 sent any particular packet. However, if the two neighbors of m are assigned unique labels t_1 and t_2, it can easily detect that path l_1 sent color $k(= t_1)$ or that path l_2 sent color $k(= t_2)$, without any uncertainty. Note that m here can represent s_1, s_2, d or any other intermediate node in the network. □

Corollary 4.1.2. *Assigning distinct labels to all neighbors of a node is a necessary condition for traceback correctness.*

Proof: Assigning distinct labels to all neighbors of a node ensures that while global namespace collisions may exist, local namespace collisions in the neighborhood of any node are eliminated. This local uniqueness of router labels ensures that any two converging path segments are always assigned distinct router label sequences, and according to Lemma 4.1.1 also necessary to guarantee traceback decoding correctness along all paths in the network. □

Corollary 4.1.3. *Assigning distinct labels to a node and all its neighbors is a sufficient condition for traceback correctness.*

Proof: Assigning distinct labels to a node and all its neighbors ensures that every edge is bi-colored, and is thus a stronger condition than the necessary condition above. □

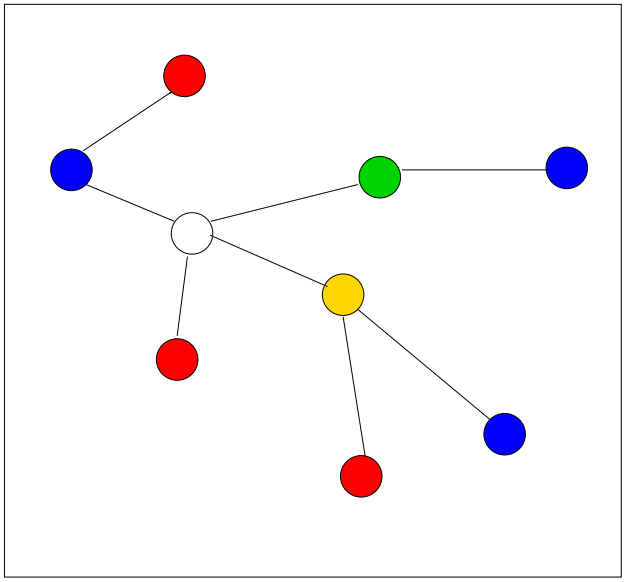


Figure 4.1 Localized Labels

Fig. 4.1 now represents an instance of router label reuse wherein randomly generated labels have been assigned to each network router. In Fig. 4.2, the need for this router labeling constraint is clearly evident due to local namespace collision at the convergence of the two network

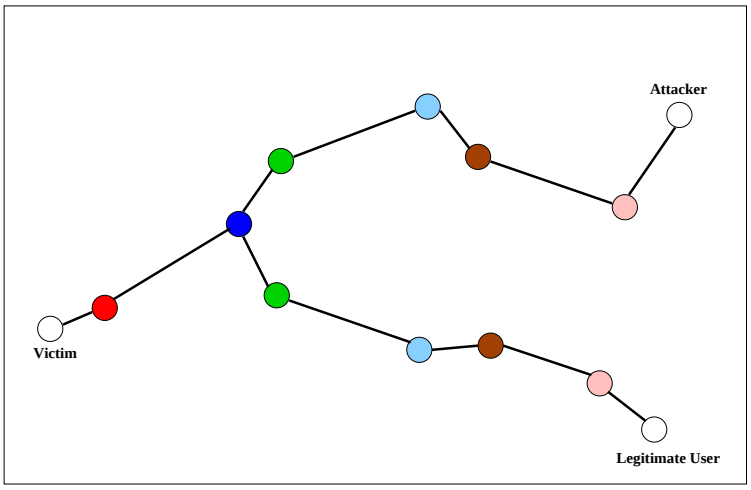


Figure 4.2 Ambiguous Traceback Decoding - Need for Star Coloring

paths. Fig. 4.1 thus presents a router labeling that adequately satisfies this constraint, thereby supporting precise attack source/path identification without any false positives/negatives during the pushback (decoding) process. Also, note that the proposed use of pseudo-identities in the logical route abstraction optimization would require each router to consume more than (or less than) a single router label during the traceback decoding process to ensure an accurate pushback along the actual physical network route.

The *sufficient condition* described above is popularly known in literature as the star coloring or distance-2 coloring problem [124]. While it is far more restrictive than the required *necessary condition* [125], we adopt it here due to the large body of prior work in designing optimal star coloring algorithms for generic graphs. As graph coloring primitives form the crux of our proposed traceback design/implementation, we study the generic coloring problem, its different variants, theoretical bounds and algorithms in great detail here.

Star Coloring: The *graph coloring* problem is defined as an assignment of colors (labels) to various objects in the graph, namely vertices, edges, faces, or any combination thereof. While node and edge coloring are both equally relevant in this context, we limit our discussion here to node coloring only - edge coloring techniques can be derived along similar lines. We now

define *proper vertex coloring* as an assignment of colors to all vertices in the graph such that no two adjacent vertices are assigned the same color. We also define *star coloring* as a proper vertex coloring wherein no path of length three is bi-colored. In other words, it is a variant of the *distance-k coloring* wherein no two vertices are assigned the same color if the shortest path between them is k-hops or less.

Theoretical Lower Bound: Consider a graph G having a maximum node degree Δ (each vertex V_i has a node degree Δ_i), and requiring $\chi(G)$ colors for a star coloring. For any vertex V_i in G, as a unique color is assigned to itself and all its 1-hop neighbors, a minimum of $\Delta_i + 1$ colors are required. Thus a minimum of $\Delta + 1$ colors are required for a star coloring of the graph G. Fig. 4.1 shows a graph requiring $\Theta(\Delta)$ colors for star coloring.

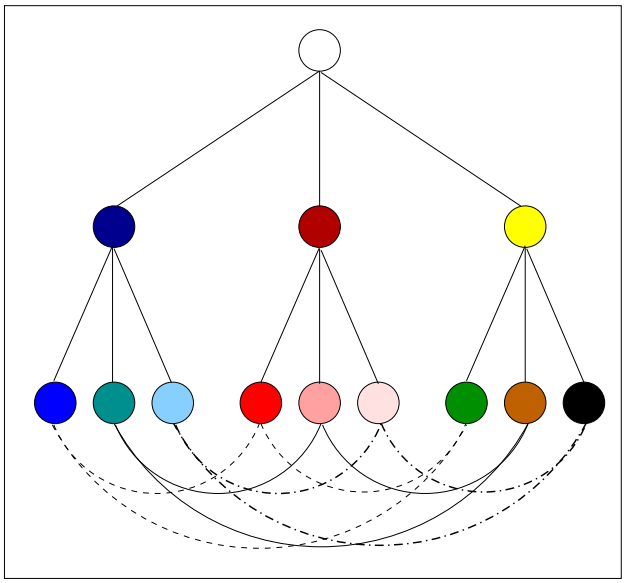


Figure 4.3 Star Coloring - $\Theta(\Delta^2)$ Colors

Theoretical Upper Bound: A star coloring of graph G, requires every path of length three in G to be assigned three distinct colors. For any vertex V_i, a path of length three consists of one of its 1-hop neighbors and one of its corresponding 2-hop neighbors. Thus, in the worst case where each such path segment has to be assigned distinct colors, the total number of colors

required to color the neighborhood of a vertex would be one more than the sum of its 1-hop and 2-hop neighbors. As the number of 2-hop neighbors is bound by $(\sum_{\forall j \in \Delta_i} \Delta_j - \Delta_i)$, the total number of colors required is bound by $(\sum_{\forall j \in \Delta_i} \Delta_j + 1)$. Thus for a star coloring of graph G, we would require $(\Delta^2 + 1)$ colors in the worst-case.

There exist an infinite class of graphs that require a minimum of $\Theta(\Delta^2)$ colors for star coloring (Fig. 4.3). These graphs are essentially $\lceil \frac{\Delta}{2} \rceil$-regular trees on three levels with leaves interconnected to form $\Theta(\Delta)$ cliques, each consisting of $\lceil \frac{\Delta}{2} \rceil$ nodes; where each leaf belongs to two different cliques, one formed by the siblings at each leaf level in the base tree, and the other formed by leaves with different parents and corresponding positions in the children ordering. Such a graph has $\Theta(\Delta^2)$ nodes and diameter 2 and requires $\Theta(\Delta^2)$ colors for star coloring. This upper bound is however a very loose bound and tighter bounds have been proven in [126]. A more formal derivation and proof of these bounds can be found in [126] [127].

For practical considerations, we express the number of colors required for star coloring as in Eqn. 4.1, and algorithmically bound it as in Eqn. 4.2. A detailed description of these theoretical bounds can be found in [126].

$$(\Delta + 1) \leq \chi(G) \leq (\Delta^2 + 1) \tag{4.1}$$

$$\Omega(\Delta) \leq \chi(G) \leq O(\Delta^2) \tag{4.2}$$

The star coloring problem has been shown to be NP-complete for general graphs and also when restricted to planar graphs in [127]. While various approximate algorithms have been proposed in literature, we prefer a simpler greedy approach (Algorithm 1), trading optimality for easy of operation - a formal analysis of this centralized algorithm being presented in [126]. Various other practically viable algorithms have also been proposed for star coloring, including parallel/distributed algorithms in [128] [129], and are discussed in more detail below.

Algorithm 1 Greedy Star Coloring Algorithm

input: A graph $G = (V, E)$
output: A coloring $c : V \rightarrow 1, 2, 3, ...$

for all u such that $u \in V$ **do**
  $\mu \Leftarrow \phi$
  for all v such that $(u, v) \in E$ **do**
    $\mu \Leftarrow \mu \cup c(v)$
    for all w such that $(w, v) \in E$ **do**
      $\mu \Leftarrow \mu \cup c(w)$
    end for
  end for
  $c(u) \Leftarrow$ least color $\notin \mu$
end for

The greedy star coloring algorithm described above runs in $O(n\Delta^2)$ time and uses $O(\Delta^2)$ colors, where n is the number of nodes in the network [124]. The number of colors is thus dependent not on the total network size, but on the maximum router degree in the network, and hence lends well to shorter packet fingerprints and a highly scalable deployment.

4.1.2 Logical Routes

The traditional probabilistic packet marking techniques have usually assigned a single identifier to each router in the network. On the other hand, our proposed logical abstraction scheme is essentially a simple graph resizing algorithm that assigns partial or multiple labels to each router, to suitably modify the effective path length for all packets in the network. For example, in Fig. 4.4, we transform a physical layer network to its underlying logical abstraction - node *F* induces a sybil node *I*, while nodes *C, D, E* collaborate to induce nodes *G, H*. We now propose multiple different logical abstraction techniques, namely expansive coloring, contractive coloring, and hybrid coloring for better abstraction of the network topology.

Expansive Coloring: Consider a node n with degree d, requiring $d+1$ router labels for star coloring. If we now replace node n with k pseudo entities $n_1, ..., n_k$, then the effective degree is reduced to $\frac{d}{k}$, assuming a uniform distribution of neighbors across all logical entities - we now require only $k + \frac{d}{k}$ router labels. Thus for large values of d and appropriate choice of k,

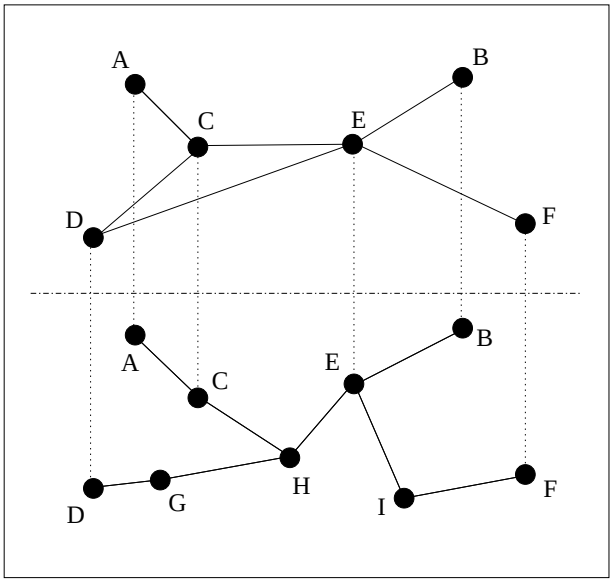


Figure 4.4 Logical Routes

we can significantly reduce the number of router labels.

The degree distribution in the Internet router-level and AS-level topologies follows a power-law distribution [130]. Thus the presence of a few high-degree nodes can significantly increase the node width for star colored router labels. In this context, the proposed expansive coloring technique can greatly reduce the skew by replacing every high-degree physical node with a fully-connected mesh of logical nodes. As it essentially increases every 3-hop path to a 4-hop path in the vicinity of any node, greater spatial reuse of colors is feasible, thereby yielding a smaller node width representation. Note that this graph transformation is fully localized to the node itself and not visible to any other entity in the network.

Contractive Coloring: Consider k nodes $n_1, ..., n_k$ with degrees $d_1, ..., d_k$ and requiring $k + \frac{d_i}{k}$ router labels for star coloring respectively. If we now replace them with a single pseudo entity n, the effective path length is now shorter by at least one hop in all directions. Also, the effective degree is now bound by $\sum d_i \pm \frac{k^2}{2}$ and for small values of d and appropriate choice of k, we can achieve significantly better node width utilization.

Along similar lines, we notice that the presence of a large number of low-degree nodes [130] significantly increases the path length and renders a large portion of their node width representation unused. In this context, the proposed contractive coloring technique can greatly reduce the effective path length by abstracting away a subgraph of these low-degree nodes. In addition to reducing every 4-hop path to a 3-hop path in the vicinity of any node, the increased node degree also yields a better node width utilization. Note that this graph transformation is again fully localized to a node group itself and is not visible to any external network entity.

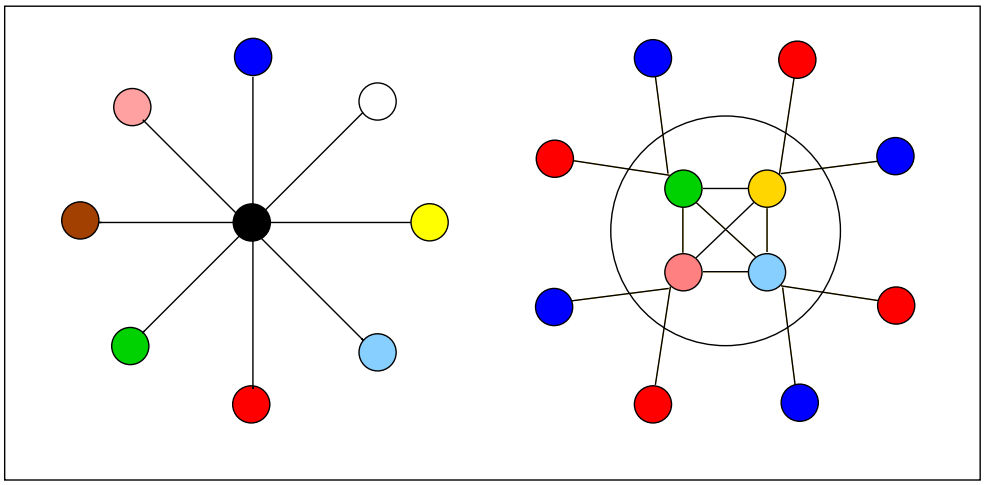


Figure 4.5 Graph Resizing

Fig. 4.5 shows a network topology with a single node requiring 9 router labels, and another network topology with a logical mesh of 4 nodes requiring 6 router labels for star coloring. While the graph transformation from the first to the second topology represents expansive coloring, the inverse transformation from the second to the first topology represents contractive coloring - thereby displaying natural duality by design.

Hybrid Coloring: These graph resizing techniques are thus complementary in nature - while one reduces the node width at the cost of increased path length, the other reduces the effective path length at the cost of a larger node width. Hence we now propose a hybrid coloring ap-

proach that employs these complementary graph resizing techniques simultaneously to achieve *router degree homogenization* in any network topology. In this context, we envision a large number of incremental variations that can be supported - multiple graph resizing iterations at each router, periodic changes to the associated labels and/or the very nature of the graph resizing techniques employed at each router, etc. However, in order to prevent the disruption of the pushback process during the transition phase, we would require each router to support both forms of traceback during this volatile transition period and ensure sufficient (mutual) isolation by not reusing identical router labels (colors) across multiple consecutive transitions.

We have thus proposed two novel concepts to achieve an optimal packet fingerprint size - while star coloring provides scalability by node width optimization for router labels, graph resizing provides scalability by path length optimization for logical route abstraction.

4.2 Practical Considerations

As discussed previously, the traceback-assisted mitigation techniques require not only the actual attack sources but also the entire routing path for their attack packets, to assist in quick attack graph construction and for efficient distributed filter placement for appropriate attack traffic throttling across the entire network. Thus the proposed packet marking technique, employing localized labels and logical routes, needs to be both scalable and incrementally deployable to be practically viable on the Internet scale.

4.2.1 Scalability

The proposed graph coloring technique due to its centralized nature induces a single point of failure for global router label assignment, and is hence not completely reliable or scalable, and is a significant performance bottleneck. Additionally, as the central processor must now be aware of the entire network topology and the individual router choices, it poses a critical

security concern. To address these diverse concerns, we now provide a truly distributed implementation of the proposed graph coloring algorithms.

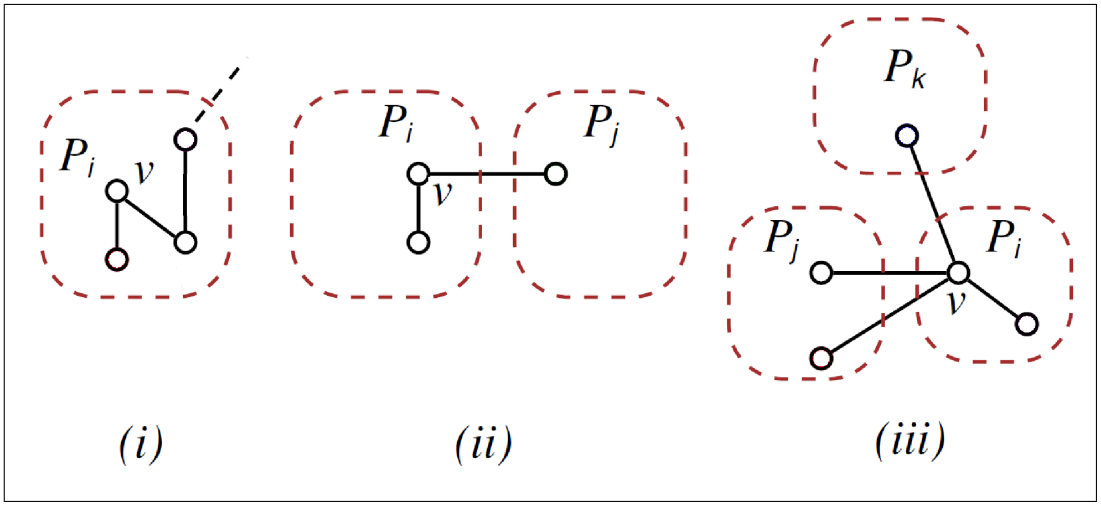


Figure 4.6 Scalable Router Labeling

Distributed Coloring: Consider a graph G distributed across p processors, where each processor P_i owns a vertex subset V_i, and also stores the list of all vertices V_j' in other processors P_j that share an edge with vertices in P_i. This induces two disjoint groups - *interior vertices* whose 1-hop neighbors are on the same processor, and *boundary vertices* which share an edge with at least one boundary vertex on a different processor.

Interior Vertices: As star coloring conflicts for all interior vertices on a single processor can be locally verified, they are easily colored using the proposed centralized greedy approach in Algorithm 1. Additionally, as interior vertices on different processors are now separated by at least a 3-hop path, delineated by a boundary vertex on each processor, all interior vertices across all mutually independent processors can thus be colored in parallel.

Boundary Vertices: As direct assignment of labels to any boundary vertex can cause star coloring conflicts due to already colored 1-hop and 2-hop neighbors on other processors, we propose a 2-phase iterative approach here. During the *speculative coloring phase*, each uncol-

ored node is assigned a random color which is then relayed to all its 1-hop neighbors. In the *conflict detection phase*, each node finalizes its speculative color if no star coloring conflicts are detected. The coloring process thus terminates when all vertices are colored - note that a processor cannot terminate once its vertex subset has been colored, as it needs to detect potential conflicts induced subsequently by other processors.

Algorithm 2 Distributed Star Coloring Algorithm

input: Graph $G = (V, E)$, divided into p subgraphs (V_i, E_i)
output: A coloring $c : V \rightarrow 1, 2, 3, ...$

▷ **at each processor** P_i *(run in parallel)*
for all v such that $v \in V_i$ **do**
  set $c(v) \Leftarrow 0$
end for
while $\exists v \in V$ such that $c(v) = 0$ **do**
  —- *Barrier sync for speculative coloring* —-
  while $\exists v \in V_i$ such that $c(v) = 0$ **do**
    set $spec(v) \Leftarrow$ *random-color*
    for all w such that $(w, v) \notin E_i$ **do**
      transmit m_w^v to $Processor(w) \Leftarrow spec(v)$
    end for
  end while
  —- *Barrier sync for conflict detection* —-
  for v such that $v \in V_i$ **do**
    $\mu \Leftarrow \phi$
    for w such $(w, v) \in E$ **do**
      $color \Leftarrow c(v) \mid spec(v) \mid m_v^w$ *(choose appropriately)*
      $\mu \Leftarrow \mu \cup color$
    end for
    if $\mu(i) \neq \mu(j), \forall i, \forall j$ **then**
      $c(v) \Leftarrow spec(v)$
      for all w such that $(w, v) \notin E_i$ **do**
        transmit m_w^v to $Processor(w) \Leftarrow c(v)$
      end for
    end if
  end for
end while
run *Greedy-Coloring*(G_i) for non-boundary vertices

The proposed distributed coloring technique thus proceeds in multiple rounds till it eventually converges to a fixed configuration (Algorithm 2). The different scenarios that may lead to conflicts in distributed star coloring are now shown in Fig. 4.6, and explained in greater detail in [129]. Finally, various optimizations to achieve faster convergence using forbidden colors for

each vertex, efficient inter-processor communication, etc. have also been studied [128].

Thus the proposed distributed coloring approach where the different processors each color a small network subgraph only, not only provides better resilience against failures, but also restricts the scope of data accumulation at each processor to its vertex subset only. In this regard, assigning a unique processor to each AS network such that the intra-AS edge (core) routers represents the boundary (interior) vertices respectively, provides a natural grouping along administrative domains, each managed by the different network operators. Alternatively, we could assign a single processor to each router in the network, thereby lending well to a truly distributed and scalable Internet-wide deployment.

To provide greater resilience against node failures, we may also replace the global barrier sync model with localized read/write locks obtained for short time intervals [131]. While we do not delve into details here, it can be easily deduced that the number of iterations that each node in the network has progressed through can significantly vary - nodes with higher computation capabilities, high speed connectivity, and fewer failures can progress much faster than other slower failure-prone nodes. In this regard, special care needs to be taken to ensure that nodes with high connectivity do not spend a large majority of their time in distributed lock management, and mutually collaborate to make steady incremental progress over time.

