Handling Large-Scale Link-Flooding Attacks in the Internet

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Min Suk Kang

B.S., Electrical Engineering and Computer Science, KAIST, Korea
M.S., Electrical Engineering and Computer Science, KAIST, Korea

Carnegie Mellon University
Pittsburgh, PA

June, 2016

Dedicated to my wife Jinhee

Acknowledgments

First of all, I would like to express my utmost gratitude to my advisor, Professor Virgil Gligor. Having him as my advisor at Carnegie Mellon has been a blessing. His endless curiosity and academic enthusiasm have always inspired me and helped me to become a better researcher. He has shown me how to become a great advisor through countless examples.

I would also like to thank my thesis committee, Professors Adrian Perrig, Vyas Sekar, and Peter Steenkiste for numerous discussions, extensive advice, helpful feedback, and constant support. Also, my special thanks to Dr. Soo Bum Lee, who always has provided great insights and encouragements.

I have been so fortunate to learn from and work with many remarkable CyLab faculty members: Professors Lujo Bauer, David Brumley, Nicholas Christin, Anupam Datta, Limin Jia, and Dr. Maverick Woo. Also, I am indebted to my friends in CyLab over the years for all of the great discussions and support. Thank you, Saurabh, Miao, Zongwei, Jun, Yueqiang, Gihyuk, Seyed, Soo-Jin, Tianlong, Billy, Niranjini, Tiffany, Sang Kil, Hyoseung, and Rijnard.

In my masters years at KAIST, I was fortunate to work with my advisor, Professor Dan Keun Sung, who taught me many research skills and encouraged me to challenge myself to go beyond limits. I also thank Professor Bang Chul Jung for being a great mentor and a great collaborator during my years at KAIST Institute.

Lastly but most importantly, I would like to thank Jinhee, who has been with me at every moment of joy and struggle. Without your encouragement and enduring faith in me, I could not have completed this thesis.

The research in this thesis was supported in part by CyLab at Carnegie Mellon under contract W911NF-09-1-0273 from the US Army Research Office, the National Science Foundation under grant CNS1040801, the National Science Foundation under grant CCF-0424422, and a grant from PNC Bank. The views and conclusions contained in this document are solely those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Abstract

Link-flooding attacks in which an adversary coordinates botnet messages to exhaust the bandwidth of selected network links in the core of the Internet (e.g., Tier-1 or Tier-2 networks) have been a powerful means of denial of service. In the past few years, these attacks have moved from the realm of academic curiosities to real-world incidents. Unfortunately, we have had a limited understanding of this type of attacks and effective countermeasures in the current Internet. In this dissertation, we address this gap in our understanding of link-flooding attacks and propose a two-tier defense approach.

We begin by identifying *routing bottlenecks* as the major cause of the Internet vulnerability to link-flooding attacks. A routing bottleneck is a small set of links whose congestion disrupts the majority of routes taken towards a given set of destination hosts. These bottlenecks appear despite physical-path diversity and sufficient bandwidth provisioning in normal (i.e., non-attack) mode of operation, and are an undesirable artifact of the current Internet design. We illustrate their pervasiveness for adversary-chosen sets of hosts in various cities and countries around the world via experimental measurements. We then present a real-time adaptive attack for *persistent flooding* of chosen links in the discovered routing bottlenecks using attack flows that are indistinguishable from legitimate traffic. We demonstrate the feasibility of these strategies and show that disruptions can scale from targeted hosts of a single organization to those of a country.

To counter the link-flooding attacks defined in this dissertation, one could remove their root cause, namely the routing bottlenecks. However, this would affect the cost-minimizing policy that underlies the current Internet, change its routing architecture, and possibly affect communication costs. Instead, we propose an attack-deterrence mechanism that represents a *first line* of defense against link-flooding attacks by cost-sensitive adversaries. In the proposed defense, most link-flooding attacks are handled by the low-cost, single-domain based mechanism. As a *second line* of defense, which targets cost-insensitive adversaries that are undeterred, we propose the use of a multi-domain coordinated defense mechanism that is harder to orchestrate in the current Internet.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

As society becomes increasingly reliant on the Internet, *availability* of Internet services becomes increasingly critical. Lack of service availability can cause serious economic, safety, and national-security problems [45, 127, 163]. Experience of the past decade shows that Internet services can become unavailable due to a variety of causes, with the following four being predominant.

- **Failures.** The Internet comprises a vast number of electronic components (e.g., routers, switches, optical cables) whose failures often cause small disruptions of operations before recovery takes place; e.g., 50 msec [85]. Large-scale failures, such as disruptions caused by natural disasters (e.g., Northeast US Internet outage due to Hurricane Sandy [37]) may require a few hours to a day for recovery.
- **Human errors.** Network-topology and control-plane setups in large networks are highly complex and hence prone to operational errors [63]. A well-known example is the Youtube outage of 2008 when a clerical mistake of a Pakistani Internet Service Provider (ISP) caused a BGP misconfiguration that disrupted Youtube access for two-thirds of the world [107].
- **Disputes.** Most of the Internet is operated by independent ISPs whose business interests are sometimes misaligned. Sometimes ISPs may deliberately harm network services of competitor ISPs (e.g., throttling bandwidth or even discontinuing services) for commercial advantage in highly competitive markets [32, 153].

- **Attacks.** Network *attackers* can degrade Internet availability by *denying* legitimate-user access to selected services. *Denial-of-service* (DoS) attacks typically generate malicious traffic to overload targeted end-points servers, network bandwidth, or both. Often, the attack traffic is generated from compromised machines, or *bots*, that create significant numbers of attack flows. The goals of DoS attacks might vary widely. Some adversaries disrupt government services for political-propaganda reasons. Others aim to damage their competitors' business. Recently, a growing number of DoS attacks have been used as instruments of extortion [20]. Furthermore, criminals can use DoS attacks as smokescreens to cover up theft or fraud [84].

This dissertation addresses Internet availability problems caused by deliberate DoS attacks. In particular, we focus on the *network-layer flooding* attacks, one of the most common types of DoS attacks [19], where adversaries attempt to exhaust bandwidth resources by flooding network-layer devices with attack packets. Traditionally, denial of service attacks have flooded end-point services by sending a large volume of traffic *directly* to the chosen servers; e.g., by sending several Gbps of traffic, an adversary can flood the direct *access link* to the targeted server. Worse yet, with larger volumes of attack traffic, adversaries can flood the entire *upstream network* that provides Internet connectivity to the targeted servers [19].

This dissertation focuses on a *non-traditional* type of network-flooding attacks, where adversaries target routers/links in the *core* of the Internet; e.g., Tier-1 or Tier-2 networks. This type of non-traditional attacks have been considered to cause much greater damage than traditional attacks; viz., early academic studies [13, 144]. Also, from recent real-world attack incidents [31, 68], it appears that the current Internet is vulnerable to such non-traditional attacks in practice.

However, we have a limited understanding of this type of non-traditional attacks and their countermeasures. For example, on the one hand, it has not been thoroughly investigated why the Internet, which has rich physical connectivity with ample redundancy, can suffer from massive connectivity degradation when a few routers/links in its core are flooded. Also, it is still unknown whether such attacks can disconnect a particular set of *adversary-chosen* hosts from the Internet, as opposed to disrupting unspecified Internet hosts [144]. On the other hand, it is also unknown whether there exists any countermeasure that protects the Internet from such attacks by removing any underlying vulnerability. Additionally, comprehensive studies have not yet been conducted on the desired properties of the Internet to counter non-traditional at-

tacks, particularly the properties that can be readily implemented in the current Internet. This dissertation addresses this gap in our understanding about the non-traditional attacks and readily-available countermeasures for the current Internet.

1.1 Link-Flooding Attacks

In this dissertation, we call these non-traditional attacks, which flood network links in the core of the Internet (e.g., IP backbone links in Tier-1 or Tier-2 ISPs) to degrade the communication of end-point servers, the *link-flooding attacks*. These attacks are *indirect* since the locus of the attack (i.e., flooded links) is different from the ultimate target; e.g., end-point servers. Recently, we have witnessed a noticeable increase in this type of attacks, which brings added real-life motivation to the research reported herein.

Research interest in link-flooding attacks appeared nearly two decades ago when Albert *et al.* conducted a theoretical study on the topological vulnerability of the Internet to autonomous system (AS) removal [13]. These authors assumed a large-scale attack that can remove several ASes entirely from the Internet and found the conditions under which the Internet would break up into smaller isolated pieces. Although AS removal attacks are subjects of only theoretical interest, they vividly illustrated the enormous potential damage that can be caused by large-scale, link-flooding attacks.

In 2004, an Internet worm, known as SQL Slammer, showed that link-flooding attacks against routers in the core of the Internet can cause severe damage. This worm spread rapidly to tens of thousand victim machines within ten minutes, creating extremely high traffic volume that originated from infected machines and inadvertently flooding numerous Internet routers [132]. Slammer caused denial of service in some Internet hosts and dramatically slowed down general Internet traffic.

In 2009, Studer and Perrig illustrated the first general link-flooding attack, called the Coremelt attack [144]. This attack demonstrated how a set of bots can send packets to each other and flood a set of backbone routers, without requiring unwanted messages. As a consequence, Coremelt eludes all defense mechanisms that filter unwanted traffic.

In the past few years, the link-flooding attacks have quickly moved from the realm of academic curiosities to real-world settings. We recently have witnessed two real-world incidents: an attack against the Spamhaus service in 2013 and another one against the ProtonMail service in 2015. Unlike previous link

flooding incidents (e.g., congestion due to SQL Slammer), these attacks *intentionally* targeted and flooded links in the Internet core.

Spamhaus. On March 16, 2013, an individual launched a denial-of-service attack against the *Spamhaus* web server[1] with attack traffic that peaked at about 10 Gbps. The initial attack successfully disrupted the web service for about two days until Spamhaus begun to use a content distribution network (CDN) service, CloudFlare[2], on March 19th. As soon as the content of the Spamhaus service was cached in multiple (i.e., 13) distributed cloud servers, the initial attack became ineffective. CloudFlare used the *anycast* mechanism to distribute the attack flows to the datacenters that are closer to the traffic generators and effectively diffused them. After few days later, the same adversary changed his strategy and began to flood the four Internet exchange points (IXPs) that provided connectivity to some of the CloudFlare data centers. This second attack used DNS-based reflection and traffic amplification to generate 350 Gbps of attack traffic. As a result of the indirect flooding attack, the Spamhaus service along with many other services protected by CloudFlare suffered regional service disruptions, particularly in some European countries. Despite the large-scale connectivity damage, this IXP flooding attack was countered after only a few hours mainly because the intense attack traffic was directly sent to the target routers and thus it became easily distinguished from legitimate traffic and dropped [31, 106].

ProtonMail. In November 2015, a hacking group called the Armada Collective launched a direct server-flooding attack against the data center of *ProtonMail*[3] – an email service located in Switzerland. The attack perpetrators asked for ransom in exchange for stopping the attack, which was paid by the company running ProtonMail shortly after the attack began. Although the server flooding stopped, a much larger-scale attack was launched by an unknown adversary almost immediately after the first attack ended.

The second attack targeted several network links connecting multiple ISPs that provide Internet connectivity to ProtonMail. The adversary carefully chooses these link targets so that the flooding disrupted most user traffic to ProtonMail. Although details of the second-attack strategy remain largely unknown, all evidence suggests that the attack flows have *not* been easily distinguishable from legitimate flows, unlike

[1] `https://www.spamhaus.org`

[2] `https://www.cloudflare.com`

[3] `https://www.protonmail.com`

in the case Spamhaus attack. As a consequence, the targeted ISPs and ProtonMail had to spend seven days to find countermeasures and still failed to identify all the attack traffic. The attack was finally countered by significant ISP-level network topology changes[4] and purchasing an on-demand cloud-based scrubbing service.

1.2 Challenges of Countering Link-Flooding Attacks

The fact that the targets of link-flooding attacks are not end-point servers makes it difficult to utilize many existing defense mechanisms already installed at these servers; e.g., firewalls, IDSs. Hence, countermeasures to link flooding need to be implemented primarily at the Internet core where these attacks could, at least in principle, be handled. We identify three fundamental challenges of countering link-flooding attacks in Sections 1.2.1 through 1.2.3 below.

1.2.1 Indistinguishable Attack Flows

From the point of view of an adversary seeking to launch a link-flooding attack, a desirable property of attack traffic is *flow indistinguishability*; i.e., the traffic characteristics of attacks flows are so similar to those of legitimate flows that a defender cannot distinguish attack flows legitimate ones. The first challenge a defender faces is that an adversary can craft indistinguishable link-flooding flows much more easily than server-flooding flows. The main reason for this adversary advantage is the inherent difference between the router and end-point server functions. That is, routers are supposed to convey *all* Internet traffic while end-point servers usually are intended to receive *only* certain types of traffic; e.g., web servers expect to see mostly web traffic. Therefore, it is much harder for routers to define and filter out protocol *non-conforming* or *unwanted* traffic than end-point servers. For example, the Coremelt attack generates only wanted flows that need not be protocol-conforming. Bots can collaborate to flood backbone links via any private protocol and their flows can remain indistinguishable from legitimate traffic in routers [144]. Another example, which we provide in Chapter 3, shows that an adversary can use different attack capabilities to generate

[4]ProtonMail simplified its Internet infrastructure by exclusively connecting to a large Tier-1 ISP. However, this appears to be insufficient in general since this network configuration can still be flooded by more advanced attacks since links of Tier-1 ISPs may still become effective targets for link-flooding attacks; viz., Chapter 2.

indistinguishable flows; e.g., by utilizing low-rate, protocol-conforming flows.

When attacks use indistinguishable flows, handling link flooding at a target router reduces to a *resource-sharing problem*, where multiple indistinguishable resource requesters (i.e., both legitimate and malicious) contend for the same resource; i.e., the network link bandwidth. In this case, the operation of any requester becomes dependent upon the operation of other – often malicious and unknown – requesters of that resource. The existence of this type of *undesirable dependency* among requesters is the necessary condition for all denial of service in resource-sharing problems [66], and can be countered only by enforcing *agreements* among requesters (i.e., constraints placed on requester behavior) *outside* the shared-resource service [170]. In contrast to requester-agreement schemes at the application layer (e.g., cryptographic client puzzles [160], ticket-based rate control [67]), establishing agreements at the network layer is much more challenging to implement. For example, adding requester-agreement functionality to the IP protocol requires significant modifications to large portions of router and host design of the current Internet; e.g., 'congestion puzzles' in the IP layer [161]. Furthermore, verification of these agreements in backbone routers would undoubtedly affect line-rate performance due to the large traffic volume processed in the backbone.

1.2.2 Cost Asymmetry

Whenever a countermeasure to link flooding reduces to finding a solution to a resource-sharing problem (viz., Section 1.2.1), the cost of the resource for both adversaries and defenders (i.e., the cost of targeted routers bandwidth) becomes a key factor in determining the effectiveness of both attacks and defenses. For example, if the cost of generating attack traffic (i.e., the cost of the shared resource requests) is extremely high, or if the cost of available bandwidth (i.e., resource provisioning) at the target network (i.e., resource manager) is negligible, attacks would become very unattractive.

Unfortunately, in the current Internet, the opposite cost relation prevails, which makes link-flooding attacks very attractive. That is, the cost of bandwidth for generating attack traffic is *orders of magnitude* lower than that of provisioning backbone-link bandwidth; viz., Chapter 4. In other words, a severe *cost asymmetry* exists that favors the adversary over the defender. Furthermore, removing this cost asymmetry is not only a matter of changing the Internet design. Instead, whether the asymmetry can be removed

depends on two independent markets: the botnet markets and backbone bandwidth markets. The former is an underground online e-commerce market [34], where bot buyers can demand and sellers supply attack bots, whereas the latter is a legitimate network-infrastructure market, where many entities compete by well-established rules that determine the market price of backbone bandwidth.[5]

We note that removing the cost asymmetry, and even reversing it to favor the defender, does *not* completely deter link-flooding attacks; e.g., cost-insensitive adversaries, such as those sponsored by a state, could still launch link-flooding attacks. However, it would change today's severely imbalanced cost structure and would certainly deter cost-sensitive (e.g., rational) adversaries. Hence, it would yield an effective first line of defense, as argued in Chapter 4.

1.2.3 A Defender's Dilemma

Many link-flooding attacks rely on the existence of *a few* link targets whose congestion would disrupt the majority of routes that pass the Internet core from a set of sources to a set of destination hosts. We call these links the *routing bottleneck* of a set of sources and destinations, and we show that its existence is an *undesirable artifact* of Internet design. Although in the attack-free mode of operation these bottlenecks are not an operational hazard, we seek to remove them since they can constitute an Internet vulnerability in the presence of a link-flooding adversary.

However, as we show in Chapter 2, removing routing bottlenecks to prevent link flooding is impractical in the current Internet because they are the result of employing a *cost-minimizing* (or revenue-maximizing) policy of the Internet routing and topology designs. In other words, the source of many link-flooding vulnerabilities is, in fact, a very *desirable* feature of the Internet business model. Hence, a defender faces the following dilemma: *how can one remove a vulnerability of a system when it is caused by a very desirable feature of the system's design and operation?*

As long as the causality between a route-cost minimization policy and the existence of flooding targets holds, any attempt to remove the latter would necessarily affect the former. However, in the highly competitive Internet transit markets ISPs would naturally be very reluctant to adopt any countermeasure

[5]The market involves many layers of businesses, including equipment companies, optical cable companies, undersea cable companies, Internet exchange points (IXPs), etc.

that would increase route cost.

1.3 Future Internet Designs for Potential Solutions

We briefly review future Internet designs that could address the challenges of link-flooding attacks, and are the subject of on-going research. In the last decade, several architectures have been proposed for more flexible, extensible, and secure future Internet [40, 72, 82, 118, 129, 173]. These "clean-slate" designs have substantial merit in solving many inherent problems of the current Internet, albeit at higher expected transition costs then mere modifications of the existing Internet. Several clean-slate design features have been proposed to address the challenges of link-flooding attacks mentioned above. The brief review of these new features helps us understand some of the limitations of the current Internet and highlight the potential benefits future designs.

- **Route controllability.** Currently, sources or destinations of the Internet have virtually no control over the routes taken by routers in the middle of the Internet. Results presented in Chapter 2 suggest that a certain level of route control (i.e., choices of routers on a path) at sources and destinations could be useful to remove the cause of many link-flooding attacks [173].
- **Source and path authentication.** The current Internet inherently lacks source and path authentication mechanisms [17]. Thus, it is hard to detect source IP spoofing accurately at the core of the current Internet. A source/path authentication mechanism embedded in the future Internet architecture (e.g., [120, 167]) would be useful for accurate attack traffic identification, which is a necessary condition for strong bandwidth guarantees to legitimate users [92].
- **Coordinated defense.** Currently, interactions among autonomous systems (ASes) are highly limited and static. Dynamic and global-scale AS coordination in the future Internet would be useful to counter large-scale, link-flooding attacks by cost-insensitive adversaries. In particular, when attacks flood multiple link targets in different ASes, a coordinated defense operation at multiple ASes becomes necessary to identify the ultimate target servers; viz., Chapter 3. A working group called DDoS Open Threat Signalling (DOTS), formed in 2015 by IETF, can motivate the technical discussions on more active AS coordinations for DoS mitigations [111]. However, a more challenging task

is to create strong incentive models that could fuel global AS coordination. For example, the creation of entire end-to-end Internet paths to protect a user's traffic has been proposed recently [26]. However, establishing tangible economic models that demonstrably justify the necessity of global AS coordination remains as an open problem.

1.4 Thesis Overview

The brief analysis of the link-flooding attacks presented above suggests the following thesis statement:

Thesis statement. *Inherent Internet design features and economic factors render it vulnerable to a large class of link-flooding attacks that can persistently degrade host connectivity in targeted areas; e.g., cities, countries. Despite this basic vulnerability, we can implement a two-tier defense approach for handling link-flooding attacks: (1) a low-cost, first-line defense deters attacks by cost-sensitive adversaries and (2) a second-line defense handles cost-insensitive adversaries by using multi-domain coordinated defenses that are harder and more expensive to orchestrate and deploy.*

1.4.1 Outline

Understanding inherent vulnerability. Initially, we focus on understanding the vulnerability of the current Internet to link-flooding attacks. In Chapter 2, we identify routing bottlenecks for hosts in various cities and countries around the world, and show that they render the current Internet vulnerable to link flooding when they become the targets of attacks. In addition to routing bottlenecks' pervasiveness, we also explain their root causes and characteristics through Internet-scale measurements. Our root-cause analysis reveals that routing bottlenecks are caused by a desirable feature of the Internet business, which suggests the defender's dilemma defined above.

Large-scale persistent link-flooding attack. Knowledge of routing bottlenecks, although useful, does not provide a full understanding of the link-flooding attacks. In order to gain a full understanding, we need to investigate the attack that can flood the network links in the discovered routing bottlenecks. The generated attack flows must be indistinguishable from legitimate ones to make the attack undetectable at the targeted routers. Also, the attack must take into account any possible reactions by the targeted

routers (e.g., reactive re-routing) to maintain its effectiveness potentially indefinitely. Thus, in Chapter 3, we propose and demonstrate a real-time adaptive attack called Crossfire that persistently degrades the connectivity of any given target geographic area by flooding link targets in the routing bottlenecks.

A first line of defense. Finally, in this dissertation, we propose a solution to the defender's dilemma defined above. That is, instead of removing the routing bottleneck vulnerability completely from the Internet by redesigning the underlying routing architecture, which would involve a significant transition cost, we propose a *first line of defense* mechanism that can be applied to the current Internet. In the proposed defense in Chapter 4, most link-flooding attacks are handled by a low-cost, single-AS based deterrence mechanism in the current Internet. This way, only attacks that are still not deterred need to be handled by higher-cost, multiple-AS coordinated defense mechanisms, which may require changes to the Internet architecture. This *two-tier* defense approach enables us to handle link-flooding attacks readily in the current Internet at low cost, and complement higher-cost defenses in future Internet routing architectures.

Chapter 2

Routing Bottlenecks in the Internet: Inherent Vulnerability to Link-Flooding Attacks

2.1 Outline of the Chapter

Several academic investigations [13, 144] and real-life attacks [31, 68] have offered concrete evidence that link-flooding attacks can severely degrade the connectivity of large numbers of hosts in the Internet. However, neither the root cause nor pervasiveness of this vulnerability has been analyzed to date. Furthermore, it is unknown whether certain network structures and geographic regions are more vulnerable to these attacks than others. In this chapter we address this gap in our knowledge about these attacks by (1) introducing the notion of the *routing bottlenecks* and its role in enabling link-flooding attacks at scale; (2) finding bottlenecks for hosts in 15 countries and 15 cities distributed around the world to illustrate their pervasiveness; and (3) measuring bottleneck parameters (e.g., size, link types, and distance to adversary-selected hosts) to understand the magnitude of attack vulnerability. We also discuss both structural and operational countermeasures and their limitations.

In principle, route diversity could enhance Internet resilience to link-flooding attacks against *large sets* of hosts (e.g., 1,000 hosts) since it could force an adversary to scale attack traffic to unattainable

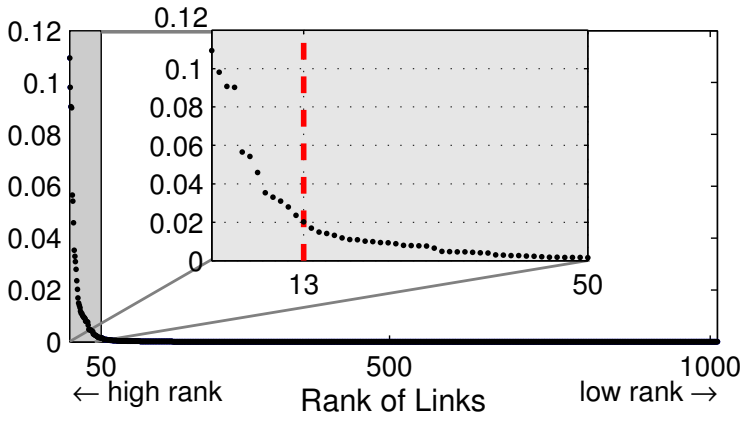

Figure 2.1: Normalized route-count distribution in routes from S = 250 PlanetLab nodes to D = 1,000 randomly selected servers in Iran

levels to flood all possible routes. In practice, however, the mere existence of many routes between traffic sources and selected sets of destination hosts cannot guarantee resilience whenever the vast majority of these routes are distributed across very few links, which could effectively become a routing bottleneck.

To define routing bottlenecks more precisely, let S denote a set of (source) IP addresses of hosts that originate traffic to a set of IP destination addresses, denoted by D. S represents any set of hosts distributed across the Internet. In contrast, D represents a set of hosts of a specified Internet region (e.g., a country or a city), which are chosen at random and independently of S. A *routing bottleneck* on the routes from S to D is a small set B of IP (layer-3) links such that B's links are found in a majority of routes whereas the remaining links are found in very few routes. $|B|$ is often over an order of magnitude smaller than both $|S|$ and $|D|$. If all links are ranked by the number of routes between S and D that traverse each link, the bottleneck links, B, have a very high rank whereas the vast majority of the remaining links have very low rank. The sharper the skew in the route counts of all the links, the narrower the bottleneck. Note that a routing bottleneck is defined for the routes *from S to D* but *not* necessarily vice versa due to Internet route *asymmetry* [73].

An Example. To illustrate a real routing bottleneck, we represent route sources S by 250 PlanetLab nodes [125] distributed across 164 cities in 39 countries. For the route destinations, D, we select 1,000 web servers in one country (i.e., *Iran* in this example) at random from a list of publicly-accessible servers obtained using the 'computer search engine' called Shodan (`http://www.shodanhq.com`). We trace

the routes between S and D for this country, calculate the number of routes that traverse each unique link IP address we see in the routes, and plot their route-count distribution, as shown in Fig. 2.1. This figure clearly shows a very skewed route-count distribution, which implies the existence of a narrow routing bottleneck; i.e., 72% of the routes are found in only $|B| = 13$ links; viz., Fig. 2.4.2.

In this chapter we argue that the pervasive occurrence of routing bottlenecks is a fundamental property of the Internet design. That is, power-law distributions that characterize the route-count distribution are a consequence of employing route-cost minimization, which is a *very desirable* feature of Internet routing; viz., Section 2.2. Fortunately, routing bottlenecks do *not* lead to traffic degradation during ordinary Internet use, because the bandwidth of bottleneck links is usually provisioned adequately for normal mode of operation. Hence, these bottelnecks should not be confused with the bandwidth bottlenecks in end-to-end paths [11, 76] since one does not always imply the other; viz., Section 2.6.3.

Problem. Unfortunately, however, bottleneck links provide a very attractive target to an adversary whose goal is to flood few links and severely degrade or cut off connectivity of targeted servers, D, in various cities or countries around the world. For example, an adversary could easily launch a traffic amplification attack using NTP monlists (400 Gbps) [108] and DNS recursors (120 Gbps) [31] to distribute an aggregate of 520 Gbps traffic across the 13-link bottleneck of Fig. 2.1. Such an attack would easily flood these links, even if each of them is provisioned with a maximum of 40 Gbps capacity, severely degrading the connectivity of the 1,000 servers of a country from the Internet; viz., Fig. 2.4.2. As we will discuss later in Chapter 3, more insidious attacks can flood bottleneck links persistently with attack traffic that is indistinguishable from legitimate traffic by routers and invisible to, and hence undetectable by, the targeted servers, D.

To counter link-flooding attacks that exploit routing bottlenecks, we first define the parameters that characterize these bottlenecks; e.g., size, link types, and average distance of bottleneck links from the targeted servers, D. Then we define a *connectivity-degradation metric* to provide a quantitative view of the risk exposure faced by these servers. The bottleneck parameters and metric are particularly important for applications in the targeted country or city where Internet-facing servers need stable connectivity; e.g., industrial control systems [47], financial [143], defense and other government services. We illustrate the usefulness of our connectivity-degradation metric in assessing the vulnerabilities posed by real life routing

bottlenecks found for hosts in fifteen countries and fifteen different cities around the world; viz., Section 2.

Analysis of routing-bottleneck exploits explains why intuitive but naive countermeasures will *not* work in practice; e.g., reactive re-routing to disperse the traffic flooding bottleneck links across multiple local links; flow filtering at routers based on traffic intensity; reliance on backup links on exposed routes. More importantly, our analysis provides a *route-diversity metric*, which is based on autonomous-system (AS) path diversity, and illustrates the utility of this metric as a proxy for the bottleneck avoidance in the Internet. Finally, we discuss operational countermeasures against link-flooding attacks, including inter- and intra-domain traffic engineering, and their limitations.