4.2.2 Incremental Deployment

As the Internet consists of a large number of legacy systems, many of them in use for more than a decade now, the complete roll-out of Internet-wide traceback would require a few years at best. Hence any practically viable traceback technique should be functional given only a few active nodes, and also support a gradual roll-out to newer nodes. A highly desirable feature in this regard would be a steady improvement in performance and accuracy as more traceback-capable nodes are deployed in the network.

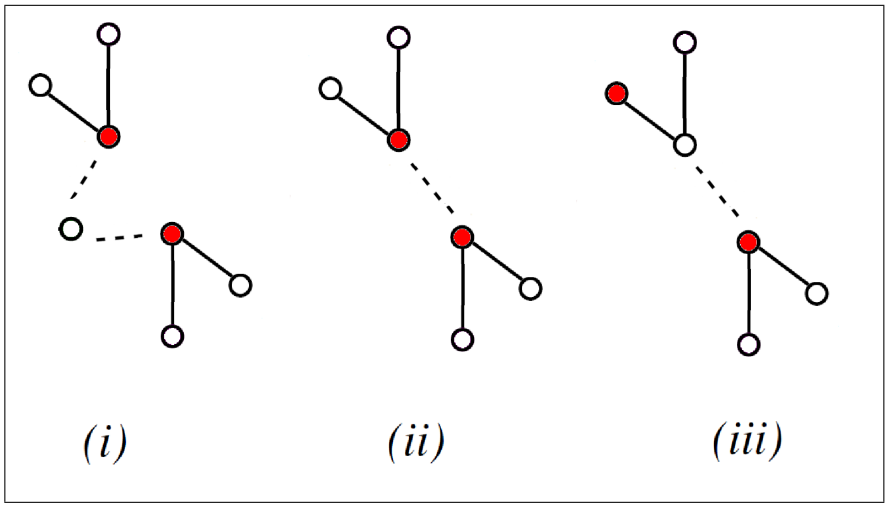


Figure 4.7 Incrementally Deployable Router Labeling

Merge Coloring: While the addition of any new router to the traceback overlay network can be easily achieved by forcing a global re-coloring, it may be practically infeasible in the long term due to the potential overhead in coloring a traceback overlay steadily increasing in size and/or the expectations of a rapid rate of traceback adoption on the Internet. Hence, we propose a novel *merge coloring* concept that lets individual nodes resolve all coloring conflicts in its local vicinity itself by mutual collaboration with other neighboring entities without any change to the global coloring configuration. In this regard, we now identify the two main scenarios that can induce a 1-hop or 2-hop star coloring conflict - adding an uncolored node to the traceback overlay, and merging two different traceback overlay networks (Fig. 4.7). In all these cases, the resulting conflicts can be locally resolved either by appropriate graph resizing and/or localized graph re-coloring of a few nodes, transparent to other nodes in the network.

We have thus extended the centralized graph coloring technique to operate in a truly distributed environment by providing greater reliability, scalability and support for incremental deployment - thereby making it practically viable for an Internet-wide deployment.

4.3 Experimental Evaluation

For the experimental evaluation of the proposed distributed graph coloring approach, we have analyzed various Internet topology maps for a more realistic characterization - an Internet AS-level topology inferred by the Skitter tool developed by CAIDA [132], and an Internet router-level topology inferred by the Internet Mapping Project from Lumeta [133], both studied in great detail in [130]. In this regard, we view the AS-level and router-level topologies as representative of a partially deployed and a fully deployed Internet-wide traceback solution respectively. To realistically emulate Internet-scale graph coloring in a distributed environment, we make use of Google's Pregel framework [134] designed for large-scale graph processing to implement our proposed distributed graph coloring algorithm for thousands of graph nodes across tens of hundreds of computing devices.

We use three different metrics here to evaluate the performance of the proposed distributed graph coloring approach, namely the packet fingerprint size, the number of colors used by the random labeling process, and finally the convergence speed of the distributed algorithm.

4.3.1 Fingerprint Size

We now analyze the effect of the node width and path length optimizations on the overall fingerprint size used for packet marking.

Node Width: As explained previously, the packet fingerprint size is directly dependent on the individual router label sizes (node width), and thus governed by the total number of unique colors in the neighborhood of any router. As it is bound by the router's degree on the lower side and its star degree on the higher side due to star coloring constraints, we now focus on their individual distributions in the Internet graph. While Fig. 4.8(a) and Fig. 4.8(b) represent the 1-hop node degree distribution, Fig. 4.9(a) and Fig. 4.9(b) represent the 2-hop node degree distribution, for the AS-level and router-level Internet topologies respectively. In each of these

graphs, the (log-scale) X-axis represents the node degree (star degree), while the (log-scale) Y-axis represents the corresponding number of nodes for the different node degree (star degree) values respectively. Finally, we plot these distributions for four different topologies here, namely those induced by the proposed simple star coloring, expansive coloring, contractive coloring, and hybrid coloring techniques for graph resizing.

Along expected lines, we note that the AS-level topology has far fewer nodes than the router-level topology, and that the degree (star degree) distribution for both the topologies follows a power-law distribution [130]. As the node width is governed by the maximum star degree and not the total network size, simple star coloring thus provides a shorter router label representation than other traceback techniques known in literature. Additionally, expansive coloring reduces the impact of the long tail of high degree (star degree) nodes by proportionately increasing the number of lower degree (star degree) network nodes. Similarly, contractive coloring reduces the impact of the large number of low degree (star degree) nodes by proportionately increasing the number of higher degree (star degree) network nodes. Finally, hybrid coloring oscillates between these two curves in the graph, thereby achieving node degree (star degree) homogenization in the network. Hence hybrid coloring not only improves node width utilization, but also limits the growth of the packet fingerprint.

Path Length: As explained previously, the packet fingerprint size is directly dependent on the number of traceback capable routers along the routing path from the packet source to its destination (path length). Fig. 4.10(a) and Fig. 4.10(b) represent the hop-length distribution for the AS-level and router-level Internet topologies respectively. In each of these graphs, the (log-scale) X-axis represents the path length, while the (log-scale) Y-axis represents the corresponding number of nodes for the different path length values respectively.

As measured by other researchers in the past, we again notice that the AS-level topology has a much smaller hop-length than the router-level topology [130]. We also notice that while

expansive coloring leads to an increased path length, contractive coloring results in a reduced path length, appropriately changing the packet fingerprint size in both cases. Finally, hybrid coloring bounds the effective path length between these two curves in the graph, thereby achieving the best of both worlds by increased router degree homogeneity.

4.3.2 Coloring Efficiency

In order to study the efficiency of the proposed graph coloring approach, we now define a new metric called "color spread" that measures the distribution of the various colors across the different nodes in the network. We also define a "color palette" as the number of unique colors required to optimally color the given graph rounded up to the closest binary representation - hence the color palette has 5 colors for the network topology in Fig. 4.1. We now analyze three different graph coloring variants, namely a centralized greedy coloring technique (Fig. 4.11), a distributed coloring technique assigned one color palette (Fig. 4.12), and a distributed coloring technique assigned four distinct color palettes (Fig. 4.13), for both the AS-level and router-level topologies. In each of these graphs, the (log-scale) X-axis represents the unique color indices assigned to the different routers, while the (log-scale) Y-axis represents the corresponding number of nodes for the different color indices respectively.

We notice that the centralized coloring approach achieves maximal reuse of lower color indices due to the greedy nature of the proposed algorithm, while the distributed coloring techniques display a more uniform distribution across the different color indices due to random color assignment during the speculative coloring phase. We also note that larger the color palette size, the fewer the number of network nodes sharing the same color index - increasing the size of the color palette by a factor of two leads to a resulting increase in the router label size by just one bit for each router, and is thereby an ideal mechanism to pre-emptively support a much larger network for future scalability concerns. In this regard, it is particularly interesting to note the effect of graph coloring on the different topologies induced due to graph resizing.

As the total number of colors is governed by the maximum star degree in the network, the expansive coloring topology requires far fewer colors than any of the other topologies. Also, as the color index reuse is governed by the minimum star degree in the network, the contractive coloring topology achieves the least color reuse than any of the other topologies. Finally, the hybrid coloring topology achieves the best of both worlds by using fewer colors and also allowing greater color reuse in the network.

Thus the color spread metric illustrates the positive effect that the proposed graph coloring approach has on the overall node width and path length metrics in the network.

4.3.3 Convergence Speed

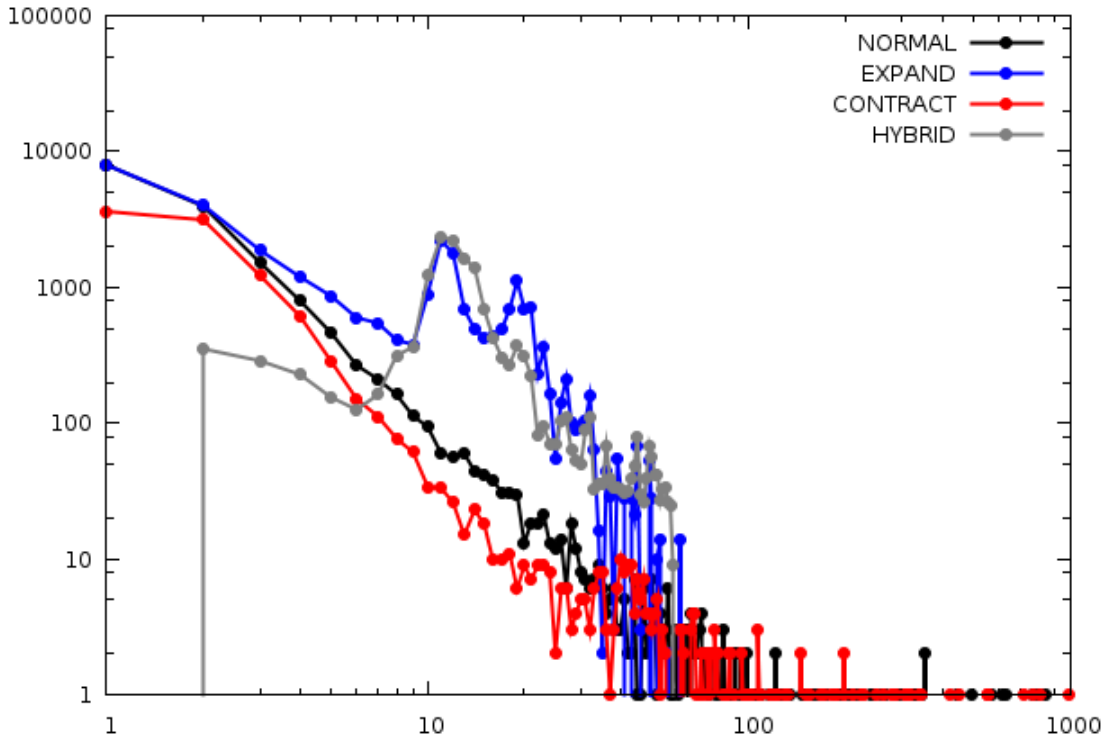
In order to study the speed of the proposed distributed graph coloring approach, we now analyze the total time required to fully label the network topology without any star coloring conflicts. In this regard, precisely calculating the actual time required is rather difficult due to myriad factors including processing speed, transmission bandwidth, number of coloring conflicts, etc. and hence we limit ourselves to measuring the total number of idempotent iterations required to obtain a fixed coloring configuration for the entire network. In this context, the very nature of speculative coloring by random selection and subsequent re-coloring due to newly discovered star coloring conflicts, essentially means that the proposed distributed coloring technique would require a sufficiently large number of iterations before it eventually converges, and is hence much slower than the centralized approach which can potentially color the entire network topology in one iteration on a single processor locally.

Fig. 4.14, Fig. 4.15 and Fig. 4.16 present the node distribution across the total number of iterations needed before any of their individual color assignments could be finalized, for the distributed coloring techniques having one, two and four color palettes respectively. In each of these graphs, the (log-scale) X-axis represents the different coloring iterations, while the

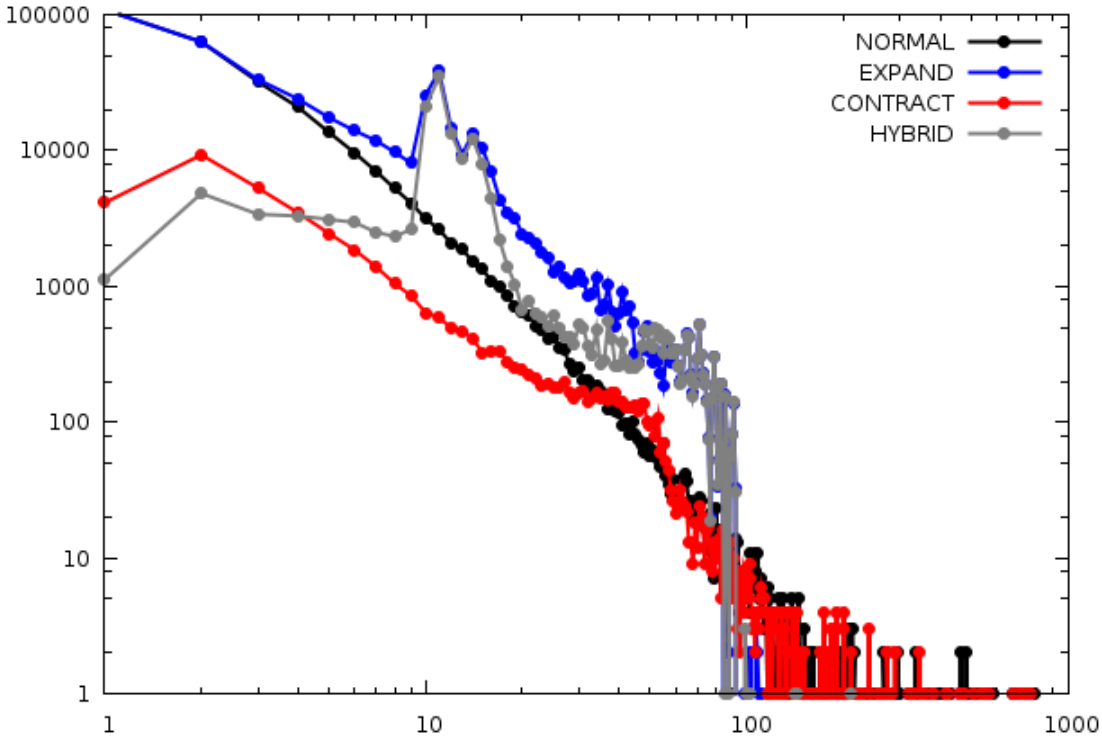
(log-scale) Y-axis represents the corresponding number of nodes successfully colored at each of those iterations respectively. Note that while the use of randomized color selection during the speculative coloring phase does alter the various data points in these graphs across multiple executions, the overall trends are essentially invariant and hence serve as fairly accurate indicators of the overall convergence speed for distributed graph coloring.

Along similar lines to the discussion on color spread, we notice the large number of iterations needed for expansive coloring due to the dense inter-connectivity of mesh-connected logical nodes that it induces. Similarly, we can explain the significantly smaller number of iterations needed for the contractive coloring topology based on the very nature of contractive coloring and the presence of fewer number of nodes in the entire network. Finally, hybrid coloring again oscillates within a small region flanked by these two curves in the graph, thereby providing heuristic bounds for the overall convergence speed of the proposed distributed graph coloring technique. A far more interesting observation here is the significant drop in the total number of iterations required as we increase the size of the color palette - while scaling the color palette size by a factor of two essentially increases the effective node width by a single bit for all routers in the network, the corresponding decrease in the convergence speed is far more significant. Thus it is probably best to run the distributed graph coloring algorithm with sufficient slack in the number of colors than needed for optimal coloring, to not only enable faster convergence of network-wide router label assignment, but also provide greater scope for future growth of the Internet topology.

Hence, the different performance metrics discussed above all reinforce the fact that the proposed hybrid coloring approach is probably the best suited for Internet-wide deployment. We also conclude that while naive star coloring can provide efficient packet fingerprints as compared to other known traceback techniques, the novel hybrid coloring approach can yield even greater improvements by restricting the effects of a few outliers in the network topology.

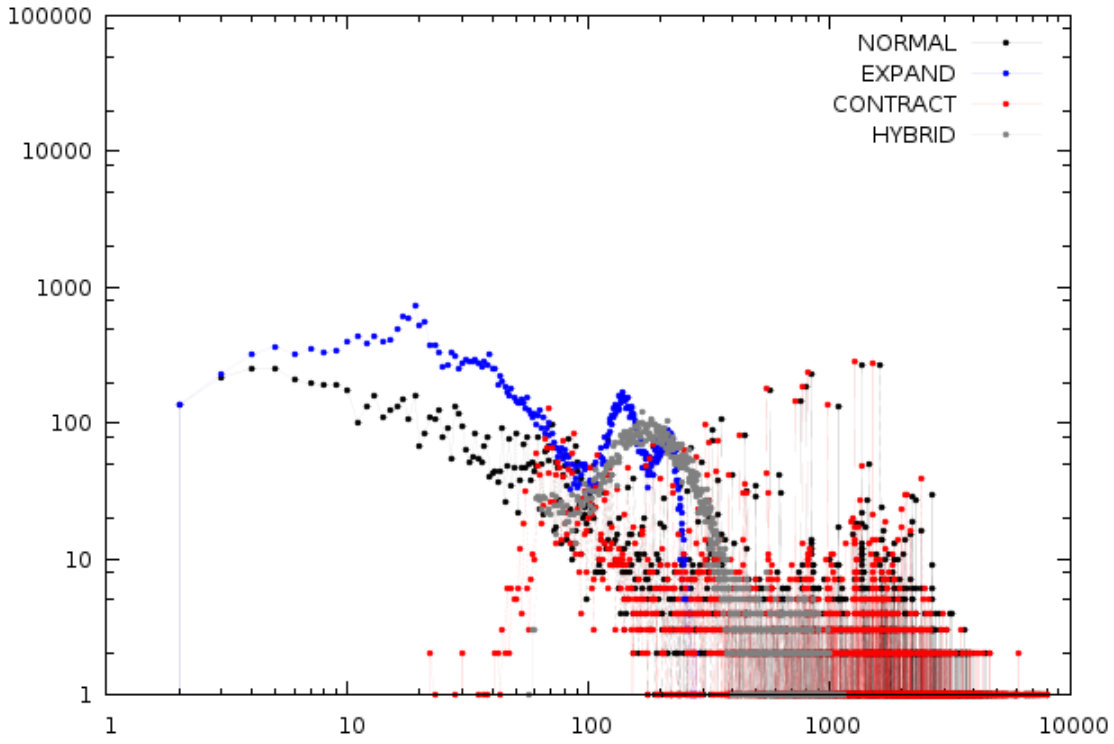


(a) AS Topology

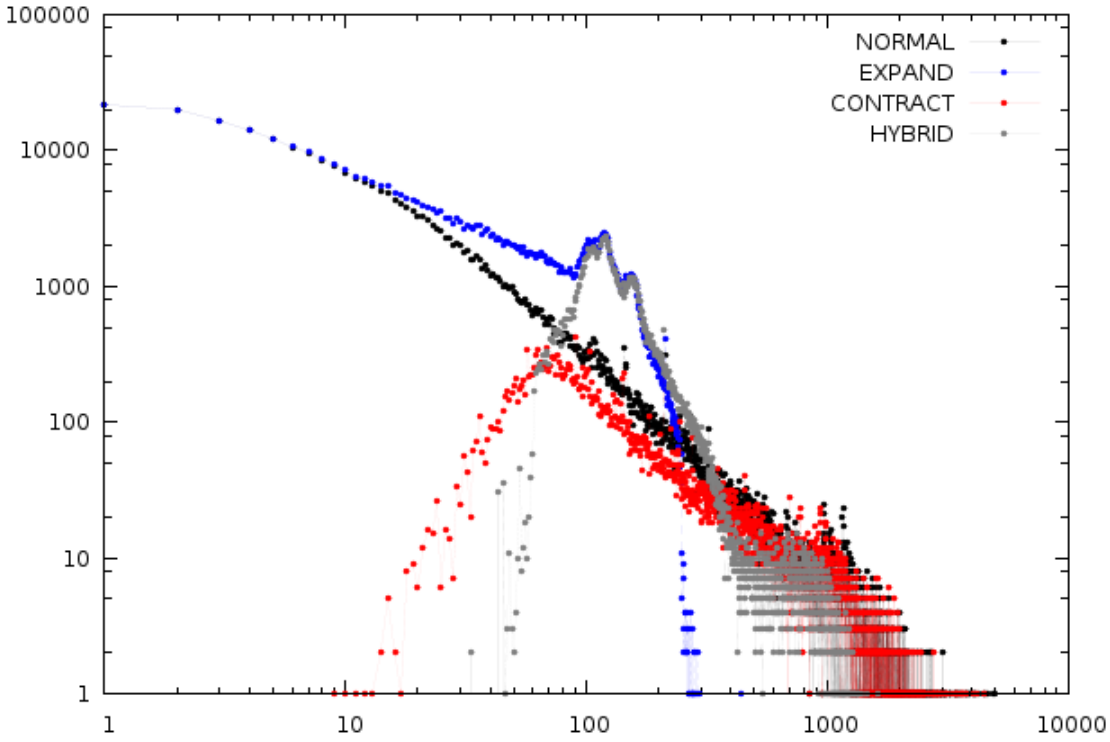


(b) Router Topology

Figure 4.8 Degree Distribution (#neighbors vs #nodes)