Contributions. In summary, we make the following contributions:

- We explain the root causes and characteristics of routing bottlenecks in the Internet, and illustrate their pervasiveness with examples found in 15 countries and 15 cities around the world.
- We present a precise quantitative measure of connectivity degradation to illustrate how routing bottlenecks enable an adversary to scale link-flooding attacks without much additional attack traffic.
- We present several classes of countermeasures against attacks that exploit routing bottlenecks, including both structural and operational countermeasures.

2.2 Routing Bottlenecks

2.2.1 Route-count measurements

To determine the pervasiveness of routing bottlenecks, we investigate the routing bottlenecks for hosts in 15 countries and 15 cities, selected by the following criteria. For the list of countries, we select 15 countries from the union of the two lists of countries: the top countries with largest IPv4 address allocation and the top attack-traffic originating countries [10]. Table 2.1 shows the list of 15 countries we measure their routing bottlenecks, where they are ordered by increasing route-count skew. Note that for the United States and China (which are the two highest-ranked countries at the both lists of countries [10]) we measure the routing bottlenecks of their major cities instead of the entire countries for more fair comparison with other countries. For the list of cities, we select 15 cities, five major cities from each of the three

Index	Country	Index	City
Country1	United Kingdom	*City1*	London
Country2	Brazil	*City2*	New York
Country3	France	*City3*	Berlin
Country4	Germany	*City4*	Rome
Country5	Italy	*City5*	Los Angeles
Country6	Russia	*City6*	Moscow
Country7	Turkey	*City7*	Beijing
Country8	India	*City8*	Paris
Country9	Japan	*City9*	Shenzhen
Country10	Taiwan	*City10*	Chicago
Country11	South Korea	*City11*	Guangzhou
Country12	Israel	*City12*	Philadelphia
Country13	Romania	*City13*	Shanghai
Country14	Egypt	*City14*	Tianjin
Country15	Iran	*City15*	Houston

Table 2.1: List of 15 countries and 15 cities use for route-count measurement.

groups: the United States, Europe, and China. In particular, we select the top five cities by population in the United States and China. Table 2.1 also shows the list of 15 cities, ordered by increasing route-count skew.

We measure the route-count distribution in a large number of the routes towards a selected destination region. We perform *traceroutes* to obtain a series of *link samples* (i.e., IP addresses at either end of layer-3 links) on a particular route from a source host to a destination host in a selected Internet region. From the collected link samples on the routes, we construct the route-count distribution by counting the number of routes for each link. Then we select the minimum set of links whose removal disconnects all routes to the destination region by removing redundant links. Section 2.4.1 describes the selection algorithm in detail. In these measurements, we trace 250,000 routes by using *traceroute* from 250 source hosts (i.e., 250 PlanetLab nodes [125]) to 1,000 randomly selected web servers in each of 15 countries and 15 cities.

Traceroute is a common network monitoring tool whose use is often fraught with pitfalls [141]. Care was taken in analysing the *traceroute* dataset so that our measurement results are not affected by the typical errors of *traceroute* use; e.g., alias resolution, load-balanced routes, accuracy of returned IP, hidden links in MPLS tunnels. For a detailed discussion, see Section 2.3.3. We perform multiple *traceroutes* for the same source-destination host pair to determine the *persistent* links; i.e., links that always show up in the multiple *traceroutes*. We collect only the samples of persistent links because non-persistent links do not lead to reliable exploitation of routing bottleneck. We have found extremely skewed route-count distribution for the 1,000 randomly selected hosts in each of the 15 countries and 15 cities, which strongly indicates the existence of routing bottlenecks for hosts in all the countries and cities in which we performed our measurements.

2.2.2 Power-law in route-count distributions

The analysis of route-count distributions helps us understand both the cause of routing bottlenecks and their physical characteristics (e.g., size, type, distance from destination hosts) as well as countermeasures against flooding attacks that attempt to exploit them. To illustrate the skew of route-count distributions, we present our measurements for 15 countries and 15 cities around the world in Fig. 2.2.2 and Fig. 2.2.2, respectively. In these figures, we illustrate the relation between the route count normalized by the total

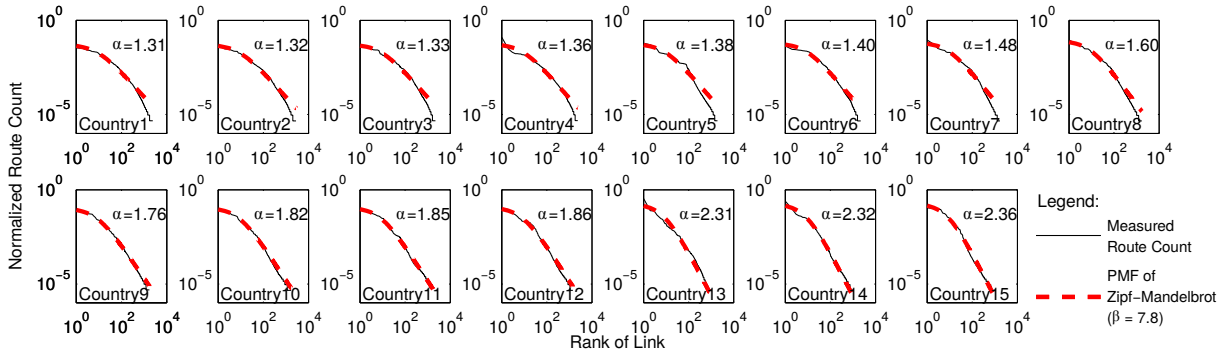

Figure 2.2: Normalized route count/rank in traced routes to 1,000 randomly selected hosts in each of the 15 countries.

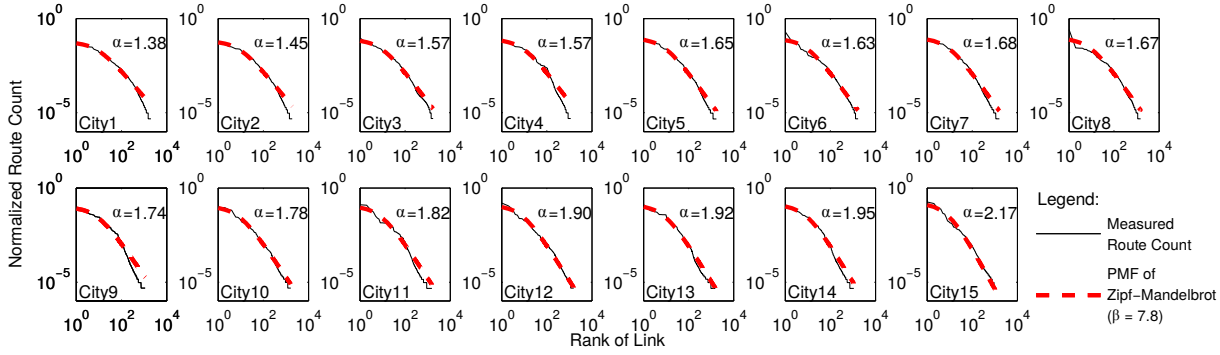

Figure 2.3: Normalized route count/rank in traced routes to 1,000 randomly selected hosts in each of the 15 cities.

number of measured routes and the rank of links on *log-log* scale, for 1,000 servers in each country and city. The normalized route count of a link is the portion of routes between S and D carried by the link; e.g., if a link carries 10% of routes between S and D, its normalized route count is 0.1.

We observe that the normalized route-count distribution is accurately modeled by the *Zipf-Mandelbrot* distribution; namely,

$$f(k) \sim 1/(k+\beta)^{\alpha},$$

where k is the rank of the link, α is the exponent of the power-law distribution, and β is the fitting parameter. Exponent α is a good measure of route concentration, or distribution skew, and hence of bottleneck size: the higher α, the sharper the concentration of routes in a few links. Fitting parameter β captures the flatter portions of the distribution in the high-rank region; i.e., lower values on the x-axis. This

region is not modeled as well by an ordinary Zipf distribution since its probability mass function would be a straight line in *log-log* scale on the entire range. The flatter portion of the distribution in high-rank region is due to the nature of link sampling via route measurement. That is, multiple links are sampled together when each route is measured and there exist no duplicate link samples in a route in general due to the loop-freeness property of Internet routes. Thus, the occurrences of extremely popular links are limited and the high-rank region is flattened. (Similarly flattened occurrence of high-ranked data samples was observed and explained in other measurements and modeling studies [70].)

To enable comparison of route concentration in a few links of different destination regions, we fix the fitting parameter β and find the values of exponent α for the best fit across the fifteen countries; i.e., $\beta = 7.8$ causes the smallest fitting error. In Fig. 2.2.2 and Fig. 2.2.2, the fifteen countries and cities are ordered by increasing value of α in the range 1.31 – 2.36.

2.2.3 Causes

What causes routing bottlenecks, or high skew/power-law distribution of route count? Often, power-law distributions (especially the Zipf-Mandelbrot distribution) arise from processes that involve some cost minimization. For example, research in linguistics shows that power laws defining the frequency of word occurrences in random English text arise from the minimization of human-communication cost [101, 155]. Thus, one would naturally expect that power-laws in route-count distributions are caused by the *cost minimization* criteria for route selection and network design in the Internet; i.e., both intra- and inter-domain interconnections and routing. Extra cost minimization is provided by the "hot-potato" routing between domains.

Cost minimization in inter-domain routing

Inter-domain routing policy creates routing bottlenecks in inter-AS links: BGP is the *de facto* routing protocol for inter-domain (i.e., AS-level) Internet connectivity. The *rule-of-thumb* BGP policy for choosing inter-AS paths is the minimization of the network's operating cost. That is, whenever several AS paths to a destination are found, the minimum-cost path is selected; e.g., customer links are preferred over peer links and over provider links. If there exist multiple same-cost paths, the shortest path is selected. This

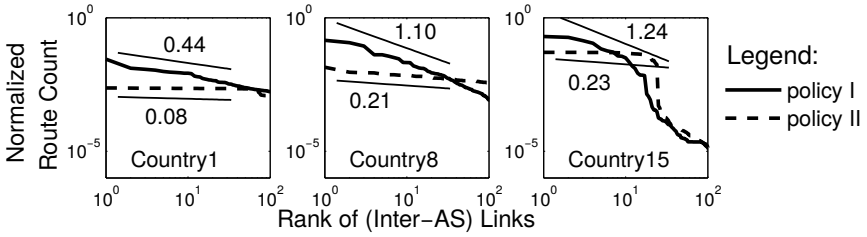

Figure 2.4: Normalized route count/rank in simulated inter-AS links in three countries.

policy is intended to minimize operating costs of routing in the Internet [61, 64].

To determine whether the rule-of-thumb routing policy (i.e., policy I) contributes to the creation of routing bottlenecks, we compare its effects with those of a *hypothetical* routing policy that distributes routes uniformly across possible inter-domain links (i.e., policy II). This hypothetical policy favors inter-domain links that serve fewer AS paths for a particular destination. To perform this comparison, we run AS-level simulations using the most recent (i.e., June 2014) CAIDA's AS relationship, which is derived by Luckie *et al.* [96]. We simulate the hypothetical policy using Dijkstra's shortest-path algorithm [48] with dynamically changing link weights, which are proportional to the number of BGP paths served.

Fig. 2.4 shows the normalized route count/rank plots for inter-AS links when we create BGP paths from all stub ASes to the ASes in *Country1*, *Country8*, and *Country15* according to the two BGP policies. To clearly see the different skew of the route count distribution of the two policies, we measure the slopes of link distributions in *log-log* scale in the high-rank region. Since the route-count distribution of policy II is not modeled by Zipf-Mandelbrot distribution, we simply measure the slope in the high rank region to compare the skew. *Country1* has a barely observable skew in this region (i.e., the slope is less than 0.1) with policy II while it has a much higher skew (i.e., a slope of 0.44) with policy I. *Country8* and *Country15* have small skews (i.e., slopes of 0.21 – 0.23) with policy II and much higher skews of 1.10 – 1.24 with policy I. This suggests that, even though inter-domain Internet topology may have no physical bottlenecks (or very few, as in *Country8* or *Country15*), the BGP cost-minimization policy creates inter-domain routing bottlenecks.

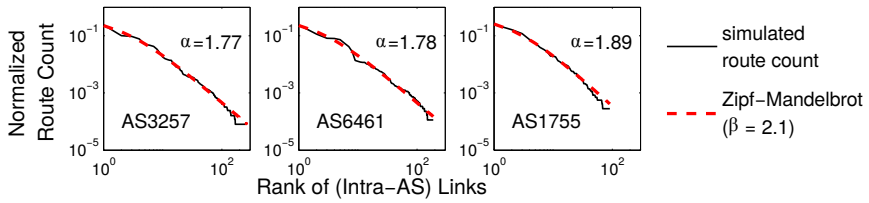

Figure 2.5: Normalized route count/rank in simulated AS-internal routes for three ASes.

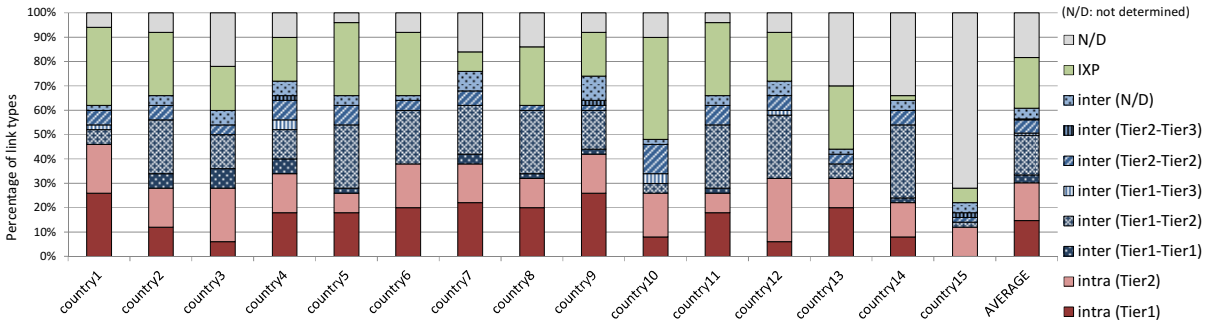

Figure 2.6: Percentage of link types of the 50 most occurred links for each of the 15 countries. Three link types (i.e., intra-AS links, inter-AS links, and IXP links) and three AS types (i.e., Tier-1, Tier-2, and Tier-3) are used for categorization.

Cost minimization in intra-domain network topology and routing

Internal AS router-level topology creates intra-domain routing bottlenecks: Most ISPs build and manage hierarchical internal network structures for cost minimization reasons [95, 139] and these structures inherently create routing bottlenecks within ISPs. An ISP is composed of multiple points of presence (or PoPs) in different geographic locations and they are connected via few high-capacity *backbone* links. Within each PoP, many low-to-mid capacity *access* links connect the backbone routers to the border routers.

In general, ISPs aim to minimize the number of expensive long-distance high-capacity backbone links by multiplexing as much traffic as possible at the few backbone links; viz., HOT network model in [95]. As a result, backbone links naturally become routing bottlenecks. To show this, we carry out simulations using Tier-1 ISP topologies inferred by Rocketfuel [139]. We construct ingress-egress routes for all possible pairs of access routers using shortest-path routing [100]. Fig. 2.5 shows the simulated normalized route count/rank for the three ASes belonging to different ISPs. In all three ASes, we find that

the Zipf-Mandelbrot distribution fits accurately for high value of skew α (i.e., 1.77 – 1.89) when β is deliberately fixed to 2.1 to yield a best fit and allow direct skew comparison. That is, a few AS internal links are extremely heavily used whereas most other internal links are very lightly used. Moreover, most of the heavily used links (i.e., 70%, 70%, and 90% of 10 most heavily used links in each of the three ISPs, respectively) are indeed backbone links that connect distant PoPs. We reconfirm the prevalence of intra-domain bottleneck links later in Section 2.2.5 where we find that a large percentage (i.e., 30%) of links in routing bottlenecks are intra-AS links.

Hot-potato routing policy in ISPs aggravates inter-domain routing bottlenecks: The *hot-potato* routing policy is another example of a cost-minimization policy used by ISPs; i.e., this policy chooses the closest egress router among multiple egress routers to the next-hop AS [148]. As already reported [154], this policy causes a load imbalance at multiple inter-AS links connecting two ASes and thus aggravates the routing bottlenecks at the inter-AS links.

2.2.4 Characteristics of Bottleneck Links

In this subsection we investigate the characteristics of the links in the routing bottlenecks in terms of link types (e.g., intra-AS links, inter-AS links, or IXP links) and distance from the hosts in the target region (e.g., average router and AS hops) as a backdrop to the design of countermeasures against attacks that exploit bottleneck links. Our investigation suggests that the variety of link types found and their distribution make it *impractical* to design a single 'one-size-fits-all' countermeasure. Instead, in Section 2.5, we discuss several practical countermeasures that account for the specific bottleneck link types.

2.2.5 Link types

We consider three link types based on their roles in the Internet topology: intra-AS links, which connect two routers owned by the same AS; inter-AS links, which connect routers in two different ASes; and IXP links, which connect routers of different ASes through a switch fabric. Although the link types are clearly distinguished in the above definitions, the determination of link types via *traceroute* is known to be surprisingly difficult and error prone due to potential inference ambiguity [103]. For example, the *AS boundary ambiguity* [103] arises because routers at AS boundaries sometimes use IPs borrowed from their

neighbor ASes for their interfaces. This is possible because the IPs at the both ends of the inter-AS links are in the same prefix. Borrowed IPs make it difficult to determine whether a link is an intra- or inter-AS link.

Our method of determining link type eliminates the AS boundary ambiguity by utilizing route diversity at the bottleneck links. Unlike previous measurements and analyses [76, 103], we measure a large number of disjoint incoming/outgoing routes to/from a bottleneck link. In other words, we gather all visible links 1-hop before/after the bottleneck link, and this additional information helps us infer the link types at AS boundary without much ambiguity.[1]

Fig. 2.6 summarizes the percentage of the link types of the 50 most frequently found links for each of the 15 countries. The average percentage of all 15 countries is presented in the rightmost bar. Notice that the intra-AS and the inter-AS links are further categorized by the AS types; i.e., Tier-1, Tier-2, and Tier-3 ASes. The list of Tier-1 ASes is obtained from the 13 selected ASes in Renesys' Baker's Dozen.[2] ASes that have no customer but only providers or peers are Tier-3 ASes. The rest of the ASes are labeled as Tier-2 ASes.

Our investigation found two unexpected results. The first is that the intra-AS links are a major source of routing bottlenecks; see the rightmost bar in Fig. 2.6 where approximately 30% of routing bottleneck links are intra-AS links while the other 30% and 20% are inter-AS links and IXP links, respectively. (The balance of 20% is not determined due to lack of *traceroute* visibility). This high percentage of intra-AS bottleneck links contradicts the common belief that ISPs distribute routes over their internal links very well using complete knowledge of, and control over, their own networks. This result motivated us to investigate the practical challenges of route distribution within individual ISPs; viz., Section 2.5.3. The second unexpected result is that the majority of both intra-AS and inter-AS bottleneck links (i.e., 100% for intra-AS type and 81.2% for inter-AS type) is exclusively owned and managed by large ASes; e.g., Tier-1 or Tier-2 ASes. This implies that the Tier-1/Tier-2 ASes are the primary sources of bottleneck links.

[1] For IP to ASN mapping, we use the public IP-to-ASN mapping database by Cymru (`https://www.team-cymru.org/Services/ip-to-asn.html`).

[2] `http://www.renesys.com/2014/01/bakers-dozen-2013-edition/`

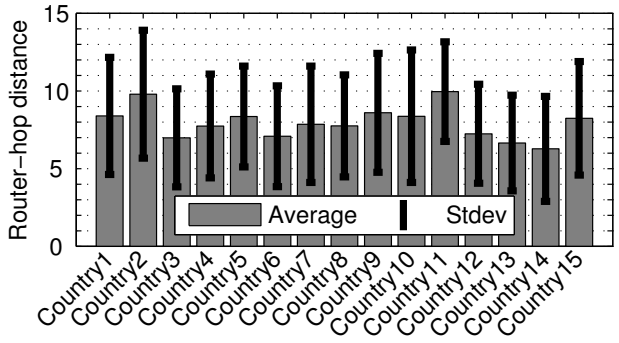

Figure 2.7: Router-hop distances of 50 bottleneck links for each of the 15 countries from the target regions.

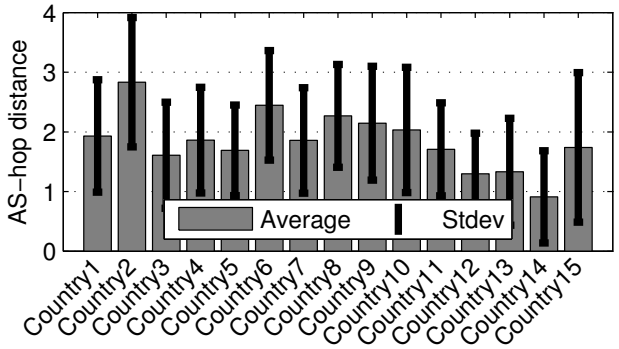

Figure 2.8: AS-hop distances of 50 bottleneck links for each of the 15 countries from the target regions.

Link distance

We also measure the *router-hop* and *AS-hop* distance of the bottleneck links from the hosts in the target regions. To measure a bottleneck link's router-hop distance, we take the average of router-hop distances from the 1000 hosts in a region. A challenge in measuring the router-hop distance via *traceroute* is that some destinations used have firewalls in their local networks, which prevents discovery of the last few router hops from the destinations. When *traceroute* does not reach a destination we assume the presence of the destination immediately past the last hop found. Thus, the measured router-hop distance is a strict *lower-bound* of the average router-hop distance from destination hosts.

Fig. 2.2.5 shows the average and standard deviation of the router-hop distance of the 50 bottleneck links for each of the 15 countries. The average router-hop distance ranges from 6 to 10 router hops with average of 7.9 router hops and no significant differences were found across the 15 countries. Considering the average length of Internet routes is approximately 17 router hops [54], we conclude that the bottle-

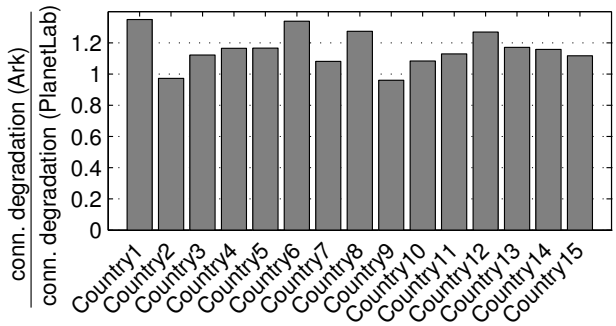

Figure 2.9: Connectivity degradation in the Ark dataset relative to the PlanetLab dataset for 50 flooding links selected from the routes measured by the PlanetLab nodes.

neck links are located in the middle to the slightly closer to the target region on the routes to the target. The distance analysis is also consistent with the observation that the most bottleneck links are within or connecting Tier-1/Tier-2 ASes.

Fig. 2.2.5 shows the average and standard deviation of the AS-hop distance for the 15 countries. The average AS-hop distance from the target to the bottleneck links ranges from 1 to 3 AS hops with average of 1.84 AS hops. Again, the measured AS-hop distances are strict lower bounds of the average AS-hop distances due to limited *traceroute* visibility.

2.3 Validation of Bottleneck Measurements

2.3.1 Independence of Route Sources

One of the common pitfalls in Internet measurements is the dependency on vantage point; that is, the location where a measurement is performed can significantly affect the interpretation of the measurement [124]. Here we argue that our routing-bottleneck results are independent of the selection of route sources S. To show this, we validate our computation of routing-bottleneck results by comparing the connectivity degradation (defined in Section 2.4.2) calculated using the original source set S (i.e., 250 PlanetLab nodes) with that calculated using an *independent* source set S' (i.e., 86 Ark monitors),[3] as shown in Fig. 2.9.

[3]The CAIDA's Ark project uses 86 monitors distributed over 81 cities in 37 countries and performs *traceroute* to all routed /24's. For consistent comparison, we use the Ark dataset that was measured on the same day when the PlanetLab dataset was obtained and select a subset of the measured traces in the Ark dataset that has the same AS destination, D, used in the PlanetLab

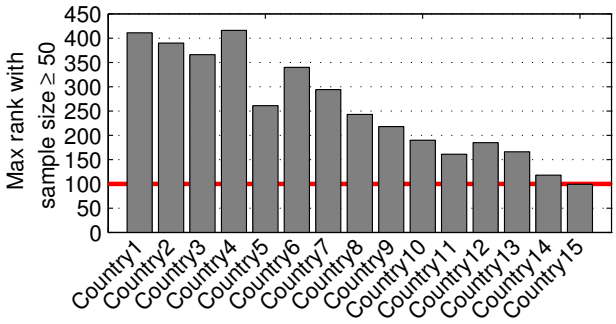

Figure 2.10: Maximum rank with sample size ≥ 50 for each of the 15 countries.

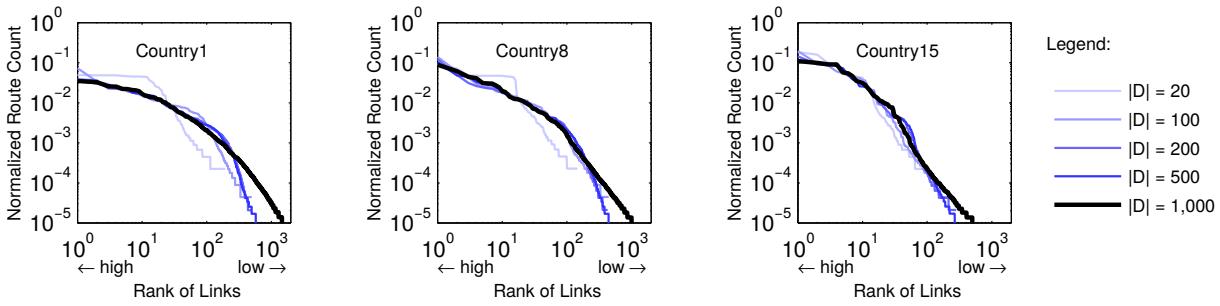

Figure 2.11: Normalized route count/rank in traced routes to 1,000 randomly selected hosts in 3 countries.

Notice that we select 50 bottleneck links for each country by analyzing the routes measured by PlanetLab nodes for *both* S and S'. The selection of these bottleneck links is discussed in Section 2.4.2. In most countries, the ratios of the two connectivity degradations are slightly higher than or very close to 1, which means that the bottlenecks of the PlanetLab dataset also become the bottlenecks of the independent Ark dataset.

2.3.2 Sufficiency of Link-Sample Size

Another common pitfall in Internet measurements aiming to discover statistical properties of datasets is the lack of a sufficiently large sample size; that is, it is possible that the sample size is insufficient to detect possible deviations from a discovered distribution. For reliable parameter estimates, the rule of thumb is that one needs to collect at least 50 samples for each element value [41]. Fig. 2.10 shows the maximum rank of the links (ordered by decreasing route count) that are observed with at least 50 link samples for the 15 countries in our measurement. The figure shows that for all 15 countries, all the high ranked links

dataset.

(i.e., 0–100 rank) are observed with more than 50 link samples and thus the parameter estimates based on these links (i.e., α and β in Fig. 2.2.2) are statistically sound.

Fig. 2.11 confirms that we have collected a sufficient number of link samples. In this figure, we illustrate the normalized route count with the various sizes of disjoint D and observe how the route count in the high rank region (i.e., rank 0–100) converges. We can conclude that $|D|$ = 1,000 is sufficient to discover the power-law distribution in the top-100 rank because it displays the same power-law distribution in the range as that observed with smaller size for D; i.e., $|D|$ <1,000. Thus, with a relatively small number of measurements one can learn the power-law distribution of the few but frequently observed high-rank links.

2.3.3 Traceroute Accuracy

In this subsection we review the common pitfalls in analyzing *traceroute* results and explain why they do not affect our measurement results.

Inaccurate alias resolution: As shown in many topology measurement studies, it is extremely important to accurately infer the group of interfaces located in the same router (or alias resolution) because its accuracy dramatically affects the resulting network topology [105, 139]. Highly accurate alias resolution still remains an open problem. Our measurements do *not* need alias resolution because we do not measure any router-level topology, but only layer-3 links (i.e., interfaces) and routes that use those links.

Inaccurate representation of load-balanced routes: Ordinary *traceroute* does not accurately capture load-balanced links and thus specially crafted *traceroute*-like tools (e.g., Paris traceroute [21]) are needed to discover these links. Our measurement does *not* need to discover load-balanced links because they cannot become the routing bottlenecks. Instead, we perform ordinary *traceroute* multiple times (e.g., 6 *traceroutes* in our measurement) for the same source-destination pair and ignore the links that do not always appear in multiple routes.

Inconsistent returned IPs: In response to *traceroute*, common router implementations return the address of the *incoming* interface where packets enter the router. However, very few router models return the *outgoing* interface used to forward ICMP messages back to the host launching *traceroute* [103, 104] and thus create measurement errors. However, our routing bottleneck measurement is not affected by this

router behavior because (1) most of the identified router models that return outgoing interfaces are likely to be in small ASes since they are mostly Linux-based software routers or lower-end routers [103], and (2) we remove all load-balanced links that can be created by the routers which return outgoing interfaces [104].

Hidden links in MPLS tunnels: Some routers in MPLS tunnels might not respond to *traceroute* and this might cause serious measurement errors [164]. However, according to a recent measurement study in 2012 [49], in the current Internet, nearly all (i.e., more than 95%) links in MPLS tunnels are visible to *traceroute* since most current routers implement RFC4950 ICMP extension and/or ttl-propagate option to respond to *traceroute* [49].

2.4 Routing-Bottleneck Exploits

Bottleneck links provide a very attractive target for link-flooding attacks; viz., attacks against Spamhaus [31] or ProtonMail [68]. By targeting these links, attacks become both *scalable* and *persistent*. Scalable because the number of targeted hosts can be increased substantially by flooding only few additional links, as shown later in this section. Persistent because adversaries can dynamically change the flooded link sets while maintaining the same targeted hosts, making the attacks undetectable by traditional anomaly detection methods. In this chapter, we focus primarily on the scalability of link-flooding attacks; i.e., we discuss the selection of target bottleneck links, the expected degradation in connectivity to the targeted hosts D, and how to increase attack targets without much additional measurement effort and attack traffic. The persistence properties of routing-bottleneck exploits is discussed in detail in Chapter 3.

To measure the strength of a link-flooding attack, we first define an *ideal attack* that completely disconnects all routes from sources S to selected hosts of destinations D. Then, we define realistic attacks that can cause very substantial *connectivity degradation*.

2.4.1 Disconnection Attacks

Let S be the 250 PlanetLab nodes and D the 1,000 randomly selected hosts in the target region; e.g., a country or a city. For efficient disconnection attacks, the adversary needs to flood only *non-redundant* links; that is, flooding of a link should disconnect routes that have *not* been disconnected by the other

already flooded links. Link redundancy can be avoided by flooding the *minimum-link set* of the routes from S to D, namely the *minimum* set of links whose removal disconnects *all* the routes from S to D, which is denoted by $M(S, D)$. Note that our notion of the minimum-link set differs from the *graph-theoretic* mincut. Our minimum-link set is a set of links that cover all routes to chosen nodes whereas the graph-theoretic mincut is a set of (physical) link cuts for an arbitrary network partitioning. Thus, one cannot use well-known polynomial-time mincut algorithms of graph theory [149] for our purpose.

Finding $M(S, D)$ can be formulated as the *set cover problem*: given a set of element $\mathcal{U} = \{1, 2, \cdots, m\}$ (called the universe) and a set $\mathcal{K}$ of n sets whose union equals the universe, the problem is to identify the smallest subset of $\mathcal{K}$ whose union equals the universe. Thus, our minimum-link set problem can be formulated as follows: the set of all routes we want to disconnect is the universe, $\mathcal{U}$; all IP-layer links are the sets in $\mathcal{K}$, each of which contains a subset of routes in $\mathcal{U}$, and their union equals $\mathcal{U}$; the problem is to find the smallest set of links whose union equals $\mathcal{U}$. Since the set cover problem is NP-hard, we run a greedy algorithm [74] to calculate $M(S, D)$. The greedy algorithm, which is explained in more detail in the description of the new attack strategy in Chapter 3, iteratively selects (and virtually cuts) each link in $M(S, D)$ until all the routes from S to D are disconnected.

Our experiments show that flooding an entire minimum-link set, $M(S, D)$, in any of the fifteen countries and cities selected would be rather unrealistic. For example, approximately 83 Tbps would be required to flood a minimum-link set of 2,066 links with 40 Gbps link capacity for a flooding attack against 1,000 servers in *Country1*. $|M(S, D)|$ can be much larger than both $|S|$ or $|D|$ since in measuring the size of $M(S, D)$ we exclude network links that are directly connected to hosts in S or D. Worse yet, Fig. 2.12 (top curves) shows that the minimum-link set size, $|M(S, D)|$, grows as $|D|$ grows. This implies that any practical link-flooding attack that disconnects *all* the hosts of a target region, D, must scale with an already large $|M(S, D)|$. However, as we show in the next section, an adversary does not need to flood an entire minimum-link set to degrade connectivity of D hosts of a targeted region very substantially. Also, by taking advantage of the power-law distributions of bottleneck links, an adversary can scale attacks to very large target sets, D.

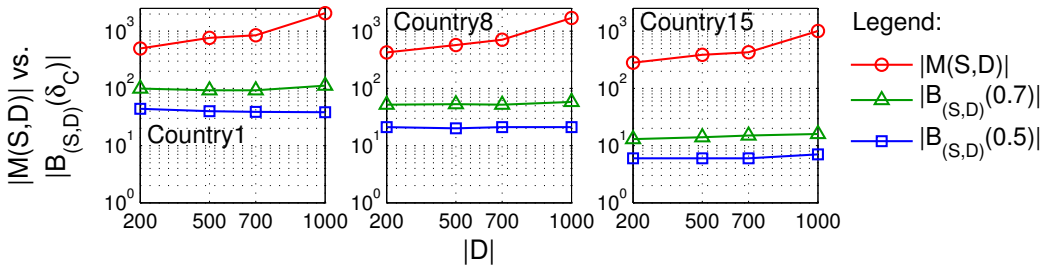

Figure 2.12: Measured sizes of minimum-link sets, $|M(S,D)|$, and selected bottlenecks sizes, $|B_{(S,D)}(\delta_C)|$, for given degradation ratios δ_C and varying $|D|$ in 3 countries.

2.4.2 Connectivity Degradation Attacks

Feasible yet powerful connectivity-degradation attacks would flood much smaller sets of links to achieve substantial connectivity degradation to the routes from S to D. To measure the strength of such attacks we define a connectivity-degradation metric, which we call the *degradation ratio*, as follows:

$$\delta_{(S,D)}(B) = \frac{\text{number of routes that traverse } B}{\text{number of routes from } S \text{ to } D}, \tag{2.1}$$

where B is the subset of the minimum-link set $M(S,D)$ links that are flooded by an attack.

The degradation-ratio metric represents the *connectivity damage* to the routes from S to D when all the routes that pass through any link in B are disrupted due to link-flooding attacks. For example, the degradation ratio of 0.5 represents that the half of the routes from S to D are damaged by the flooding attacks against the links in B. We believe that this connectivity-damage metric also faithfully represents the *bandwidth damage* of the communication from S to D; i.e., a portion of bandwidth from S to D that is disrupted by the attack. For this, we make the assumption that network links *provisioned adequately* for normal mode of operation; i.e., when there are no attacks. That is, a network link that carries large numbers of routes is equipped with *proportionally* large link bandwidth to provide reliable services when not under attacks. With this assumption, this connectivity damage becomes equivalent to the bandwidth damage.[4]

B's size, $|B|$, is determined by an adversary's capability. Clearly, the maximum number of links

[4]Notice that accurate measurements of the bandwidth damage require massive bandwidth measurements of all the links in $M(S,D)$, which would be challenging for existing link-bandwidth measurement techniques (such as BFind [11], Pathneck [76]) due to the large size (e.g., 2,000) of $M(S,D)$.

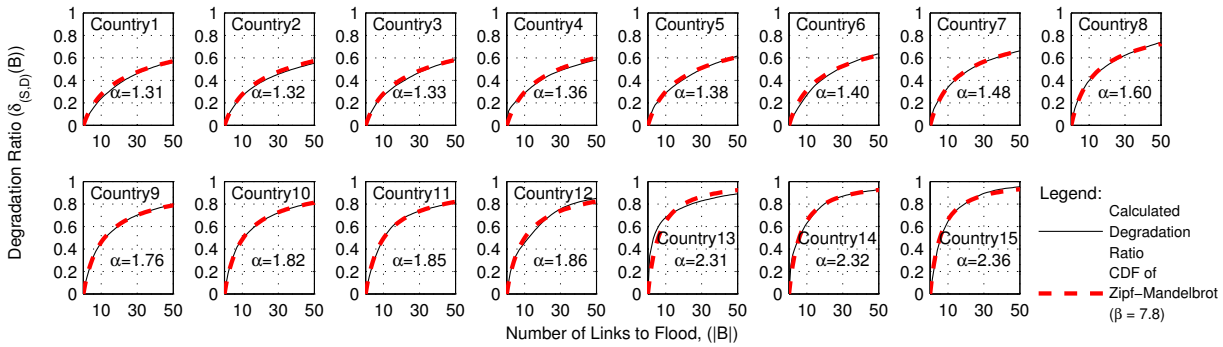

Figure 2.13: Calculated degradation-ratio/number-of-links-to-flood for 1,000 servers in each of the 15 countries.

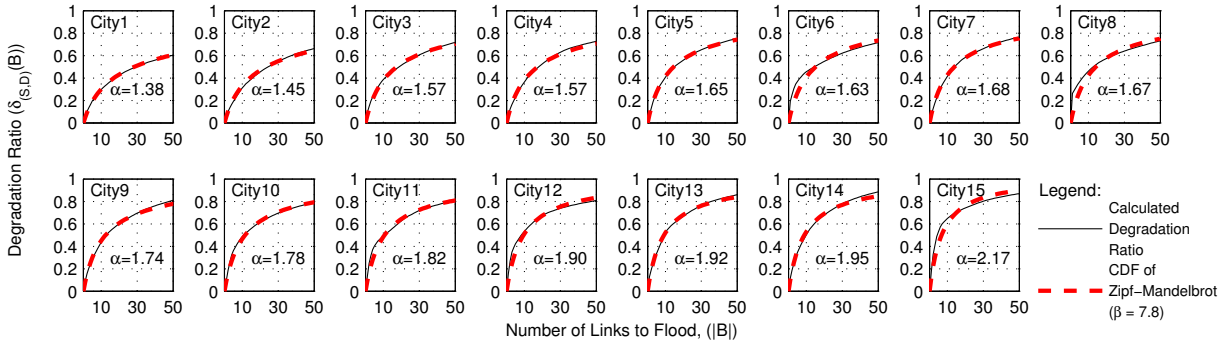

Figure 2.14: Calculated degradation-ratio/number-of-links-to-flood for 1,000 servers in each of the 15 cities.

that an adversary can flood is directly proportional to the maximum amount of traffic generated by attack sources controlled by the adversary; e.g., botnets or amplification servers. Here, we assume that the required bandwidth to flood a single link is 40 Gbps and thus the adversary should create $40 \times n$ Gbps attack bandwidth to flood n links concurrently. Links with larger physical capacity (e.g., 100 Gbps) have recently been introduced in the Internet backbone but the vast majority of backbone links still comprises links of 40 Gbps or lower capacity [81].

Fig. 2.4.2 and Fig. 2.4.2 show the expected degradation ratio calculated for each of the 15 countries and 15 cities for varying number of links to flood, or $|B|$, respectively. These countries and cities are ordered by increasing the averaged degradation ratio over the interval $1 \leq |B| \leq 50$. By definition, the degradation ratio for B (i.e., $\delta_{(S,D)}(B)$) is the sum of normalized route counts of the links in B. Thus, degradation ratio can be accurately modeled by the cumulative distribution function (CDF) of the Zipf-

Mandelbrot distribution since the normalized route count follows this distribution. Parameters α and β are listed in both figures. We note that the ordering of the degradation ratios in Fig. 2.4.2 and Fig. 2.4.2 is exactly the same as the ordering of the values of the distribution skew, α, of the 15 countries and 15 cities in Fig. 2.2.2 and Fig. 2.2.2, respectively. That is, countries and cities with low α have a lower degradation ratio (i.e., are less vulnerable to flooding attacks) whereas countries and cities with high α have high degradation ratio; i.e., are more vulnerable to flooding attacks. This confirms that the skew of the route-count distribution, α, of the Zipf-Mandelbrot distribution is a good indicator of vulnerability to link-flooding attacks.

Fig. 2.4.2 and Fig. 2.4.2 also show that the adversary can easily achieve significant degradation ratio (e.g., 40% - 82%) when flooding only few bottleneck links; e.g., 20 links. Given the proliferation of traffic amplification attacks achieving hundreds of Gbps or the extremely low costs of botnets, flooding several tens of bottleneck links of selected hosts in different countries around the world seems very practical.

Sizes of Bottlenecks

The size of a bottleneck selected for attack clearly depends on the *chosen degradation ratio* δ_C sought by an adversary. This size is defined as:

$$|B_{(S,D)}(\delta_C)| = \text{minimum } |B|, \text{ such that } \delta_{(S,D)}(B) \geq \delta_C.$$

The bottlenecks selected for attack, $B_{(S,D)}(\delta_C)$, are substantially smaller than their corresponding minimum-link sets, $M(S, D)$. Fig. 2.12 shows the set sizes of the minimum-link sets and the selected bottlenecks for chosen ratios δ_C of 0.7 and 0.5 for varying sizes of D. The plots for the three countries show that $|M(S, D)|$ is one to two *orders of magnitude* larger than $|B_{(S,D)}(\delta_C)|$ in the entire range of measured $|D|$ and δ_C. In other words, the attack against the selected bottlenecks requires a much lower adversary's flooding capability than for a minimum-link set while achieving substantial connectivity degradation; e.g., 70%.

Scaling the Number of Targets

Our experiments suggest that an adversary need not scale routing measurements and attack traffic much beyond those illustrated in this analysis for much larger target-host sets (i.e., $|D| \gg 1,000$) in a chosen

region to obtain connectivity-degradation ratios in the range illustrated in this analysis. This is the case following two reasons. First, our measurements for multiple disjoint sets of selected hosts in a target region yield the *same* power-law distribution for different unrelated sizes of D; viz., Fig. 2.11. Hence, increasing the number of routes from S to a much larger D will not increase the size of the bottlenecks appreciably. In fact, we have already noted that, unlike the size of minimum-link sets, $|M(S,D)|$, the size of the selected bottlenecks for a chosen degradation ratio δ_C, $|B_{(S,D)}(\delta_C)|$, does *not* change as $|D|$ increases, as shown by the lower two curves of in Fig. 2.12. Second, we showed that routing-bottleneck discovery is independent of the choice of S, where $|S| \gg |B|$; viz., Section 2.3.1. This implies that, to flood the few additional bottleneck links necessary for a much increased target set D, an adversary needs not increase the size of S and attack traffic appreciably.

2.5 Countermeasures

Defenses against attacks exploiting routing bottlenecks range from simple but naïve approaches to far-reaching structural countermeasures and operational countermeasures. We summarize these countermeasures, discuss their deployment challenges, and briefly evaluate their effectiveness. Naturally, defense mechanisms for *server-flooding* attacks (viz., [62]) are irrelevant to this discussion.

2.5.1 Naïve Approaches

The naïve approaches presented here are the most probable responses that the current networks would perform once the degradation attacks hit the routing bottleneck for hosts in any target region.

Local rerouting: Targeted networks can reactively change routes crossing flooded links so that the flooding traffic (including both legitimate and attack flows that are indistinguishable from legitimate flows) is distributed over multiple other local links. However, this might cause more collateral damage on the other local links after all.

Traffic-intensity based flow filtering: Typical mitigations for volumetric DDoS attacks detect and filter long-lived large flows *only* because otherwise they cannot run in real-time in large networks [89]. This countermeasure cannot detect nor filter attack flows in bottleneck links because these could be low-rate and thus indistinguishable from legitimate.

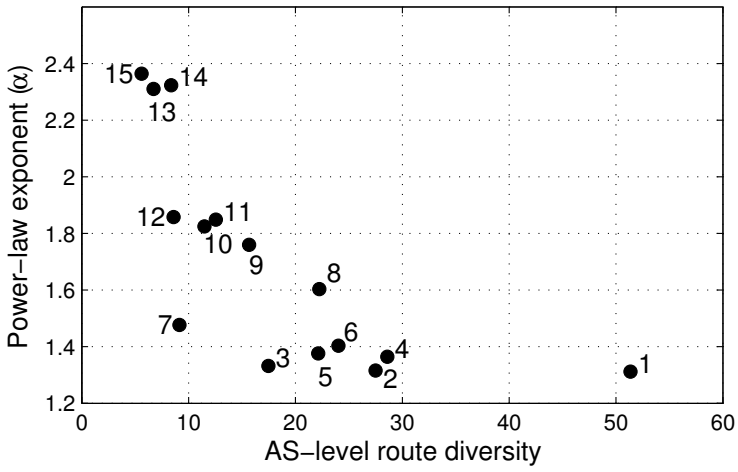

Figure 2.15: Correlation between power-law exponent and AS-level route diversity in 15 countries. (Legend: number i = Country$i$, for $i = 1, \cdots, 15$)

Using backup links: Typical backbone links are protected by the backup links, such that whenever links are cut, the backup links seamlessly continue to convey traffic. However, backup links cannot counter link-flooding attacks because they could be flooded too.

2.5.2 Structural Countermeasures

Structural countermeasures range from changes of physical Internet topology to those of inter-AS relationships. Although this type of countermeasures might require significant time (e.g., months) to implement, it could widen routing bottlenecks and decrease link-flooding vulnerability significantly. For example, if a country is connected to the rest of the world via only a handful of market-dominating ISPs, no matter how well routes are distributed, the country would inevitably experience routing bottlenecks. To remove these bottlenecks, the country would have to increase its route diversity through structural changes to its connectivity to the outside world.

Fig. 2.15 illustrates how AS-level structural changes could solve the routing-bottleneck problem. The x-axis is the metric called *AS-level route diversity* and it is calculated as

$$\frac{\{\text{number of intermediate ASes}\}}{\{\text{average AS hops from Tier-1 ASes to target region}\}}, \tag{2.2}$$

where the intermediate ASes are the ASes that connect the Tier-1 ASes with ASes located within each target region. The list of ASes within a target region are obtained from `http://www.nirsoft.`

net/countryip/. Ratio (2.2) is a good proxy for measuring the AS-level route diversity because it represents the average number of possible ASes at every AS hop from the Tier-1 ASes to the target region. The y-axis is the power-law exponent α obtained in Section 2.2.2. We see the clear correlation between the AS-level route diversity and the power-law exponent for the 15 countries, which supports our claim that the more AS-route diversity, the lower the power-law exponent.

We find that the Western European countries use significantly higher AS-level route diversity (i.e., 24.5 on average) than the rest of the countries (i.e., 16.5 on average), and thus are much less vulnerable to link-flooding attacks. This is undoubtedly due to long-standing policies (e.g., local-loop unbundling [29]) in European Union to stimulate ISP competition; e.g., to lower the cost of entry in ISP markets [60]. We believe that similar policies that promote ISP competition will increase route diversity and ultimately reduce the vulnerability to link-flooding attacks in other parts of the Internet.

2.5.3 Operational Countermeasures

Operational countermeasures could improve the management plane of various routing protocols (e.g., BGP or OSPF) to either decrease the skew of route-count distribution or better react to the exploits. Although most of the countermeasures discussed here have been proposed in different contexts before (e.g., [75, 113, 154, 166]) their effectiveness in reducing routing bottlenecks is unknown. Hence, we briefly analyze these countermeasures and their limitations here.

Inter-domain traffic engineering

When an inter-AS link is flooded, at least one of the ASes should be able to quickly redirect the flooding traffic to relieve congestion. To do this, an AS would need to update its BGP announcements to its neighbors that use the flooded links. However, inbound traffic redirection via updated BGP announcements [158] is not guaranteed since upstream ASes may have no positive incentives to re-route; i.e., upstream ASes would ignore these announcements whenever re-routing increases traffic cost. Even if neighbour ASes followed the updated BGP announcements, the long BGP convergence time (e.g., up to an hour [102]) would render them ineffective for timely response to link-flooding. Further delays would be incurred because outbound inter-AS level redirection requires human intervention in the current Internet.

That is, inter-AS traffic redirection can only be manually configured since inter-AS links are selected by the coupling of BGP and IGP (e.g., OSPF) protocols [148]. Hence, added costs would become necessary to diffuse flooding traffic [154]. Therefore, timely and cost-effective reaction to inter-AS link flooding requires a dynamic mechanism that adaptively utilizes multiple parallel inter-AS links [154] and/or multiple AS-level route with different next-hop ASes [166]. The specific design of such mechanisms is beyond the scope of this paper.

Intra-domain traffic engineering

- *Proactive load balancing.* Many of today's networks, especially those of large ISPs, deploy intra-domain load-balancing mechanisms based on the Equal-Cost Multi-Path (ECMP) algorithm [75]; e.g., approximately 40% of Internet routes [21] are load balanced by it. We call a particular intra-domain link is *load-balanced* when most of intra-domain flows crossing the link have alternative equal-cost routes that do not include the link. Note that such load-balanced links *cannot* be the routing bottleneck since flooding at the link cannot congest all the routes that it serves; viz., Section 2.2.1. Thus, to prevent intra-domain routing bottlenecks, we can make all the intra-domain links to be load-balanced. However, this requires a network condition which cannot always be satisfied in practice unless an ISP has a symmetrical network topology and weight configuration. Thus, in practice ISPs should identify potential link targets and perform real-time reconfiguration of their networks (which appears to be a non-trivial network configuration task) so that the identified link targets become load-balanced.
- *Reactive MPLS traffic engineering.* One of the most widely used traffic engineering mechanisms is Multi-Protocol Label Switching (MPLS). As of 2013, at least 30% of Internet routes travel through MPLS tunnels [49] and they are mostly deployed in the large ISPs. Unlike the local rerouting solution discussed in Section 2.5.1, MPLS reconfiguration can perform fine-grained traffic steering to avoid collateral damage on the other links. However, the widely used offline MPLS reconfiguration cannot be very effective since it can reconfigure tunnels only on a time scale ranging from tens of minutes to hours and days [55, 159]. Worse yet, the online MPLS reconfiguration, such as the auto-bandwidth mechanism [113], which automatically allocates required bandwidth to each tunnel

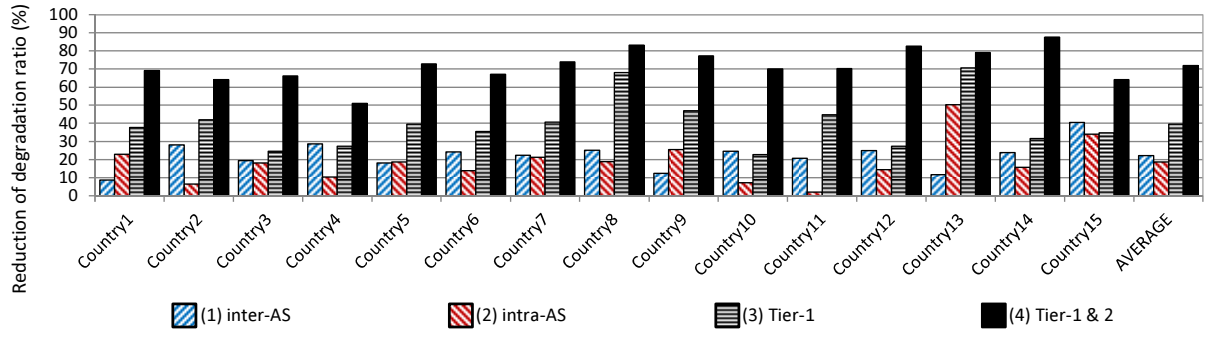

Figure 2.16: Reduction of degradation ratios due to four defense strategies when 20 bottleneck links are flooded for each of the 15 countries.

and change routes, is susceptible to sustained congestion. This is because it cannot detect congestion *directly* but only via reduced traffic rates caused by congestion. Thus, even an auto-bandwidth mechanism would require human intervention to detect link-flooding attacks [142] thereby slowing reaction time considerably.

Moreover, even if an MPLS traffic engineering mechanism can quickly and reliably balance the flooding traffic at the targeted link, its effectiveness can be very limited. For example, in typical ISP networks, the aggregate bandwidth of all the new MPLS tunnels after optimal online MPLS traffic engineering tends to have only about 2 or 3 times more bandwidth than the initial target link bandwidth; viz., Section 4.7 for simulated results. This small bandwidth provisioning by MPLS traffic engineering is ineffective since the network can be easily flooded again as adversaries simply double or triple their attack traffic.

We utilize the recently proposed real-time traffic engineering techniques that leverage software-defined networking (SDN) architecture in ISP networks to implement traffic-engineering based bandwidth expansion mechanism that tests and ultimately deters cost-sensitive adversaries; viz., Chapter 4.

Effectiveness of operational countermeasures

We assume that all the operational countermeasures discussed above can be effectively implemented, albeit the provided practical challenges above. Then we evaluate the reduction of degradation ratios due

to the following four defense strategies using the operational countermeasures: (1) inter-domain load balancing at all inter-AS links, (2) intra-domain load-balancing and traffic engineering at all intra-AS links, (3) all operational countermeasures at all Tier-1 ASes, and (4) all operational countermeasures at all Tier-1 and Tier-2 ASes. In this evaluation, the types of all flooded links are known. Fig. 2.16 shows the reduction of degradation ratios in percentage for 15 countries. It shows that the defense strategies that protect a specific type of links (i.e., strategy (1) and (2)) are not very effective in general (approximately 20% reduction on average) because adversaries can still find bottleneck links from the other types of links. However, the defense strategies deployed by Tier-1 and/or Tier-2 ASes (i.e., strategy (3) and (4)) show much higher effectiveness: when all Tier-1 ASes implement all the operational countermeasures, the degradation ratio is reduced by 40%; and when all Tier-2 ASes also join the defense, 72% of reduction is achieved on the average. This confirms our previous observation that the large Tier-1 and Tier-2 ASes are primarily responsible for routing bottlenecks.

2.5.4 Application Server Distribution

One might distribute application servers in different geographic locations, possibly using content distribution (e.g., Akamai [119]) and overlay networks (e.g., RON [16], SOS [86]) to distribute routes. The application servers have to be distributed in such a way that inherent route diversity is fully utilized; i.e., analysis must show that no routing bottleneck arises. However, this might not be practical for some domains such as industrial process systems, financial services, or defense services where constrained geography may restrict application distribution.

2.6 Related Work

2.6.1 Internet Topology Studies

A large body of research investigates the topology of Internet. Two long-term projects have measured the router-level Internet topology via *traceroute*-like tools: CAIDA's Archipelago project [35] and DIMES project [137]. Rocketfuel [139] is another project that use approximately 800 vantage points for *traceroute* to infer major ISP's internal topology. Together these studies provide important insights into the layer-3

topology of the Internet.

Our routing bottleneck measurement differs from the topology studies in two important ways. First, we do not measure or even infer the router-level topology but simply observe *how the routes are distributed* on the underlying router-level topology. Second, we do not need nor attempt to observe all the routes covering the entire address space but focus on the route-destination regions of potential adversary interest.

2.6.2 Topological Connectivity Attacks

Faloutsos *et al.* analyzed *traceroute* data and concluded that the node degree of the routers and ASes have power-law distribution [50]. Albert *et al.* confirmed the power-law behavior of the node-degree distribution and concluded that the Internet suffers from an 'Achilles' heel' problem; i.e., targeted removal attacks against the small number of hub nodes with high node degree will break up the entire Internet into small isolated pieces [13].

The Achilles' heel argument has triggered several counter-arguments. Some find that node-removal attacks are unrealistic because the number of required nodes to be removed is impractically high [98, 162]. Li *et al.* argue that the power-law behavior in node-degree distribution does not necessarily imply the existence of hub nodes in the Internet by showing that power-law node-degree distribution can be generated without hub nodes [95].

Our routing-bottleneck study discovers a new power-law distribution in the Internet. However, this power-law is *completely different* from that of the above-mentioned work for two reasons. First, we measure a power law for the link usage in Internet routes whereas the above-mentioned work finds power laws in the node-degree distribution. Second, the scope of our power-law analysis is different; i.e., it is focused on, and limited to, a *chosen* route-destination region whereas the above-mentioned work analyzes the power-law characteristics of the entire Internet.