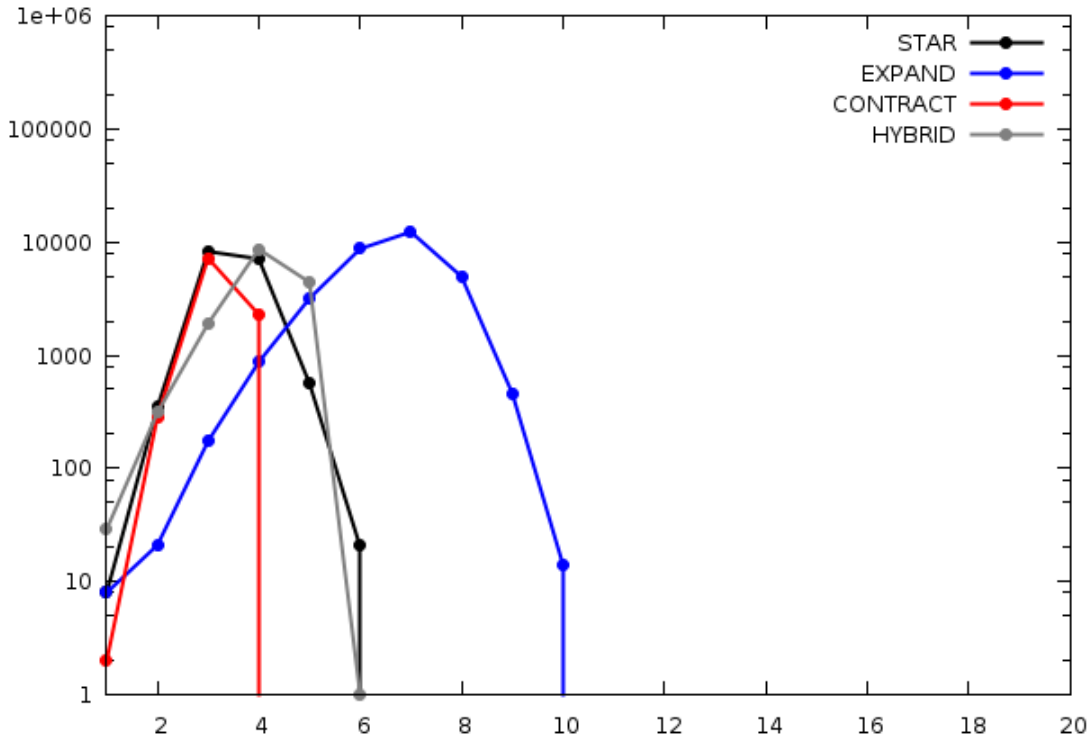


(a) AS Topology

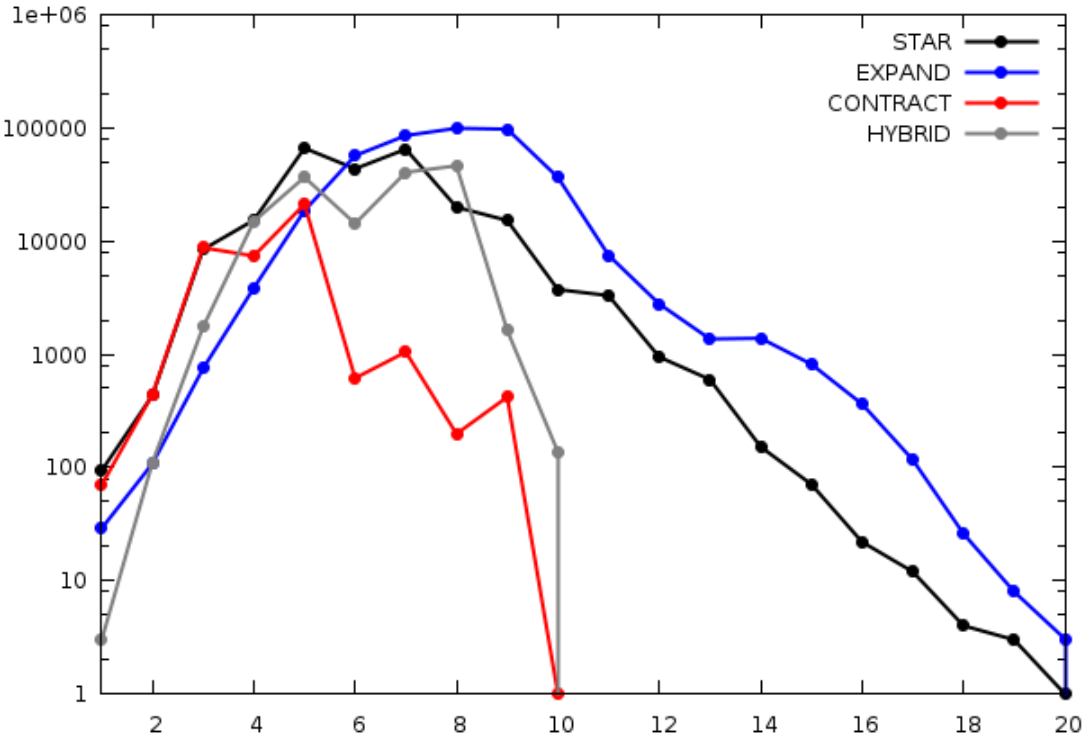


(b) Router Topology

Figure 4.9 Star Degree Distribution (#star-neighbors vs #nodes)

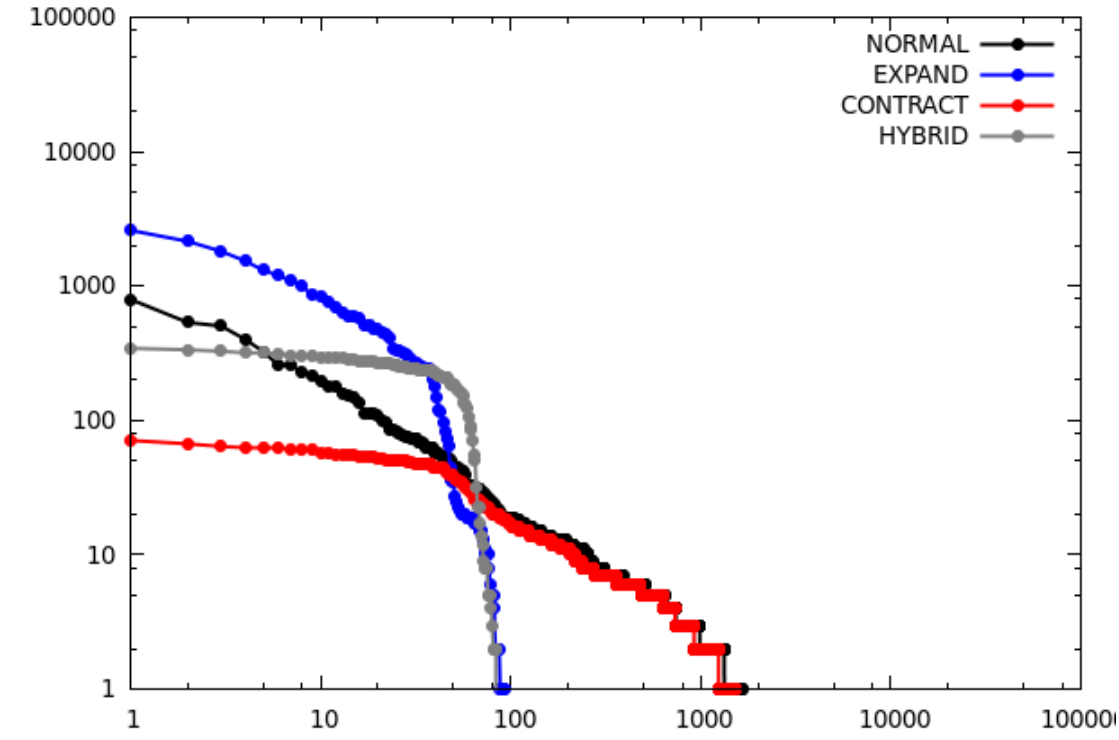


(a) AS Topology

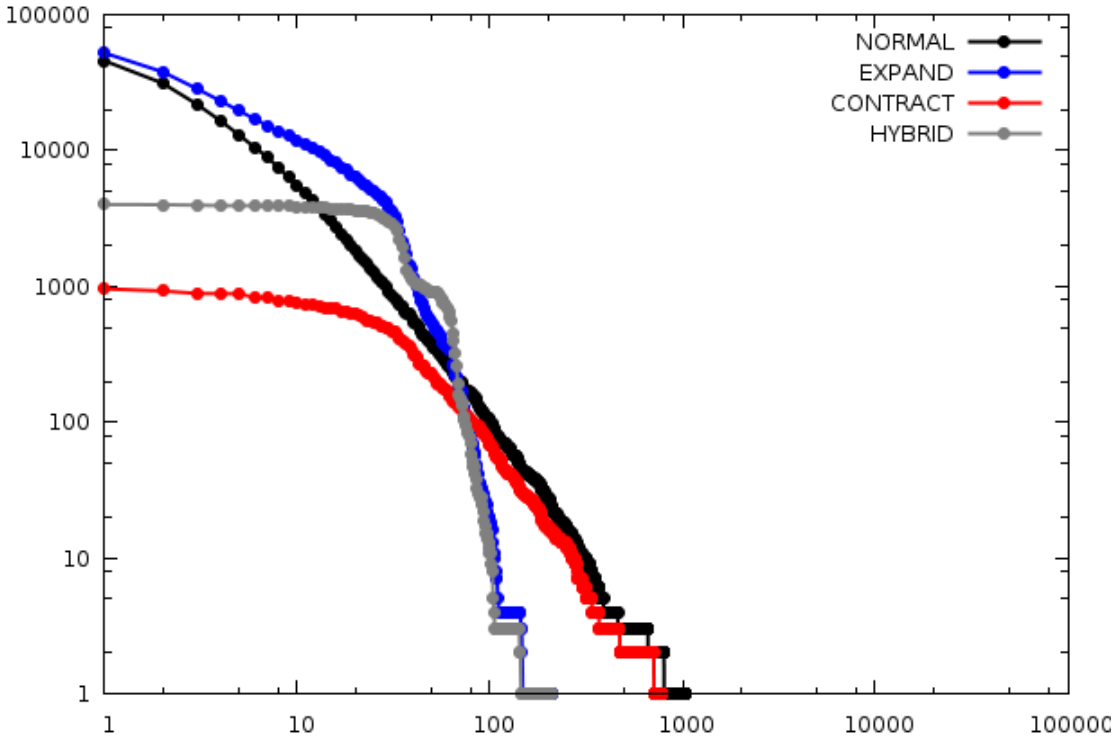


(b) Router Topology

Figure 4.10 Path Length Distribution (#hops vs #nodes)

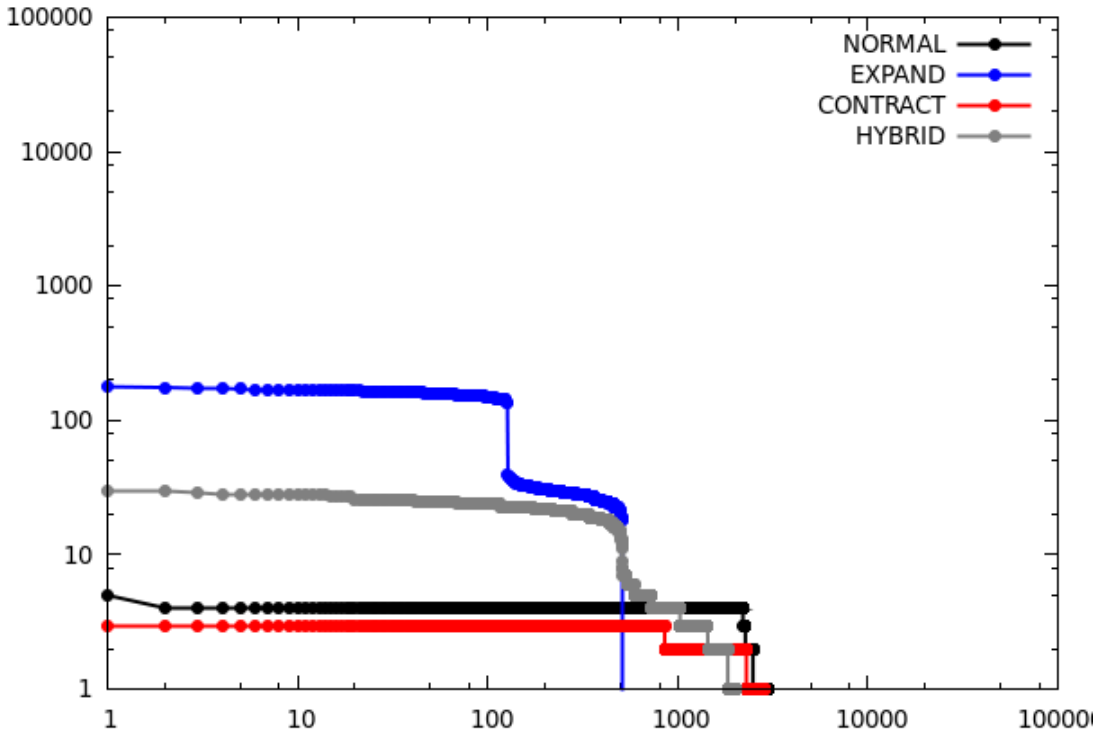


(a) AS Topology

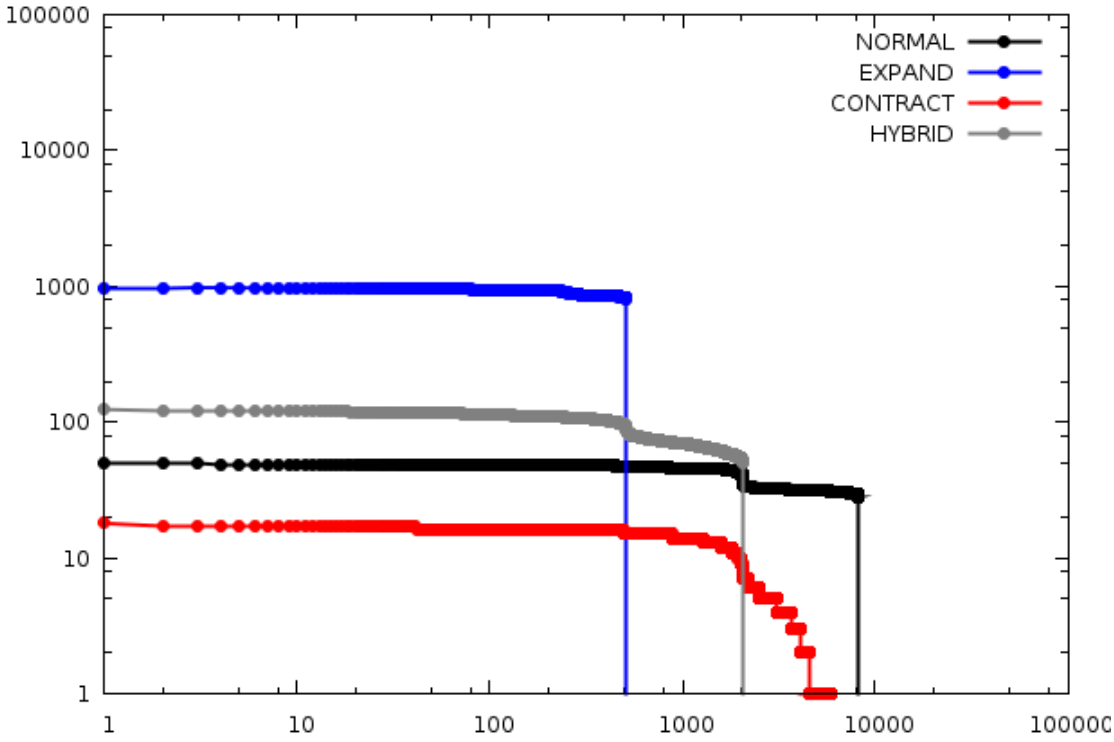


(b) Router Topology

Figure 4.11 Coloring Efficiency (Centralized) (#colors vs #nodes)

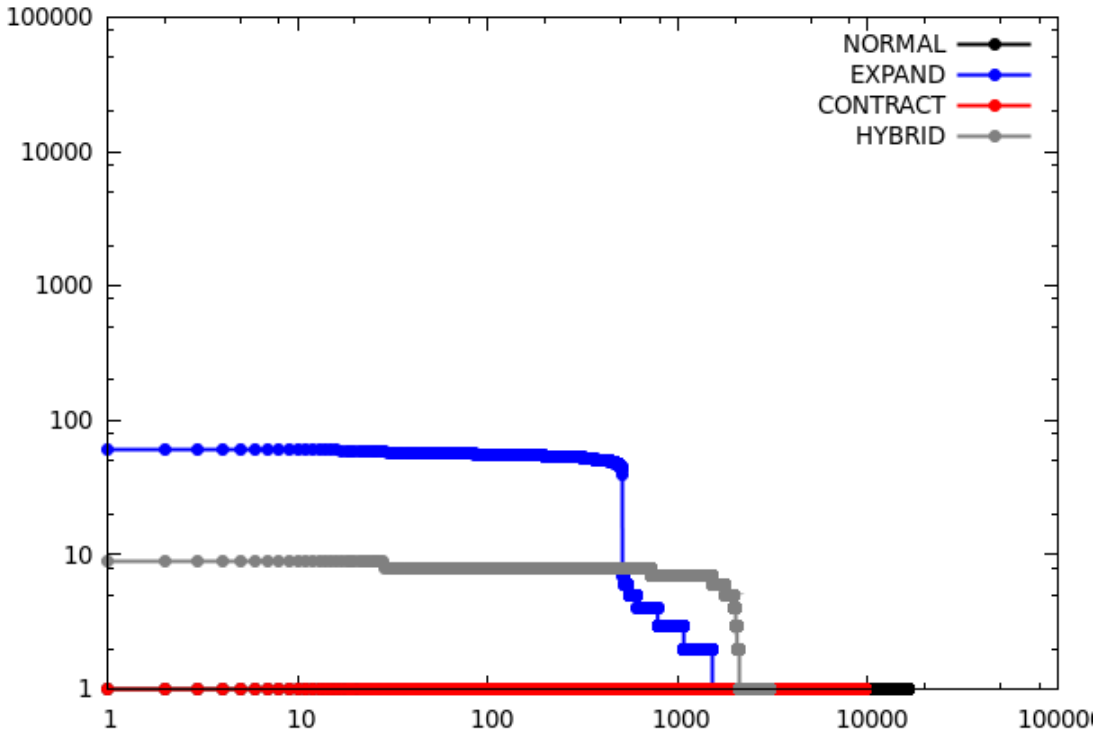


(a) AS Topology

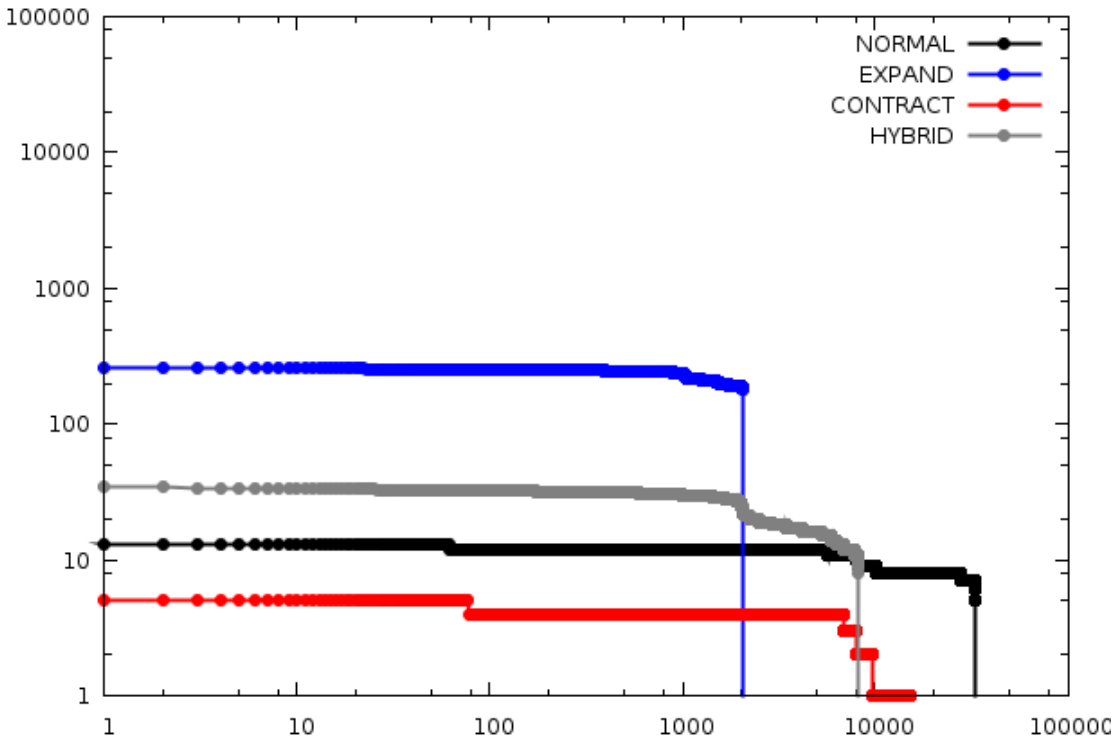


(b) Router Topology

Figure 4.12 Coloring Efficiency (One Palette) (#colors vs #nodes)

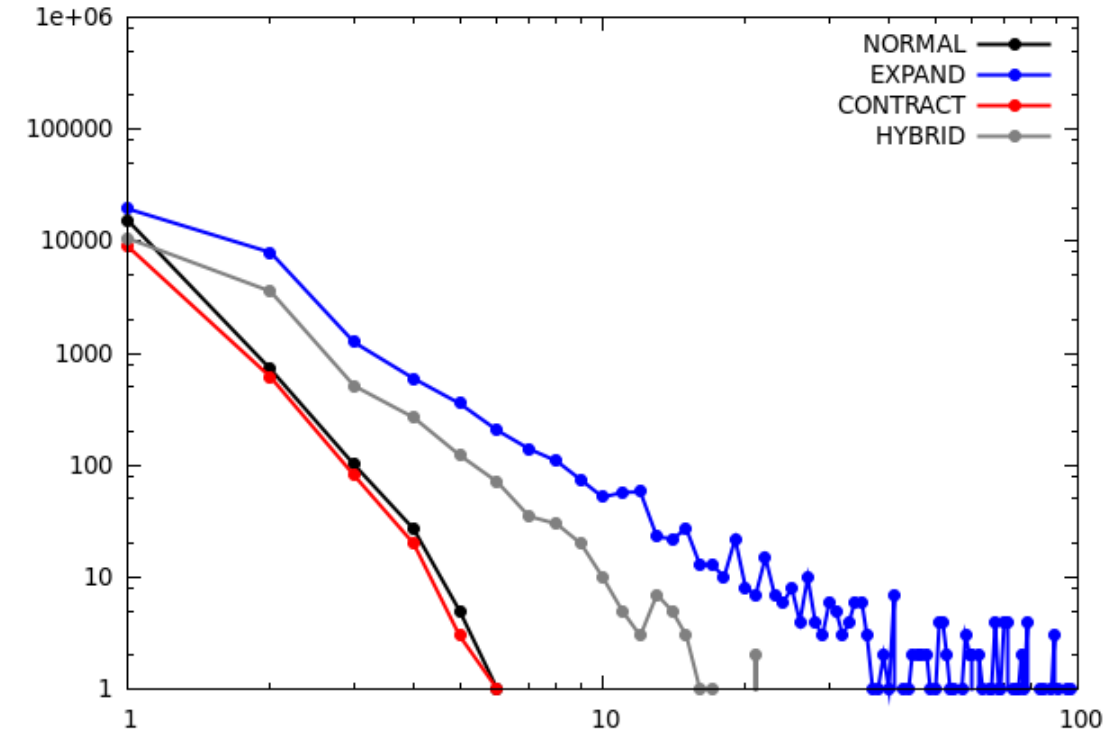


(a) AS Topology

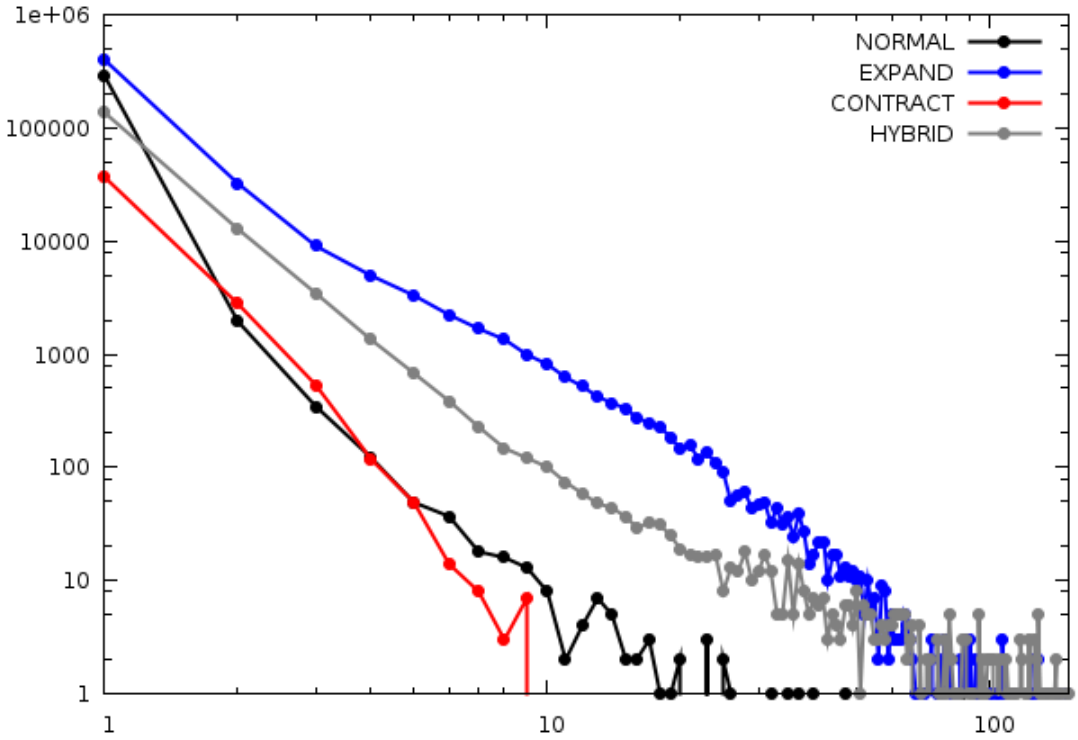


(b) Router Topology

Figure 4.13 Coloring Efficiency (Four Palettes) (#colors vs #nodes)

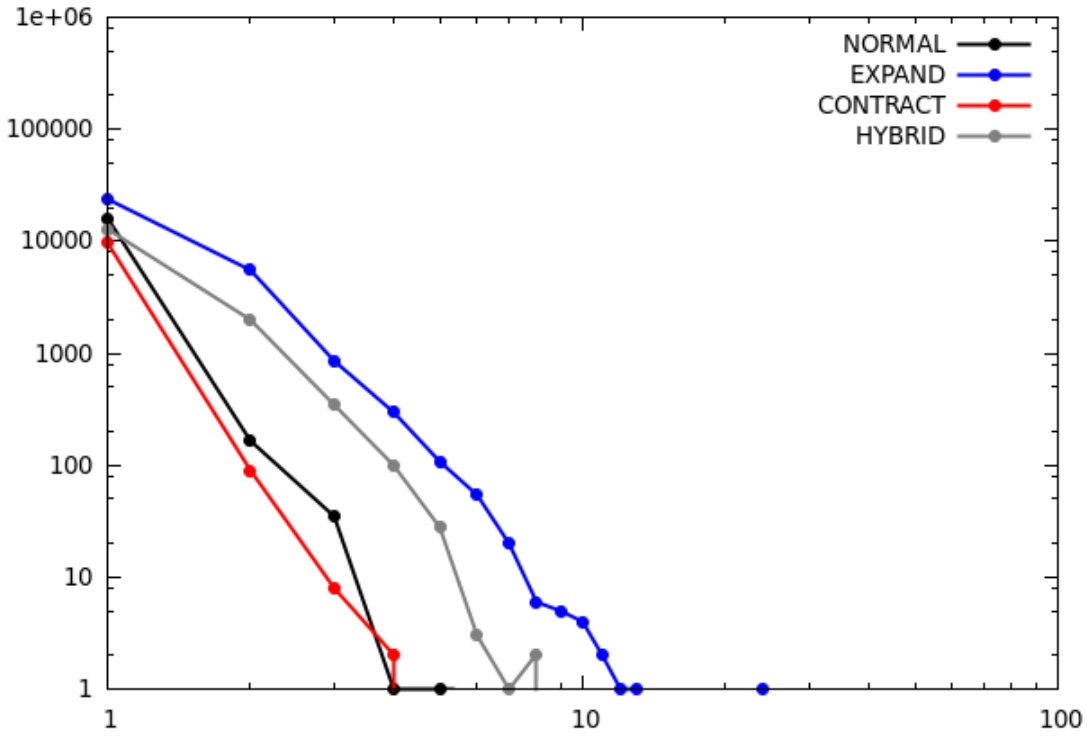


(a) AS Topology

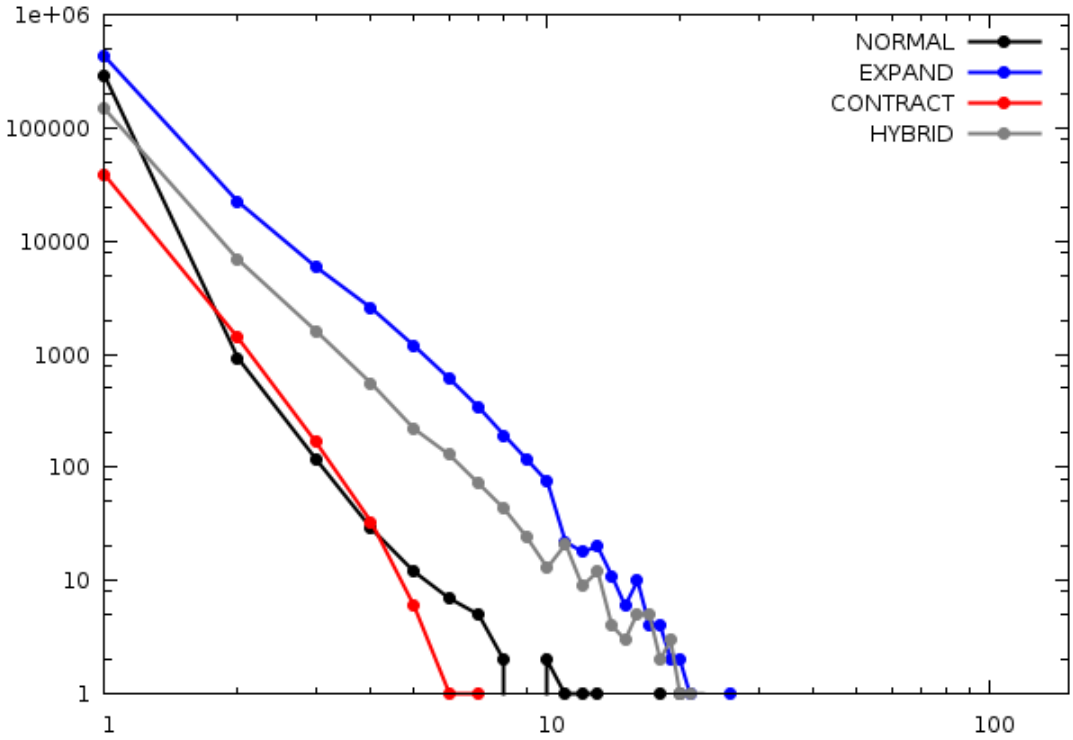


(b) Router Topology

Figure 4.14 Convergence Speed (One Palette) (#iterations vs #nodes)

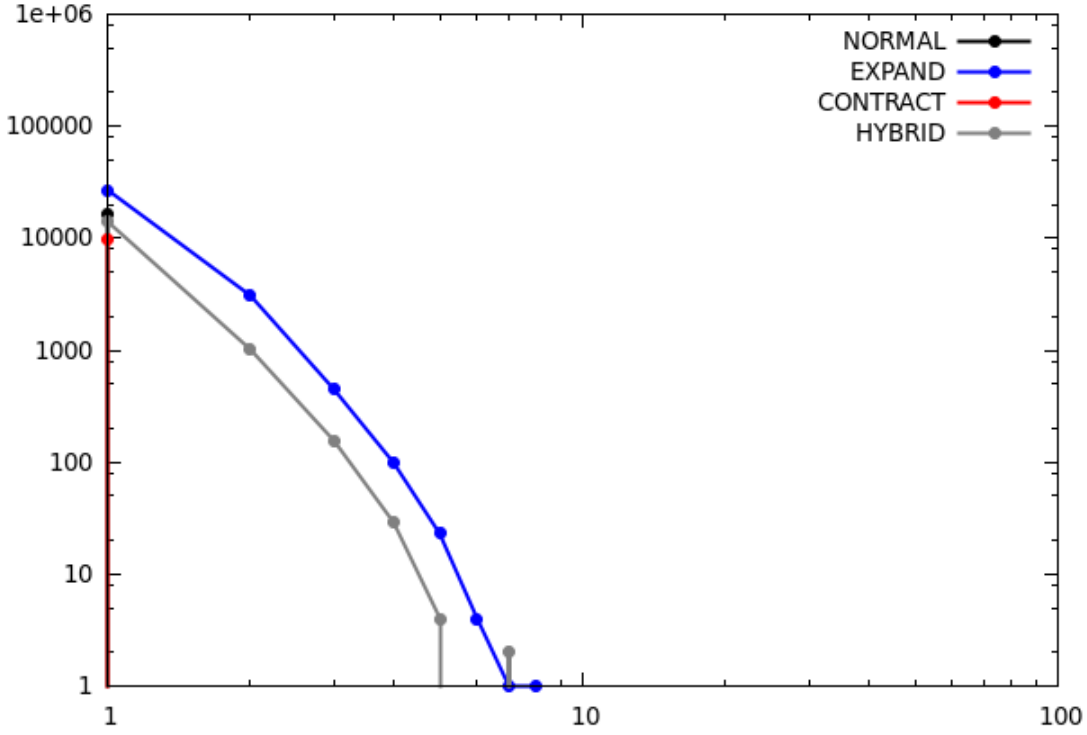


(a) AS Topology

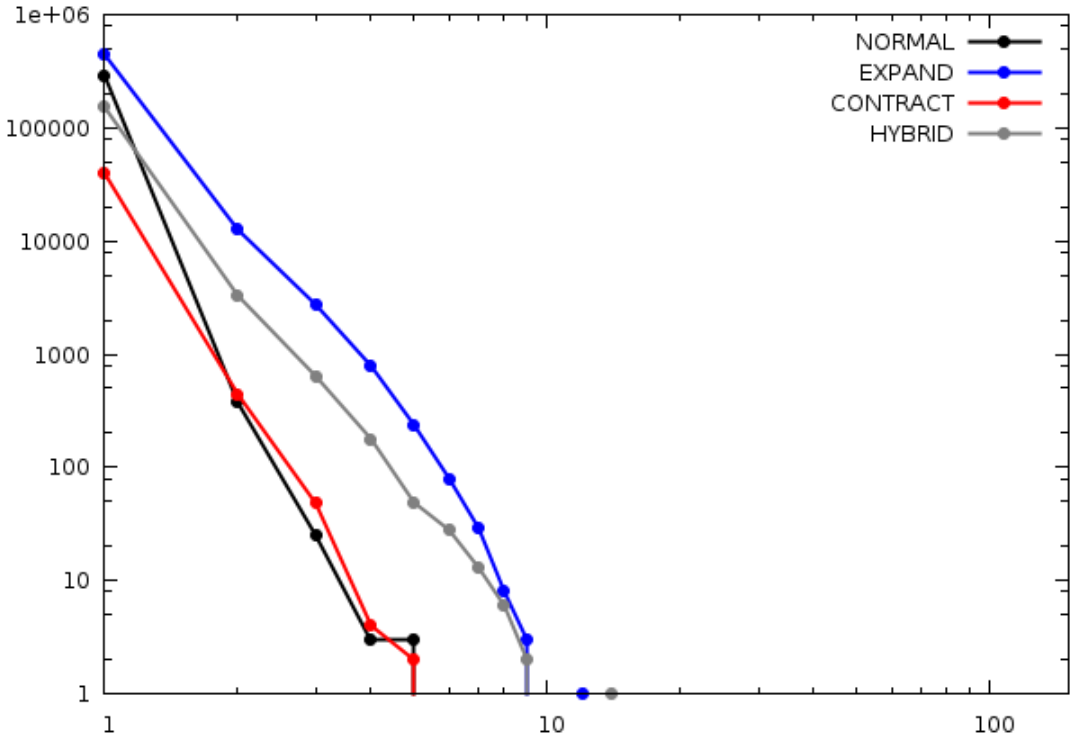


(b) Router Topology

Figure 4.15 Convergence Speed (Two Palettes) (#iterations vs #nodes)



(a) AS Topology



(b) Router Topology

Figure 4.16 Convergence Speed (Four Palettes) (#iterations vs #nodes)

CHAPTER 5. PATH LABELING: A Divide-and-Conquer Approach

As discussed in great detail thus far, the probabilistic packet marking traceback techniques tag each packet with a router's identity while it is being forwarded. The entire fingerprint aggregated across multiple packets to a single destination is then used for various computations, namely building the attack tree in the network, detecting attack traffic volumes from multiple sources, and finally correlating the many attack packets with the different attack paths identified for appropriate filtering in the network. Thus there is a clear lack of *separation of duties* with respect to each of these individual actions, and the same in-band packet marking process is expected to satisfy these myriad requirements, often in a sub-optimal manner. We now seek to improve the state of the art by treating each of these requirements as mutually independent tasks and thereby try to solve them in an optimal manner - we employ novel distributed divide-and-conquer techniques to not only provide greater security and performance guarantees, but also support a practically viable Internet-wide deployment.

5.1 Proposed Concepts

We now identify three critical operations underlying all DDoS defense mechanisms, namely attack tree construction, attack path frequency detection, and packet to path association.

Attack Tree Construction: It is defined as the process of obtaining an abstraction of the router-level Internet graph, called the attack tree, where the attack victim is the tree root, and the different traffic sources (or their egress routers) are the many tree leaves.

Attack Path Frequency Detection: It is defined as the process of obtaining a weighted attack tree, where each edge is annotated by the number of packets that traversed that network link, in the specified time period. This metric often serves as a differentiator while classifying traffic sources as attack/legitimate for subsequent filtering.

Packet to Path Association: It is defined as the process of associating any data packet to a particular traffic source (and hence a routing path) in the attack tree.

Attack tree construction has traditionally been achieved by requiring the different intermediate routers to probabilistically/deterministically send in-band/out-of-band edge information, and then using a path reconstruction algorithm at the victim that collates all these different fragments (packet marks) to reconstruct the original attack tree. The use of multiple packets by the individual routers and the potential namespace collision across multiple routers, makes the path fragments unreliable, thereby leading to inefficient, possibly incorrect, attack tree construction. While attack path frequency detection has traditionally been inferred indirectly by counting the number of path fragments from each intermediate router, the probabilistic nature of data collection and the unreliability of the underlying attack tree constructed, often lead to high false positives/negatives in attack source attribution. Also, the probabilistic nature of the packet marking process makes it extremely slow for any immediate use in traffic filtering. Finally, packet to path association is also error-prone as fingerprint fragmentation across multiple packets requires multiple buffers at the victim to accumulate the path information, and the lack of efficient inter-packet fragment correlation often leads to incorrect ordering of the different router fingerprints, and thereby incorrect path identification.

We believe that the lack of clear separation of duties in the traditional designs for DDoS defense, with often the same feature satisfying the many requirements simultaneously, either implicitly or explicitly, has led to their sub-optimal performance. Our major contribution here has been to show that addressing these issues as three disjoint problems, not only reduces

the complexity of their individual implementations, but also achieves higher efficiency across a wide variety of metrics. In this regard, we propose to use single packet in-band path identifiers for unique packet to path association, while handling attack tree construction and attack path frequency detection as independent out-of-band processes. Our proposed use of a novel *distributed divide-and-conquer* approach, casting the individual actions as simple *recurrence relations* makes it easily extensible for implementing future distributed network defenses and early warning systems, in an incremental fashion.

We now argue that the proposed design additionally enables the use of significantly smaller packet fingerprints - the significance of this has been discussed at length in Chapter 4. Consider a sample attack tree $T(n, l, d, p)$, where n, l, d, p represent the total number of routers (tree nodes), the average hop length (tree depth), the average router degree (tree node degree), and the average number of routing paths (tree leaves) respectively. An approximate mapping of the modeled tree to a *balanced k-ary tree* [135] is shown in Eqn. 5.1 and Eqn. 5.2, while assuming Internet-measured values of $d \approx 4^{+}$, $l \approx 16^{+}$ for analysis [96] [136].

$$p = d^{l} \tag{5.1}$$

$$n = \frac{d * (d^{l} - 1)}{d - 1} = \frac{\sqrt[l]{p} * (p - 1)}{\sqrt[l]{p} - 1} \tag{5.2}$$

$$F_{\ global} = l * \lceil log_2(n) \rceil \approx 512\ bits \tag{5.3}$$

$$F_{\ local} = l * \lceil log_2(d) \rceil \approx 48\ bits \tag{5.4}$$

$$F_{\ proposed} = \lceil log_2(p) \rceil \approx 32\ bits \tag{5.5}$$

As Internet end-hosts far out-number the routers in use today, the use of globally unique

IP addresses (or their hash fragments) as router identifiers leads to a large namespace scatter (Eqn. 5.3). The recent use of multiple local small-worlds employing lazy path discovery, that requires the unique mapping from a router to its label only within some closed domain [96], reduces the overall packet fingerprint size significantly (Eqn. 5.4) - note that the proposed graph coloring approach in Chapter 4 falls under this category. We now propose to reduce the scope of the problem even further by assigning unique path identifiers to only the traffic sources or tree leaves (Eqn. 5.5), thereby achieving a much higher degree of compactness. Note that the equations above represent the per-packet router identifier size (F), and not the entire set of fingerprints that represent a routing path in the attack tree.

We now discuss these ideas in greater detail, justifying our specific design choices and their resulting impact on the performance of any DDoS defense mechanism that employs them.

5.1.1 Attack Tree Construction

The attack tree rooted at the victim is essentially an abstraction of the Internet router-level graph based on the different packet flows to the victim, and hence remains static over moderately short intervals of time. For the victim to obtain a comprehensive view of the various attack sources/paths, the path insertions in the attack tree need to be immediate, while path removals can possibly be lazy. Stated differently, it suffices to refresh the attack tree either infrequently or in an interrupt-driven fashion, where triggering happens only when some critical structural modification occurs. Thus, overloading in-band packet marking to handle attack tree construction, would lead to unnecessary repetitive transmissions on a per-packet basis. Hence we propose to use out-of-band packet marking as the preferred means of data transmission for the attack tree. We also avoid the use of independent communication channels between the victim and the intermediate routers as in [71], as it is grossly inefficient not only due to redundant connections and repetitive transmissions, but also due to the inability of the intermediate routers to infer their attack sub-trees without expensive computations.

Recursive Approach: We propose to use a *distributed divide-and-conquer* approach by recursively breaking down the attack tree construction problem at each router into multiple sub-problems, each in turn handled by that router's neighbors (attack tree children) respectively. The solutions to the sub-problems are then combined and propagated up the attack tree from the different traffic sources to the victim. Thus we adopt a *bottom-up* approach rather than the traditional *top-down* approach controlled by the victim. Each router essentially aggregates the attack sub-trees (T_{R_i}) of its direct neighbors (tree children), and forwards it to its immediate upstream neighbor (tree parent). This when implemented by every router in the attack tree, leads to an incremental attack tree evolution in a bottom-up yet distributed fashion.

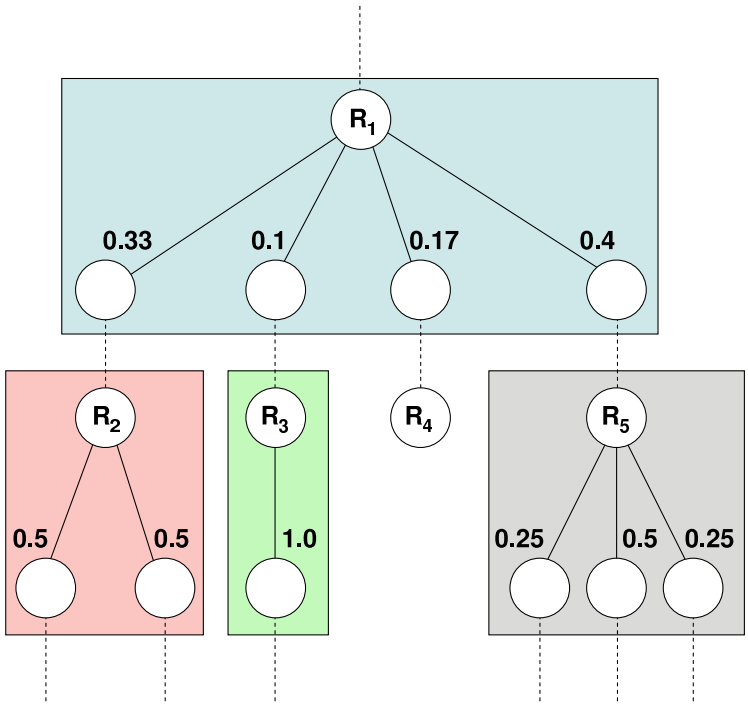


Figure 5.1 Modular Path Tree

Logical Representation: Consider an abstraction of the attack tree as shown in Fig. 5.7, showing the attack sub-tree of some router R_1, having 4 different tree children, namely R_2,

R_3, R_4, and R_5. The logical representation of this sub-tree is then given by Eqn. 5.6, where D_{R_i} and T_{R_i} represent the degree and attack sub-tree of router R_i respectively. Eqn. 5.7 then generalizes this expression for every router in the attack tree, thus representing the proposed distributed divide-and-conquer approach as a succinct *recurrence relation*, where Δ_{R_i} represents the immediate children of R_i.

$$T_{R_1} = D_{R_1} \cup T_{R_2} \cup T_{R_3} \cup T_{R_4} \cup T_{R_5} \tag{5.6}$$

$$T_{R_i} = D_{R_i} \bigcup_{R_j \in \Delta_{R_i}} T_{R_j} \tag{5.7}$$

$$T_{R_1} = \underline{4} \cup \underline{2 \circ \circ} \cup \underline{1 \circ} \cup \underline{0} \cup \underline{3 \circ \circ \circ} \tag{5.8}$$

Thus the proposed attack tree construction employing distributed divide-and-conquer techniques, ensures that each router not only performs minimal computation with no redundant messaging, but also obtains a global view of its entire attack sub-tree.

Physical Representation: The attack sub-tree of router R_1 is now expressed as in Eqn. 5.8, where $\circ$ represents further recursive expansion not shown due to abstraction. Interestingly, if we tag every node in the attack tree with its degree, then Eqn. 5.8 also represents the *pre-order traversal* (prefix notation) of the attack tree. In [137], the authors discuss the standard technique used to reconstruct the original k-ary tree from its (prefix) *Polish Notation*, if the arity of all the intermediate nodes are known. As we tag the attack sub-tree representation of each router with its degree information, the victim (or any intermediate router) can uniquely reconstruct the attack tree from its pre-order traversal with relative ease.