2.6.3 Bandwidth Bottleneck Studies

In networking research, the term 'bottleneck' has been traditionally used to represent the link with the smallest available bandwidth on a route; i.e., the link that determines the end-to-end route throughput. To distinguish it from a routing bottleneck, we call this link the *bandwidth* bottleneck. Several attempts have

been made to measure bandwidth bottlenecks in the Internet; viz., BFind [11] and Pathneck [76]. However, routing and bandwidth bottlenecks are fundamentally different as they do not necessarily imply each other. That is, routing bottlenecks are unrelated to the available bandwidth or provisioned link capacity, but closely related to the number of routes served by each link. Conversely, bandwidth bottlenecks can occur in the absence of any routing bottlenecks.

2.6.4 Control-Plane and Link-Flooding Attacks

Attacks that cause instability of the control plane in Internet routing [133] and link-flooding attacks [144] have been recently proposed and launched in real life already [31]. In contrast to the previous link-flooding attacks, where we focus on the feasibility of flooding a small set of critical links, here we explore a *fundamental vulnerability* of today's Internet, namely, pervasive routing bottlenecks that can be exploited by any flooding attack. We show the ubiquity of routing bottlenecks for hosts in various countries and cities around the world via extensive measurements, and identify their root cause. We also explore the characteristics of bottleneck links; e.g., link type and distance to targets. Last, we provide several practical countermeasures.

Chapter 3

Crossfire: Large-Scale Persistent Link-Flooding Attacks

3.1 Outline of the Chapter

The previous chapter shows that routing bottlenecks are pervasive in today's Internet and that in principle link-flooding attacks targeting routing bottlenecks can effectively disconnect large geographic areas (e.g., cities, counties) from the rest of the Internet. However, studies on attack strategies that disconnect particular *adversary-chosen* Internet servers have been uncommon, possibly because of the complexity of selective server targeting and difficulty of flooding arbitrary bottleneck links in the Internet core. Instead, most of these attacks cause route instabilities [133] and Internet connectivity disruption [144] that affect unspecified servers rather than selective end-server disconnection (reviewed in Section 3.7). Nevertheless, when disconnection of critical infrastructure (e.g., energy distribution, time-critical finance, command and control services) from the Internet is the aim of an attack, link flooding can be extremely effective; e.g., current peak rates of a single botnet-driven attack can easily exceed 100 Gbps [116], making it possible to flood the vast majority Internet links.

Link flooding by botnets cannot be easily countered by any of the traditional Internet defense methods for three reasons. First, bots can use *valid IP addresses*, and thus defenses based on detecting or preventing use of spoofed IP addresses become irrelevant; e.g., defenses based on ingress filtering [56], capability

systems [165, 168], or accountable protocol designs [17, 112]. Second, and more insidiously, botnets can flood links *without using unwanted traffic*; e.g., they can send packets to each other in a way that targets groups of routers [144]. Third, a botnet can launch an attack with *low-intensity traffic* flows that cross a targeted link at roughly the same time and flood it; e.g., a botnet controller could compute a large set of IP addresses whose advertised routes cross the same link (i.e., *decoy* IPs), and then direct its bots to send low-intensity traffic towards those addresses. This type of attack, which we call the *Crossfire attack*[1] and describe in this chapter, is undetectable by any server located at a decoy IP address, and its effects are invisible to an ISP until (too) late[2]. Furthermore, current traffic engineering techniques are unable to counter these attacks. The latency of offline traffic engineering is impractically high (e.g., hours and days [55, 159]) whereas online traffic engineering techniques cannot offer strong stability guarantees [94], particularly when multiple ISPs need to coordinate their responses to counter an attack, and hence cannot be deployed in the Internet backbone. Worse yet, even if online techniques could be deployed, an adversary attack could change the set of target links in real time thereby circumventing online traffic engineering defenses; viz., discussion in Section 4.8.

In this chapter, we present the Crossfire attack. This attack can effectively cut off the Internet connections of a targeted enterprise (e.g., a university campus, a military base, a set of energy distribution stations); it can also disable up to 53% of the total number of Internet connections of some US states, and up to about 33% of all the connections of the West Coast of the US. The attack has the hallmarks of *Internet terrorism*[3]: it is low cost using legitimate-looking means (e.g., low-intensity, protocol conforming traffic); its locus cannot be anticipated and it cannot be detected until substantial, persistent damage is done; and most importantly, it is *indirect*: the immediate target of the attack (i.e., selected Internet links) is not necessarily the intended victim (i.e., an end-point enterprise, state, region, or small country). The low cost of the attack (viz., Section 4.8) would also enable a perpetrator to *blackmail* the victim.

The main contributions of this chapter can be summarized as follows:

[1]This attack should not be confused with that of Chou *et al.* [38], which also uses the term "crossfire" for a different purpose; i.e., to illustrate *unintentionally* dropped legitimate flows.

[2]Of course, an adversary could easily change the set of bots used in the attack; e.g., typical networks of 1M bots would allow one hundred disjoint, and a very large number of different sets of 10K bots.

[3]Although common agreement on a general definition of terrorism does not exist, the means of attack suggested here are common to most terrorist attacks in real life.

1) We introduce the Crossfire attack in the Internet and show how it can isolate a target area by flooding carefully chosen links. In particular, we show that it requires relatively small botnets (e.g., ten thousand bots) and is largely independent of the bot distribution. It has no effective countermeasure at either target routers or end-point servers, and as a result, it can degrade and even cut off connections to selected Internet areas ranging from a single organization to several US states, for a long time.

2) We show the feasibility of the Crossfire attack with data obtained from large-scale experiments. In particular, our analysis of Internet traffic to targets confirms our previous discussion on routing bottlenecks (i.e., the existence of very few links that are responsible for delivering the vast majority of all traffic to a specific area) which makes this attack fairly easy to launch. Traffic concentration in a small set of links located a few (e.g., three to four) hops away from a targeted area is intuitively attributable to the shortest path routing by the Internet IGB/BGP protocols, and easily discoverable by common tools such as *traceroute*. We show that the attack traffic on these links follows a power-law distribution that depends on the targeted servers and cannot be anticipated by generic Internet-connectivity metrics; e.g., metrics based on router connectivity [50, 90] or betweenness centrality [117]. Note that flooding routing-bottleneck links degrades connections *towards* target areas (e.g., connections made from remote clients to servers in a target area) but not necessarily the reverse connections.

3) We show that the Crossfire attack is *persistent* in the sense that it cannot be stopped either by individual ISPs or by end-point servers, which are effectively disconnected by flooded links at least three hops away, for a long time. Attack persistence is caused by three independent factors. First, the selected attack routes become stable after the removal of all load balancing dynamics (which is consistent with prior observations [43]). Second, the attack traffic is shaped such that (i) only a data plane of a link is flooded while the control plane remains unaffected, and hence dynamic re-routing can be initiated only after data-plane flood detection, which gives an adversary ample time to select alternate sets of links for the same target area; and (ii) early congestion of links located upstream from a targeted link is avoided by *a priori* estimation of the bandwidth available on the route to that link. Third, the availability of multiple, disjoint sets of target links distributed across multiple ISPs implies that no single ISP can unilaterally detect and handle this attack.

4) We argue that collaborative on-line, rather than offline, traffic engineering techniques would become

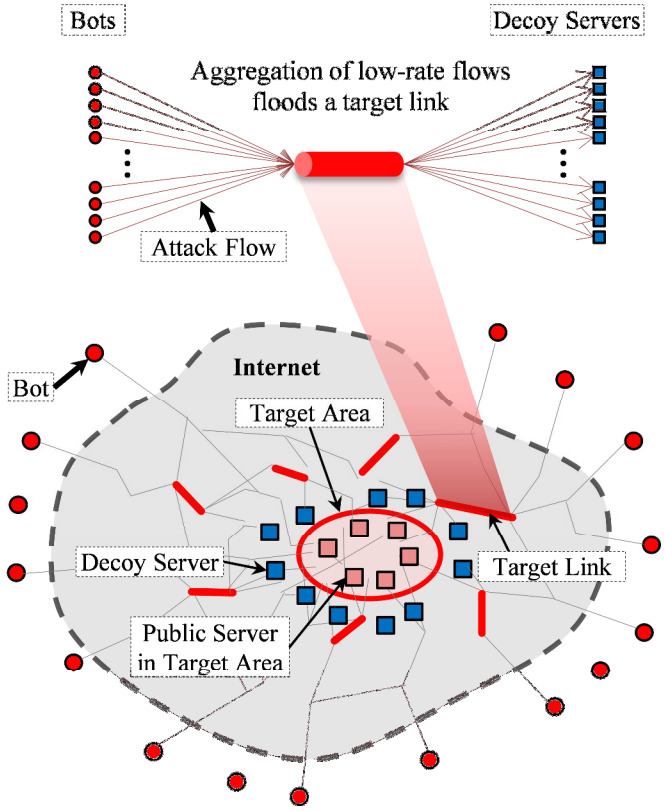

Figure 3.1: The Elements of the Crossfire Attack

necessary to reduce the persistence of such attacks. In the absence of such measures, the Crossfire attack must be handled by application protocol layers; e.g., overlays that detect effective host disconnection from the Internet and re-route traffic via different host routes [15, 16, 86]. Botnet market disruption and international prosecution of attack perpetrators may complement technical countermeasures against these attacks.

3.2 The Crossfire Attack

In this section, we present the steps of the Crossfire attack. The adversary's goal is to prevent legitimate traffic from flowing into a specific geographic region of the Internet, and the capability she needs to accomplish that goal is to flood a few network links in and around that region. We begin by defining the two most common terms used in this chapter: the *target area* and *target link*. Then, we describe how an

adversary designs an attack using the bots she controls. Fig. 3.1 illustrates the concept of the Crossfire attack.

Target Area: A target area is a geographic region of the Internet against which an adversary launches an attack;[4] viz., the area enclosed by the circle in Fig. 3.1. A typical target area includes the servers of an organization, a city, a state, a region, and even a country, of the adversary's choice.

Target Link: A target link is an element of a set of network links the adversary needs to flood so that the target area is cut off from the rest of the Internet. These carefully chosen network links are the actual target of the flooding attack whereas the target area is the real, intended target.

To launch a Crossfire attack against a target area, an adversary selects a set of public servers within the target area and a set of *decoy servers* surrounding the target area. These servers can be easily found since they are chosen from publicly accessible servers (viz., Section 3.5.2). The set of public servers is used to construct an attack topology centered at the target area, and the set of decoy servers is used to create attack flows. Then, the adversary constructs a *"link map"*, namely the map of layer-3 links from her bot addresses to those of the public servers. (The differences between a link map and a typical router-topology map are discussed below.) Once the link map is created, the adversary uses it to select the best target links whose flooding will effectively cut off the target area from the Internet. Next, the adversary coordinates the bot-decoy (server) flows to flood the target links, which would eventually block most of the flows destined to the target area. This can be easily done since target links are shared by flows to the decoy servers and target area. A *flow* is defined by *5-tuple*, which is a stream of packets having the same source and destination IP addresses, same source and destination port numbers, and same protocol number. Finally, the adversary selects multiple disjoint sets of target links for the same target area and floods them one set at a time, in succession, to avoid triggering bot-server route changes. The three main steps needed to launch the Crossfire attack consist of the link map construction, attack setup, and bot coordination, as shown in Fig. 3.2. Note that, to extend the duration of the attack, the last step, namely the bot coordination step, is executed repeatedly by dynamically changing the sets of target links, which we will explain in detail in Section 3.2.4. We describe each of the adversary's steps below.

[4]The attack may have side effects and affect other non-targeted areas. However, these side effects do not increase attack's detectability. They can be a desired feature whenever the adversary's goal is to cut off most of the traffic at and around a target area, rather than to surgically isolate a small number of specific servers.

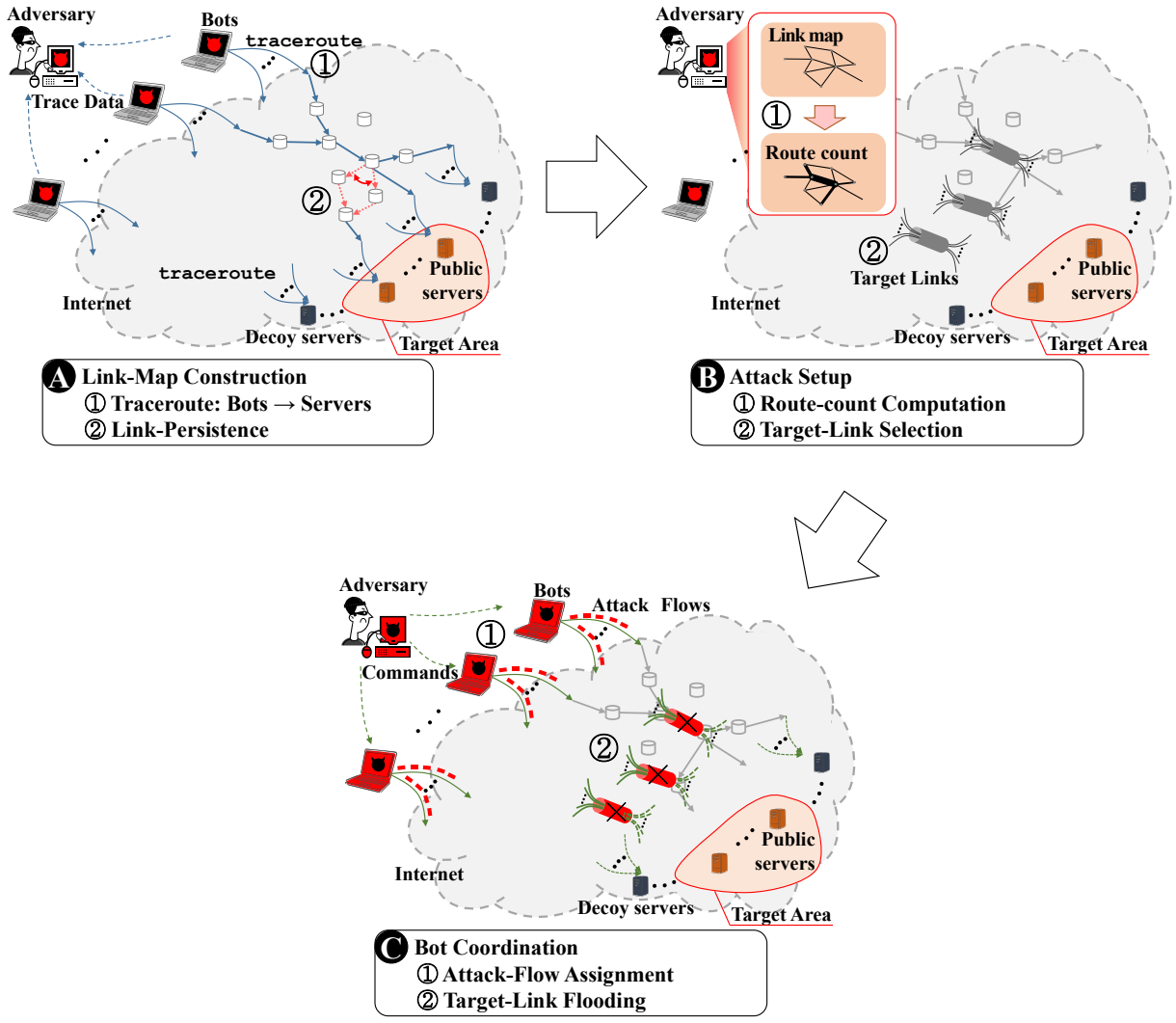

Figure 3.2: The steps of the Crossfire attack.

3.2.1 Link Map Construction

To flood links leading to a target area, an adversary needs to construct a link map of the Internet surrounding that area.

Traceroute from Bots to Servers

To construct the link map, the adversary instructs her bots to run *traceroute* and find all the router-level routes to the public servers in the target area and the decoy servers. The result of a traceroute is a sequence of IP addresses that are assigned to the interfaces of the routers on the route, where a link is identified by the

IP address of the adjacent router's interface. Thus, the sequence of IP addresses represents the sequence of layer-3 links[5] that the attack traffic would travel.

A link map for the Crossfire attack is *different* from a typical router-topology map [50] that attempts to build a router-level connectivity to analyze topological characteristics (e.g., node degree). This attack only needs the list of layer-3 links and their relationships to compute a set of target links on the bot-to-target area routes, while each link's membership to a specific router is *irrelevant*. Note that the link map construction does *not* require IP alias resolution [138]; i.e., determining the set of IP interfaces owned by the same router is unnecessary. As a consequence, an adversary can use the ordinary traceroute for the link map construction regardless of how inaccurate its IP alias resolution may be [164].

A bot runs multiple traceroutes to the same server in order to determine the stability and multiplicity (or diversity) of a route, both of which are used for selecting effective target links (discussed in Section 3.5.4 in detail). The traceroute results are collected by the adversary and used to construct the link map.

Link-Persistence

The link map obtained in the previous step cannot be directly used to find target links since some of the routes obtained may be unstable. Unstable routes would complicate the attack since the adversary may end up chasing a moving target. Route instability is primarily caused by ISPs' load balancing processes (i.e., forwarding traffic through multiple routes), which are supported by most commercial routers [21]. A consequence of load balancing is that, for the same bot-to-server pair, some links do not always appear on the trace of the route produced by multiple invocations of traceroute (viz., the arrowed links of step A-② in Fig. 3.2). These links are said to be *transient*, whereas those that always appear on a route are said to be *persistent*. The adversary identifies transient links and *removes* them from the set of potential target links. Our Internet experiment shows that 72% of layer-3 links measured by traceroute are persistent[6].

[5] Although a single layer-3 link consists of several lower layer connections that are invisible to the adversary, the flooding on the layer-3 link is still effective whenever the adversary's maximum bandwidth assumption (e.g., 40 Gbps in our experiments) is correct along the layer-3 link.

[6] The link map obtained may not include *backup* links since these links typically do not show up in traceroutes. The existence of such links is largely immaterial to the effectiveness of Crossfire. If attack traffic spills over onto backup links and its

3.2.2 Attack Setup

The adversary uses the obtained link map to discover the set of target links whose flooding cuts off the largest number of routes to the target area. Clearly, the larger the proportion of cut routes out of all possible routes to the target area, the stronger the attack. The attack-setup step consists of the following two sub-steps.

Route-Count Computation

The adversary analyzes the link map for a target area and computes *'target-specific route count'*, or simply route count henceforth, for each network link in the link map. The **route count** of a persistent link is the number of flows between bots and target-area servers that can be created through that link, as defined in Section 2.2. Hence, route count is a target-area-specific metric and can vary widely from one target area to another (viz., Section 3.3.1). It is a very different metric from those used for Internet connectivity, such as the "betweenness centrality" [117] and the degree of routers [50, 90] (viz., Section 3.3.1), and should not be confused with them.

A high route count for a link indicates that the link delivers both a large number of attack and legitimate (or non-attack) to a specific target area, and thus the link becomes a good attack target. We confirm that the route count follows a *power-law* distribution in a link map (viz., Section 3.3.1) as we discussed in Section 2.2, and this enables an adversary to easily discover a set of high route count links that delivers most traffic to a target area.[7] Furthermore, the computed route count remains largely unchanged for at least several hours due to the well-known, long-term stability of Internet routes [43, 123]. Hence, route count can be used as a stable and reliable metric by the adversary in selecting target links.

Target-Link Selection

In this step, the adversary finds multiple disjoint sets of target links to be flooded. The adversary selects at least two disjoint sets of target links and uses them one at a time, in succession, to achieve attack

intensity dampens appreciably, the adversary could easily switch to a new set of target links for the same server area, as shown in Section 3.2.4.

[7]The power-law of route count should *not* be confused with *connectivity properties* derived from traceroute, such as those for the degree of router level topology [90].

persistence (viz., Section 3.2.4). The goal of this step is to maximize the amount of disrupted traffic flowing into the target area by optimal selection of target links using the link map and route count.

To quantify how much of the traffic to a target area can be cut off by a chosen target-link set, the adversary computes the degradation ratio for that target area. The **degradation ratio** is the fraction of the number of bot-target area routes cut by the attack over the number of all possible bot-target area routes; viz., Eq. (2.1) in Section 2.4. We say that a route is cut by an attack if the route contains a target link that is flooded by the attack.

To select the target links that maximize the degradation ratio to a target area, the adversary must solve the *generalized maximum coverage problem*, which is a well-known NP-hard problem. Instead of finding an exact solution, the adversary uses an efficient heuristic, namely a greedy algorithm [42], presented in Section 3.4.4. The execution time of our heuristic is very small, namely less than a minute in all experiments (viz., Section 3.4.4). This enables the adversary to adapt to dynamic route changes, if necessary. The output of this algorithm shows that flooding a few target links can block a majority of the connections to a target area. For example, flooding ten target links causes a 89% degradation ratio for a small target area; flooding fifteen target links can block 33% of connections flowing to the West Coast of the US (viz., Section 3.5.4).

3.2.3 Bot Coordination

Once target links are selected at step B-② (Fig. 3.2), the adversary coordinates individual bots to flood the target links. To create flooding flows for a given set of target links, the adversary assigns to each bot (1) the list of decoy servers and (2) the send-rates for packets destined to individual decoy servers. The send-rates are assigned in such a way that individual attack flows have low intensity (or low bandwidth) while their aggregate bandwidth is high enough to flood all target links. This step consists of two sub-steps.

Attack-Flow Assignment

The goal of the attack-flow assignment is to make the aggregate traffic rate at each target link slightly higher than the link bandwidth so that all the legitimate flows are severely degraded in those links. Two assignment constraints must be satisfied. The first is that the adversary must keep each per-flow rate

low enough so that none of the network protection mechanisms in routers or intrusion detection systems (IDS) at or near a server can identify the flow as malicious. The second is that the aggregate attack traffic necessary to flood all the targeted links is relatively evenly assigned to multiple bots and decoy servers. The first constraint ensures *indistinguishability* of attack flows whereas the second addresses *undetectability* both at servers in the target area and at decoy servers; viz., Section 3.6 for details. The adversary first sets the maximum target bandwidth for each target link and exhausts it with attack flows. Then, she assigns individual flows for each target link.

The rate of an attack flow at a target link is lower-bounded by the route count. The average per-flow rate for the target link should be higher than the target bandwidth divided by the maximum number of available attack flows on the link, which is proportional to its route count. Moreover, the assignment of the per-flow rate must take into account the maximum flow rate a decoy server can handle without triggering traffic alarms. For example, if a decoy server is a public web server, one web click per second on average (a HTTP GET packet per second $\simeq$ 4 Kbps) would not be classified as abnormal traffic at the server. Therefore, the adversary can easily assign a large enough number of attack flows with low per-flow rates. The adversary also has to assign per-bot and per-decoy server rates that are evenly distributed. For enhanced undetectability of attack traffic at the bots and the decoy servers, the adversary must account for all previously assigned traffic rates at the bots and decoy servers whenever assigning new attack flows. The adversary conservatively sets the target bandwidth to 40 Gbps, which is the most widely used link bandwidth currently deployed (OC-768) for high bandwidth backbones.

Despite an adversary's careful attack flow assignment, non-target links located upstream of the target links could still become congested, which we call *early congestion*, if they have limited bandwidth and/or the bot density in a certain area is too high. The adversary can avoid potential early congestion using *a priori* link bandwidth estimation, which we discuss in detail in Section 3.4.3.

Target-Link Flooding

The adversary directs her bots to start generating the attack flows. Each bot is responsible for multiple attack flows, each of which is assigned a distinct decoy server with the corresponding required send-rate. Bots slowly increase the send-rates of their attack flows up to their assigned send-rates, which makes the

attack flows indistinguishable from the traffic patterns of typical ”flash crowds” [83]. Bots can adjust the intensity of their flow traffic dynamically, based on the state of each target link; i.e., if the actual bandwidth of a target link is less than the assigned attack bandwidth (set in Section 3.2.3), the bots stop increasing the rates of attack flows as soon as the target link is flooded.

3.2.4 Rolling attacks

The adversary can dynamically change the set of target links (among the multiple sets found previously) and extend the duration of the Crossfire attack virtually indefinitely. Continuous link flooding of the same set of target links would lead to bot-server route changes since it would inevitably activate the router's failure detection mechanism. Hence, changing the set of target links assures attack persistence and enables the attack to remain a pure data-plane attack. The adversary can also dynamically change the set of bots to further enhance the undetectability of the Crossfire attack. These dynamic attack execution techniques are called *rolling attacks* in Section 3.4.2 where they are described in more detail.

3.3 Technical Underpinnings

In this section, we discuss the two characteristics of the current Internet which enable the Crossfire attack, namely (1) the power law of route-count distribution, which is target-area specific, and (2) the independence of the geographical distribution of bots from target links and attack targets, which gives an adversary has a wide choice of bots in different locations on the globe.

3.3.1 Characteristics of Route-Count Distribution

Before analyzing the distribution of route count, we must distinguish between the attack-specific route count and connectivity-specific metrics, such as the *betweenness centrality* [117] and the *degree of routers* [50], which characterize an Internet topology. Recall that the route count of a link represents the number of source-to-destination (i.e., bot-to-server in the target area in the Crossfire attack) pairs whose traffic crosses the link *persistently*. In contrast, betweenness centrality, which is the number of shortest routes among all vertices that pass through an edge in a graph, does not reflect actual traffic flows and their dynamics. Similarly, the connectivity degree of a router, which represents the router's layer-3 direct

connections to neighbor routers, namely the topological connectivity of the router, does not capture any dynamics of traffic flows. Thus, neither of these metrics could be used to evaluate the feasibility of the Crossfire attack.

Our analysis on the route-count distribution is two-fold; first, we show that it is easy to find target links that have extremely high route count for a selected target area; and second, we show that route count of a link is not a constant but varies depending on a selected target area (i.e., route count is a target-area specific metric).

Universal power-law property of route-count distribution

Notice first that the discussion on power-law property here confirms our previous power-low analysis in Section 2.2. A power-law distribution exhibits a heavy-tail characteristic, which indicates that extreme events are far more likely to occur than they would in a Gaussian distribution. More formally, a quantity x obeys a power-law if it follows a probability distribution

$$p(x) \propto x^{-\alpha} \text{ for } x > x_0, \tag{3.1}$$

where α is a constant parameter of the distribution known as the scaling parameter [41]. The power-law property appears in the tail of the distribution (i.e., $x > x_0$)[8]. If a power-law distribution holds for route count, that would imply that an adversary could easily find links whose route count is many orders of magnitude higher than average. These links would become good targets for attack for a particular target area.

We use the rigorous statistical test proposed by Clauset *et al.* [41][9] to show that a *power-law holds for route-count distributions*. We first estimate the parameters (i.e., x_0 and α) of power-law distribution on our route-count datasets and test the power-law hypothesis with the estimated parameters. Fig. 3.3

[8]Some past research relied on simple data-fitting methods to conclude that their datasets follow a power-law distribution [50, 115]; i.e., if a histogram of empirical datasets is well fitted to a straight line on log-log scale, a power-law behavior would be ascribed to the datasets. However, recent studies [145, 164] show that these data-fitting methods are insufficient to conclude the power-law compliance of empirical data. According to Clauset *et al.* [41], the majority of purported power-law datasets fail to pass the rigorous statistical hypothesis test on their power-law distribution.

[9]The statistical tools, proposed by Clauset *et al.* [41], are available at `http://tuvalu.santafe.edu/~aaronc/powerlaws/`.

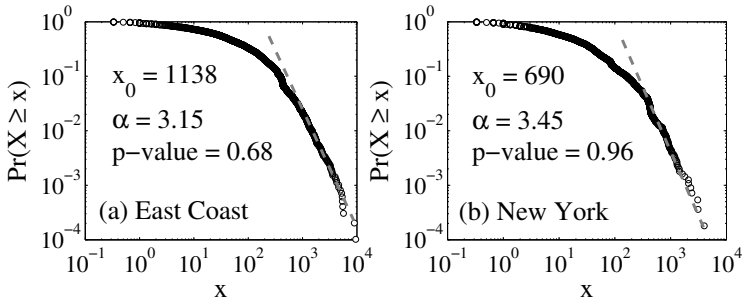

Figure 3.3: Flow-density distributions for various target areas: (a) East Coast and (b) New York. The complementary cumulative distribution functions (CCDFs) (i.e., Pr($X \geq x$)) of route count (x) for both areas are plotted on log-log scale.

shows the route-count distributions of two different target areas: (a) East Coast and (b) New York. The complementary cumulative distribution function (CCDF) (i.e., Pr($X \geq x$), where x is route count) of the route-count datasets is plotted on a log-log scale. As the graphs show, both distributions are well fitted to the diagonal lines at the tail. More precisely, we apply the power-law hypothesis test proposed by Clauset *et al.* [41] to the measured route-count dataset and obtain the p-value, which indicates the degree of plausibility of a hypothesis, for each test. The p-values for the two target areas (i.e., 0.68 and 0.96) are much higher than the significance level, which is often set to 0.05. Hence, the plausibility of the null hypothesis (i.e., the route-count distribution follows a power law) is accepted [130].