The power of this recurrence relation lies in its modularity. Any structural modification to the attack tree thus supports a simple *plug-and-play* design that can propagate up the tree to the victim, without needing a complete re-computation of the entire attack tree or affecting

other independent attack sub-trees, as shown by different shaded regions in Fig. 5.7. Thus we can closely model the dynamic Internet routing characteristics, by periodic or triggered update messages containing only the attack sub-trees that have been structurally modified. It is to be noted in this context, that no other scheme in literature provides such robust and explicit support for dynamic changes to the attack tree, without a complete re-transmission.

$$TreeSize_{\ max} = n * \lceil log_2(d) \rceil \tag{5.9}$$

$$T_{R_i} = X_{R_i}\left(D_{R_i} \bigcup_{R_j \in \Delta_{R_i}} T_{R_j}\right) \tag{5.10}$$

Tree Pruning: The maximum size of the attack tree rooted at the victim is represented by Eqn. 5.9, and for high traffic web portals such as Google, Yahoo, etc., the attack tree size can potentially reach unmanageable levels. Hence, we propose a *tree pruning* technique (X_{R_i}) to dynamically reduce the attack tree size to more manageable levels. As the tree size grows linearly with n and hence with p, we choose to limit the number of actively tracked traffic sources in the attack tree to bound its size to more practical limits. Eqn. 5.10 thus represents the new form of the *recurrence relation* with pruning possibly supported at multiple intermediate routers in the network. The distributed nature of the pruning mechanism thereby enables individual service/network providers to use independent strategies to ensure local optimality in tree pruning, a few design parameters incorporating the relative resource utilization, feedback from the victim, feedback from distributed monitoring systems, round-robin dropping of randomly chosen paths, white/black listing using local signature databases, etc.

The deployment of tree pruning techniques is completely optional and does not adversely affect the proposed attack tree construction technique. However harnessing it provides significant gains, as the maximum tree size for monitoring one million packet (attack traffic) sources simultaneously is only $\approx$ 507KB - derived by using a few realistic numbers in the modeled attack tree $T(n, l, d, p)$, $p = 1,000,000$ and $l \approx 16^{+} \Rightarrow d \approx 2.4$.

5.1.2 Attack Path Frequency Detection

During a highly disruptive distributed DoS attack, it is probably easy to weed out persistent high volume attackers. However, DDoS attacks employing large botnets with a low median traffic volume per source, often make it difficult to classify packet sources as legitimate or those with malicious intent. A traffic hot-spot due to a sudden spike in Internet activity geared towards a particular destination (say, *Slashdot effect*), also leads to an unintended denial-of-service due to unprovisioned capacity problems, both at the end-host and in the network. Thus the problem of DDoS defense reduces to that of prioritizing certain traffic flows/sources over others, the granularity of such demarcation being determined by the destination under heavy traffic load. Any such traffic classification would require both packet-level metrics (content signature, protocols/flags in use, etc.), and flow/path-level metrics (frequency, flow pulse characteristics, source subnet, etc.). As packet-level metrics can easily be gleaned off the packets themselves, we focus on the flow/path-level metrics here.

The attack tree we have obtained thus far using out-of-band packet marking, is essentially an *attack path tree*, embedding only the router connectivity information. We propose to overload this attack path tree to also embed path frequency information, to construct a novel *attack path frequency tree*. Along similar lines of the proposed distributed divide-and-conquer tree construction mechanism, we now *encode* path frequency distributions as simple tree data structures, and then *embed* them into the original attack tree, to yield a novel bottom-up attack path frequency tree construction mechanism. Although any frequency encoding and embedding techniques can be used, we limit our scope here to Huffman encoding [138] and the "balanced parentheses" embedding [139] techniques.

Frequency Encoding: Huffman encoding [138] is an entropy encoding data compression algorithm that assigns variable-length prefix-free codes to symbols so as to approximately match

code lengths with the probabilities of the symbols themselves. Consider three symbols with probabilities 0.4, 0.3, 0.3 respectively. While a naive compression technique would map them to the binary representations 00, 01, 10 respectively, the Huffman code would map them to 0, 10, 11 respectively, so as to achieve a smaller average output size for transmitting a large number of these symbols. For a sorted set of m symbol probabilities, the linear-time Huffman code generator uses a *bottom-up tree construction* mechanism to map the different symbols to efficient binary codes ((2m-1)-node binary tree), rounding the given probabilities to the closest negative powers of two. Huffman codes can thus structurally represent any rounded frequency distribution as a binary tree, and vice-versa. Thus Huffman codes provide a means of translating the path frequency distribution at any router into an equivalent binary tree representation.

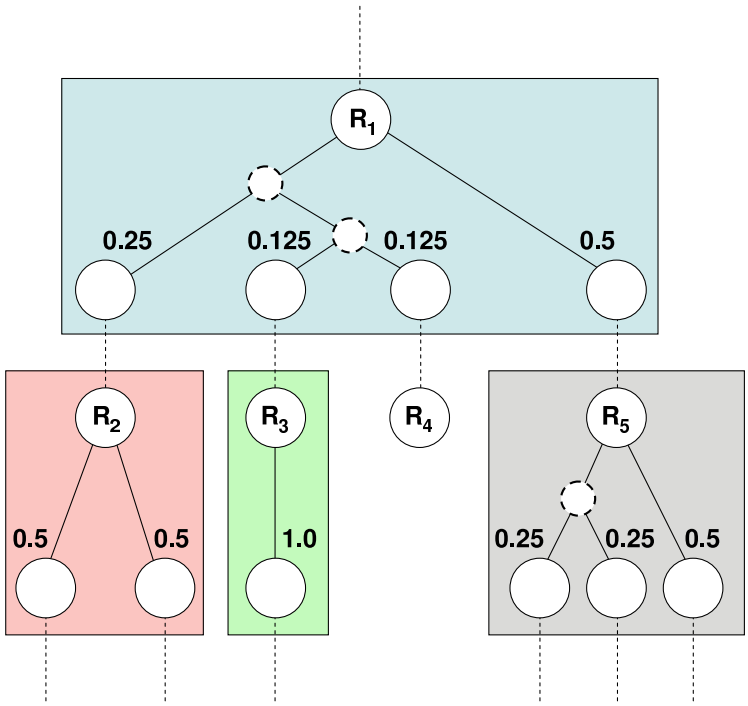


Figure 5.2 Modular Path Frequency Tree

Frequency Embedding: Every intermediate router in the attack tree deploys path frequency counters to measure the number of packets it has received from each of its immediate children

in the attack path tree, the total number of counters now being linearly dependent on that router's degree only, and not on the total number of paths in its attack sub-tree. The periodic snapshots of the frequency distribution are then run through a on-board Huffman encoder to generate the corresponding frequency-encoded Huffman binary trees. Finally, the edges linking any router to its immediate children in the attack path tree are replaced by these time-varying Huffman trees (H_{R_i}) to obtain the path frequency tree. Fig. 5.8 shows the frequency embedding for the attack tree in Fig. 5.7, with the path frequencies rounded off to the closest negative power of two at each depth in the tree. Eqn. 5.11 then represents the modified *recurrence relation* for the bottom-up approach to the path frequency tree construction problem.

$$T_{R_1} = H_{R_1} \cup T_{R_2} \cup T_{R_3} \cup T_{R_4} \cup T_{R_5} \tag{5.11}$$

Physical Representation: The ordering that the Huffman encoding imposes in every iteration might potentially vary from the original ordering of its immediate children for any router - see the ordering of the children of router R_5 in Fig. 5.7 and Fig. 5.8. Hence the physical representation of the attack path frequency tree should be capable of correctly associating the different attack sub-trees it has received and the computed frequency embedding with its immediate children in the attack path tree. We propose to use the balanced parentheses optimization here for both strict ordering and a compact representation.

In [139], the authors propose an optimal tree representation, needing just $2m$ bits for compactly representing a m-node tree structure. This optimal representation is obtained by a *pre-order* tree traversal, producing a "(" while visiting a node for the first time, and a ")" while visiting the node after completely visiting its sub-tree. The two parentheses can easily be replaced by bits 0 and 1 respectively, for a compact $2m$-bit tree structure representation. For a node-labeled tree, the authors propose to also use a pre-order traversal of the node labels, easily represented in $m * log(m)$ bits. Thus, the information-theoretic lower bound for representing a node-labeled binary tree is $\approx m * (2 + log(m))$ bits, as explained in [139]. We now propose to use the node-labeled balanced parentheses representation to explicitly ensure

correct association and ordering of its attack tree children, for every intermediate router in the network. The physical representation of the attack sub-tree of router R_1 is thus expressed as in Eqn. 5.12, where $\circ$ represents further recursive expansion not shown due to abstraction.

$$\begin{aligned} T_{R_1} &= \underline{((()(()()))()) \ 1234} \\ &\cup \ \underline{(()()) \ 12 \ \cup \circ \cup \circ} \\ &\cup \ \underline{(()) \ 1 \ \cup \circ} \\ &\cup \ \underline{()} \\ &\cup \ \underline{((()())()) \ 132 \ \cup \circ \cup \circ \cup \circ} \end{aligned} \tag{5.12}$$

The maximum attack path frequency tree size is shown in Eqn. 5.13, while Eqn. 5.14 represents the generalization of the bottom-up attack path frequency tree construction mechanism at each intermediate router with optional tree pruning, as a succinct *recurrence relation*, where Δ_{R_i} represents the immediate children of R_i.

$$TreeSize_{\ max} = n * (d * (4 + \lceil log_2(d) \rceil)) \tag{5.13}$$

$$T_{R_i} = X_{R_i}\left(H_{R_i} \bigcup_{R_j \in \Delta_{R_i}} T_{R_j}\right) \tag{5.14}$$

The bottom-up path frequency computation thus helps us infer tree edge frequencies at any upstream router, as measured by its downstream routers in the attack tree. We thus achieve efficient frequency detection at each intermediate router with far fewer path frequency counters, at the expense of a minor increase in the attack tree size, and a minor perturbation in inferred frequencies due to Huffman rounding.

5.1.3 Packet to Path Association

As discussed above, the attack tree is built in a bottom-up fashion, and hence each router has a snapshot of only its attack sub-tree and not the global view of the entire attack tree rooted at the victim - thereby making it infeasible for the intermediate routers to predict the global path identifier for any packet in the overall attack tree. On the other hand, the packet to path association mandates a simple means of inspecting a single packet and thereby determining its routing path in the network. In other words, our proposed solution essentially guarantees a single packet traceback in the Internet, by efficient packet marking to convey a single path identifier to the victim. In order to support this functionality, our proposed solution now employs both in-band packet marking and minimal network/router storage.

Recursive Approach: We now exploit the recursive nature of the proposed distributed divide-and-conquer approach to support single packet traceback in the network. Each of the different intermediate routers marks only the local (not global) unique path identifier in its attack sub-tree for a particular packet, and its upstream router (tree parent) during aggregation then translates this in-band packet marking information to its own local namespace of unique path identifiers. Thus the different path identifiers evolve from some local namespace to a globally unique namespace, as they propagate through multiple different router from the tree leaf to the victim in the attack tree. We propose to use simple hash lookup tables (L_{R_i}) at each intermediate router for per-hop namespace translation of the path identifiers (P_{R_i}). Eqn. 5.15 thus expresses the path identifiers as *recurrence relations*, while Eqn. 5.5 bounds the maximum size of the different path identifiers in the victim's attack tree.

$$P_{R_i} = L_{R_i}(P_{R_j}), \;\; R_j \in \Delta_{R_i} \qquad leaf \rightsquigarrow R_j \rightarrow R_i \rightsquigarrow root \tag{5.15}$$

Consider a packet received by router R_1 from router R_3 (Fig. 5.7) with an in-band packet mark of k. Then assuming that the attack sub-tree of R_2 has t unique paths, the new path identifier at R_1 would now be $(t+k)$. Thus each intermediate router stores the initial offset for

any packet received from each of its immediate children in a local hash lookup table, whose size is linearly dependent on its degree. It is to be noted that when used in conjunction with the tree pruning algorithm discussed previously, we need to provide an extra layer of indirection for appropriate *namespace scaling*. In Fig. 5.7, if R_1 prunes the attack sub-tree of R_2 from t to t' unique paths, $(t + k)$ would lead to unnecessary namespace scatter and potential size explosion due to inefficient utilization. Hence optimal namespace utilization is achieved by translating k to $(t' + k)$ instead of $(t + k)$. Thus we see that unique packet to path association with single packet traceback guarantees and no false positives/negatives is easily achieved using the proposed recursive approach to namespace translations.

Note that it is feasible that we skip the entire attack path frequency encoding step completely, and instead use the relative distribution of the different unique path signatures at the victim to infer the traffic volumes along each edge of the attack tree. However, we do not support such a design for multiple reasons: pre-emptive traffic filtering at any of the intermediate routers may skew the distribution at the victim, a large number of frequency counters are needed at the many intermediate routers for each of them to obtain a complete view of the traffic-weighted attack sub-tree, and finally dynamic changes to the Internet routing topology may further skew the distribution due to the transient nature of path signatures in such cases.

Router	Attack Path Tree	Attack Path Frequency Tree
R_1	$2 \cup R_2 \cup R_3$	$(()())\ 12 \cup R_2 \cup R_3$
R_2	0	()
R_3	$3 \cup R_4 \cup R_5 \cup R_6$	$(((())()))\ 132 \cup R_4 \cup R_6 \cup R_5$
R_4	0	()
R_5	$1 \cup R_7$	$(())\ 1 \cup R_7$
R_6	$2 \cup R_8 \cup R_9$	$(()())\ 12 \cup R_8 \cup R_9$
R_7	0	()
R_8	0	()
R_9	0	()

Figure 5.3 Illustration: Path Tree & Path Frequency Tree

Illustration: We now illustrate the working of the proposed scheme at each of the interme-

diate routers in the attack tree. Consider a sample tree as in Fig. 5.4, where the node and edge labels indicate the different intermediate routers and the actual traffic distribution respectively. The path labels are derived based on the natural (say, sorted) ordering imposed by any router on its immediate children. Table 5.3 illustrates the path tree and path frequency tree representations at each of the intermediate routers, while Table 5.5 illustrates the in-band path identifiers as they evolve across different depths, for packets along paths 2 and 4 respectively.

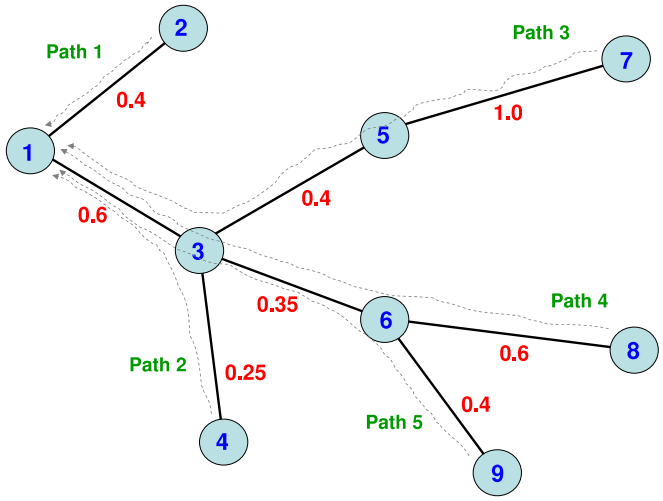


Figure 5.4 Traffic-Annotated Attack Tree

5.2 Experimental Evaluation

We now evaluate the feasibility and the potential overhead associated with an Internet-wide deployment of the proposed approach, as measured on real-life Internet topologies. We analyze the three different layers for data management, namely out-of-band packet marking, in-band packet marking, and network/router storage, by measuring the attack path tree size, attack path frequency tree size, the unique path identifier size, and the router lookup table size respectively as the evaluation criteria in our analysis.

Path 2			Path 4			
R_1	R_3	R_4	R_1	R_3	R_6	R_8
2	1	0	4	3	1	0

Figure 5.5 Illustration: In-Band Path Identifiers

We evaluate multiple attack scenarios here tracking variable number of attack sources at each router, thereby evaluating different tree pruning limits of 8K, 32K, and 128K at each tree depth. We also capture both the average (A) and maximum (M) values for all the metrics at the different tree depths, as they realistically indicate the *utilization* and *provisioning* requirements respectively. Finally, we define an attribute called "estimate" that shows the average value of these metrics across all the different tree depths. We have assumed, in this estimation, that an intermediate router has an equal probability of being present at any of the different tree depths, when viewed globally for all the potential attack victims in the Internet. Although this assumption might seem inaccurate, it helps us realistically estimate different benchmarks for any router in today's Internet. We also use two real-life Internet topologies for our performance evaluation: the datasets obtained from CAIDA's Skitter [140] and Lumeta's Internet Mapping projects [133]. In Fig. 5.6, we show the average and maximum router degree at each depth in the attack tree, for the two datasets. While characterizing the high variability in the degree distribution, we also notice that the Skitter dataset shows high clustering near the victim, while the Lumeta dataset shows a more uniform distribution.

5.2.1 Out-of-band Traffic

As we use an out-of-band channel for transmitting the attack tree from the different routers to the victim, we need to ensure that the resulting increase in traffic volume does not overwhelm the victim during an attack. Figs. 5.7(a), 5.7(b) represent the periodically transmitted attack path tree size, while Figs. 5.8(a), 5.8(b) represent the periodically transmitted attack path frequency tree size, for the two topologies respectively. In each of these graphs, the X-axis represents the tree depth, while the Y-axis represents the attack sub-tree representation (in

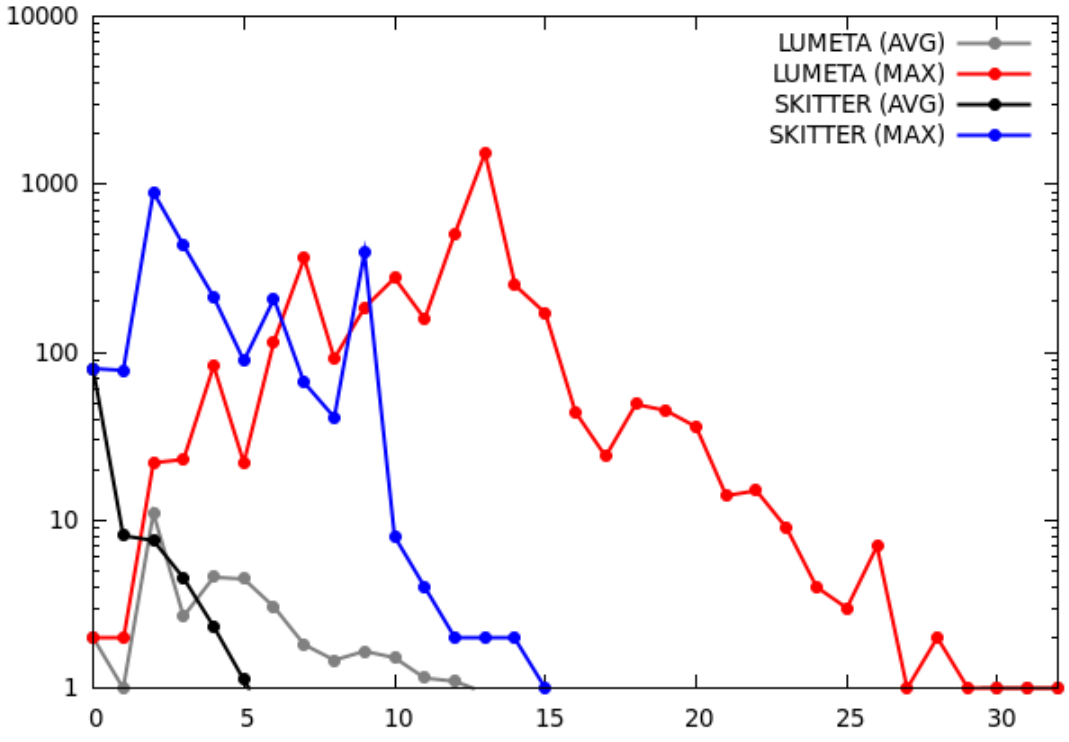


Figure 5.6 Degree Distribution (#hops vs #nodes)

KB) at each router respectively. We notice that the average values at each tree depth closely model the maximum values at each of those depths. We also notice the *Long Tail* phenomenon, indicating the large number of packet sources, and also the relative sparsity of routers at depths closer to the packet source (rather than the victim). Also distinctly visible is the effect of the different attack tree pruning limits at multiple intermediate routers. While the maximum attack tree size is $\approx$ 100 & 150KB (highest at depth 0) for the two models respectively, its average *estimate* for any router is a mere $\approx$ 15 & 25KB respectively. Thus the average data transmission per router per victim in every refresh interval, is a few kilobytes of control plane traffic, for multiple gigabytes of data plane attack traffic.

5.2.2 In-band Traffic

We use in-band packet marking for establishing unique packet to path association in our proposed approach. Figs. 5.9(a), 5.9(b) represent the in-band packet marking size for the two

topologies respectively. In each of these graphs, the X-axis represents the tree depth, while the Y-axis represents the packet marking size (in bits) respectively. We notice that the attack tree is dense for the top one-third of its depth, and is significantly sparse for further depths from the victim. An in-band packet marking field of 17 bits thus suffices to track 100K packet sources, while 20 bits could track upward of a million sources, thereby realistically providing single packet traceback guarantees in today's Internet.

As the different packet marking schemes in literature use varying number of packets each with different bit-sizes for in-band marking, we propose to use the total number of bits marked across all those packets for obtaining a certain attack path detection probability [81], as our evaluation criterion. Fig. 5.11(a) shows the total in-band data transmission for path detection probabilities of 50% and 95%, for PPM [81], FIT [82], EPM [90], TPM [96], Huffman [88], and our proposed scheme. In this graph, the X-axis represents the different traceback mechanisms being studied and the different path detection probabilities, while the Y-axis represents the total packet marking size (in bits) respectively. It is to be noted that while both TPM and Huffman additionally require a pushback mechanism, our proposed scheme requires an out-of-band periodic ($\frac{1}{100K}$ packets) transmission of the attack path frequency tree. Thus we see that our proposed approach not only provides single packet traceback guarantees with no false positives/negatives, but also achieves multiple orders of magnitude improvement with respect to network traffic (aggregate of in-band and out-of-band) overhead, over other well-known schemes in the literature.

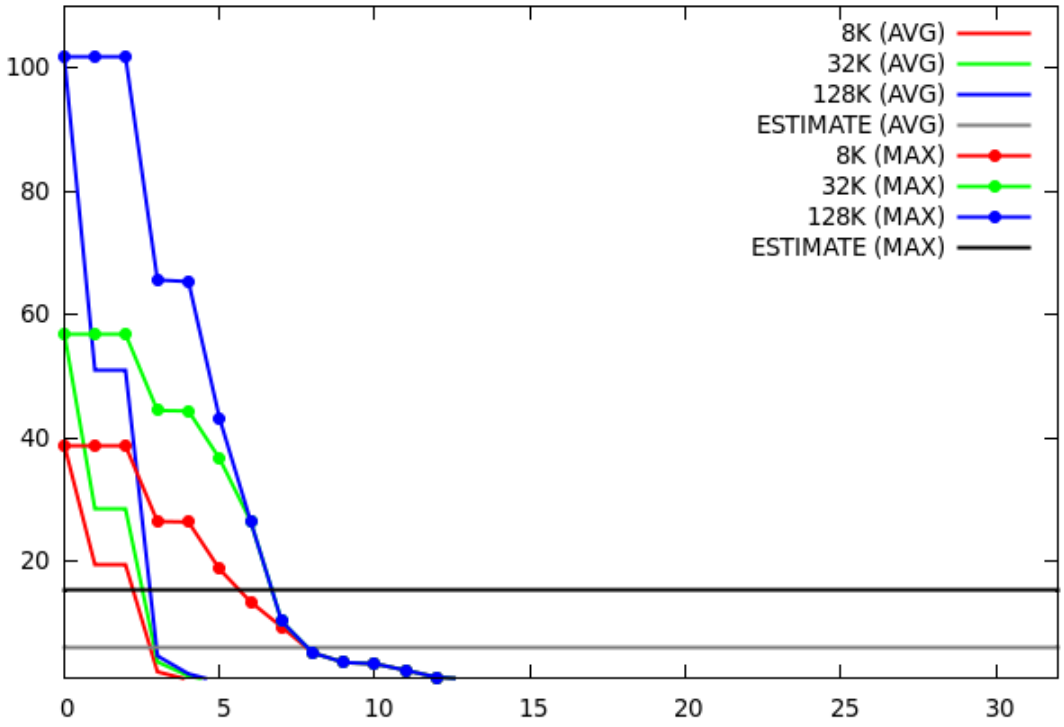
5.2.3 Router Storage

In our proposed approach, we use a router hash lookup table for namespace translation to ensure correctness of the unique packet to path association. Figs. 5.10(a), 5.10(b) represent the router lookup table size at each depth in the attack tree, for the two topologies respectively. In each of these graphs, the X-axis represents the tree depth, while the Y-axis represents the router storage size (in bytes) respectively. While the maximum lookup table size is $\approx$ 4KB, its

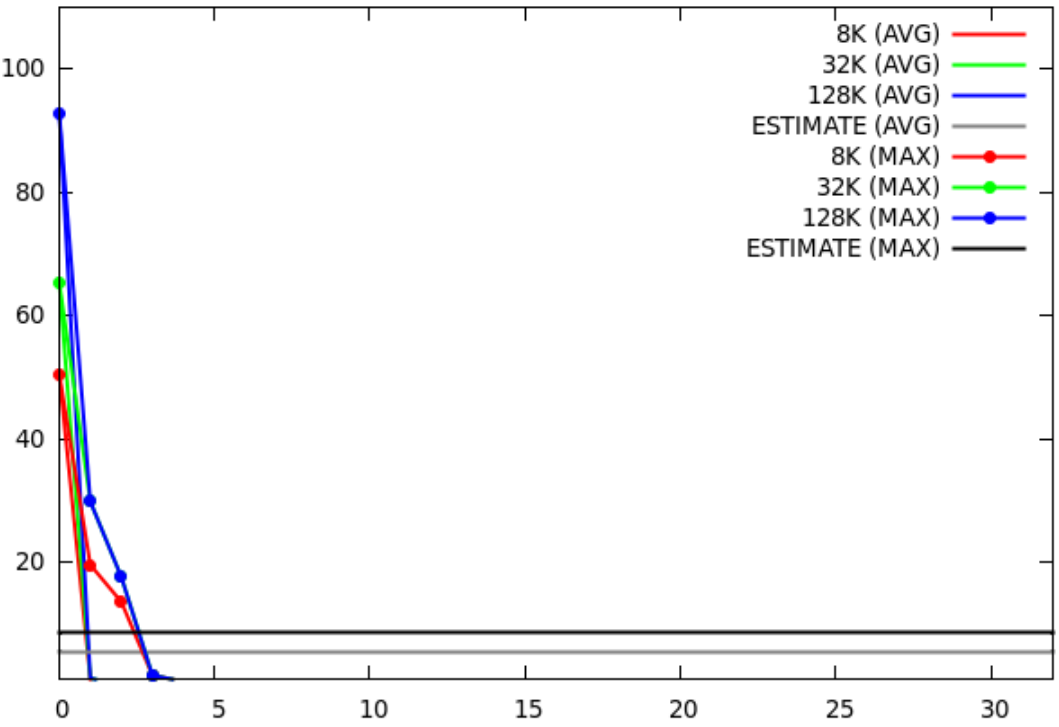
average *estimate* for any Internet router is just $\approx$ 0.4KB.

As any router needs to store both the lookup table and the attack path frequency tree in local memory, we now compare the total router storage requirements of the proposed scheme with other well-known packet logging schemes, namely HASH [76], RMS [79], Huffman [88]. In Fig. 5.11(b), we notice that the router storage requirements for the other schemes are dependent on the link speeds, and also the duration for which they are cached. In this graph, the X-axis represents the different traceback mechanisms being studied and the different router link speeds, while the Y-axis represents the total router storage size (in GB/day) respectively. The proposed scheme however depends only on the number of victims being simultaneously supported (100K in Fig. 5.11(b)) and the individual tree pruning limits (8K and 32K in Fig. 5.11(b)) - thus providing greater scalability and flexibility by being link speed and time agnostic unlike other well-known schemes. Additionally, this analysis shows that the proposed approach can defend 100K different victims each targeted by 32K independent high-bandwidth DDoS attackers simultaneously, and this far exceeds the maximum number of parallel attack flows that other well-known schemes can conceivably handle. We can further use *statistical multiplexing* of the router storage across multiple victims, thereby easily achieving more than an order of magnitude improvement over other well-known defense strategies.

We thus see that our proposed approach employing *separation of duties*, while using a bottom-up tree construction mechanism with optional tree pruning, and utilizing both packet marking and packet logging paradigms, provides significant improvement over other well-known schemes in the literature. Most importantly, it realistically provides single packet traceback guarantees in today's Internet and serves as a practically viable DDoS defense mechanism.

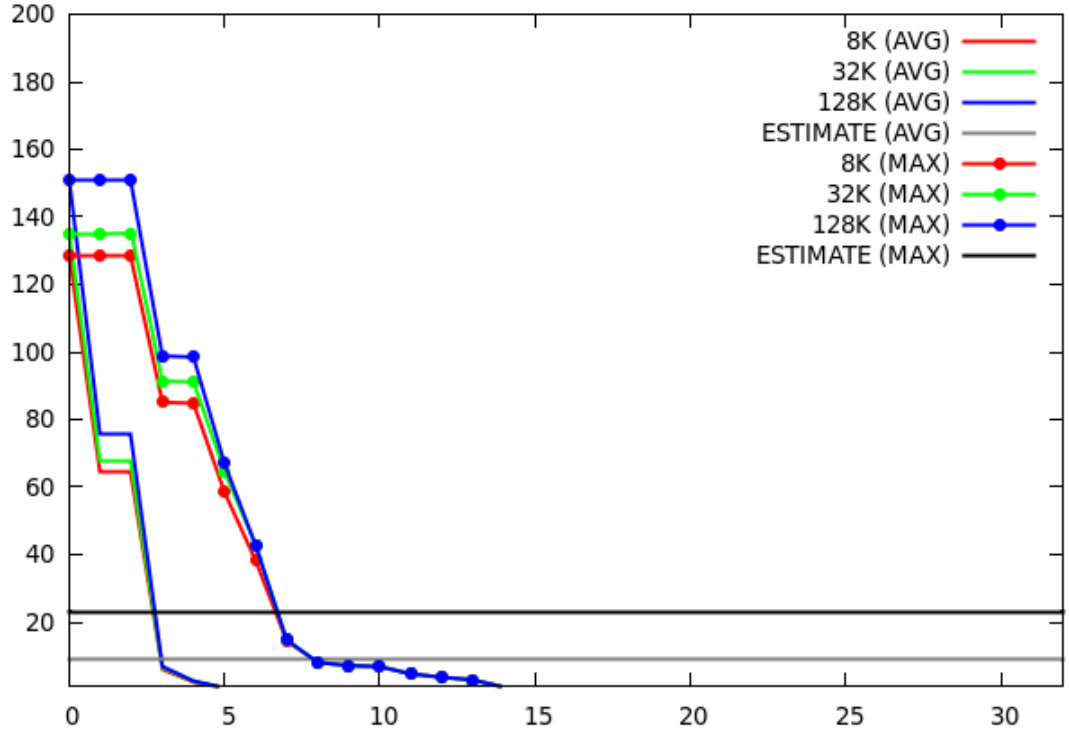


(a) Lumeta Topology

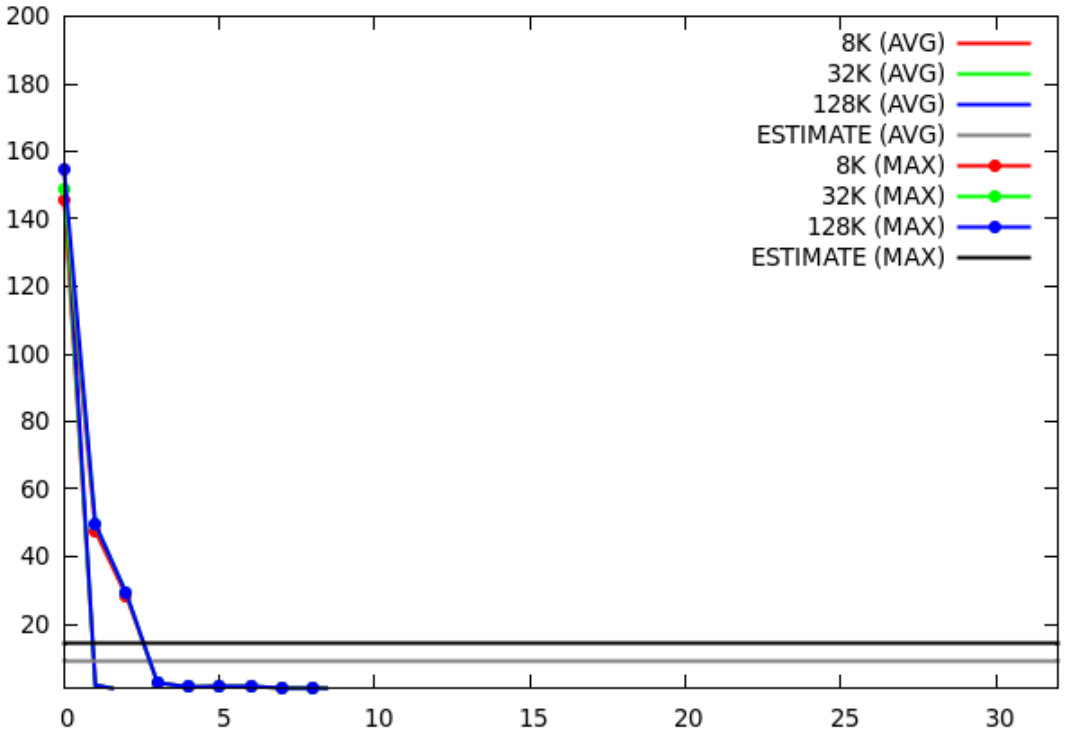


(b) Skitter Topology

Figure 5.7 (Out-of-band) Path Tree Size (#hops vs #KBs)

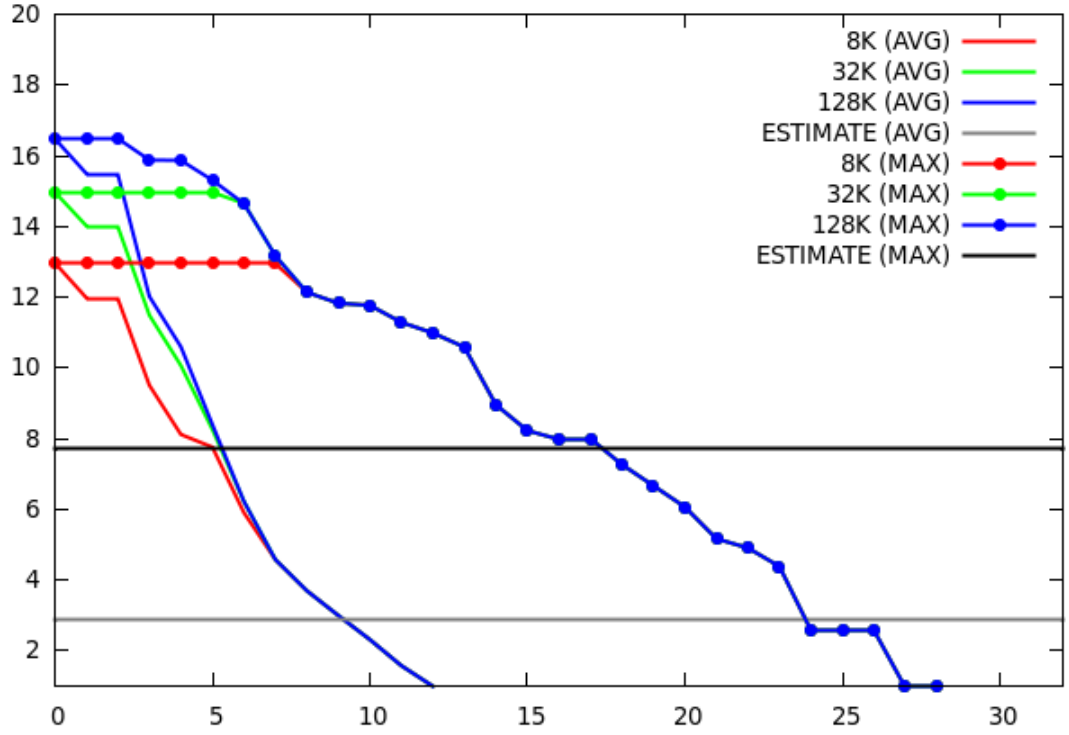


(a) Lumeta Topology

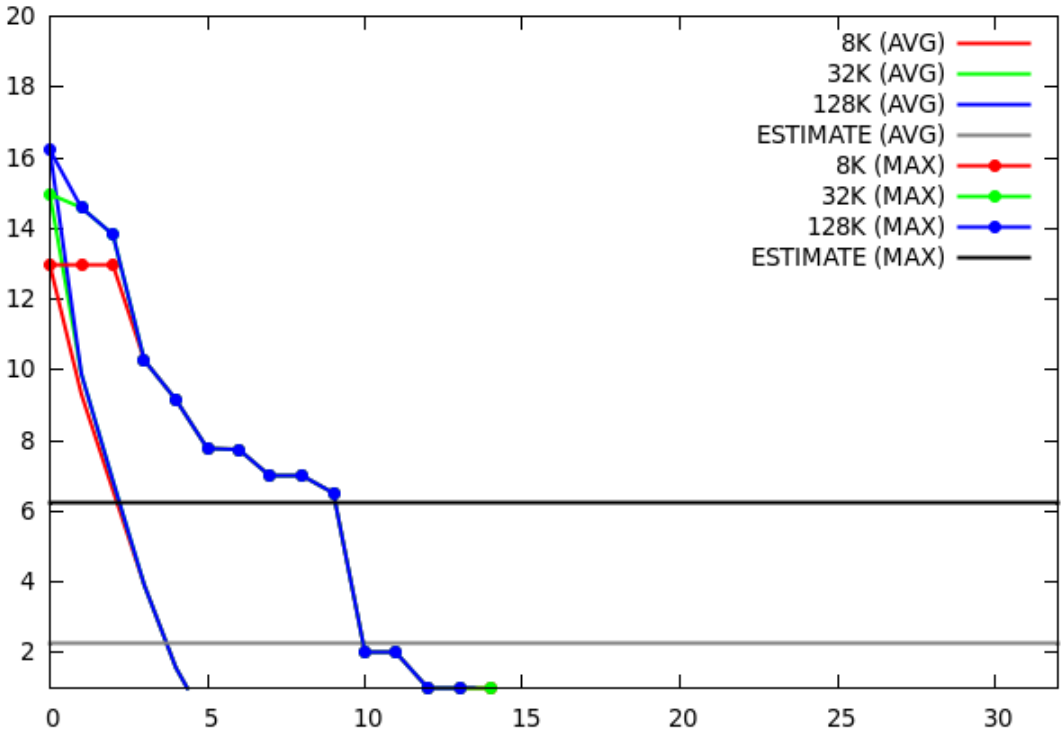


(b) Skitter Topology

Figure 5.8 (Out-of-band) Path Frequency Tree Size (#hops vs #KBs)

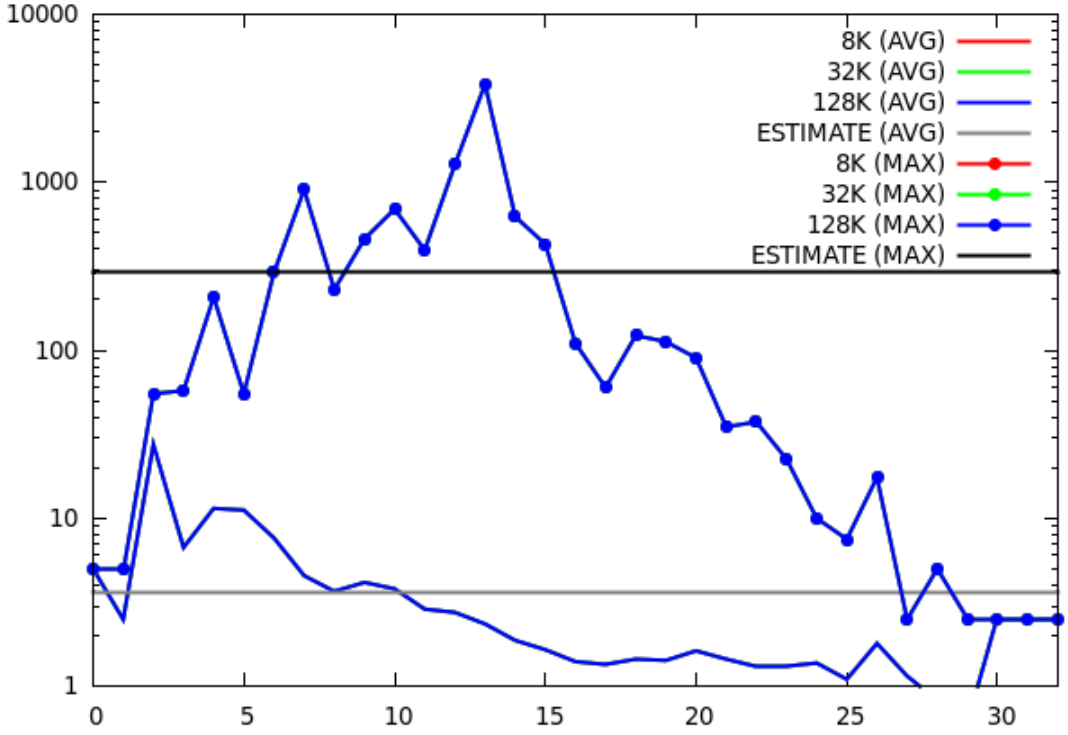


(a) Lumeta Topology

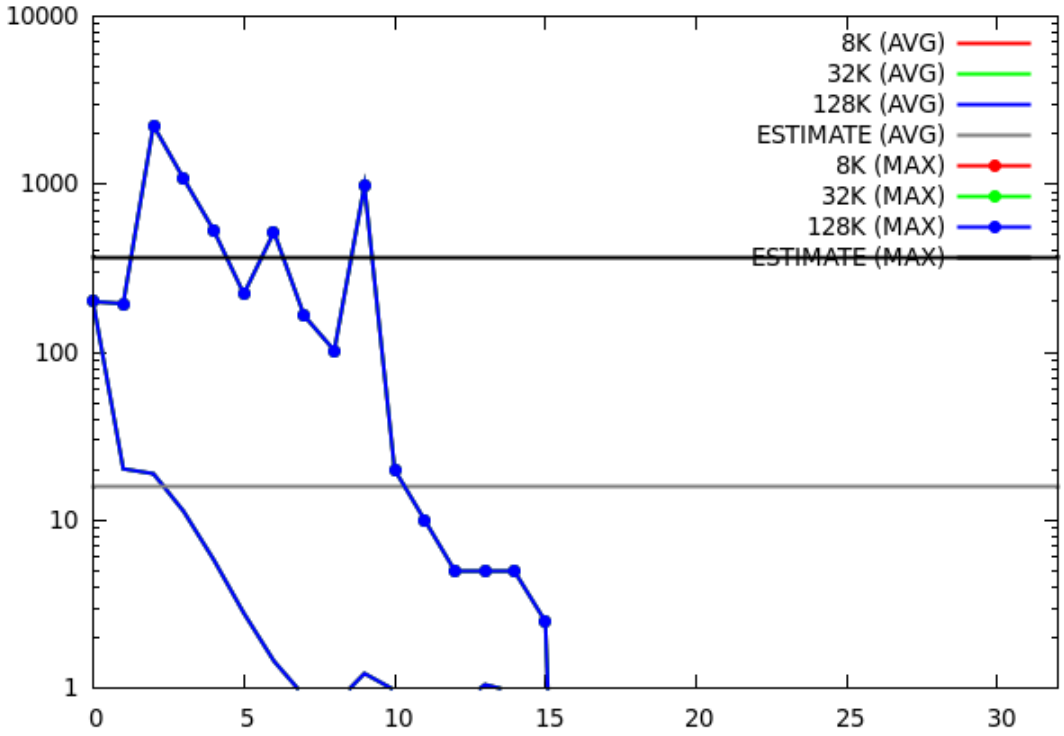


(b) Skitter Topology

Figure 5.9 (In-band) Packet Marking Size (#hops vs #bits)