Target-area dependency of route count

Unlike connectivity-related metrics, which are dependent only on *physical* network connectivity but independent of attack targets, route count is an attack-specific metric; i.e., a target link that has high route count for a target area may have a very different density for other areas.

Table 3.1 illustrates the top 20 links ordered by flows densities for three target areas of different sizes: the East Coast, Massachusetts, and Univ2. Naturally, one would expect that the links' flow densities would follow the obvious link-map inclusion relations, namely the link map of Univ2 $\subset$ link map of Massachusetts $\subset$ link map of the East Coast. However, Table 3.1 shows that the top 20 links for these related target areas are very different: not only these links do not follow the link-map inclusion, but also whenever some are shared between areas they have different density ranks. For example, link **19** has the

Target area	Indices of top 20 flow-density links
East Coast	01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, **13**, 14, 15, 16, 17, 18, **19**, 20
Massachusetts	**19**, 21, **13**, 22, **23**, 24, 25, 26, 27, 28, 29, **30**, 31, 32, 33, 34, 35, 36, 37, 38
Univ2	39, 40, **30**, 41, 42, **23**, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56

Table 3.1: Top 20 route-count links for three different target areas: the East Coast of the US, Massachusetts, and Univ2. Each link IP address is mapped to a link index. Bold indices denote the links shared by different areas.

highest flow-density rank when the state of Massachusetts is targeted, and yet it only ranks next to the last for the East Coast. Furthermore, link **19** does not even appear in the top 20 link densities of Univ2. This clearly shows that route count is a target-area specific metric, which reveals a link's usefulness in an attack that targets a specific area.

3.3.2 Geographical Distribution of Bots

Although the selected target links are highly dependent on the target area of the attack, they are nearly *independent* of the choice of bot distributions; i.e., even if an adversary uses different sets of bots that have different geographic distributions to flood a target area, the effectiveness of the Crossfire attack would remain nearly unchanged. To show this, we performed the following experiment. First, we partitioned the set of bots into several subsets based on bots' geolocation (viz., subsets denoted by S_j, $j = 1, ..., 8$ in Table 3.2). Then, we selected different subsets to form six different bot distributions (viz., distributions denoted by $Distr_i$, $i = 1, ..., 6$ in Table 3.2), and simulated a separate Crossfire attack for each distribution against three different target areas; i.e., East Coast, Pennsylvania, and Univ1. Finally, we analyzed how the different distributions affect the degradation ratios.

The geographical distributions of 620 PlanetLab nodes and 452 LG servers are as follows: 42% were located in Europe, 39% in North America, 13% in Asia, and 6% in the rest of the world (viz., Figure 3.5).

	North America			Europe			Asia	Others
	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
Baseline	✓	✓	✓	✓	✓	✓	✓	✓
$Distr_1$	✓			✓	✓	✓	✓	✓
$Distr_2$		✓		✓	✓	✓	✓	✓
$Distr_3$			✓	✓	✓	✓	✓	✓
$Distr_4$	✓	✓	✓	✓			✓	✓
$Disrt_5$	✓	✓	✓		✓		✓	✓
$Distr_6$	✓	✓	✓			✓	✓	✓

Table 3.2: Different geographic distributions of bots ($Distr_i$) created using different subsets of PlanetLab nodes and LG servers (S_j).

Since the distributions of PlanetLab nodes and LG servers in North America and Europe cover wider areas than those in the rest of the world, we (1) assigned three disjoint subsets to each; i.e., S_1, S_2, and S_3 to North America and S_4, S_5, and S_6 to Europe; and (2) constructed the bot distributions such that $Distr_1$, $Distr_2$, and $Distr_3$ cover a similar number of bots in North America and Asia, and $Distr_4$, $Distr_5$, and $Distr_6$ a similar number of bots in Europe and Asia.

Fig. 3.4 shows the degradation ratios for the six different bot distributions shown in Table 3.2 and three different-size target areas chosen; i.e., East Coast, Pennsylvania, and Univ1. For each target area, we defined a baseline degradation ratio (denoted by "Baseline" in Fig. 3.4) as the degradation ratio given by an attack launched by *all* bots available. The six degradation ratios are computed using the same total number of routes as that used in the baseline ratio. Thus, if the degradation ratio of a certain distribution is close to the baseline, that distribution of bots is as damaging to the target area as the baseline (i.e., as all available bots). As shown in Fig. 3.4, the choice of the six different distributions does not diminish the effectiveness of the attack in a measurable way. That is, the effectiveness of an attack is nearly independent of the geographical distribution of bots. This is particularly noticeable in the case of the small and medium target area where the degradation ratios are almost indistinguishable from the baseline.

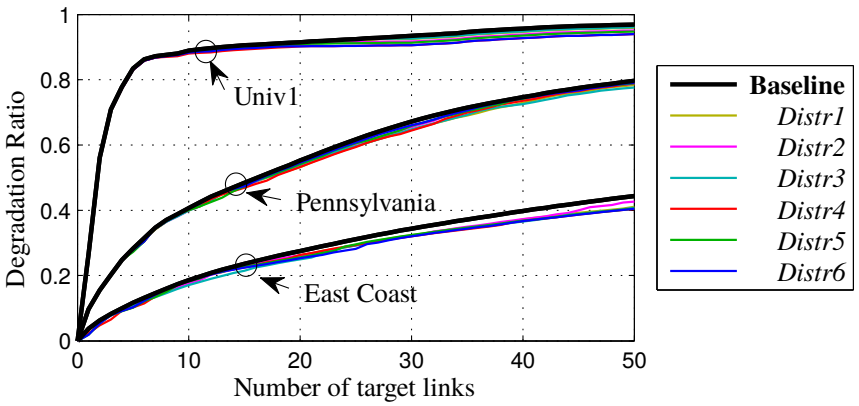

Figure 3.4: Degradation ratios for different geographic distributions of PlanetLab nodes and LG servers.

3.4 Attack Persistence and Cost

3.4.1 Data-Plane-Only Attack: Indefinite Duration

In this subsection, we discuss how the Crossfire attack maintains its effectiveness, namely a high connection degradation ratio for selected target areas caused by link flooding (data plane only), by avoiding any route change (by the control plane) in the Internet. Clearly, the goal of the adversary is to avoid control plane reaction since that would cause routes to change dynamically in response to any unexpected network-state variations (e.g., due to link failures or high traffic load akin to link flooding).

The Crossfire attack takes advantage of the fact that the current Internet's dynamic response to link flooding is too slow for an adaptive adversary. That is, if the adversary periodically changes the set of predetermined target links in less than 3 minutes, she can maintain a very high connection degradation ratio without inducing any Internet route changes. Thus, the attack duration can be extended virtually indefinitely. The technique of changing the set of target links, namely the rolling attack, is discussed in detail in Section 3.4.2. The following two subsections illustrate how slowly the current Internet would react to the Crossfire attack.

Link failure detection

Link-failure detection refers to a function of a routing protocol that enables a router to assess the physical connectivity of its network link to its neighbor router [135]. A router which misses several consecutive

control packets (e.g., *hello* packets for OSPF or *keepalive* messages for BGP) in a specific time interval (default 40 seconds for OSPF or default 180 seconds for BGP) will conclude that the link failed and broadcast the link failure to other routers. The consequence of the link failure is two-fold. First, if an intra-AS link fails, the failure notification is sent to all the routers within the same AS, which leads to internal topology changes. In contrast, if a link between two neighbor ASs (i.e., an inter-AS link) fails, the failure, in the worst case, could propagate to all the BGP speaking routers in the Internet and cause a global topology change. These topology changes would redirect the attack traffic to alternate routes and invalidate the route counts computed for the on-going Crossfire attack.

To measure Internet reaction to link failures, Shaikh *et al.* [135] inject traffic that consumes 100% of the capacity of a link and measure the time for the router to detect the link failure. This experiment shows that it takes 217 seconds for a IGP router (that runs OSPF or IS-IS) and 1,076 seconds for a BGP router to diagnose congestion as a failure[10]. Note that failure detection takes much longer than its default waiting time interval for the control packets, namely 40 seconds for OSPF and 180 seconds for BGP. This is because some control packets that are queued at the congested interface at a router can successfully reach a neighbor router even in severe link congestion. Clearly, the congestion diagnostic times are too long to enable rapid reaction to the Crossfire attack where the adversary can change the set of target links for an area in much less than 3 minutes; viz., the rolling attacks of the next subsection.

Traffic engineering

Most commonly, ISPs use *offline* traffic engineering techniques, whereby network parameters are periodically re-optimized based on the estimated traffic matrix among the ingress/egress points of their networks [159]. The network parameters can be the link weights of IGP protocols (e.g., OSPF or IS-IS) in pure IP networks [57] or bandwidths of LSP (label switched path) tunnels in MPLS networks [46, 114]. Offline traffic engineering produces new routes on a time scale ranging from tens of minutes to hours and days [55], though more commonly in days and weeks [46, 114, 159]. Even though it is not frequently

[10]We assume that the OSPF and BGP protocols do not use shorter intervals for fast failure detection [58], but use default timers (HelloInterval & RouteDeadInterval for OSPF and KeepaliveTimer & HoldTimer for BGP). Since most optical fiber connections (e.g., SONET or SDH) provide failure reports in less than 50 ms, additional system configuration for faster link failure detection at layer-3 is obviously unnecessary [78].

used by ISPs due to its potential instability problem, *online* traffic engineering occurs on a smaller time scale, namely from minutes to hours [94, 159]. Given that the adversary can repeatedly relaunch the Crossfire attack for new routes, neither current offline nor online traffic engineering can offer effective countermeasures.

3.4.2 Proactive Attack Techniques: the Rolling Attack

A Crossfire attack is said to be *rolling* if the adversary changes the attack parameters (e.g., bots, decoy servers, and target links) dynamically while maintaining the same target area. A rolling attack can be employed by an adversary to further increase indistinguishability of attack traffic from legitimate traffic and undetectability of all target links by target area. Based on the types of attack parameters that can be dynamically changed, rolling attacks can be categorized into two types: one that changes bots and decoy servers while maintaining the same target links, and the other that changes target links while maintaining the same target area.

The main advantage of the first type of attack is that it further increases the indistinguishability of the Crossfire flows from legitimate flows while maintaining the same attack effects. Since the source and the destination IP addresses seen at the selected target links change over time, the ISPs cannot easily identify the source and the destination IP addresses that contribute to the attack. A potential disadvantage is that this attack requires more bots and decoy servers than the minimum necessary to flood the target links. However, the current cost of bots suggests that this disadvantage is insignificant (viz., discussion of bot costs below).

The second type of rolling attack uses multiple sets of *disjoint* target links for the same target area. To find the multiple disjoint sets, the adversary executes the target-link selection algorithm (viz., Section 3.2.2) successively; i.e., the n-th best set of the target links is selected after removing the previously selected links. The use of multiple disjoint sets of the target links enhances attack undetectability by ISPs since ISPs could not anticipate the adversary's choice of targets with certainty. More importantly, this type of rolling attack enables Crossfire to remain a pure data plane attack, as discussed in the previous subsection. A potential disadvantage is that this type of rolling attack may degrade the effectiveness of the Crossfire attack since the degradation ratio caused by attacking a non-best target set can be lower than that

Target area	Target link set		
	Best set	2nd best set	3rd best set
Univ1	89%	77%	63%
Pennsylvania	42%	30%	24%
East Coast	21%	16%	14%

Table 3.3: Degradation ratios for different disjoint target link sets. Each set has 10 target links.

of attacking the best set. However, the degradation ratios of different sets of target links shown in Table 3.3 indicate that this degradation is minimal. In order to maximize attack effects while being undetected, the adversary can alternate the target sets; she would use the best set for the most of attacks and switch to the non-best sets only for a short time interval. For example, if the adversary repeatedly schedules 3 minutes for the attack on the best set and next 30 seconds for the second-best set, she can maintain the attack towards a target area indefinitely while limiting the reduction of the degradation ratio less than 4%.

3.4.3 Bandwidth Bottlenecks at Non Targeted Links

The Crossfire attack floods a set of targeted routing-bottleneck links and turns them into *bandwidth-bottleneck* links; i.e., the links with the smallest available bandwidth on end-to-end routes. While flooding the targeted links, the Crossfire must ensure that *non targeted* links do *not* become bandwidth bottlenecked; otherwise, attack packets cannot reach the targeted links since they could be dropped at non targeted links.

In general, the Crossfire attack can prevent *bandwidth bottlenecks at non targeted links* by *dynamic assignment* of attack flows. First, bots can easily detect bandwidth bottlenecks at non targeted links by regularly performing traceroutes to the target area. Then, the adversary can re-assign some attack flows to other bots and remove the bandwidth bottlenecks at non targeted link. In other words, the adversary *adaptively* assigns attack flows to geographically distributed bots, so that a sufficient number of attack packets reach the target links and flood them without flooding non targeted links.

Note, however, that in some *unusual* network configurations, particularly when network bandwidth is *poorly* provisioned, the proposed dynamic attack-flow assignment strategy might *not* be able to avoid

bandwidth bottlenecks at non targeted links. For example, let us consider an ISP in which the sum of the bandwidth of its inter-domain links is *smaller* than a target internal backbone link bandwidth.[11] This unusual network bandwidth configuration can make it *impossible* to flood the target backbone link by the Crossfire attack since the inter-domain links would become bandwidth bottlenecks before the target link does. When facing such undesirable situations, adversaries can quickly find other link targets from the routing bottleneck.

3.4.4 Execution Time of Target Selection Algorithm

The greedy algorithm of selecting a set of T target links runs as follows:
Let $\mathcal{R}$, $\mathcal{L}$ and $\mathcal{T}$ be the set of all bot-to-target area routes, the set of candidate links for the target area, and the set of target links, respectively. Let l_i be a link on a route.

1. Add all distinct links ($l_i's$) of $\mathcal{R}$ to $\mathcal{L}$.
2. Take out the highest route count link, l_i^{max}, from $\mathcal{L}$ and add it to $\mathcal{T}$.
3. Recompute the route count for all l_i's in $\mathcal{L}$.
4. Repeat (2) and (3) until T target links are selected, i.e., until $|\mathcal{T}| = T$.

The above algorithm finds the T best target links that disconnect the target area in terms of the degradation ratio, in T iterations of steps (2) - (3). Step (3) re-evaluates route counts after removing all routes of $\mathcal{R}$ that include l_i^{max} and as a consequence, the step ensures that the adversary selects the target link that maximally disconnects the target area at each iteration. Table 3.4 shows the execution times taken by our experiments. As expected, the execution time is proportional to the number of target links (T) for all target areas, and grows significantly for a large target area (e.g., 52 seconds in selecting 50 target links for the East Coast of the US), since more unique links can be found in large target areas. However, the number of unique links is bounded by a *limited* number of routes. This number is limited because bot-decoy pairs in the same source and destination subnets produce a *single* unique route. Hence, the execution time of the algorithm is short enough (e.g., at most a couple minutes) for an adversary to adapt to all potential route changes even for a large target area, in practice.

[11] Such bandwidth provision would be *uncommon* since typical ISPs provision the internal bandwidth with high oversubscription ratio (e.g., 10–100) [69].

Target area	$T = 10$	$T = 20$	$T = 30$	$T = 40$	$T = 50$
Univ1	0.94	1.87	2.79	3.72	4.65
Pennsylvania	3.10	5.46	7.38	8.99	10.38
East Coast	13.44	24.93	35.13	43.96	52.05

Table 3.4: Execution time (in seconds) to select T target links for different target sizes.

3.4.5 The Cost of the Crossfire Attack

To launch a Crossfire attack, an adversary needs bots. To get them, she can either infect user machines and install her own bots or buy the bots from *Pay-Per Install* (PPI) botnet markets [34]. For cost estimation, we assume that the adversary buys the bots from the markets. Our cost estimates are based on a recent analysis of PPI botnet markets [34]. A possible option would be to rent cloud services for bot operation from many, say one hundred, providers around the world. Given the low computation and communication requirements of Crossfire bots and the high-bandwidth connectivity of data centers to the Internet, the bots' behavior during an attack would not trigger providers' alarms.

PPI botnet markets have region-specific pricing plans. Generally, bots in the US or the UK are most expensive and cost \$100-\$180 per thousand bots. Bots in continental Europe cost \$20-\$60 whereas bots in the rest of the world cost less than \$10 per thousand bots. The mix of bots used in our experiments (presented in Section 3.3.2) has 49% of bots in the US or UK, 37% in continental Europe, and 14% in the rest of the world. If we assume the size of a bot cluster (β) is 500, the total cost of the Crossfire attack is roughly \$46K. Our experiments also show (viz., Section 3.5.4) that the minimum number of required bots that can flood 10 target links can be as low as 107,200 bots, and hence the attack cost can be as low as \$9K. This implies that a single organization or even an individual can launch a massive Crossfire attack. If the attack is state- or corporate-sponsored, many more bots can be purchased and a much larger number of links can be targeted. In this case, the Crossfire attack could easily disconnect almost 100% of the Internet connections to a large target area.

3.5 Experiment Setup and Results

In this section, we demonstrate the feasibility of the Crossfire attack and its effects on various target areas using real Internet data. In particular, we show how one sets up the bots, decoy servers, and target area for a Crossfire attack.

3.5.1 Bots

Instead of using real bots to perform our experiments, which would raise ethical [8, 79] and/or legal concerns [53], we use *PlanetLab nodes* [125] and *Looking Glass (LG) servers* as attack sources. PlanetLab is a global research testbed that supports more than one thousand nodes at 549 sites. An LG server is a publicly available router that provides a Web-interface for running a set of commands, including *traceroute* [150]. They have been used as vantage points for discovering Internet topology [22, 146, 157].

The PlanetLab and LG server networks provide a faithful approximation of a globally distributed bot network. As seen in Fig. 3.5, the 620 PlanetLab nodes and 452 LG servers are located 309 cities in 56 countries. In Section 3.3.2, we will show that different bot distributions created using PlanetLab nodes and LG servers, result in practically the same attack effectiveness. Hence, the Crossfire attack using real bots (e.g., leased from botnet markets) would experience similar attack effects as in our experiments. A single PlanetLab node or LG server represents several hundred bots, given (1) the high degree of clustering observed in real-bot distributions [44, 140], and (2) the fact that bot-originated traffic from the same AS domain would converge at a router and then follow the same route, due to the BGP's single best route selection policy. Hence, the routes we trace from the PlanetLab nodes or LG servers to the public servers in the target area, allows us to build the actual Internet link map of the target area. We call the group of bots represented by the same PlanetLab node or LG server a *bot cluster*, and experiment with cluster sizes of 100, 200, and 500 bots.

3.5.2 Decoy servers

Decoy servers, which are the destinations for attack traffic, can be any public server whose physical location is nearby a target area. Among various possible ways an adversary could select decoy servers, one way is to find servers of public institutions (e.g., universities and colleges) physically located surrounding

Figure 3.5: A map of geographic locations of the 620 PlanetLab nodes (red pins) and 452 LG servers (blue pins) used in our experiments.

the target area. For example, the servers of a university or college are typically located on their campus[12].

We found 552 institutions (i.e., universities and colleges) on both the East Coast (10 states) and West Coast (7 states) of the US, which can provide large numbers of decoy servers. The list of institutions in a specific US state is easily found on the Web[13]. An adversary can find a minimum of 1,000 public servers within an institution. For example, we found 2,737 and 7,411 public web servers within Univ1 in Pennsylvania and Univ2 in Massachusetts, respectively, via port-scanning. Had we used real bots, port scanning duties would be distributed to each bot and would be performed over a period of time, to avoid triggering IDSs or firewall alarms at those institutions. Similarly, an adversary could use 351,000 public servers located in 351 institutions on the East Coast of the US, and 201,000 public servers in 201 institutions on the West Coast.

[12]The adversary might use a public search engine, such as SHODAN (`http://www.shodanhq.com`), to gather a large number of publicly accessible IPs at a geographical location. However, use of SHODAN would require cross-validation of the IP addresses in a geolocation due to possible search inaccuracies. Cross-validation would be a fairly simple matter of comparing results of multiple IP geolocation services for a certain target area

[13]`http://www.4icu.org/`

3.5.3 Target area

A target area is the geographic location where an adversary wants to block Internet traffic. To establish that the Crossfire attack works for various target-area sizes, we used three different configurations: small, medium, and large. For the small area size, we set a single organization as the target area. Specifically, we set Univ1 and Univ2 as examples of small-sized target areas. As examples of medium-sized areas, we picked four US states, namely New York, Pennsylvania, Massachusetts, and Virginia. Finally, we picked ten states on the East Coast and seven on the West Coast as two examples for large target areas. Note that the large target areas' sizes could conceivably represent a medium-size country. For a small or medium target area, we chose decoy servers outside the target area for the undetectability of attack flows. However, for a large target area, we chose decoy servers inside the target area since the wide array of decoy servers within the area would not diminish the Crossfire's undetectability.

Table 3.5 illustrates the extrapolated numbers of public servers in the target areas and decoy servers used for attacking those areas. Note that the extrapolation is based on that an adversary can find 1,000 public servers within an institution.

3.5.4 Results

We performed Internet-scale experiments to verify the feasibility and the impact of the Crossfire attack based on the steps described in Section 3.2. For each attack target area illustrated in Table 3.5, we construct a link map (Step 1, viz., Section 3.2.1) and select the target links (Step 2, viz., Section 3.2.2), using the PlanetLab nodes and LG servers, and public servers in the target area. Bot-coordination (Step 3, viz., Section 3.2.3) is performed via simulations, for obvious ethical and legal reasons. However, the simulations use the real link map and data obtained from the first two attack steps illustrated in Fig. 3.2. In this section, we summarize the results of our experiments.

Link map. We gather traceroute data from all the PlanetLab nodes and LG servers (i.e., sources) to all the institutions in the target areas (i.e., destinations) and construct the link maps centered on the target areas of the East and West Coasts of the US. For each source-destination pair, we run a traceroute six times to diagnose link persistence. Since multiple traceroute packets (i.e., ICMP packets) to the same destination are independently load-balanced at a load-balancing router [21], running six traceroutes is

Target area	Number of public servers in target area	Number of decoy servers
Univ1	1,000	350,000
Univ2	1,000	350,000
New York	86,000	265,000
Pennsylvania	82,000	269,000
Massachusetts	54,000	297,000
Virginia	34,000	317,000
East Coast (US)	351,000	351,000
West Coast (US)	201,000	201,000

Table 3.5: The extrapolated numbers of public servers in target areas and decoy servers used for attacking each target area in our experiments

enough to determine whether a link on the route is persistent or transient. We classify a link as persistent if the link appears in all six traceroute results. The false positive probability, namely the probability that we falsely determine a transient link as persistent, is at most 0.016 ($\simeq 2^{-6}$). This is the case because the highest false positive probability is reported when a router, which has *two* load-balancing links to the next hop router, happens to select the *same* link in forwarding six traceroute packets originated from the same source. If the router has more load-balancing links, the false positive probability becomes lower.

We summarize the percentages of persistent links found by traceroutes in Table 3.6. Regardless of the size of a target area, the majority of the discovered links are persistent and hence can be used for the Crossfire attack. This result shows that even though traffic load-balancing through multiple links is widely implemented by ISPs in the current Internet, a large portion of Internet links are persistent. This enables the adversary to easily find (persistent) target links. In the following subsection, we discuss how the adversary finds the target links whose congestion would effectively disconnect a target area.

Link Coverage. Although one could not demonstrate that all links leading to a target area can be found by traceroute, one could show that *all critical links* can be found for a target area. To show this

Target area	Percentage of persistent links
Univ1	79.99 %
Univ2	70.37 %
New York	69.70 %
Pennsylvania	75.68 %
Massachusetts	74.11 %
Virginia	70.32 %
East Coast (US)	71.78 %
West Coast (US)	72.37 %

Table 3.6: Percentage of persistent links per target area

we selected different uniformly-distributed subsets of the 1,072 bots used (i.e., PlanetLab nodes and LG servers); e.g., subsets of 10%, 20%,..., 90% of all bots. We computed their degradation ratios for three target areas and plotted those against the *baseline degradation ratio* produced by all 1,072 bots. Figure 3.6 shows that, for each target area, beyond a certain bot-subset size, the differences in deviations from the baseline degradation ratios taper off, indicating that additional critical links which would increase degradation ratios can no longer be found; i.e., that size is approximately 10% of all bots for Univ1, 20% for Pennsylvania, and 50% for the East Coast. In similar experiments, if we vary server-subset sizes beyond a certain target-area related *threshold*, additional critical links that would increase the degradation ratios could not be found any longer. These two experiments suggest that the critical links we find adequately cover the flows toward a target area.

Route count. To compute route counts of all persistent links of the link map, we count the number bot-to-target area routes on those links. As expected, the distribution of route counts is highly non-uniform, namely it follows a power-law distribution; i.e., a few links have unusually high route counts while most of the other links have much lower route counts (viz., Section 3.3.1). The power-law distribution of route counts makes the Crossfire attack very effective indeed. That is, flooding only a few high flow-density links would effectively disconnect a large number bot-target area routes.

After computing the route counts of all persistent links, we select a set of target links using the greedy

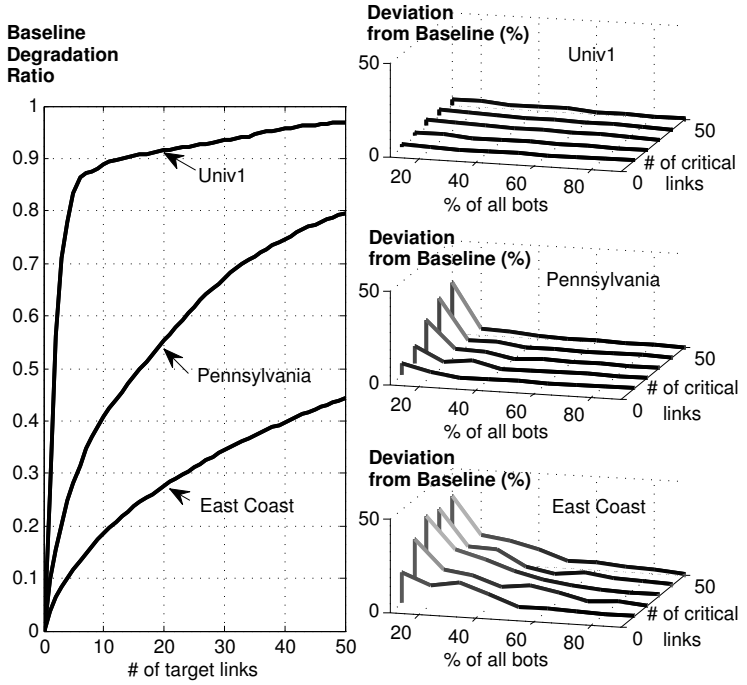

Figure 3.6: Deviations from baseline degradation ratios for different bot subsets.

algorithm specified in Section 3.4.4. Recall that we do not select links that are located close to a target area (more precisely, links whose distance from the target area is less than or equal to three hops) to avoid attack detection by any servers in the target area. For example, the average hop distance from the selected target links to Univ1 and Univ2 are 3.67 and 4.33, respectively[14]. Note that even though we eliminate links that are less than three hops away from the target area, we can effectively find target links with sufficiently large route counts as discussed in the following subsection.

Degradation ratio. Fig. 3.7 shows the degradation ratios for various target areas with different numbers of target links. As shown in this figure, the increase in the degradation ratio achieved by flooding additional target links diminishes as we flood more links; e.g., flooding the first five target links for attacking Univ1 results in an 83% degradation ratio whereas flooding five additional target links increases the degradation ratio by only 6%. This trend clearly shows that the power-law distribution of the route count enables the adversary to achieve a high degradation ratio by flooding only a few target links. In general, the smaller the target area, the higher degradation ratio, because smaller target areas have relatively few

[14]For medium and large areas, the hop distance can be measured relative to the peripheral servers.