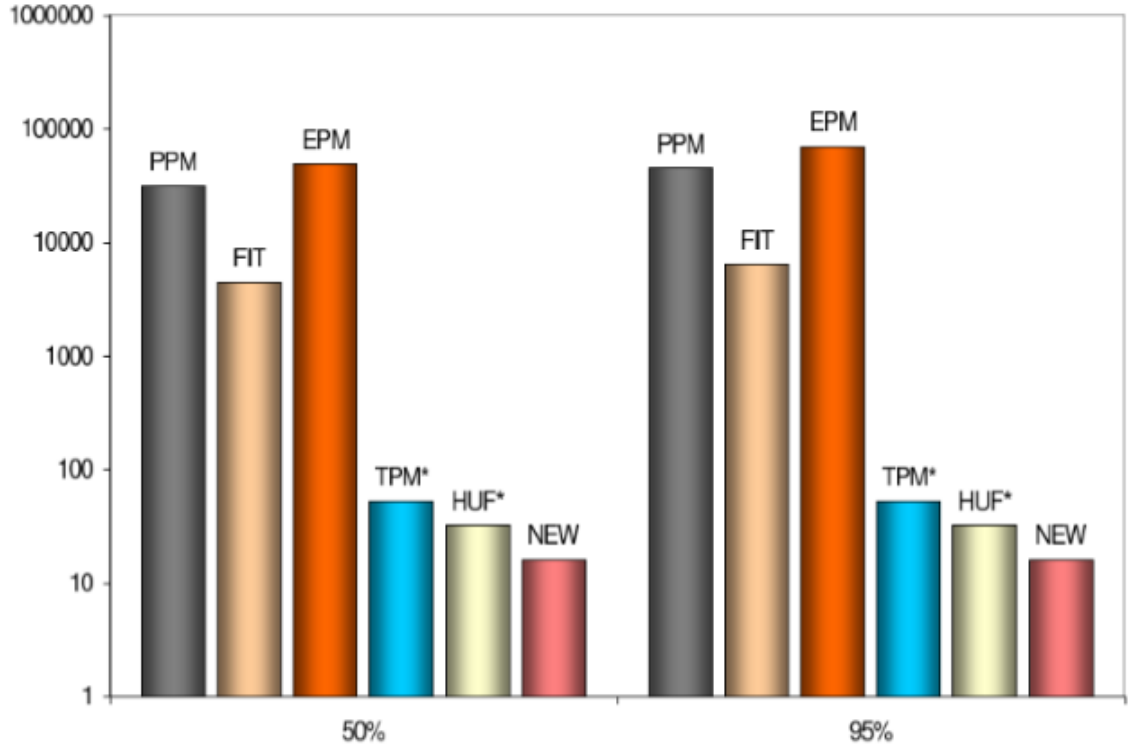


(a) Lumeta Topology

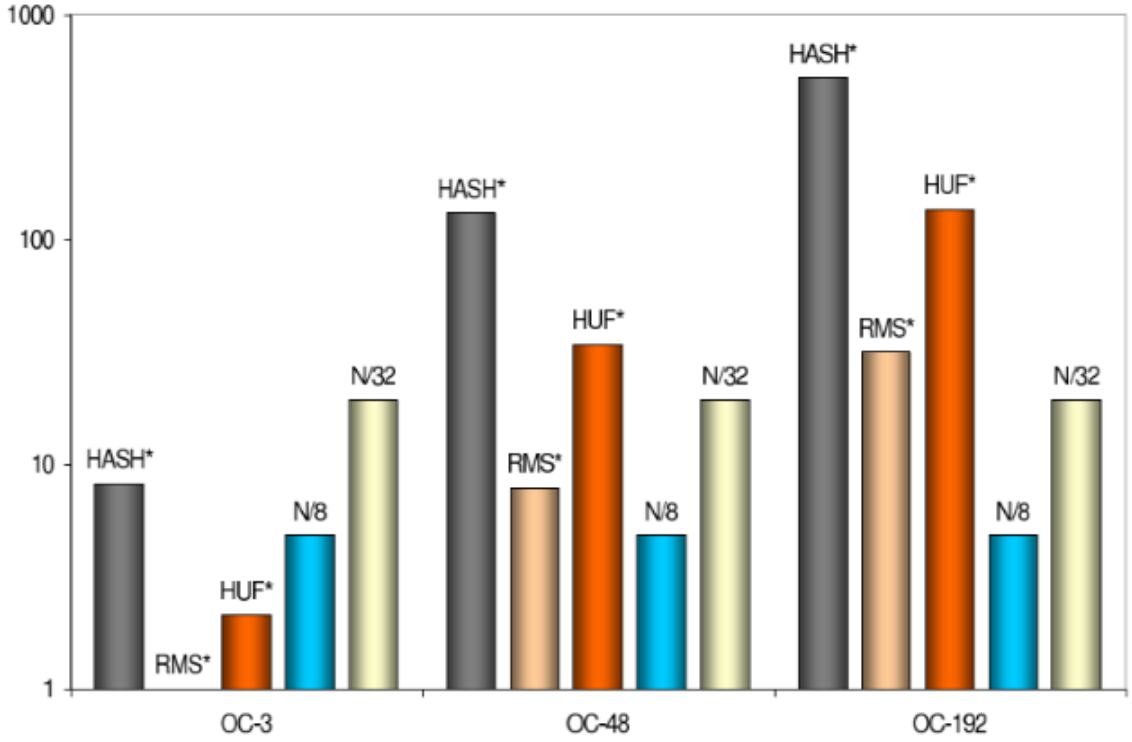


(b) Skitter Topology

Figure 5.10 (Router Storage) Lookup Table Size (#hops vs #bytes)



(a) Network Traffic Overhead



(b) Router Storage Overhead

Figure 5.11 Comparison: Network Traffic and Storage Overhead

CHAPTER 6. SOCIAL ACCEPTANCE OF ATTACK ATTRIBUTION

We have thus far discussed our proposed composable data model in great detail, and have also harnessed this novel abstraction to yield two concrete implementations that are both efficient in design and practically viable in scope - the first approach employing advanced graph theory concepts for router labeling, and the second approach employing distributed divide-and-conquer techniques for path labeling in the Internet. We have thus successfully advanced the state of the art - the near-optimal performance guarantees and easy deployability making them both practically feasible and economically viable for an Internet-wide deployment.

While the focus of the entire research community has been on a cheap, effective, and non-disruptive solution to the denial-of-service problem, not much attention has been paid to the social aspects including widespread user acceptance. The many DDoS defense proposals under consideration today, while both practical and economical to deploy, violate some of the key tenets of Internet use. The highly invasive and non-anonymous nature of wide-spread data gathering, in the name of network traceback, would significantly hinder the fundamental ability of users to stay anonymous (online) and operate freely without retribution. While these principles have never been explicitly incorporated in any discussions about the Internet as a whole, the widely popular concepts of free speech and popular dissent have received a substantial nod of approval from various individuals, human rights organizations, and even many democratic governments. Thus it is vital that we pro-actively address potential user privacy concerns with respect to the quick archival and easy retrieval of daily user activities and their proxy routing by multiple entities, while still supporting network-wide traceback of attack traffic.

The many network traceback techniques today try to construct an attack graph rooted at the victim, with or without network support, to assist in attack attribution and incremental attack mitigation. More specifically, the different packet logging techniques store the individual packet fingerprints at the different intermediate routers that forwarded that packet, thereby making it easily accessible to multiple disjoint entities - it would now be practically infeasible to enforce access restrictions and proper oversight across all ISP networks in the Internet. On the other hand, the different packet marking techniques store the path fingerprint in the data packets themselves, thereby rendering them vulnerable to interception by any strategically placed eavesdropping device in the network core. Additionally, the different Internet service providers (ISP) are probably not too keen on revealing their internal network topology to the external world, primarily to their competitors and hackers, as it could potentially lead to competitive pressure and further security concerns. Thus, in addition to addressing the various performance constraints, we also need to provide substantial usability guarantees to make any proposed DDoS defense remotely realistic. The need of the hour is thus a simple technique to fingerprint network data flows anonymously, such that information about specific attack instances can be easily gleaned, but massive data collection is impractical if not infeasible.

In this regard, we propose anonymity in the traceback context to serve as a benchmark for the design and analysis of the different traceback techniques, old and new alike. While network traceback seeks to trace the origins of every data packet on the Internet, anonymity constraints prohibit us from tracking any such attack packet in the first place. Our critical contribution here has thus been to reconcile these mutually divergent requirements to make Internet-wide DDoS defenses not only economically feasible and politically viable but also socially acceptable.

6.1 Anonymity

We now provide a formal definition of anonymity, and also motivate the need for anonymity in the traceback context on the Internet.

Definition: It is theoretically defined as the state of being not identifiable within a set of objects, the *anonymity set*, such that larger the anonymity set and more even the distribution of the various actions across its many objects, the stronger the anonymity [141] - a qualitative measure of anonymity being proposed in [142]. Our goal here is to thus achieve *possible innocence* - a provable non-trivial probability that any action was possibly performed by another object. Note that we do not seek *absolute privacy* here as it is not exactly clear how that can be supported without extensive changes to the existing Internet architecture itself.

As we now study anonymity in the traceback context, we realize that the different objects in the anonymity set represent the many network routers, while the individual actions represent their routing behavior respectively. Hence given the path fingerprint of any network data packet, while traceback techniques expect to identify all intermediate routers, the anonymity constraints require that it be impossible to flag any such router - thereby representing mutually disjoint expectations that seem hard to reconcile.

While it is potentially feasible to provide a secure and authenticated means for traceback encoding/decoding, enforcing access restrictions and oversight across all ISP networks would be practically infeasible. Additionally, it would be open to potential abuse by covert eavesdropping by various governmental organizations. Hence a longer term plan would need to support anonymity within the basic framework of the traceback process itself such that additional security measures are not needed. We now present the anonymous in the traceback context that can possibly help us bridge this gap between anonymity and traceback.

Note that anonymity is a popular concept that has been studied in multiple contexts and at multiple different layers of online communications, and subsequently represents different requirements for researchers in different domains. The primary focus of anonymity has been in the application-layer, as evident by the large amount of research devoted to it [143]-

[147]. However, we interpret anonymity here to be applicable at the IP-layer, and not at the application-layer - not much research is available in this context [148], and we presume that fundamental support at the IP layer can yield far better results at the application layer too, thereby potentially enhancing other anonymous techniques in use today.

6.2 Anonymity in the Traceback Context

The many traceback techniques in use today construct the attack graph by gathering an exhaustive list of unique router labels and the corresponding physical routers that they represent. We make a simple observation here - the feasibility of obtaining such a unique mapping for all network entities is a fundamental design bottleneck, from an anonymity perspective. Hence, we now impose an additional design constraint to provide stronger anonymity guarantees - *no single entity must be individually capable of decoding a stored path fingerprint in its entirety.* This would ensure that successful decoding of any path fingerprint would require participation from multiple co-operating network entities, thereby limiting the scope of any potential abuse.

In other words, our proposed design principle dictates that the ability to interpret the path fingerprint of any network data packet be distributed amongst multiple network entities such that no single entity can easily harvest all available information in the network. Such a robust mechanism would ensure that no covert action by any single entity would be feasible without notification and approval of other network entities in the public domain.

We now analyze the proposed router labeling technique and its use of localized labels for node width optimization and logical routes for path length optimization, to verify if it indeed satisfies the design constraint stated above. Note that while the proposed path labeling technique can also exploit the logical routes abstraction, we do not delve into further details as its effect on overall performance is rather easily understood by any reader. We do however analyze the two different optimizations here in greater detail, and subsequently argue that they

provide significantly greater anonymity guarantees due to the very nature of their design.

Lemma 6.2.1. *The greater the router label reuse in the network, the stronger the anonymity guarantees that can be provided.*

Proof: Consider a network having n nodes appropriately labeled using t labels. If $n \approx t$, there exists a unique mapping from a physical router to its canonical label. However, if $n \gg t$, any single label is shared by $\approx \frac{n}{t}$ routers, assuming a uniform router label distribution. In such cases, given any router label, the probability of accurately detecting the router which issued that label is greatly reduced to $\frac{t}{n}$. For a small t and large n, it is essentially reduced to a random selection, with no implicit bias whatsoever. □

While many traceback techniques today allocate distinct router labels to minimize namespace collisions, our proposed use of localized labels intentionally causes a large number of such collisions thereby providing plausible deniability to all routers in the network - it subsequently uses a *pushback mechanism* to work-around these collisions across multiple routing paths.

Lemma 6.2.2. *The greater the router mobility along the path length, the stronger the anonymity guarantees that can be provided.*

Proof: Consider a node m at position k along a path of length l. If l and k are fixed for all packets along that path, it is possible to infer the router label of node m by monitoring all traffic along that path. Additionally, for t distinct paths $l_1, ..., l_t$ with node m at positions $k_1, ..., k_t$ respectively, triangulation techniques can easily help infer its identity. However, if l and k are not guaranteed to be the same for all traffic along any path, the apriori knowledge of the network topology would prove useless in inferring the individual router identities. □

While many traceback techniques today operate at a higher abstraction by uniquely mapping a physical router to its logical representation, our proposed use of logical routes intentionally allows each router to independently choose its own logical representation. In this regard,

a large number of network topologies are possible - a single router can masquerade as one or more logical entities, a group of routers can aggregate to form one or more logical entities, etc. These decisions are fully localized to the participating routers and their presence/behavior is unknown to other network entities, thereby providing stronger anonymity guarantees. Note that this optimization can also be applied to the proposed path labeling technique, if desired.

Corollary 6.2.3. *The degree of anonymity is unaffected by any change in node width or path length in the network.*

Proof: Consider a n-node network having t labels, each of node width w - an optimal binary representation requiring $w = \lceil log_2(t) \rceil$. However, if w is sub-optimally chosen to be independent of t, then according to Lemma 6.2.1 the degree of anonymity is unaffected by any node width change. Similarly, consider a node m at position k along a path of length l. According to Lemma 6.2.2, router mobility such that $0 \leq k \leq l$ provides stronger anonymity than a fixed k along the path. While continual adaptations of the path length l may yield even better anonymity guarantees, it is not essential for our stated goal of *possible innocence* and plausible deniability. Hence, we consider any node width or path length changes to be purely performance oriented, and not governed by any anonymity constraints. □

Theorem 6.2.4. *Maximal router mobility along the routing path and maximal router label reuse in the network together provide anonymity for any network entity.*

Proof: While Lemma 6.2.1 proves that router label reuse protects against short-term path fingerprint inversion attacks, Lemma 6.2.2 proves that employing pseudo-entities protects against long-term traffic analysis attacks. Thus the unique (global) mapping from a physical router to its label can now be replaced with a newer many-to-many (local) mapping from a physical router to its logical entities and a subsequent many-to-many (local) mapping from this logical entity to its canonical labels, thereby providing plausible deniability for all entities. □

Theorem 6.2.5. *Better performance & stronger anonymity need not be mutually exclusive, but can be achieved simultaneously by any network traceback technique.*

Proof: Theorem 6.2.4 shows that stronger anonymity can be easily achieved by maximal router label reuse and increased router mobility. A critical observation here is that the node width now depends only on the cardinality of the set of routers assigned the same label, and not on the total network size. Additionally, appropriate mobility choices at each router can potentially reduce the overall path length. As discussed at length previously, appropriate reduction of the node width and path length sizes can lead to faster network traceback. We thus realize that these mutually independent optimizations thereby help us achieve the best of both worlds - stronger anonymity and better performance. □

Collusion Attacks: The proposed use of star coloring for network-wide router label assignment requires that the true identity of any router label be revealed to other routers in its immediate vicinity. While this might suffice for plausible deniability against any single entity attacks, the bigger class of multi-entity collusion attacks can significantly affect the overall degree of anonymity. In this regard, it it critical to note that the pushback mechanism works via incremental per-hop discovery, and hence requires the cooperation of all intermediate routers for successful traceback. Any refusal to comply with traceback decoding directives by any router along the routing path essentially renders the path fingerprint useless - thus collusion attacks are limited in scope to discover only *partial path segments* directly visible to the many colluding entities. It is vulnerable only when all nodes along the routing path from the source to the destination have been compromised, which represents a rather unlikely attack scenario.

Thus our proposed anonymous in the traceback context is not only resistant to massive data gathering by individual entities, but also provides plausible deniability against collusion attacks. It is to be noted here that no known traceback techniques today provide any anonymity guarantee whatsoever, let alone strong guarantees of multi-entity attack resistance.

6.3 Theoretical Analysis

As defined previously, anonymity is the state of being unidentifiable within a set of entities. We now provide a quantitative measure of anonymity, thereby formalizing the notion of *possible innocence* and *absolute privacy*.

Consider a network having n nodes with a probability mass function p and a corresponding entropy H [141]. Let p now provide the probability p_i that a node n_i can be uniquely identified as having performed some action, say issued a particular router label, forwarded a certain packet, etc. Thus H now provides a measure of the information contained in this probability distribution. Eqn. 6.1 now bounds the range of p_i values, while Eqn. 6.2 measures the corresponding entropy of the system.

$$0 \leq p_i \leq 1 \quad \& \quad \sum_{i=1}^{n} p_i = 1 \quad \forall i \in [1, n] \tag{6.1}$$

$$H = -\sum_{i=1}^{n} p_i \, log(p_i) \quad \vdash \quad H_{max} = log(n) \tag{6.2}$$

The maximum entropy is achieved when all nodes seem equally likely to have performed a certain action, and thus represents the state of *absolute privacy* (Eqn. 6.3). In this regard, *possible innocence* represents a weaker guarantee where each node has a provably non-trivial probability to have not performed that action (Eqn. 6.4). Finally, we define *degree of anonymity* (η) as a measure of the information leaked by the system, some node having performed that action (Eqn. 6.5).

$$H = H_{max} \;\Rightarrow\; Absolute\ Privacy \tag{6.3}$$

$$\forall i,\ p_i < 1 \;\Rightarrow\; Possible\ Innocence \tag{6.4}$$

$$\eta = 1 - \frac{H_{max} - H}{H_{max}} = \frac{H}{H_{max}} \tag{6.5}$$

Let us now consider a node at position k along a path of length l. Let k' and l' now represent the corresponding node position and path length after appropriate logical abstraction respectively. Also, let r represent the graph resizing threshold - we assume that r is a global constant for simplicity. Eqn 6.6 thus represents the mobility of any router along the path length. It is critical to note that router mobility is thus not feasible along the entire path length, and is restricted to a smaller range to account for the graph resizing limits of nodes situated on either side along that path. The *degree of anonymity* is also reduced proportionately (Eqn. 6.7), thereby providing *possible innocence* guarantees only.

$$0 \leq \left\lceil \frac{k}{r} \right\rceil \leq k' \leq l' - \left\lceil \frac{l-k}{r} \right\rceil \leq l' \tag{6.6}$$

$$\eta \approx \frac{l' - \frac{l}{r}}{l'} = 1 - \frac{l}{rl'} \tag{6.7}$$

$$\eta \approx 1 - \frac{1}{r} \quad \Leftarrow \quad l \approx l' \tag{6.8}$$

The degree of anonymity is thus dependent only on the graph resizing threshold if the effective path length is unchanged (Eqn. 6.8) - thus greater the graph resizing, stronger the anonymity. Additionally, as it is independent of the original node position, we can provide *similar* anonymity guarantees for all routers along the path from the source to its destination.

Thus our proposed graph resizing techniques when incorporated by the different trace-back techniques can additionally provide stronger anonymity guarantees for all Internet users, thereby making their network-wide deployment more socially acceptable.

CHAPTER 7. CONCLUSIONS

The phenomenal growth of the Internet from a few machines in a research lab some decades ago to extensive world-wide adoption today, for a wide array of activities including communications and electronic commerce, has also exposed it to a large number of security attacks. The steady evolution of distributed denial-of-service (DDoS) attacks as a vehicle for achieving political, economic and commercial gains, and the relative ease, low costs, and limited accountability in launching such attacks, have rendered them one of the top threats to today's Internet services. Although various DDoS attack prevention, mitigation, and traceback techniques have been proposed, their relative uptake has been minimal at best, due to the lack of a robust, fool-proof, and universal defense mechanism. The need of the hour is thus a defense mechanism that can not only handle sophisticated attacks with ease, but also provide definitive attack source attribution for critical deterrence against launching such attacks.

The primary focus of our research work here has been on network-wide support for defending against DDoS attacks. In this regard, we focus on the technical challenges for practical feasibility and also the social aspects for widespread acceptance. Our main contributions here can be succinctly stated as follows: (1) design of a simple framework for the unified representation of the different DDoS defense approaches, (2) design of a composable data model to leverage multiple independent DDoS defense techniques to address their individual shortcomings, (3) design of newer router labeling techniques based on graph coloring primitives for faster traceback guarantees, (4) design of newer path labeling techniques based on distributed divide-and-conquer concepts for single packet traceback guarantees, and finally, (5) subtle design changes to augment the proposed techniques to also provide enhanced user privacy and

anonymity guarantees. In short, we have focussed on making DDoS defense mechanisms economically feasible, practically viable, and socially acceptable, thereby significantly improving the state of the art for realistic Internet-wide deployment today.

We now provide a short summary of our research work as described in the different chapters of this dissertation, to serve as an easy reference for quick readers.

In Chapter 1, we discussed the problem of DDoS attacks and the great impact they have on the Internet economy, by enumerating the various attack instances publicly reported over the past decade. We also described the overall attack mechanism and the different automated tools at the attacker's disposal today. We have also identified a few distinct characteristics of the Internet design that readily enable such attacks, and finally provided a detailed classification of DDoS attacks based on multiple attributes, namely the target of the attack, the motive for the attack, and the precise steps taken in launching these attacks.

In Chapter 2, we discussed the various DDoS defense approaches in great detail, starting with the most basic approach describing the various best practices for network setup/maintenance. We have then enumerated the many challenges in defending against such attacks, including the flawed design of the Internet with no regard for security, the widespread use of IP spoofing, the lack of readily accessible test-beds for evaluation, etc. We have also compiled a list of desirable features for any DDoS defense mechanism, so as to make it withstand newer sophisticated attacks and also to render it practically feasible for Internet-wide deployment. Finally, we studied the various DDoS defense approaches proposed by researchers in the industry and academia, while broadly classifying them as prevention, mitigation, and traceback strategies.

In Chapter 3, we have described in detail the inability of the different DDoS defense approaches to operate in isolation and have subsequently proposed the notion of traceback-enabled mitigation and its availability as a subscription-based service. Our primary contribution though

has been the design of a composable data model that can not only represent the different DDoS defense mechanisms in use today, but can also provide a single framework for its mutual comparison with other pseudo-similar techniques. The proposed use of multi-axis data representation, newer data transmission schedules, and a basic understanding of the data utility concept, helps us combine various individual optimizations across multiple different techniques, so as to achieve a comprehensive design that is both provably better and practically viable.

In Chapter 4, we proposed a new router labeling algorithm based on advanced graph theory optimizations, to address the various shortcomings of traditional probabilistic packet marking traceback techniques in literature. We introduced the concept of localized labels, based on sophisticated graph coloring primitives, that explored the spatial reuse of router identifiers as a means of obtaining shorter binary representations for each of them. We then introduced the concept of logical routers, based on advanced graph resizing techniques, that explored the clustering and de-clustering of router groups as a means of obtaining shorter representative Internet routing paths for the different attack packets. In addition to addressing the scalability and incremental deployability constraints here, we have also analyzed its performance on real-life Internet topologies, across multiple attributes including coloring efficiency, overall size of the packet fingerprint, and the total convergence speed of the distributed computation.

In Chapter 5, we proposed a new data dissemination architecture, wherein we looked at the problem of DDoS defense as three disjoint issues, namely attack tree construction, attack path frequency detection, and packet to path association, and addressed them independently in a locally optimal manner. We then proposed a novel distributed divide-and-conquer approach to represent their individual implementations as succinct recurrence relations - we employed a combination of in-band packet marking, out-of-band packet marking, and router storage for computing the different path signatures in an evolutionary manner across the many routers on the Internet. Using performance evaluation on real-life Internet topologies, we were then able to show that the proposed technique can thus help us realistically achieve single packet

traceback guarantees for a large number of victims under heavy traffic loads simultaneously, with very high efficiency and practically no false positives/negatives.

In Chapter 6, we specifically called out the highly invasive nature of wide-spread data gathering being considered today in the name of network traceback, thereby potentially violating one of the key principles of Internet use today - the ability to stay anonymous and operate freely without retribution. In this regard, we have precisely formulated anonymity in the traceback context to serve as a benchmark for the design and analysis of the different traceback techniques, old and new alike. In precisely quantifying the degree of anonymity that can be achieved by any network entity, we have also shown that anonymity and network traceback need not be mutually exclusive goals, but goals that can be simultaneously achieved.

Finally, our proposed work does open up several directions for future research:

- a comprehensive analysis of all existing attack attribution techniques to identify further scope for improvement in a systematic manner, unlike the ad-hoc style in use today,
- the design of newer attack attribution techniques that can exploit the potential synergies with other seemingly independent and unrelated approaches described in literature, so as to comprehensively integrate and unify the various attack prevention, mitigation and traceback techniques in an efficient manner,
- the use of the proposed data utility concept to not only design newer incremental DDoS defense mechanisms, but also to provide a basic foundation for the design of secure protocols that can operate in the face of inadequate or imprecise information,
- the use of traceback-assisted mitigation as a powerful paradigm for designing newer attack mitigation algorithms to effectively counter emerging sophisticated DoS attacks,
- and finally, the incorporation of newer attributes (such as anonymity, scalability, etc.) into the design framework of the composable data model abstraction in a mutually orthogonal fashion so as not to impact the current performance that can be harnessed.

BIBLIOGRAPHY

[1] History of the Internet, *http://en.wikipedia.org/wiki/History_of_the_Internet*

[2] History of the World Wide Web, *http://en.wikipedia.org/wiki/History_of_the_World_Wide_Web*

[3] A. Chakrabarti and G. Manimaran, *Internet Infrastructure Security: A Taxonomy*, IEEE Network, vol.16, no.6, pp.13-21, Dec 2002

[4] Trends in Denial of Service Attack Technology, *http://www.cert.org/archive/pdf/DoS_trends.pdf*

[5] J. Nagle, *Congestion Control in IP/TCP*, RFC 896, 1984

[6] S. Floyd, *Congestion Control Principles*, RFC 2914, 2000

[7] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, *Recommendations on Queue Management and Congestion Avoidance in the Internet*, RFC 2309, 1998

[8] DoS attacks, *http://en.wikipedia.org/wiki/Denial-of-service_attack*

[9] Leading websites under attack, *http://news.cnet.com/Leading-Web-sites-under-attack/2100-1017_3-236683.html*

[10] Register.com suffers week long DDoS attack on DNS servers, *http://www.secure64.com/news-register-reflective-ddos-dns*

[11] Massive DDoS Attack Hit DNS Root Servers, http://www.internetnews.com/dev-news/article.php/1486981

[12] Outages for RIAA and SCO sites, http://news.netcraft.com/archives/2004/02/28/outages_for_riaa_sco_sites.html

[13] Online payment firm in DDoS drama, http://www.theregister.co.uk/2004/11/03/protxddosattack/

[14] Sasser infections hit hard, http://www.pcworld.com/article/115979/sasser_infections_hit_hard.html

[15] BlueSecurity folds under spammers' wrath, http://www.securityfocus.com/news/11392

[16] Massive DDoS attack target Estonia; Russia accused, http://arstechnica.com/security/news/2007/05/massive-ddos-attacks-target-estonia-russia-accused.ars

[17] Georgia President's web site under DDoS attack from Russian hackers, http://blogs.zdnet.com/security/?p=1533

[18] South Korea hit by cyber attacks, http://www.bbc.co.uk/news/technology-12646052

[19] Operation Payback cripples MasterCard sites in revenge for WikiLeaks ban, http://www.guardian.co.uk/media/2010/dec/08/operation-payback-mastercard-website-wikileaks

[20] Arbor Networks - 2010 Worldwide Infrastructure Report, http://www.arbornetworks.com/dmdocuments/ISR2010_EN.pdf

[21] All about Stuxnet, http://www.stuxnet.net/

[22] The DoS Project's trinoo distributed denial of service attack tool, *http://staff.washington.edu/dittrich/misc/trinoo.analysis*

[23] The Tribe Flood Network distributed denial of service attack tool, *http://staff.washington.edu/dittrich/misc/tfn.analysis.txt*

[24] The Stacheldraht distributed denial of service attack tool, *http://staff.washington.edu/dittrich/misc/stacheldraht.analysis*

[25] N. Long, S. Dietrich and D. Dittrich, *An Analysis of the Shaft distributed denial of service tool*, LISA, 2000

[26] CERT Advisory - Denial of Service Tools, *http://www.cert.org/advisories/CA-1999-17.html*

[27] Hacking for Dollars, *http://news.cnet.com/Hacking-for-dollars/2100-7349_3-5772238.html*

[28] Cs3 Inc - MANAnet DDoS White Papers, *http://www.cs3-inc.com/mananet.html*

[29] Mazu Networks - Mazu Technical White Papers, *http://www.mazunetworks.com/whitepapers/*

[30] Arbor Networks - The Peakflow Platform, *http://www.arbornetworks.com*

[31] Asta Networks - Vantage System Overview, *http://www.astanetworks.com/products/vantage/*

[32] How Carnivore Email Surveliance Worked, *http://email.about.com/od/staysecureandprivate/a/carnivore.htm*

[33] G-8 Talks Traceback in Moscow, *http://www.ewa-canada.com/Papers/CANCV2N5.htm*

[34] U.N. Agency Eyes Web Anonymity Controls, *http://www.cbsnews.com/stories/2008/09/12/tech/cnettechnews/main4443738.shtml*

[35] Inside Australia's Data Retention Proposal, *http://www.zdnet.com.au/inside-australia-s-data-retention-proposal-339303862.htm*

[36] Early Warning System, *http://smile29.eu/doc/DS29EN.pdf*

[37] Does averting cyberwar mean giving up web privacy? *http://www.npr.org/templates/story/story.php?storyId=127575960*

[38] GENI - Exploring networks of the future, *http://www.geni.net/*

[39] NSF NeTS FIND Initiative, *http://www.nets-find.net/*

[40] T. Peng, C. Leckie and R. Kotagiri, *Detecting reflector attacks by sharing beliefs*, IEEE Global Communications Conference, October 2003

[41] H. Wang, D. Zhang, and K. G. Shin, *Detecting SYN Flooding Attacks*, INFOCOM, 2002

[42] P. Ferguson and D. Senie, *Network ingress filtering: Defeating denial-of-service attacks which employ IP source address spoofing*, RFC 2827, 2000

[43] J. Merkovic, G. Prier and P. Reiher, *Attacking DDoS at the Source*, ICNP 2002

[44] H. Wang, A. Bose, M. El-Gendy, and K. G. Shin, *IP Easy-pass: Edge Resource Access Control*, IEEE INFOCOM, 2004

[45] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, *An architecture for differentiated services*, RFC 2475, 1998

[46] K. Park and H. Lee, *On the Effectiveness of Route-Based Packet filtering for Distributed DoS Attack Prevention in Power-Law Internets*, ACM SIGCOMM, pp. 15-26, 2001

[47] J. Li, J. Mirkovic, M. Wang, P. Reiher, and L. Zhang., *SAVE: Source address validity enforcement protocol*, IEEE INFOCOM, 2002

[48] J. Mirkovic and P. Reiher, *A Taxonomy of DDoS Attacks and Defense Mechanisms*, ACM SIGCOMM Computer Communications Review, vol.34, no.2, pp.39-54, April 2004

[49] G. Yang, M. Gerla, and M. Y. Sanadidi, *Defense against Low-rate TCP-targeted Denial-of-Service Attacks*, ISCC 2004

[50] A. Bremler-Barr and H. Levy, *Spoofing Prevention Method*, IEEE INFOCOM, 2005

[51] A. Kuzmanovic and E. Knightly, *Low-rate TCP-targeted denial of service attacks*, ACM SIGCOMM, 2003

[52] R. Beverly, S. Bauer, *The Spoofer Project: Inferring the extent of Source Address Filtering on the Internet*, USENIX SRUTI, 2005

[53] J. Ioannidis and S. M. Bellovin, *Implementing Pushback: Router-Based Defense Against DDoS Attacks*, Network and Distributed System Security Symposium, 2002

[54] R. Mahajan, S. Floyd, and D. Wetherall, *Controlling High-Bandwidth Flows at the Congested Router*, IEEE ICNP 2001

[55] T. Peng, C. Leckie and K. Ramamohanarao, *Protection from Distributed Denial of Service Attack Using History-based IP Filtering*, IEEE ICC, 2003

[56] Y. Kim, W. Lau, M. Chuah, J. Chao, *PacketScore: A statistical-based overload control against DDoS attacks*, IEEE INFOCOM, 2004

[57] C. Jin, H. Wang, and Kang G. Shin, *Hop-Count Filtering: An Effective Defense Against Spoofed DDoS Traffic*, ACM CCS, 2003

[58] A. D. Keromytis, V. Misra, and D. Rubenstein, *SOS: An Architecture for Mitigating DDoS Attacks*, IEEE JSAC, vol.22, no.1, pp.176-188, 2004

[59] A. Yarr, A. Perrig and D. Song, *Pi: A path Identification Mechanism to Defend against DDoS Attacks*, IEEE Symposium on Security and Privacy, 2003

[60] M. Sung and J. Xu, *IP Traceback-based Intelligent Packet Filtering: A Novel Technique for Defending Against Internet DDoS Attacks*, IEEE Transactions on Parallel and Distributed Systems, vol.14, no.9, pp.861-872, 2003

[61] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, *Controlling High Bandwidth Aggregates in the Network*, ACM SIGCOMM CCR, vol.32, pp.62-73, 2002

[62] D. K. Yau, John C. S. Lui, and F. Liang, *Defending Against Distributed Denial of Service Attacks with Max-Min Fair Server-centric Router Throttles*, IEEE IWQoS, 2002

[63] M. Chuah, W. Lau, Y. Kim and J. Chao, *Transient Performance of PacketScore for Blocking DDOS attacks*, IEEE ICC, 2004

[64] C. M. Cheng, H. T. Kung, and K. S. Tan, *Use of spectral analysis in defense against DoS attacks*, IEEE GLOBECOM, 2002

[65] A. Kulkarni and S. Bush, *Detecting Distributed Denial-of-Service Attacks Using Kolmogorov Complexity Metrics*, Journal of Network and Systems Management, vol.14, no.1, pp.69-80, 2006

[66] J. B. D. Cabrera, L. Lewis, X. Qin, W. Lee, R. K. Prasanth, B. Ravichandran and R. K. Mehra, *Proactive detection of distributed denial of service attacks using MIB traffic variables - a feasibility study*, IFIP/IEEE International Symposium on Integrated Network Management, 2001

[67] T. M. Gil and M. Poletto, *MULTOPS: a data-structure for bandwidth attack detection*, USENIX Security Symposium, 2001

[68] D. Xuan, S. Chellappan, X. Wang, and S. Wang, *Analyzing the Secure Overlay Services Architecture under Intelligent DDoS Attacks*, IEEE ICDCS, 2004

[69] H. Burch and B. Cheswick, *Tracing anonymous packets to their approximate source*, USENIX LISA, 2000

[70] J. Postel, *Character Generotor Protocol*, RFC 864, 1983

[71] Internet Draft: ICMP traceback messages, *http://tools.ietf.org/id/draft-bellovin-itrace-00.txt*, 2000

[72] A. Mankin, D. Massey, C.L Wu, S.F Wu and L. Zhang, *Intention-driven ICMP Traceback*, IEEE ICCCN, 2001

[73] F. Hsu and T. Chiueh, *A Path Information Cashing and Aggregation Approach to Traffic Source Identification*, IEEE ICDCS, 2003

[74] S. Kandula, D. Katabi, M. Jacob and A. Burger, *Botz-4-Sale: Surviving DDos Attacks that Mimic Flash Crowds*, USENIX NSDI, 2005

[75] R. Stone, *Centertrack: An IP overlay network for tracking DoS floods*, USENIX Security Symposium, 2000

[76] A. C. Snoeren, C. Partiridge, L. A. Sanchez, C. E. Jones, F. Tchhakountio, S. T. Kent, and W. T. Strayer, *Hash-Based IP Traceback*, ACM SIGCOMM, 2001

[77] B. H. Bloom, *Space-time trade-offs in hash coding with allowable errors*, Communications of ACM, vol.13, no.7, pp.422-426, 1970

[78] L. A. Sanchez, W. C. Milliken, A. C. Snoeren, F. Tchakountio, C. E. Jones, S. T. Kent, C. Partridge, and W. T. Strayer, *Hardware Support for a Hash-Based IP Traceback*, DARPA Information Survivability Conference and Exposition, 2001.

[79] J. Li, M. Sung, J. Xu, L. Li and Q. Zhao, *Large-Scale IP Traceback in High-Speed Internet: Practical Techniques and Theoretical Foundation*, IEEE Symposium on Security and Privacy, 2004

[80] A. Belenky and N. Ansari, *IP Traceback With Deterministic Packet Marking*, IEEE Communication Letters, vol.7, no.4, pp.162-164, 2003

[81] S. Savage, D. Wetherall, A. Karlin and T. Anderson, *Practical network support for IP traceback*, ACM SIGCOMM, 2000

[82] Abraham Yaar, Adrian Perrig and Dawn Song, *FIT: Fast Internet Traceback*, IEEE INFOCOM, 2005

[83] A. Belenky and N. Ansari, *Accommodating fragmentation in deterministic packet marking for IP traceback*, IEEE GLOBECOM, 2003

[84] I. Hamadeh and G. Kesidis, *Performance of IP Address Fragmentation Strategies for DDoS traceback*, IEEE IPOM, 2003

[85] D. Song and A. Perrig, *Advanced and authenticated marking schemes for IP traceback*, IEEE INFOCOMM, 2001

[86] D. Dean, M. Franklin, and A. Stubblefield, *An algebraic approach to IP traceback*, NDSS, 2001

[87] K. Park and H. Lee, *On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack*, IEEE INFOCOM, 2001

[88] K. H. Choi and H. K. Dai, *A marking scheme using Huffman codes for IP Traceback*, IEEE ISPAN, 2004

[89] C. Bai, G. Feng and G. Wang, *Algebraic Geometric Code Based IP Traceback*, IEEE IPCCC, 2004

[90] M. T. Goodrich, *Efficient Packet Marking for Large-Scale IP Traceback*, ACM CCS, 2002

[91] M. Ma, *Tabu Marking Scheme for Traceback*, IEEE IPDPS, 2005

[92] M. Adler, *Tradeoffs in Probabilistic Packet Marking for IP Traceback*, STOC, 2002

[93] M. Waldvogel, *GOSSIB vs Traceback Rumors*, ACSAC, 2002

[94] Y. Sawai, M. Oe, K. Iida and Y. Kadobayashi, *Performance Evaluation of Inter-Domain IP Traceback*, IEEE ICT, 2003

[95] B. Al-Duwairi and G. Manimaran, *Novel Hybrid Schemes employing Packet Marking and Logging for Traceback*, IEEE TPDS, vol. 17, no.5, pp.403-418, 2005

[96] B. Al-Duwairi and T. Daniels, *Topology-based Packet Marking*, IEEE ICCCN, 2004

[97] A. Yaar, A. Perrig and D. Song, *StackPi: New Packet Marking Filtering Mechanisms for DDoS and IP Spoofing Defense*, JSAC, vol.24, no.10, pp.1853-1863, 2006

[98] A. Castelucio, A. Ziviani and R. Salles, *An AS-level overlay network for IP traceback*, IEEE Network, vol.23, no.1, pp.36-41, 2009

[99] B. Armbruster, J. Cole Smith and K. Park, *A packet filter placement problem with application to defense against spoofed denial of service attacks*, European Journal of Operations Research, vol.176, no.2, pp.1283-1292, 2005

[100] XianGhui Liu, Jianping Yin, Zhiping Cai and Shaohe Lv, *On the Placement of Active Monitor in IP Network*, LNCS, vol.3619, pp.1181-1187, 2005

[101] C. E. Shannon, *A mathematical theory of communication I*, Bell System Technical Journal, vol.27, pp.379-423, 1948

[102] C. E. Shannon, *A mathematical theory of communication II*, Bell System Technical Journal, vol.27, pp.623-656, 1948

[103] H. von Schelling, *Coupon Collecting for Unequal Probabilities*, American Mathematical Monthly, 1954

[104] S. Lu and S. Skiena, *Filling a Penny Album*, CHANCE, 2000

[105] N. Spring, R. Mahajan and D. Wetherall, *Measuring ISP Topologies with Rocketfuel*, SIGCOMM, 2002

[106] M. Muthuprasanna and G. Manimaran, *Space-Time Encoding Scheme for DDoS Attack Traceback*, IEEE GLOBECOM, 2005

[107] M. Muthuprasanna and G. Manimaran, *Traceback-Assisted Denial-of-Service Mitigation in Wireline-Wireless Networks*, IEEE INFOCOM Poster, 2005

[108] M. Muthuprasanna, G. Manimaran, M. Alicherry and V. Kumar, *Coloring the Internet: IP Traceback*, IEEE ICPADS, 2006

[109] M. Muthuprasanna and G. Manimaran, *Secure DVR Protocol using Factual Correctness*, IFIP NETWORKING, 2006

[110] K. Wei, M. Muthuprasanna and S. Kothari, *Preventing SQL Injection Attacks in Stored Procedures*, IEEE ASWEC, 2006

[111] M. Muthuprasanna, K. Wei, S. Kothari, *Eliminating SQL-Injection Attacks: A Transparent Defense Mechanism*, IEEE WSE, 2006

[112] M. Muthuprasanna, M. Alicherry and V. Kumar, *High Speed Pattern Matching for Network IDS/IPS*, IEEE ICNP, 2006

[113] M. Muthuprasanna, G. Manimaran and Z. Wang, *Unified Defense against DDoS Attacks*, IFIP NETWORKING, 2007

[114] M. Muthuprasanna, *Factual correctness guarantees to secure distance vector routing protocols*, Master's Thesis, 2007

[115] M. Muthuprasanna and G. Manimaran, *Distributed divide-and-conquer techniques for effective DDoS attack defenses*, IEEE ICDCS, 2008

[116] M. Muthuprasanna, *Detecting and Defending Against Denial Of Service Attacks*, *submitted to* USPTO, 2008

[117] M. Alicherry, V. Kumar, G. Manimaran and M. Muthuprasanna, *Two-Tiered Packet Labeling for Data Network Traceback*, USPTO #7619990, 2009

[118] M. Muthuprasanna and V. Balaji, *Scalable Workflow Design for Automated Service Management*, *submitted to* USPTO, 2009

[119] M. Muthuprasanna and G. Manimaran, *Optimizing the update packet stream for web applications*, ICST BROADNETS, 2010

[120] M. Muthuprasanna and G. Manimaran, *Anonymous Traceback of Network Data Packets*, *submitted to* TPDS, 2010

[121] M. Muthuprasanna and M. Alicherry, *Method and System for Multi-Character Multi-Pattern Pattern Matching*, USPTO #7725510, 2010

[122] M. Muthuprasanna, *Refreshing Application State based on User Migration Patterns*, *submitted to* USPTO, 2010

[123] M. Muthuprasanna, Matt Cutts and Paul Haahr, *Automatic extraction of sensitive personally-identifiable information from web pages*, *submitted to* USPTO, 2010

[124] E. Lloyd and S. Ramanathan, *On the complexity of distance-2 coloring*, IEEE ICCI, 1992

[125] Necessary and sufficient conditions, *http://en.wikipedia.org/wiki/Necessary_and_sufficient_condition*

[126] Sajal Das and Irene Finocchi, *Star-coloring of graphs for conflict-free access to parallel memory systems*, IEEE IPDPS, 2004

[127] S. T. McCormick, *Optimal approximation of sparse hessians and its equivalence to a graph coloring problem*, Mathematical Programming, 1983

[128] A. Gebremedhin, F. Manne and A. Pothen, *Parallel distance-k coloring algorithms for numerical optimization*, Euro-Par, 2002

[129] D. Bozdag, U. Catalyurek, A. Gebremedhin, F. Manne, E. Boman and F. Ozguner, *A parallel distance-2 graph coloring algorithm for distributed memory computers*, HPCC, 2005

[130] P. Mahadevan, D. Krioukov, M. Fomenkov, B. Huffaker, X. Dimitripoulos, K. Claffy and A. Vahdat, *The Internet AS-Level Topology: Three Data Sources and One Definitive Metric*, ACM SIGCOMM Computer Communications Review, 2006

[131] Distributed Lock Management, *http://en.wikipedia.org/wiki/Distributed_lock_manager*

[132] Comparative Analysis of Internet Topologies, *http://www.caida.org/research/topology/as_topo_comparisons/*

[133] Internet Mapping Project, *http://www.cheswick.com/ches/map/index.html*

[134] G. Malewicz, M. Austern, A. Bik, J. Dehnert, I. Horn, N. Leiser and G. Czajkowski, *Pregel: a system for large-scale graph processing*, ACM PODS 2009

[135] T. Cormen, C. Leiserson, R. Rivest and C. Stein, *Introduction to Algorithms*, MIT Press, 2001

[136] A. Fei, G. Pei, R. Liu and L. Zhang, *Measurements on Delay and Hop-Count of the Internet*, IEEE GLOBECOM, 1998

[137] C. Hamblin, *Translation to and from Polish Notation*, Computer Journal, vol.5, pp.210-213, 1962

[138] D. A. Huffman, *A method for the construction of minimum redundancy codes*, IRE 40, vol.10, pp.1098-1101, 1952

[139] V. Arya, T. Turletti and S. Kalyanaraman, *Encodings of Multicast Trees*, IFIP Networking, 2005

[140] CAIDA: Router-level Topology Measurements, *http://www.caida.org/tools/measurement/skitter/router_topology/*

[141] A. Serjantov and G. Danezis, *Towards an Information Theoretic Metric for Anonymity*, LNCS, vol.2482, pp.41-53, 2003

[142] M. Reiter and A. Rubin, *Crowds: Anonymity for Web Transactions*, ACM Transactions on Information and System Security, vol.1, no.1, pp.66-92, 1998

[143] M. Ripeanu, *Peer-to-peer architecture case study: Gnutella network*, International Conference on Peer-to-Peer Computing, 2001

[144] M. G. Reed, P. F. Syverson and D. M. Goldschlag, *Anonymous connections and onion routing*, IEEE JSAC, vol.16, no.4, pp.482-494, 1998

[145] A. Serjantov and P. Sewell, *Passive Attack Analysis for Connection-Based Anonymity Systems*, Computer Security, ESORICS, 2003

[146] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, Peter and D. Wallach, *AP3: cooperative, decentralized anonymous communication*, ACM SIGOPS European workshop, 2004

[147] M. Freedman and R. Morris, *Tarzan: a peer-to-peer anonymizing network layer*, CCS, 2002

[148] Z. Brown, *Cebolla: Pragmatic IP Anonymity*, Ottawa Linux Symposium, 2002