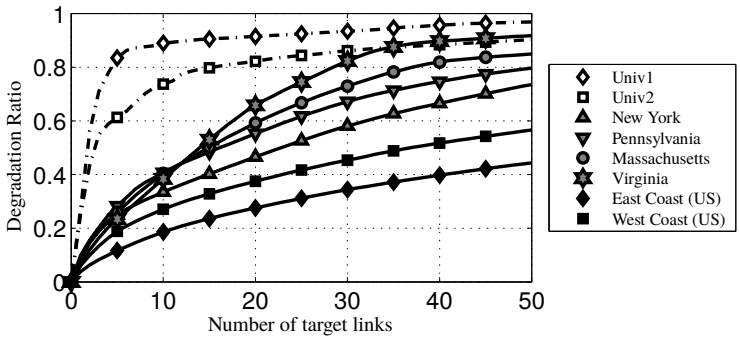

Figure 3.7: Degradation ratios for various target areas for different numbers of target links.

links that deliver most of the traffic to them. For example, when flooding 15 target links, the degradation ratio of a large area (i.e., West Coast of US) is as high as 32.85%, that of a medium area (i.e., Virginia) is as high as 53.05%, and that of a small area (i.e., Univ1) is as high as 90.52%. This result may be misinterpreted and conclude that the Crossfire attack would damage only small target areas. In reality, when the attack effects are measured in terms of the *total number* of effectively disconnected end-users (or hosts) in a target area, the attack appears to be far more lethal to a large target area than a small one. For example, a Crossfire attack against West Coast using 15 target links effectively disconnects only 32.85% of traffic, yet the number of affected servers is huge.

Attack bots and flows. To flood the selected target links, we assign attack flows to bots by providing the list of decoy server IPs and corresponding flow rates. In our experiments, we set a 4 Kbps per-flow rate, which can be achieved by sending one HTTP GET message per second, for the indistinguishability of the Crossfire attack. While maintaining the low per-flow rate, we assign the attack flows evenly to the multiple bots and decoy servers. We conservatively assume that the bandwidth of target links is 40 Gbps, which ensures the presence of at least 10^7 (i.e., 40 Gbps$/4$ Kbps) attack flows through each target link.

Fig. 3.8 shows the per-bot and per-decoy server average send-rates for three target areas of different sizes when flooding ten selected target links. Notice that for the large bot cluster size (β), we achieve lower per-bot send-rate since the attack flows can be more evenly distributed. An important observation is that for any target area, the per-bot average send-rate can be much lower than 1 Mbps when the bot cluster size (β) equals 500 (i.e., 536,000 bots in total). This shows that the adversary can aggregate a sufficiently large number (i.e., 10^7) of low-rate (i.e., 4 Kbps) attack flows at each selected target link and

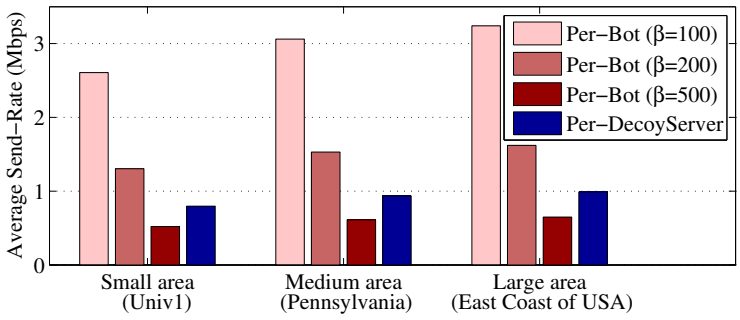

Figure 3.8: Per-bot, per-decoy server average send-rates for different bot cluster sizes (β).

thus successfully exceed the bandwidth (i.e., 40 Gbps) of the target link while maintaining low per-bot and per-decoy server average send-rates. If the adversary uses more bots and decoy servers in practice, these average rates would become even lower.

3.5.5 Approximation of Real-World Attacks

We claim that our attack evaluation is a good approximation of real-world Crossfire attacks despite some measurement limitations. In particular, our evaluation lacks the measurement (or accurate estimation) of the bandwidth of targeted and non targeted links. Without the link bandwidth measurement, it is difficult to analyze whether it is possible to flood a particular link target in real-world attacks without causing any bandwidth bottleneck at non targeted links; viz., Section 3.4.3. Therefore, some link targets found in our evaluation can be in fact difficult to flood in real-world attacks and there could exist a gap between our evaluation and real-world attacks in terms of degradation ratio.

However, this gap should be small and our evaluation is a good approximation of real-world attacks because the real-world Crossfire attacks can *adaptively* change the link targets whenever a link is determined as a hard-to-flood target. That is, the Crossfire attacks can quickly identify if a link target is hard be flooded due to the problem of bandwidth bottlenecks at non targeted links and then easily change its link target to another link in the routing bottleneck. Crossfire can achieve effectively the same degradation ratio even after changing the link target, since there are plenty of other link-target candidates in the routing bottleneck for a given target area.

3.6 Attack Characteristics

The Crossfire attack has four distinct characteristics which distinguish it from ordinary DDoS attacks, namely *undetectability*, attack-flow *indistinguishability*, *flexibility* in the choice of targets, and *persistence* in terms of attack duration.

Undetectability at the Target Area. The Crossfire attack uses all *legitimate* flows to flood target links. Each bot creates *ordinary* connections (e.g., HTTP) with a set of decoy servers following the adversary's (i.e., the bot-master's) assignments, and hence individual connections do not trigger an attack alarm at the servers. Since a target area is not directly attacked and the decoy servers near the target area do not see any suspicious traffic, the servers in the target area would be unable to detect the attack. Even decoy servers would be unable to detect the attack since the well-coordinated flows to the decoy servers would cause only a few Mbps bandwidth increase to each server. Furthermore, the adversary can easily select target links among the links in the target set that are several hops (i.e., at least 3 hops in our experiments) away from the target area since links with high route count are usually located in the core backbone networks. This makes it difficult even for the target links to identify an attack.

Indistinguishability of Flows in Routers. In the Crossfire attack, a large number of low-rate attack flows pass through a target link. Hence, a router connected to the target link cannot distinguish the attack flows from legitimate ones. In other words, since all the attack flows carry different source IP addresses and destination IP addresses, the high bandwidth aggregation mechanisms (e.g., Pushback [99], PSP [38]) become ineffective even if they are employed at all routers along the attack routes. Inspecting the payload of each packet would not help either because the attack flows carry the same payload as that of legitimate flows. Moreover, flooding target links with different sets of bots (e.g., the rolling attack, viz., Section 3.4.2) would further enhance this inherent indistinguishability of attack flows in routers.

Persistence. The Crossfire attack is able to disconnect a target area persistently by controlling the bot traffic so as not to trigger any control plane changes (e.g., route changes). This is achieved by using stable routes in rolling attacks, which change an active set of target links dynamically (viz., Section 3.4.2). In essence, a rolling attack makes the Crossfire attack a *pure data plane attack*, thereby leaving the control plane of the Internet unchanged. This extends the attack duration virtually indefinitely. The details of the attack persistence are presented in Section 3.4.1.

Flexibility. The Crossfire attack can be launched against *any* target area (regardless of its size) since an adversary can usually find a large number of public servers inside that target area and decoy servers near it; e.g., the adversary can select any of the many publicly accessible servers without needing permission from that server. This offers a great deal of flexibility in the adversary's choice of a target area, which is one of the most important characteristics that distinguish the Crossfire attack from other link-flooding attacks (viz., Related Work in Section 3.7). Our adversary's choice is enhanced by its low-rate flows used by the bots since the resulting attack flows would not trigger individual alarms in any potential target area.

3.7 Related Work

3.7.1 Control Plane DDoS Attacks

DDoS attacks against a network link, even if launched with low-rate traffic, can disrupt a routing protocol (e.g. BGP) and ultimately trigger instability in the Internet. This class of attacks, which we call Control Plane DDoS attack, first proposed by Zhang *et al.* [174], exploits the fact that the control plane and data plane use the same physical medium. This fate-sharing allows an unprivileged adversary to convince a BGP speaking router that its BGP session has failed. Schuchard *et al.* in [133] extended this attack to multiple BGP sessions, which were selected based on the betweenness centrality measures of the network topology. They showed that their CXPST attack can generate enough BGP updates to cripple the Internet's control plane.

In contrast, the Crossfire attack is pure data plane attack, which maintains the effects of the attack persistently by suppressing any control plane reaction.

3.7.2 Attacks against Links

The recent Coremelt attack [144] demonstrates how a set of bots can send packets to each other and flood a set of AS backbone routers. The key characteristic of Coremelt is that it creates only wanted traffic and thus it eludes all defense mechanisms that filter unwanted traffic. Furthermore, this traffic is not subject to the congestion-control mechanisms of TCP and can thus exceed typical TCP traffic bounds. This unique advantage cannot be exploited in Crossfire, since the ends of its attack flows are not bots.

Design Goal	Crossfire	Coremelt
Flexibility of targeting server areas	High	N/G
Bot-distribution independence	Y	N
Persistence · Data vs. control plane distinction · Robustness against route changes	Higher	Lower
Distribution of target links across multiple ISPs	Y	N/G
Indistinguishability at routers	Y	Y*
Undetectability at target area servers	Y	N/G
Reliance on wanted flows only	N	Y

(* only if bot-to-bot flow intensity does not exceed router bounds.

N/G = "Not a design Goal")

Table 3.7: Crossfire vs. Coremelt [144] Differences

Thus, Crossfire uses protocol messages that are unencumbered by congestion control; e.g., HTTP GET requests. In contrast with Coremelt, Crossfire creates very low intensity traffic (e.g., 4 Kbps flows) to decoy servers, which can be *any* public IP addresses. Furthermore, it can flood any of the *selected* target links regardless of the distribution of bots, and its server-disconnection effects at a target area are easily predictable. Crossfire is more *persistent* than Coremelt, since it does not trigger control-plane reaction (e.g., BGP route changes [133]) and it can easily evade route-change countermeasures produced by online traffic engineering. Finally, unlike Coremelt, which targets the backbone routers of an AS, Crossfire aims to select routers and links that are distributed across ASs of different ISPs, such that no single ISP could counter the attack. In short, the Crossfire attack is different from Coremelt as it shares neither all the goals nor the attack techniques of Coremelt. Table 3.7 summarizes the key differences between Crossfire and Coremelt.

3.7.3 Large-Scale Connectivity Attacks

The technical underpinnings of the Crossfire attack are also related to research on the robustness of Internet connectivity to attacks that disable routers or links [13, 98, 162]. Albert *et al.* [13] illustrate that if an adversary disables 4% of the highly connected routers, the entire Internet would break up into small isolated pieces. However, later work by Magoni [98] and Wang *et al.* [162] concludes that all such attacks would be *infeasible* because of the huge number of routers or links that need to be disconnected.

The main distinction between the Crossfire attack and this line of work is that our notion of (dis)connectivity captures the practical realities of the Internet; we say that a node A is (effectively) disconnected from a node B whenever the persistent route from A to B is severely congested (viz., Section 3.2.1).

The Crossfire attack also has a clearly different goal from the routing attack proposed by Bellovin and Gansner that cuts multiple network links to attract a certain traffic to compromised routers for eavesdropping purposes [28].It is also different from the DoS source-detection technique proposed by Burch and Cheswick [33] whereby a victim server attempts to flood various routers and measure decreasses in attack traffic – a telltale sign indentifying attack sources on the router's path.

3.7.4 Brute-Force DDoS Attacks

The goals of the Crossfire attack are fundamentally different from those of conventional brute-force DDoS attacks [67, 83, 109, 110] in at least three respects. First, it has a flexible choice of targets in a much more scalable range than those of DDoS attacks (e.g., from servers of a single enterprise, to those of a state or country). Second, its attack sources (i.e., bot hosts) are undetectable by any targeted servers, since they do not receive attack messages, and by network routers, since they receive only low-intensity, individual flows that are indistinguishable from legitimate flows. Third, its persistence against the same set of targets can be extended virtually indefinitely by changing attack parameters. The Crossfire advantage of the flexible choice of targets in a geographic area is shared by the geo-targeted DDoS attacks in cellular networks proposed by Traynor *et al.* [151]. However, these attacks are less relevant for the Internet.

Chapter 4

SPIFFY: A First Line of Defense that Deters Cost-Sensitive Link-Flooding Attacks

4.1 Outline of the Chapter

As we have seen in the previous chapters, link-flooding attacks can cause massive connectivity degradation when the attacks flood routing-bottleneck links. To *prevent* the link-flooding attacks, one could remove the root cause of the problem, namely the routing bottlenecks. However, since routing bottlenecks are the results of employing a cost-minimizing policy of the Internet routing and topology designs (viz., section 2.2.3), any attempt to remove routing bottlenecks would face a *defender's dilemma*: How can one remove a vulnerability of a system when it is caused by a very desirable feature of the system's design and operation? This dilemma suggests that a fundamental countermeasure would require major changes to the underlying Internet design and economic models.

As an alternative, one can consider a countermeasure that mitigates the effectiveness of link-flooding attacks at the individual routing-bottleneck links. However, as we saw with the example of the Crossfire attack in Chapter 3, it is also challenging due to the *indistinguishability* property of the attack flows. That is, advanced attacks can easily use low-rate, protocol-conforming attack flows that are indistinguishable

from legitimate flows by the targeted links, rendering existing filtering mechanisms (e.g., elephant-flow detection [89, 171]) ineffective. In this chapter, *flows* are defined by 5-tuple (i.e., source and destination IP addresses, source and destination port numbers, and protocol number) and thus each flow is individually rate controlled by senders TCP congestion control algorithms. A few recent proposals (e.g., CoDef [93], SIBRA [26]) appear to be effective in handling link-flooding attacks, but they require global AS coordination, which would not be readily available in the current Internet.

In essence, prevention or mitigation of link-flooding attacks require large-scale adoption of new protocols and extended deployment period. However, as we discussed in Section 1.1, link-flooding attacks are the *current* problem of the Internet and thus we need a solution to handle link-flooding attacks in the current, not the future, Internet.

To that end, we propose a *two-tier* defense approach, where (1) a *first-line* defense *deters* only cost-sensitive adversaries using low-cost defense operations that can be readily deployed in the current Internet and (2) a *second-line* defense handles the undeterred, cost-insensitive adversaries by using a multi-domain coordinated defense mechanism that is hard to orchestrate in the current Internet. In particular, in this chapter, we focus on designing a *low-cost deterrence* mechanism as our first line of defense.

Our first-line defense mechanism targets *rational* adversaries *only*; i.e., *cost-sensitive* adversaries who wish to remain *undetected*. All other (e.g., irrational, cost-unbounded) adversaries would not be deterred by our first-line defense mechanism and need to be handled by a second-line of defense mechanism. We believe that the majority of link-flooding attackers are rational in the current DDoS attack landscape. According to a recent study in behavioral economics [126], there is a strong evidence that cyber criminals are economically motivated. Also, rational adversary behaviors in DDoS attacks are observed in a study that analyzed real DDoS attack incidents in 240 countries over 5 years [77]. Note that even though only a fraction of adversaries turn out to be rational, our proposed first-line defense mechanism can be useful since it has low-cost deployment and operation cost.

To deter economically motivated (or cost-sensitive) adversaries, we focus on one economic property that is exploited by link-flooding attacks; namely, attackers have a fundamental *cost-asymmetry* advantage with respect to defenders. To be specific, on the one hand, the cost of flooding a 10 Gbps network link can be as low as US $80 and averages US $920, assuming 1 Mbps upload bandwidth per bot [34]. On

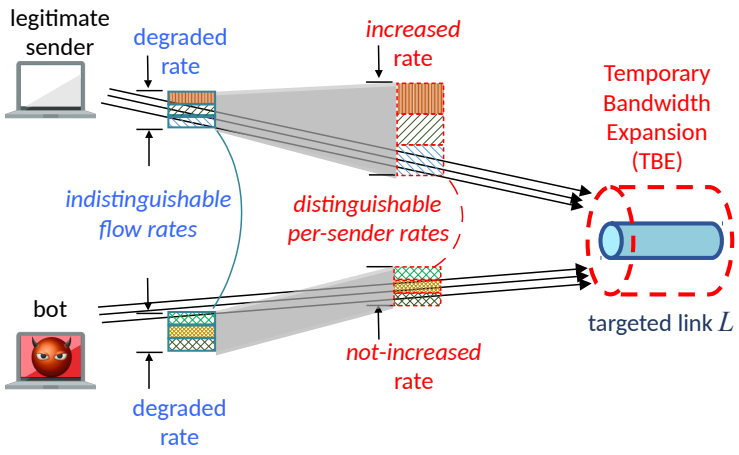

Figure 4.1: Intuition for distinguishing legitimate senders from bots via temporary bandwidth expansion.

the other hand, the cost of the backbone link bandwidth is *orders of magnitude* higher. For example, 10 Gbps bandwidth in the Internet transit costs about US $6,300 as of 2015 [3]. This is approximately 7–80 times more expensive relative to the equivalent attack bandwidth. Unfortunately, removing the attack-defense cost asymmetry is very difficult to achieve since these costs are determined by two fundamentally independent markets, namely, pay-per-install bot markets [34] and Internet transit markets [152]. This cost asymmetry enables attackers to easily launch link-flooding attacks. Our approach is to reduce or even reverse the cost asymmetry to *deter* cost-sensitive adversaries. To that end, we design a low-cost, single-AS mechanism that significantly increases the attack cost of flooding a link target.

Our countermeasure creates an untenable tradeoff between the cost and detectability. By definition, any countermeasure that can either substantially increase the attack cost relative to the defense cost or induce detectability will deter attacks by *rational* adversaries; i.e., cost-sensitive adversaries who wish to remain undetected. In contrast, countermeasures for cost-insensitive, irrational adversaries are known to be harder and more expensive to orchestrate and deploy; e.g., CoDef [93], SIBRA [26]. Thus, our efficient deterrence is a very desirable first-line defense for cost-sensitive, rational adversaries.

We show, perhaps surprisingly, that is indeed possible to force the adversary into an untenable tradeoff that either increases the attack cost or forces detectability. The high-level intuition behind our approach is as follows; viz., Figure 4.1. Suppose we know the locus of the attack L (i.e., a specific ISP link) and we have some capability to *logically* increase the bandwidth of L by some factor M temporarily; viz., Section 4.5 for detailed techniques for implementing this logical bandwidth increase in Tier-1 or Tier-2

ISP networks. After the increase, we observe the response of the traffic source IPs that were traversing L. Now, legitimate sources running TCP-like flows will naturally see a corresponding increase in their throughputs as the bandwidth of their bottleneck link has increased. Attack sources, however, will not observe this increase as a rational cost-sensitive attacker would have chosen to *fully utilize* the available bandwidth of the upstream links of the sources in the first stage; i.e., before the temporary bandwidth expansion. Thus, the bottleneck bandwidth increase will induce no increase in the effective throughput of the attack sources. Alternatively, to avoid detection, the attacker could choose to keep each bot's attack traffic rate much lower than the available bandwidth of its upstream link. Note, however, that this will increase the number of required bots and thus *increase attack cost* proportionally. In essence, adversaries are forced to either allow their attack sources to be detected (via rate-change measurements) or accept an increase in attack cost. Note that the key requirement is to monitor the *change in throughput* for traffic sources after the bottleneck bandwidth increase; measuring the raw throughput itself alone will *not* help detection as the attack flows are indistinguishable from normal flows.

However, there are three practical challenges that need to be addressed before this high-level intuition can turn into a practical defense mechanism:

(1) **Implementing bandwidth expansion:** First, we need some mechanism for increasing the logical bandwidth of L with a sufficiently large expansion factor. Note that a larger expansion factor will: (a) make it easier to distinguish bots vs. legitimate sources (e.g., to create a clear separation accounting for measurement noise) and (b) equivalently increase the effective attack cost. However, it is infeasible and uneconomical for ISPs to have spare dark fibers for each link, and thus we need deployable mechanisms to virtually increase the bandwidth, if only temporarily.

(2) **Fast workflow:** Second, we need the defense workflow to be fast and responsive to be effective against real attacks. If the temporary bandwidth expansion and detection takes several hours, then the damage is already done.

(3) **Robust rate-change detection:** Third, we need per-sender rate change measurements at scale, which may in turn require high processing requirements on monitoring routers as well as high control overhead for reporting these measurements. Furthermore, this detection must be robust to TCP effects, especially given that many legitimate flows on the Internet are short flows.

We address these practical challenges and present the design and implementation of *SPIFFY*.[1] To address (1), SPIFFY presents a new traffic engineering [57] technique based on software-defined networking (SDN) whereby one can virtually increase the bandwidth by routing around the bottleneck. To address (2), we develop fast greedy algorithms to solve a traffic optimization problem, which would otherwise take several hours even with state-of-art solvers [2]. Finally, to address (3), we suggest simple sketch-based change detection algorithms that can measure rate changes with low overhead [88, 171]. We develop a proof-of-concept prototype using `POX` [7] and use a combination of real testbed evaluation and large-scale simulations to validate the effectiveness of SPIFFY against link flooding attacks. SPIFFY relies on SDN's centralized control and traffic visibility to develop the first-known defense against such link-flooding attacks.

Contributions: In summary, this chapter makes the following contributions:

- A low-cost solution to force link-flooding adversaries into an untenable tradeoff between cost and detectability, which provides an effective first-line defense;
- A bandwidth expansion mechanism for SDN via traffic engineering based on a fast heuristic for solving the underlying optimization problem;
- An SDN-based implementation of SPIFFY and an extensive evaluation demonstrating its robustness with realistic TCP effects.

4.2 Background and Threat Model

In this section, we review the types of link-flooding attacks we address in this chapter and then formally characterize the attacker goals and constraints.

Background: The link-flooding attacks we consider in this chapter target network links in the *core* of the Internet (e.g., backbone links in large ISPs or inter-ISP links) and create a large number of attack flows crossing the targeted links to flood and virtually disconnect them; viz., Figure 4.2. This is in sharp contrast to traditional DDoS attacks that aim to choke the resources of the end target; e.g., computation, memory, or access link bandwidth. Recent research (e.g., Coremelt [144] and Crossfire in Chapter 3) and real-life

[1] SPIFFY stands for handling 'Scalable Persistent Indistinguishable link-Flooding attacks with reduced cost asymmetrY.'

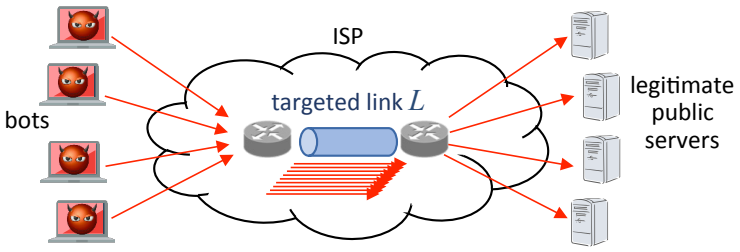

Figure 4.2: An example of link-flooding attacks. Legitimate looking connections between the bots and the legitimate public servers cross the link L in the ISP; viz., Chapter 3.

attacks against core routers of upstream networks (e.g., ProtonMail attack [68], Spamhaus attack [31]) are the examples of such attacks.

In the general case, link-flooding attacks may flood multiple link targets, as exemplified in Crossfire in Chapter 3. For simplicity of presentation, we focus on the link-flooding attacks against a single infrastructure link throughout this chapter. However, our system is also robust to multiple link-flooding attacks; viz., Section 4.8.3 for a detailed discussion.

Threat model: We consider a *rational* adversary who wants to inflict as much damage as possible on legitimate flows of the target network link using as few resources as possible, and while remaining indistinguishable. Formally, our link-flooding adversary pursues three goals:

- **Attack-Strength Maximization** ($G_{strength}$): Suppose the network a mechanism to guarantee a per-flow rate under "normal" network operation when there are no attacks; e.g., through a combination of link capacity provisioning and traffic engineering [23]. Let this *guaranteed rate* be denoted by r_g. The adversary aims to reduce the per-TCP-flow fair-share rate of flows traversing the target link to the *degraded rate*, denoted by r_d. The degraded rate will be much smaller than the guaranteed rate (i.e., $r_d \ll r_g$); otherwise (e.g., $r_d \sim r_g$) the attack would fail to degrade a legitimate flow much beyond the guaranteed rate r_g.

  The degraded rate r_d is an adversary-chosen parameter that measures the *attack strength*. A smaller r_d indicates a stronger attack since legitimate flow rates would be degraded more.

  Flow *rates* are defined as the number of bytes transferred within each time window divided by the time window size. We mainly focus on the rate changes in the order of few seconds (e.g., 5 second)

and thus unless noted otherwise we use the time window of size 1 second in this chapter.

- **Attack Persistence** ($G_{persistence}$): To circumvent detection and hence be persistent, a link-flooding attack needs to mimic legitimate traffic patterns. That is, the attack flows are indistinguishable from legitimate ones via traffic analysis of headers/payloads and/or intrusion detection at the target link. For instance, this can be achieved by using legitimate looking web sessions to decoy servers; viz., Chapter 3. To this end, the adversary uses TCP-based flooding attacks. Because TCP traffic constitutes the majority of the Internet backbone traffic, as it represents about 90 – 98% of the byte volume of the backbone links, these attacks are more difficult to detect and filter than UDP-based flooding attacks [172].
- **Attack-Cost Minimization** (G_{cost}): A *rational* adversary will seek to minimize the cost of the attack. In this chapter, we assume that the cost of the attack is proportional to the *number of bots* necessary for the attack; thus, the number of bots is a good proxy for the attack cost. This assumption is based on the observation that, in general, bots are sold in bulk (e.g., several thousands) in the pay-per-install markets [34].

We assume that the network follows a *per-flow fair-share* allocation of link bandwidth to all flows served. This is already widely observed in today's Internet since TCP flows adjust their rates in response to congestion, and thus approximate per-flow max-min fair rates [9]. If senders do not conform to the TCP flow control (i.e., they send flows faster than the fair-share rates) they can be detected by other mechanisms [89].

Based on this threat model defined, we develop and evaluate SPIFFY in the following sections.

4.3 SPIFFY Intuition and Security Analysis

In this section, we provide the intuition behind SPIFFY and the security analysis showing why it forces adversaries to either increase costs (G_{cost}) or forgo indistinguishability ($G_{persistence}$) while achieving rate degradation for legitimate flows ($G_{strength}$).

4.3.1 High-level Idea

The key reason why link-flooding attacks are so successful and dangerous is that they are affordable and indistinguishable. Thus, our overall goal is to force attackers to compromise on either $G_{persistence}$ or G_{cost} for a given $G_{strength}$; i.e., the attackers either become detectable or pay an increased cost.

The intuition behind our approach is as follows. During a link-flooding attack, a legitimate sender would only be able to send traffic at a much lower per-host rate compared to the desired application-layer data rates. This is because the attack with the rate-reduction goal ($G_{strength}$) decreases legitimate flow rates significantly. However, an attacker who is trying to optimize cost (G_{cost}) would have all its bots send at their highest per-host send rate (i.e., saturate its upstream bandwidth), by creating additional attack flows whenever its upstream bandwidth allows it. Due to these fundamental goal differences, a legitimate sender and an adversary's bot would react very differently when the congestion is *relieved*; viz., Figure 4.1. A legitimate sender would very likely increase its send rate to meet its rate demand (e.g., buffered traffic from application layer) due to TCP rate control while a bot would have no available bandwidth left for further rate increase.

We can implement the controlled congestion relief by what we call the *temporary bandwidth expansion* (TBE). That is, we temporarily increase the virtual bandwidth of the target link by some factor M to allow senders suffering from congestion to increase their send rates. TBE enables us to measure the rate increases of senders and ultimately distinguish bots from legitimate senders.

In order to prevent bots from being detected, a link-flooding adversary must give up fully utilizing the upstream bandwidth of bots and *mimic* the legitimate senders' rate increase when congestion is relieved, as we will see below. However, this rate-increase mimicry will lower the bandwidth utilization of each bot and in turn cause the link-flooding adversary to significantly increase the number of attack bots. As a result, the link-flooding adversary faces an *untenable choice*: (1) she could maintain the low-attack cost while allowing her bots to be detected, or (2) she could make bots indistinguishable while increasing the attack cost significantly.

4.3.2 Security Analysis

We begin by formulating the optimal attack strategy for a scenario *without* the SPIFFY defense and then argue why SPIFFY creates a fundamental cost-detectability tradeoff for link-flooding adversaries. For simplicity of presentation and without loss of generality, the following analysis assumes a *homogeneous* bot deployment where each bot has the same upstream bandwidth u. Let B be the bandwidth of the target link which is under the link-flooding attack.

Optimal adversary strategy without SPIFFY ($AS_{\neg spiffy}$): An adversary can optimally satisfy the attack goals $G_{strength}$, G_{cost}, $G_{persistence}$ by using B/u bots, where each bot creates u/r_d attack flows and saturates its upstream bandwidth u.

Proof. To congest the target and reduce the per-flow TCP fair-share rate $G_{strength}$, the attack first has to flood the target link. Thus, the minimum number of bots required for the attack G_{cost} is $n_b = B/u$. Also, due to TCP per-flow fairness, the fair-share rate provided by the target is $r_{FS} = \frac{B}{N_b}$, where N_b represents the total number of attack flows. This assumes no legitimate flows in the target link: the attack strategy designed without considering legitimate flows guarantees meeting the attack goals even when legitimate flows exist. Also, due to the rate-reduction goal the fair-share rate is reduced to the degraded rate $G_{strength}$, $r_{FS} = r_d$. Since N_b attack flows are created by n_b bots, on average each bot creates N_b/n_b attack flows, which is $N_b/n_b = (B/r_d)/(B/u) = u/r_d$.[2] □

An adversary with attack strategy $AS_{\neg spiffy}$ has already saturated the upstream bandwidth of each bot. As a result, bots cannot increase their sending rate by a factor of M and cannot avoid being detected by SPIFFY. We argue that for the adversary to evade the test, it must satisfy a property we call *rate-increase mimicry*.

- **Rate-increase Mimicry (RM):** Bots are capable of instantly increasing their send rate by a factor of M when congestion is relieved at the bottlenecked link. This implies that bots must use only u/M of their upstream bandwidth when congesting the target link.

The **RM** property enables the adversary to simultaneously satisfy $G_{strength}$ and $G_{persistence}$ while compromising G_{cost}. If all bots are capable of rate increase with a factor of M, they pass the SPIFFY test

[2] For simplicity, we ignore small errors generated when converting real values to integers.

and thus bots remain undetected. This leaves the adversary with the following new attack strategy under SPIFFY.

Optimal attack strategy with SPIFFY (AS_{spiffy}): The attack strategy must satisfy the two conditions to achieve the two attack goals $G_{strength}$ and $G_{persistence}$ under SPIFFY.

(1) the attack utilizes $M \cdot (B/u)$ bots and

(2) each bot creates $(u/r_d)/M$ attack flows by utilizing only $1/M$ of its upstream bandwidth u.

Proof. The proof is similar to that of the optimal attack strategy with SPIFFY ($AS_{\neg spiffy}$). However, due to the **RM** property, when attacking the target link, each bot uses only $1/M$ of its bandwidth limit u. Therefore, the attack requires $B/(u/M) = M \cdot (B/u)$ bots, where each bot creates $u/r_g = u/(M \cdot r_d)$ attack flows. □

Thus, a link-flooding adversary now faces the following mutually-exclusive options forcing a fundamental tradeoff between cost and detectability:

1. Adversary follows $AS_{\neg spiffy}$ and requires (B/u) bots, potentially allowing detection of his/her bots by SPIFFY.
2. Adversary follows AS_{spiffy} and requires $M \cdot (B/u)$ bots, circumventing the bot detection.

We argue that an adversary has no other options than those listed above. To see why, let us consider two attack strategies that differ from these: (1) *Per-flow rate increase strategy*: In this strategy, bots saturate their bandwidth to attain the cost-minimization goal G_{cost}. They quickly detect the bandwidth expansion and instantly allocate increased bandwidth to a set of selected flows by pausing (or terminating) other attack flows, making the selected flows look legitimate. However, since SPIFFY measures *per-sender* (**not** per-flow) rate changes, such bots would be detected due to their unchanged per-sender rates. (2) *Bot replacement strategy:* This strategy also saturates the bots to achieve G_{cost}. The adversary replaces his/her bots in operation with new bots whenever the current bots are detected by SPIFFY. Although this strategy can be efficient for a short period of time, the cost of maintaining the attack persistence grows linearly with attack duration increases since bots need to be replaced repeatedly.

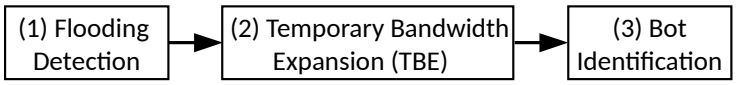

Figure 4.3: Workflow of SPIFFY

4.4 SPIFFY System Overview

In this section, we describe an end-to-end view of SPIFFY and highlight key practical challenges that we need to address to realize it. We envision SPIFFY being run by an ISP where the target link L is located, since the end customer who is the eventual target of the attack cannot detect or respond to link-flooding attacks. We believe that ISPs have a natural economic incentive to protect their immediate customers (e.g., as a value-added service [1]) and offer such capabilities on demand to create new revenue streams.

To understand the key challenges in this deployment model, let us consider the three logical stages in the SPIFFY workflow as seen in Figure 4.3:

1. **Flooding detection.** SPIFFY detects the existence of a link-flooding attack against a link (e.g., via SNMP-based link-utilization measurements [39]) and estimates the degraded rate r_d for the attack by measuring the fair-share flow rate of the target link.

2. **Temporary bandwidth expansion (TBE).** For *all* senders that use the target link, SPIFFY provides a temporarily expanded bandwidth $M\times$(current per-sender rate), where bandwidth expansion factor $M \gg 1$. For the time being, let us imagine an *ideal* TBE that increases the target link's physical bandwidth B to $M \times B$. Section 4.5 explains how TBE can be implemented in real networks. Note that the bandwidth expansion is *temporary* (e.g., $<$ 5 seconds) and the bandwidth of the link returns to B after TBE. The bandwidth expansion factor M is set to the ratio of the guaranteed rate r_g to the degraded rate r_d, namely, $M = r_g/r_d$, to let the legitimate senders increase rates from r_d to r_g in response to TBE. This value of M enables SPIFFY to identify senders with the per-sender rate change close to M as legitimate ones.

3. **Bot identification.** SPIFFY measures the *per-sender* rate changes of all the senders that use the target link. It starts measuring the per-sender rate before TBE and stops measuring after TBE. The frequency of measurements should be high enough to capture the rate increase and calculate the

ratio of the increase; e.g., every 1 second. Before TBE, flows from a legitimate sender will have the flow rate r_d (viz., $G_{strength}$), but during TBE the majority of the legitimate flows increases their rates at least up to r_g, and thus the total per-sender rate increases by a factor close to or higher than M ($= r_g/r_d$). In contrast, a bot would *not* increase its send rate even if the bandwidth allocated to it is expanded due to its saturated upload bandwidth; viz., Attack Strategy $AS_{\neg spiffy}$.

Challenges: Our focus in this chapter is on steps (2) and (3) of this workflow. We assume that existing monitoring mechanisms are used for (1); e.g., [39]. Our two key challenges arise for steps (2) and (3).

First, the challenge in designing TBE is to provide the senders significantly expanded bandwidth. Ideally, we want to physically increase the target link bandwidth, but this may not be viable unless the target network has spare dark fiber. Instead, our goal is to find an immediate solution that does not rely on spare optical fibers. Moreover, the operation of TBE has to be real-time to quickly react to the flooding attacks; e.g., in a few seconds.

Second, bot identification is challenging because it requires real-time per-sender rate measurements for *all* senders at the target link. In practice, it is difficult to keep track of these rate changes because the number of senders might easily go up to tens or hundreds of thousands. Finally, the rate-change estimation must be robust to real-world considerations; e.g., TCP effects in reacting to changes in the RTT or the impact on short legitimate flows.

Key ideas: We address these two challenges as follows. SPIFFY can leverage recent advances in software-defined networking (SDN) to implement the above workflow. An SDN's central controller provides new capabilities for network management [52, 71, 122, 128]. While we do not claim that SDN is *necessary* for countering link-flooding attacks, we consider it to be a natural enabler for realizing the SPIFFY workflow. The overall system is illustrated in Figure 4.4.

- *Practical TBE:* To enable practical TBE, we develop a traffic engineering application that dynamically changes traffic routing to meet desired goals [57]. At a high level, we increase the effective bandwidth of the link-flooding target link by routing flows around the bottleneck. We also provide practical techniques to work around the constraints of real networks where the bandwidth expansion factor (M) might be low. Finally, we develop fast heuristics to solve the traffic engineering optimization.

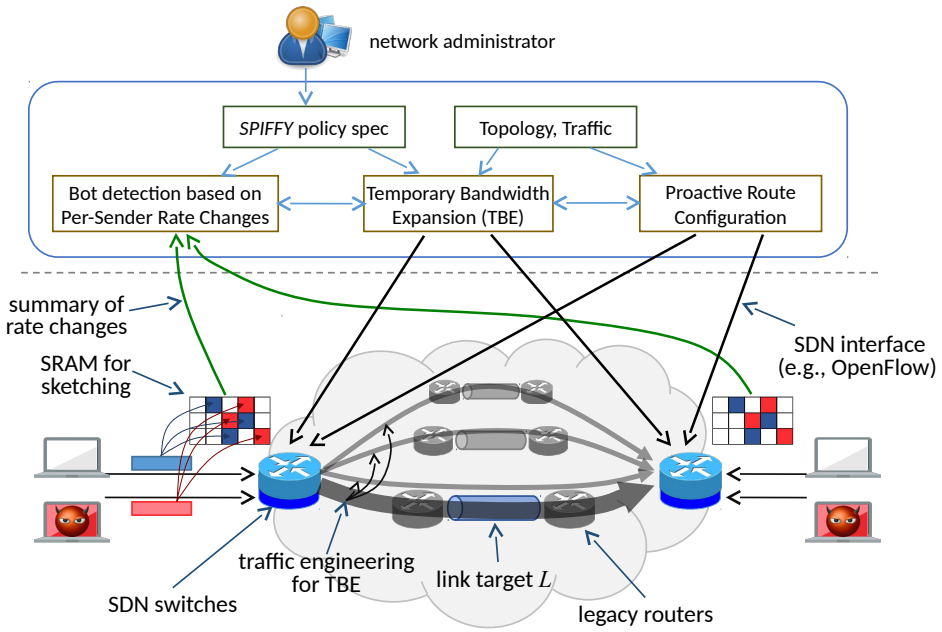

Figure 4.4: Overview of the SPIFFY using an SDN in the Internet core.

- *Robust bot detection:* First, to detect bots, we provide a scalable monitoring mechanism that relies on simple "sketching" algorithms running in the edge switches [171]. This algorithm guarantees the accurate per-flow rate change measurement with only small size of SRAM and few hash computations. Second, to obtain the robust bot-detection results, we develop strategies that yield very low false-positive rate. We investigate several cases where legitimate senders might be misidentified as bots (e.g., legitimate senders that do not react to TBE or TCP effects in response to changed RTT measurements) and propose solutions to remove such undesirable events.

4.5 Scalable and Practical TBE

In this section, we focus on a practical implementation of TBE. As discussed earlier, there are two key challenges here. First, given that networks do not have spare fibers lying around, we need a network-layer solution for TBE. Second, we need this step to be fast because it ultimately impacts our ability to rapidly test and detect bots.

Our *network-layer* TBE approach dynamically reroutes the flows traversing the target link through other under-utilized links in the network. It computes the new routes, which provide large bandwidth

expansion to all senders using the target link simultaneously, *as if* the target link bandwidth is physically expanded. The new routes are calculated at a central controller and installed in SDN-supported switches at the edge. Note that for ease of explanation we refer to the physical bandwidth expansion as the *ideal* TBE.

The goal of the network-layer TBE is to emulate the ideal TBE with large bandwidth expansion factor. The ideal bandwidth expansion factor we wish to achieve is $M_{ideal} = r_g/r_d$, as described earlier. Then the question that arises is how the network-layer TBE can achieve this high M_{ideal}. To answer the question, we first look at how much bandwidth expansion can be achieved by the network-layer TBE for a given network. Then we evaluate whether the bandwidth expansion factor is large enough for M_{ideal}.

We formulate the routing problem of finding the maximum bandwidth expansion factor, denoted as $M_{network}$, for a given a network configuration. Let us assume that we are given a network graph $G = (V, E)$, where V represents the set of routers and E represents the set of links between the routers. We denote by $b(x, y)$, where $(x, y) \in E$, the bandwidth that is not used at the time of TBE; i.e., residual bandwidth. We define a *flooding traffic matrix* T where each ingress/egress pair (s, t) denotes the total traffic rate $T(s, t)$ between s and t that contribute to the flooding at the target link. Note that we assume that the residual bandwidth $b(x, y)$ and the flooding traffic matrix T are unchanged during TBE operation. We associate a variable $f_{(x,y)}^{(s,t)}$, with each pair (s, t) and each link $(x, y) \in E$, that denotes the fraction of the traffic flow from s to t over the link (x, y). The problem of finding the maximum bandwidth expansion factor for network-layer TBE is defined by following linear program:

Linear Programming 1 (LP.1)

$$\text{maximize} \quad m \tag{4.1}$$

$$\text{subject to} \quad \sum_{y:(x,y)\in E} f_{(x,y)}^{(s,t)} = \begin{cases} -m \cdot T(s,t), & \text{if } y = s \\ m \cdot T(s,t), & \text{if } y = t \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

$$\sum_{(s,t)} f_{(x,y)}^{(s,t)} + f_{(y,x)}^{(s,t)} \leq b(x,y),\ (x,y) \in E \tag{4.3}$$

We denote the objective value of this linear program by $M_{network}$. The objective (4.1) in the above

optimization is to maximize the bandwidth expansion factor via rerouting. Conditions (4.2) represent the flow conservation constraints that ensure the m-times expanded traffic $T(s,t)$ is routed from the attack relevant ingress/egress router pairs (s,t). Constraints (4.3) define the load on each link and ensure it is smaller than its residual link bandwidth.[3]

Here, we face two challenges in solving the LP.1 and applying it for the ideal TBE emulation.

- *Scalability:* Although the LP.1 can be solved in polynomial time, the size of the problem becomes impractically large when the number of routers R is large (e.g., $R > 100$). The time to solve the problem grows rapidly with the network size and becomes unrealistic for real-time operations; e.g., few thousand seconds in a network of size $R = 196$.
- *Small $M_{network}$ compared to M_{ideal}:* In practice, the value of $M_{network}$ is small compared to M_{ideal}. As we will see in detail in Section 4.7, $M_{network}$ is typically in the range of 2 – 3 and it is likely that $M_{ideal} = r_g/r_d$ is 5 – 10 times larger than $M_{network}$.

We solve the TBE scalability problem by greedy algorithm (Section 4.5.1) and the small $M_{network}$ problem by randomized sequential TBE (Section 4.5.2).

4.5.1 Greedy algorithm for TBE Scaling

As a solution to the scalability problem, we propose a simple greedy routing algorithm that runs much faster than solving LP.1 with off-the-shelf solvers. One can also use other efficient ways of solving such problems in general; e.g., [24, 134]. The greedy algorithm presented here is one possible way for calculating an approximate solution.

The greedy algorithm takes as inputs the set of ingress/egress pairs and the number of their flows that cross the target link, the desired bandwidth expansion factor m, and the network graph $G = (V, E)$ and the residual link bandwidth $b(x, y)$ for each link $(x, y) \in E$. Then it outputs a feasible routing solution $R(s, t)$ for all ingress/egress pairs (s, t). The pseudocode of this algorithm is given in the following Algorithm 1.

We use a binary search procedure over the value of m in Algorithm 1 and obtain the estimate of the maximum bandwidth expansion factor $\widehat{M}_{network}$ and the corresponding routing solution. We show in

[3] Since we use the undirected graph G, for any link $(x, y) \in E$ we set $b(x, y) = b(y, x)$ but we activate only one them for the constraints (4.3).

Algorithm 1 Greedy algorithm for TBE

Inputs: Set of ingress/egress pairs (s, t) crossing target link,
    Number of flows on each ingress/egress pair $n(s, t)$,
    Desired bandwidth expansion factor m,
    Network topology graph $G = (V, E)$, and
    Residual link bandwidth $b(i, j)$ for $(i, j) \in E$.
while ($\exists(s, t)$ that has not yet selected) **do**
    Select ingress/egress pair (s, t) at random w/o replacement.
    Calculate the available network graph $G' = (V, E \setminus E')$,
    where E'=$\{$links w/ available bandwidth $\geq m \cdot n(s, t) \cdot r_d\}$.
    Calculate new route $R(s, t)$ in G'.
    if $R(s, t) \neq NULL$ **then**
        Move all flows in (s, t) to $R(s, t)$.
Output: New routes $R(s, t)$ for all ingress/egress pairs (s, t).

Section 4.7 that the difference between $M_{network}$ and $\widehat{M}_{network}$ is negligible in practice.

4.5.2 Randomized Sequential TBE

To solve the problem of small value of $M_{network}$, we use a randomized sequential TBE approach. That is, we test only a subset $R_{TBE} = M_{network}/M_{ideal}$ of senders at each TBE round so that a subset of senders can have feasible routes that provide M_{ideal}-times bandwidth expansion. We then repeat this process until most of the senders are tested. If the obtained $M_{network}$ from LP.1 is larger than or equal to M_{ideal}, no more than a single TBE is required.

Random sender selection. We randomly select a fraction R_{TBE} of senders in each ingress/egress pair and reroute them using the solution of LP.1. The obtained routing solution provides the selected senders M_{ideal}-times expanded bandwidth. Since we randomly sample the senders at every TBE, an adversary *cannot* anticipate when a particular bot will be tested. The number of TBE rounds that needs to be performed to test the majority (e.g., 90%) of the senders depends on the fraction R_{TBE}.

4.6 Rate-change measurement tests

In this section, we focus on the rate-change measurement test. As previously mentioned, there are two key challenges in designing the test. First, stateful per-sender rate monitoring could be expensive and induce high control overhead at the SDN controller. Second, the robustness can be undermined by real world TCP effects; e.g., prevalence of short-lived TCP flows or reaction to RTT changes.

4.6.1 Sketch-based Per-Sender Rate Change Detection

As mentioned, SPIFFY requires rate change detection for all senders that cross the target link. This raises concerns about the computation and memory complexity of such "stateful" operations. Another concern is that when the real-time per-sender rate measurements are reported back to the SDN controller the control channel could be easily congested due to the large volume of control messages.

SPIFFY can address these challenges by utilizing sketch-based measurements [80]. Sketch is a memory-efficient data structure that stores summaries of streaming data. In particular, a simplified variant of sketch-based rate change detection [88] can be used for efficiently and quickly detecting per-sender rate changes. With the sketch-based rate change detection, the edge switches report *only* the measurement summary to the SDN controller, such as list of bot IPs, and significantly minimize the control channel overhead.

Sketch-based rate-change measurement: We use the original sketch-based change detection mechanism by Krishnamurthy *et al.* [88] for measuring per-sender rate changes. In fact, SPIFFY needs a simpler version of the original sketch-based change detection since it measures with the granularity of a sender (i.e., source IP), which is coarser than the granularity of a flow. The three basic components are the sketch module, the forecasting module, and the change-detection module [88]:

1. The sketch module creates a sketch; i.e., a $H \times K$ table of SRAM memory. When a packet arrives at an edge switch, the source IP is fed into the H independent hash functions. Based on the $\texttt{mod}\ K$ of the H hash outputs, H registers in the H rows are updated by the packet size u. By updating all packets in a time interval t, we obtain a sketch $S(t)$ at the end of the interval t.
2. The forecasting module uses the observed sketches in the past intervals $S(t')$ ($t' < t$) to compute

the forecast sketch $S_f(t)$.

3. The change-detection module constructs the forecast error sketch $S_e(t) = S(t) - S_f(t)$. For each sender's IP_{src}, this module calculates the forecast error.

Estimated measurement complexity: We analyze the estimated memory size and the sketch computations. The required memory size is determined by the number of independent hash functions H and the size of sketch bins K for each hash function. For a real Internet trace dataset with more than 60 million flows, $H = 5$ and $K = 32K$ produce very accurate rate-change measurement; e.g., 95% accuracy for top 1000 flows with the maximum rate changes [88]. Our rate-change measurement will also be accurate with these parameters, since each edge switch will not need to measure more than 60 million senders in most cases. When we assume 3 bytes for each register in the sketch memory, each edge switch requires 480 KB SRAM memory space.

Sketch-based measurement requires H hash operations for individual incoming packets. Also, each SRAM access requires few tens of nano seconds. However, since the hash operations and SRAM access can be implemented in parallel in hardware, these per-packet operations can be very efficiently implemented and thus do not affect the data plane throughput [171].

At every rate-change detection interval, each edge switch calculates the forecast sketches $S_f(t)$ and the forecast error sketches $S_e(t)$. These and the final rate-change calculation requires a computational overhead of about 1.91 seconds, when for example $H = 5$, $K = 64K$, and 10 million flows are monitored [88]. Our per-sender rate-change measurement would require much shorter (e.g., $\ll 1$ sec) time for the computation at each edge switch since today's commodity CPUs are at least 3-4 times faster than the one used (900 MHz CPU clock speed) more than a decade ago [88].

Bot-detection summary reports: Instead of reporting the rate-changes of all senders, each edge switch can report only the subset of senders that are determined as bots (or legitimate senders). Thus, the aggregate bandwidth for control channel can be limited to a few Mbps or less; e.g., only 4 MB data transfer is required even when 1 million bot IPs are reported. Notice that the reports are made only when the TBE is performed.

4.6.2 Bot Detection Robustness to TCP Effects

The robustness of SPIFFY's bot detection relies on the prompt and fast rate increase of legitimate senders when TBE is performed. The rate increase is mainly determined by TCP operations at the senders since they control the maximum flow rates at a given time. However, achieving robust bot detection can be challenging due to the two following TCP effects: (1) short-lived flows (e.g., few packets in a flow) in the Internet terminate before TCP increases their rates; (2) when TBE's route changes cause sudden increase of RTT values, TCP might decrease the send rates by decreasing congestion windows and/or causing spurious timeouts.

To achieve robust bot detection, our primary focus is to maintain low false-positive rate because false-positive events misidentify legitimate senders as malicious senders. In contrast, false-negative rate (i.e., the rate in which bots are misidentified as legitimate senders) is not a particularly useful metric since SPIFFY allows false-negative events to happen for the cost-detectability tradeoffs. For example, if an adversary is determined to remain undetected, she can make the false-negative rate to be practically one at a highly increased attack cost.

Robustness to short TCP flows

Unlike long-lived flows, short-lived flows might *not* increase their rates in response to TBE because they may not last long enough (e.g., few seconds) when the bottlenecked bandwidth is expanded. Therefore, when the majority of flows are short-lived (as is the case of today's Internet traffic), per-sender rate of *legitimate senders* could be almost unchanged when TBE is performed, causing false-positive events.

Here, we first observe that the prevalence of short-lived flows does *not* affect the rate changes of senders that create realistic traffic with the mixture of short- and long-lived flows. Moreover, we show that SPIFFY can maintain false-positive rate as low as 1% or less by exempting senders with per-sender rates lower than *minimum per-sender rate* from the bot detection process regardless of their rate-change ratios.

To test our claims, we perform simulations with a synthetic web-traffic generator. We use the *ns2* simulator with PackMime-HTTP web-traffic generator to construct diverse network environments and simulate accurate TCP operations with realistic HTTP application traffic demand [4, 36]. Approximately

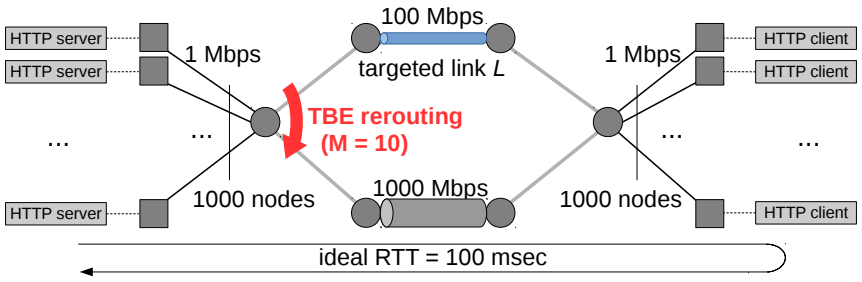

Figure 4.5: Simulation setup.

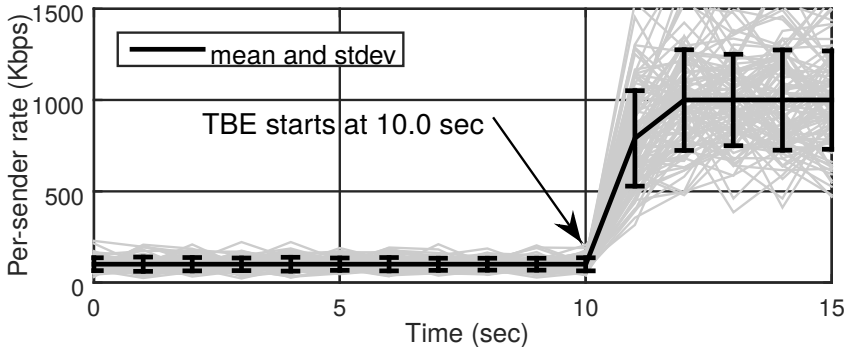

Figure 4.6: An example per-sender rate-change measurements of randomly selected 100 legitimate senders with mean and standard deviation when the bandwidth expansion factor $M = 10$.

70% of the synthetic web-traffic flows have size smaller than a single IP packet's maximum size (1,500 Bytes) while a small number of large flows exist. We determine the queue size based on a rule-of-thumb practice; i.e., $QueueSize = \overline{RTT} \times C$, where $\overline{RTT}$ is the average round-trip time of the flows crossing the link and C is the data rate of the link [18]. For TBE, we assume that the bandwidth of the target link is expanded by a factor of $M = 10$. As shown in Figure 4.5, we simulate 1000 pairs of clients/servers exchanging HTTP traffic through a target network link. We set the ideal (i.e., when no traffic on the path) round-trip time of 100 msec and the application-layer data rate of 1000 Kbps for the purpose of clear illustration.

Figure 4.6 shows a rate measurements of 100 randomly selected senders. Before TBE starts at $t = 10$ seconds, all senders achieve approximately 100 Kbps with small standard devidations; however, after TBE starts, most senders achieve 10 times higher rates within 2 seconds. This result shows that the rate change detection is robust for the legitimate senders with realistic flows, in particular with large portion

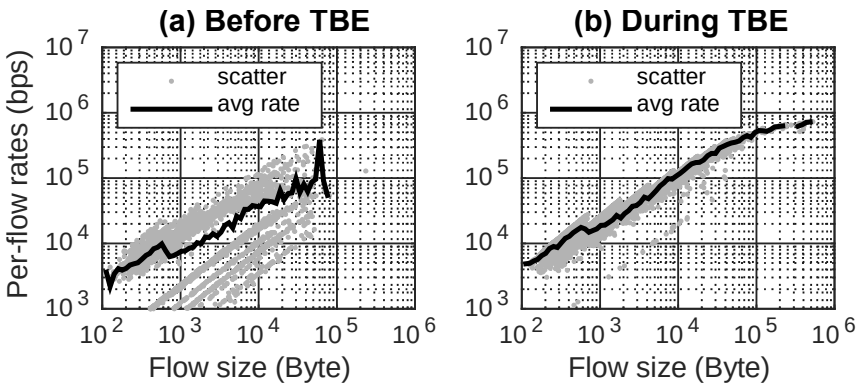

Figure 4.7: Simulated per-flow rates of flows in realistic HTTP web traffic (a) before and (b) during TBE with bandwidth expansion factor $M = 10$.

of short-lived flows.

The reason for the negligible effect of short-lived TCP flows on the effectiveness of rate-change detection is that a few long-lived flows from senders increase their rate significantly once TBE is performed and thus induce the overall per-sender rate change. Figure 4.7 shows the simulated flow rates versus flow sizes. Notice that before TBE only short-flows (i.e., small flow size) are observed. They achieve low rates and long-lived flows are not even able to complete their TCP connections. This is because short-lived flows spend most of their life in the TCP slow start and thus they can rapidly capture a greater proportion of resources than long-lived flows in TCP congestion avoidance, often driving the long-lived flows into timeouts. After TBE starts, long-lived flows achieve much higher rates whereas short-lived flows achieve only slightly higher rates than before. This is because long-lived flows now have enough time to increase the congestion windows.

Next, we evaluate the false-positive rate of the SPIFFY's bot detection with realistic traffic and propose a mechanism to maintain low false-positive rate. To simulate various types of realistic legitimate senders in different locations with different traffic rates, we vary the end-to-end propagation delays (in msec) and the application-layer data rate (in Kbps) per sender.

Figure 4.8 shows the measured rate-change ratio (RC) when the ideal round-trip time (RTT) (i.e., RTT measured when no traffic on the path) or the application-layer data rate (i.e., average HTTP data rate) vary. Figure 4.8a shows that the vast majority of measured rate-change ratios are close to the bandwidth

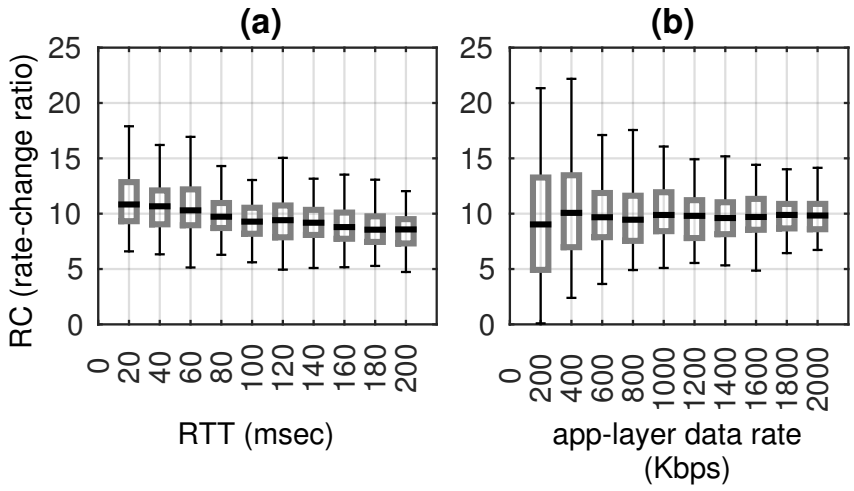

Figure 4.8: Whisker plots representing the rate-change ratios for varying RTT/application-layer data rates when the bandwidth expansion factor $M = 10$.

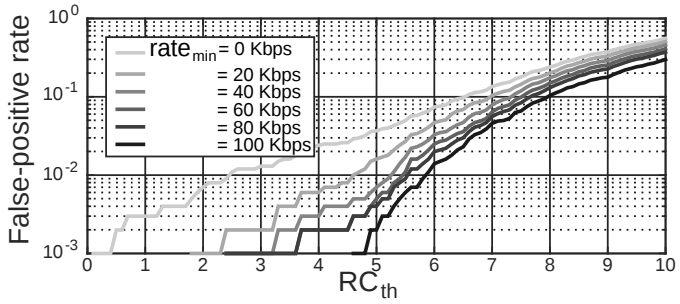

Figure 4.9: False-positive rate for varying rate-change ratio thresholds (RC_{th}) and minimum per-sender rates ($rate_{min}$).

expansion factor $M = 10$ and largely independent of the ideal RTT of the flows. This suggests that bot detection can achieve low false-positive ratio when it uses a rate-change ratio threshold RC_{th} close to M to identify senders with $RC < RC_{th}$ as bots. However, as shown in Figure 4.8b, the rate-change ratio RC is heavily affected by the application-layer data rate. While senders with high application-layer data rates show rate-change ratios very close to the bandwidth expansion factor $M = 10$, senders with low rates result in rate-change ratios that are spread over a large range. This would potentially induce non-negligible false-positive ratios when bots are identified by thresholding the rate-change ratios.

From this observation, we set the *minimum per-sender rate* ($rate_{min}$) and *exempt* the senders with per-sender rate lower than $rate_{min}$ from bot detection. In other words, senders with per-sender rate lower than

$rate_{min}$ are not tested by the target network regardless of their rate change ratios. By exempting these low-rate senders from the bot detection, we can also protect the legitimate, inherently low-rate senders from being misidentified as bots; e.g., legitimate users with slow legacy cellular connections or casual web surfing users with light activity are protected by this exemption.

Figure 4.9 shows the false-positive rate for varying rate-change ratios RC_{th} and for several values minimum per-sender rate $rate_{min}$. We first observe that the larger threshold ratio RC_{th}, the higher false-positive rate is expected because small rate fluctuations can cause false positive when the threshold ratio RC_{th} is high. We also notice that as we exclude more low-rate senders (i.e., set higher minimum per-sender rate $rate_{min}$), we can reduce the false-positive rate. As shown in Figure 4.9, with proper parameters we can easily maintain very low false-positive rate; e.g., 1% or less. Note that the exemption of low-rate senders could contribute to some false-negative errors; i.e., indicating bots as legitimate. However, the influence of the non-detected bots is limited since they do not send at the rate higher than the minimum per-sender rate $rate_{min}$, which is the chosen small rate value.

Note that adversaries *cannot* exploit this exemption of low-rate senders. An adversary might configure her bots to send at a rate lower than the minimum per-sender rate $rate_{min}$ to avoid detection, but this only increases the attack cost significantly because more bots are needed to create the same amount of attack traffic to congest the target link.

Robustness to sudden RTT increase

Our TBE mechanism reroutes traffic around the target link. Rerouting may find a new route longer than the initial one. According to our experiments (Section 4.7.2), TBE increases the route length (i.e., number of routers in a route) on average by up to 24%. This raises the question of whether this suddenly increased RTT adversely impacts the false-positive rates of SPIFFY. We list two possible cases where sudden RTT increase might cause false-positive events: (1) Some delay-based TCP variants (e.g., Compound TCP [147] and TCP Vegas [30]) use RTT measurements at receivers to adjust TCP congestion windows. These TCP variants consider RTT increase as the sign of congestion and reduces their sending rates; (2) TCP senders might experience spurious timeouts and drop sending rates significantly. A spurious timeout occurs when RTT suddenly increases and exceeds the retransmission timer that had been

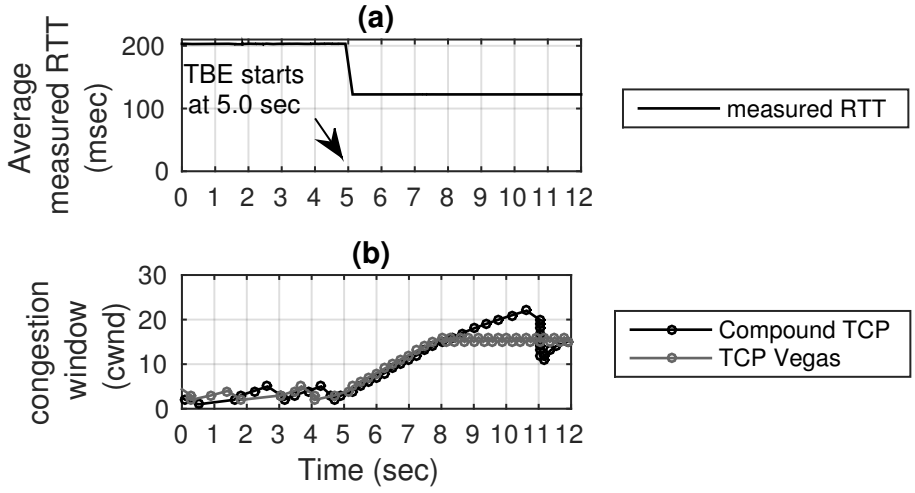

Figure 4.10: RTT and congestion window changes when TBE is performed.

determined a priori [97].

Here, we claim that such rate decrease due to RTT increase is *not* likely to happen becuase RTT will actually be reduced significantly when TBE is performed. The rationale behind this is that TBE removes high queueing delay at the (almost) full buffer of the target link. The RTT reduction due to this congestion relief is in general much larger than the RTT increase due to TBE rerouting, ultimately causing RTT reduction.

To support our claim, we measure RTT changes when TBE is performed in a simulation. We set the ideal (i.e., when no congestion on a path) RTT to 100 msec and assume 25% increase of the RTT when rerouting takes place. We assume the rule-of-thumb queue size (i.e., RTT times link capacity [18]) at the target link. As shown in Figure 4.10a, as soon as TBE is executed at time 5.0 sec, the measured RTT is significantly reduced to the near ideal RTT value. The new measured RTT is 25% higher than the ideal RTT due to TBE's rerouting, but it is still significantly smaller than the RTT measurements before TBE.

We also test how the two delay-based TCP variants, Compound TCP and TCP Vegas, adjust their congestion window in response to TBE. Figure 4.10b shows that both TCP variants increase their congestion window promptly when TBE is performed and reach the converged points less than 3 seconds.

4.7 Evaluation

In this section, we evaluate SPIFFY in an SDN testbed to show its effectiveness (viz., Section 4.7.1). Then we evaluate it using flow-level simulations to show its feasibility in large ISP networks (viz., Section 4.7.2). Our current testbed and simulation based study can be extended later in real Tier-1/2 ISPs with real attack traces in future work.

4.7.1 Testbed Experiments

Our evaluations are executed on a server-grade Dell R720 machine with 20-core 2.8 GHz Xeon CPUs and 128 GB of memory, which runs the KVM hypervisor on CentOS 6.5 (Linux kernel v2.6.32). We use Open vSwitch (OVS v2.3), virtual switches [6]. OVS v2.3 supports the OpenFlow v1.3 [5] specification. We use OpenFlow-enabled switches only at the edges of our test network and traditional switches inside the network. Note that we will interchangeably use switches and routers in this chapter. We implement SPIFFY as a `POX` application [7] on the centralized network controller. Notice that in these SDN testbed experiments, we test only long-lived TCP flows generated by `iperf3`. The effects of short-lived flows are studied in packet-level simulations, as discussed in Section 4.6.2.

Effectiveness of Bot Detection

We evaluate how effective SPIFFY is in identifying bots when they are mixed with legitimate senders. We implement the bots based on the Attack Strategy $AS_{\neg spiffy}$. Bot upstream is saturated by attack flows, each of which have the degraded rate, r_d. Note that the adversary in this evaluation does not apply the rate-increase mimicry (**RM**) and thus her bots have no available bandwidth to demonstrate the rate increase.

In our simplified ISP network with two edge switches (one ingress and one egress) and 10 parallel links that connect the two edge switches (one of them is the target link of the attack), we reroute traffic crossing the target link to other parallel links and provide ten-times expanded bandwidth (i.e., $M = 10$) to the two senders. For this, the SPIFFY application installs rules and MPLS labels (which are prevalently used in large ISPs [49] and can be implemented by SDN switches [136]) at the edge switches.

Figure 4.11 shows the general parameters for the SPIFFY experiments. A bot b_i $(1 \leq i \leq m)$ has upstream bandwidth u_i^b and a legitimate sender l_j, $(1 \leq j \leq n)$ has upstream bandwidth u_j^l. We set the

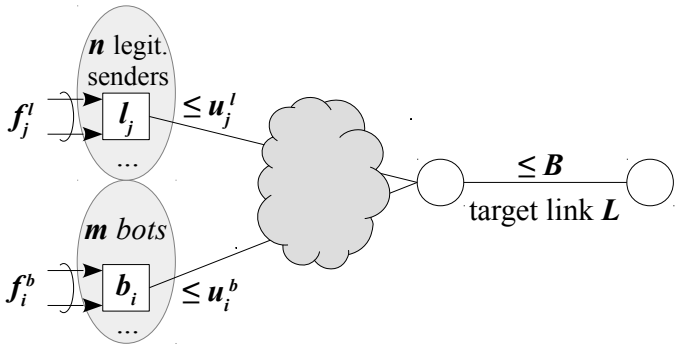

Figure 4.11: Parameters for SPIFFY experiments.

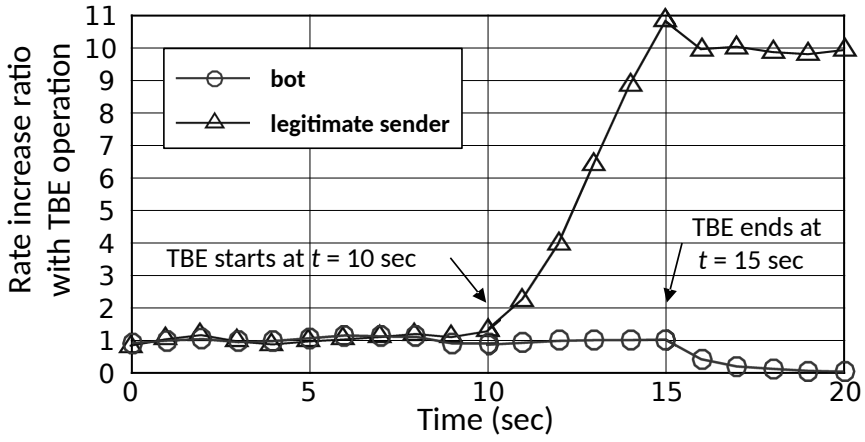

Figure 4.12: Rate-increase ratios of a bot and a legitimate sender in the SDN testbed.

number of bots m and their upstream bandwidths u_i^b in such a way that the fair-share per-flow rate at the target link L equals r_d (i.e., $\frac{B}{\sum_{i=1}^{m} f_i^b + \sum_{j=1}^{n} f_j^l} = r_d$) to achieve the attack goal $G_{strength}$. Notice that all bots generate $f_i^b = u_i^b / r_d$ flows of rate r_d to saturate their upstream bandwidth.

In our experiments, we set all senders (both bots and legitimate senders) to send 50 long-lived TCP flows to make them indistinguishable to any per-host rate filtering mechanisms. Accordingly, all bots are set to have upstream bandwidth $u_i^b = f_i^b \times r_d = 0.5$ Mbps when $r_d = 10$ Kbps. We set all legitimate senders' bandwidth to accommodate all 50 legitimate flows with guaranteed rate r_g. That is, $u_j^l = f_j^l \times r_g = 5$ Mbps when $r_g = 100$ Kbps. Note that the upstream bandwidth parameters for bots and legitimate senders are selected for illustrative purpose only. SPIFFY is effective for any practical upstream link bandwidth. RTTs are set to be 200 msec to experiment the practically worst-case rate-change responsive time for TBE operation.

Figure 4.12 shows the per-sender rate changes of the two senders measured every second by the edge switches. The rate is measured from $t = 0$ to $t = 20$ seconds, when the TBE operation is performed at $t = 10$ second. Notice that before TBE (i.e., at $t < 10$), the two senders' rates are almost identical. However, once TBE is performed, within less than 5 seconds (i.e., at $t < 15$), the two senders show very different rate changes; the legitimate sender's rate increases by almost 10 times whereas the bot's per-sender rate remains the same. At the legitimate sender TCP adapts to the expanded bandwidth in less than 5 seconds. Note that after TBE ends (i.e., at $t = 15$), SPIFFY immediately starts the bot identification. The target network notices the difference in the rate changes, identifies the bot, and filters its source IP at the corresponding ingress switch of the network. As a result, after $t = 15$ the bot's rate tapers off quickly while the legitimate sender achieves the guaranteed rate $r_g = 100$ Kbps $= 5$ Mbps / 50 flows.

Effectiveness of Increasing Attack Cost

Unlike the previous experiment, in this evaluation an adversary decides to follow the rate-increase mimicry (**RM**) and increases her attack cost. To demonstrate how the number of bots required to achieve $G_{strength}$ differ for defense strategies, we implement a simple adversary program that manages the bots and adapts to the defense changes at the target network. This program *increases* the number of bots in the attack; i.e., if the attack is unsuccessful (i.e., the average per-flow rate, r_{avg}, at the target network is larger than r_d), it adds more bots at the rate of one additional bot per second.

We evaluate the effectiveness and the cost of the attack against the three different defense strategies: (a) *no defense*: a strategy that only provides per-flow fairness, which is automatically achieved by TCP's congestion control mechanism; (b) *ordinary traffic engineering (TE)*: a strategy that *provisions* additional bandwidth by rerouting traffic crossing the target link (both malicious and benign) persistently as long as the flooding continues; and (c) *SPIFFY*: a strategy that performs TBE and rate-increase measurement *on demand* to test the bots. Note that ordinary TE provisions the additional bandwidth persistently without attempting to detect the bots, while the TBE operation is temporary and only for testing bots.

We utilize 130 bots and each of them have 1 Mbps of upload bandwidth limit. The per-flow rate demand for legitimate senders is $r_g = 100$ Kbps while bots have the rate demand of $r_d = 10$ Kbps for no defense and ordinary TE. However, since the attack against SPIFFY has the demand-rate mimicry goal

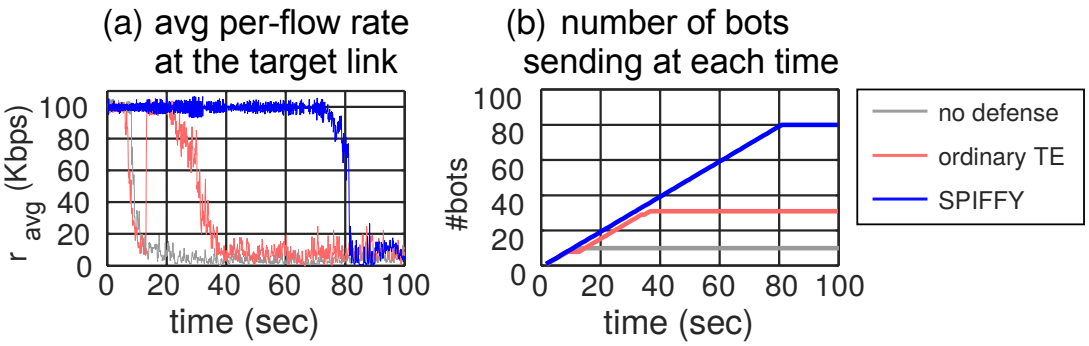

Figure 4.13: Measured average per-flow rates (r_{avg}) and the number of used bots ($\#bots$) for the three defense strategies.

(**RM**), its bots have the rate demand of $r_g = 100$ Kbps. The target link bandwidth is set to be 8 Mbps. The number of senders and the bottleneck bandwidth are limited by our experiment setup. Through additional packet-level simulations (Section 4.6.2) and flow-level simulations (Section 4.7.2), we show that the results from these limited-bandwidth experiments scale to large configurations.

Figure 4.13 shows the results of the evaluation over the three defense strategies. In the two plots, the x-axis represents the wall-clock time of the experiment. The adaptive adversary program starts from time $t = 0$, increasing its number of bots by 1 every second, if the adversary goal $G_{strength}$ is not satisfied. Figure 4.13a shows the average per-flow rate changes over time. Figure 4.13b illustrates the number of bots used in the attack, which represents the attack cost. The number of required bots varies widely for different defense strategies. For no defense, only 10 bots are needed to achieve the goal. For ordinary TE, initially the attack needs only 8 bots to achieve its goal. However, as soon as the target network is flooded at around $t = 12$ seconds, ordinary TE expands its defense bandwidth by a factor of three and the average per-flow rate of the target recovers the initial $r_g = 100$ Kbps. As a result, the adversary needs to further increase the number of bots up to 31. Notice that both the attack and defense costs increase roughly three times, which suggests that there is no reduction in cost asymmetry. For SPIFFY, we observe that the adversary requires 80 bots in total to achieve the rate-reduction goal $G_{strength}$ while the defense does not use additional bandwidth.[4] This shows that the rate-increase mimicry (**RM**) costs the adversary use roughly $M = 10$ times more bots to achieve the attack goal $G_{strength}$.

[4]The TBE operations use additional bandwidth at the target only *temporarily*. Thus, the increase in defense cost on average is negligible.

	Cogent	Tata	UUNET	NTT	Deutsche Telekom
#routers	196	144	48	46	38
#links	245	194	84	63	55

Table 4.1: Five ISP networks used for large-scale simulations and their number of routers and links.

(in seconds)

	Cogent	Tata	UUNET	NTT	Deutsche Telekom
LP solution $M_{network}$	2,039.06	435.79	0.79	0.27	0.27
Greedy algorithm solution $\widehat{M}_{network}$	14.71	9.07	0.65	0.35	0.26

Table 4.2: Execution times for LP solution $M_{network}$ and greedy algorithm solution $\widehat{M}_{network}$.

4.7.2 Large-Scale Flow-Level Simulations

In this section, we evaluate the feasibility of SPIFFY in large-scale flow-level simulations with up to about 200 routers. In particular, we focus on the implementation of the TBE to show that its proposed design (Section 4.5.2) can be implemented in practical ISP networks. For scalable evaluation (e.g., millions of flooding flows and hundreds of routers), we developed a simulator that models TCP flows (defined by srcIP, dstIP, srcPort, dstPort, and protocol) as fluid flows [25]; i.e., each flow at each time epoch has its flow rate and occupies the same amount of bandwidth at all the network links it travels. We model the behavior of TCP flows by implementing the ideal fair-share rate property (i.e., allocating equal bandwidth to all competing flows) at every attack-targeted link in the network. We examine the TBE algorithm using the flow simulator with millions of flows. Our simulator models the five large ISP topologies from the Topology Zoo database [87] as shown in Table 4.1. We use the uniform link-bandwidth model, where all links have the same bandwidth, and non-uniform model, where links in the center of the ISP topology have higher bandwidth. The simulation proceeds in discrete time ticks. At each tick, the simulation updates the rates of all flows in the network by visiting each network link and updating the rates of all flows on the link.

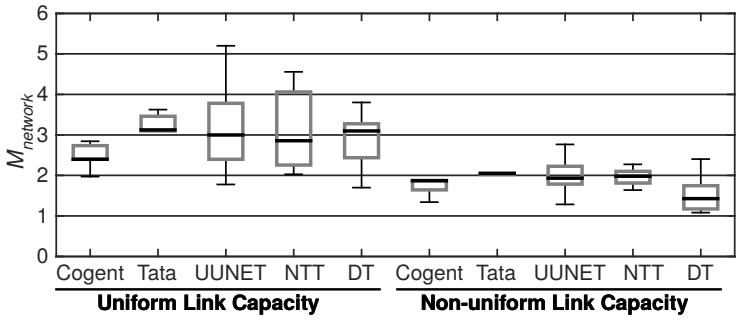

Figure 4.14: $M_{network}$ values for the five ISPs in two link-bandwidth models.

Real-time operation of TBE in large networks. The TBE operation needs to calculate the new route sets in real-time; e.g., within few seconds. We evaluate the execution time to calculate the new routes using the greedy routing algorithm (i.e., Algorithm 1) and show how time efficient it is, compared to solving the optimal LP.1. When solving the greedy algorithm solution $\widehat{M}_{network}$, we apply a binary search; viz., Section 4.5.1. We utilize the multi-core architecture of our SDN controller for the binary search and evaluate 12 values of m concurrently at each iteration. Table 4.2 shows the execution time for the LP solution $M_{network}$ and the greedy algorithm solution $\widehat{M}_{network}$. LP.1 is solved with the CPLEX solver in a server-grade machine with 20 cores. As explained, LP.1 requires an impractical amount of time for networks with large number of routers R. In contrast, the greedy algorithm with binary search requires only few seconds in general to calculate $\widehat{M}_{network}$. Even in the largest network we evaluate (i.e., Cogent), it takes only 14.7 seconds, which is less than 1 percent of the time taken by the LP solution, which is 2,039 seconds.

Optimal LP solutions $M_{network}$ and effectiveness of the TBE algorithm. We solve LP.1 in the five ISP networks with two different link-bandwidth models. The uniform link-bandwidth model assumes the same bandwidth of 40 Gbps for all the links. To model more realistic network bandwidth provisioning, we also use the non-uniform model that assigns link bandwidth based on the *betweenness centrality* of each link. The betweenness centrality of a link is the number of shortest-path routes between all pairs of edge routers that include the link [59]. This metric represents how (logically) central the link is in the network topology. We assign 40 Gbps link bandwidth to the 33% of links with the highest centrality, 5 Gbps bandwidth to the 33% of links with the smallest centrality, and 10 Gbps bandwidth to all other links in

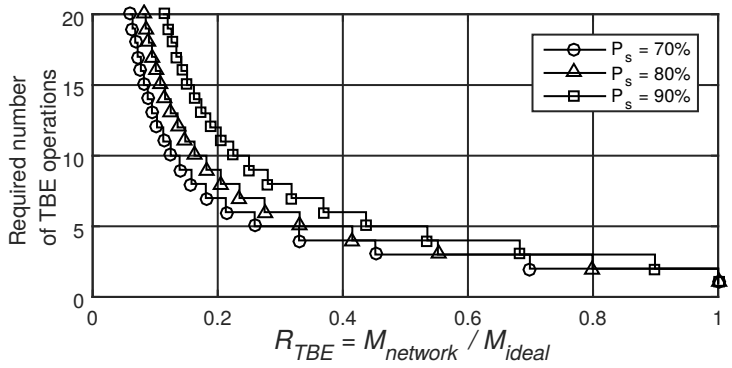

Figure 4.15: Required number of TBE operations for varying $R_{TBE} = M_{network}/M_{ideal}$ and P_s.

the middle. For each case, we setup 10 different attacks, which target 10 different links for flooding. The targeted links are chosen from the 10 links with the highest betweenness centrality in each ISP topology. We assume that 30% of bandwidth of each link is already used for underlying traffic that is unrelated to the link-flooding attack. Figure 4.14 shows the distribution of $M_{network}$ in the box plots. We achieve $M_{network}$ close to 3 with the uniform model while $M_{network}$ is close to 2 with the non-uniform model. The non-uniform model has smaller $M_{network}$ since it provides less bandwidth for alternative paths for TBE than the uniform model. As we will see later in this section, a small value of $M_{network} \simeq 2$ can still be effective when used with the sequential TBE. Moreover, we also evaluate the accuracy of the greedy algorithm solution $\widehat{M}_{network}$ compared to the LP solution $M_{network}$. We find that in all five ISP networks the greedy algorithm solution $\widehat{M}_{network}$ is nearly identical to the LP solution $M_{network}$, showing a difference of only few percentages (almost 1–2%). Also, we find that the new routes due to TBE need just 1 to 3 more router hops (or 4 – 24% longer average route length in the target network) compared to the original routes before TBE.

Operation of randomized sequential TBE. We also evaluate how many times the TBE operations need to be performed to test the majority of all senders contributing to the congest on the target link. As explained in Section 4.5, the required number of randomized sequential TBE operations, n, depends on the ratio $R_{TBE} = M_{network}/M_{ideal}$ and the percentage P_s of senders that must be tested at least once. Figure 4.15 shows the required number of TBE operations for various R_{TBE} and P_s. As expected, the higher R_{TBE}, the lower n. Moreover, the lower P_s, the smaller the number of TBE operations are required. Based on the observation that the five ISPs we evaluate have $M_{network}$ in between 2.21 and

3.19, we conclude that roughly 4 – 10 TBE rounds are required.

4.8 Discussion

4.8.1 Dynamic acquisition of bots

To avoid SPIFFY's bot detection mechanism, an adversary might attempt to *add* a significant amount of *temporary* attack bandwidth that is purchased from botnet markets only for very short duration (e.g., 5 seconds) in response to temporary bandwidth expansion (TBE). However, this dynamic acquisition of bots is hard to be realized in practice for the following reason. Botnet markets cannot provide significantly lower rates for such temporary botnet provisions since the operation of TBE, which is determined by the network operators of the target network, *cannot* be anticipated by adversaries or botnet markets. This unpredictable bandwidth resource needs render low-cost temporary attack bandwidth purchase impractical in botnet markets.

4.8.2 Legitimate Senders with Application-layer Rate Adaptation

Although the rate of a legitimate sender is mainly determined by its TCP window control (as discussed and evaluated in Section 4.6.2), application programs might also adapt their data rates and thus affect the send rate of the legitimate senders. Such application-layer rate adaptation can potentially reduce the effectiveness of bot detection. For example, let us assume a legitimate sender that has suffered from severe congestion for few minutes and its application program has adapted (i.e., reduced) its data rate to a low degraded rate. In such a case, if the adaptation of the application-layer data rate is slow (e.g., few minutes), our bot detection mechanism might miss the send rate increase of the sender and false identify the sender as a bot.

In practice, video streaming is one of the most popular examples of application-layer rate adaptation. Today's most video streaming services periodically (e.g., 1–10 seconds) adjust the quality of a video stream to provide continuous playback under various range of available network bandwidth. An experimental evaluation study with major video streaming services showed that the rate adaptation algorithm can adapt its bitrate very quickly [12]. In particular, Netflix, the most popular video streaming service, is

shown to quickly adapt to the sudden spikes of short-term (e.g., 2, 5, and 10 seconds) bandwidth expansion; viz., Figure 12 in [12]. Based on this experimental evidence, we believe that SPIFFY is effective to most legitimate senders with application-layer rate adaptation.

4.8.3 Robustness against Multiple Link-Flooding Attacks

Rational, cost-sensitive adversaries might also target multiple links concurrently to achieve higher damage to end targets. In such cases, individual links interact with the SPIFFY test at each ISP. Thus, all the security analyses in Section 4.3.2 are applicable to the multiple link-flooding attacks. That is, the attacks must satisfy the rate-increase mimicry goal **RM** to circumvent the tests launched by each link target and this causes the attacks to increase the number of bots by a factor of M for flooding each target link. If multiple link targets are located in the same network, SPIFFY can simply measure the per-sender rate changes as if single link in the network is being targeted.

Multiplexed link attack: Although simple extensions of single link attacks can be easily handled, when an adversary carefully multiplexes the attack flows across her bots and a small number (e.g., 10) target links, SPIFFY can detect the bots only *probabilistically*. The steps of a multiplexed link attack are as follows. Each bot floods multiple (up to M) link targets concurrently while using only $1/M$ of its upload bandwidth (say u) for each link target. When one of the link targets starts testing a bot, the bot allocates all of its upload bandwidth u to the flows that are dedicated to that link by pausing all the other attack flows. As a result, the bot can pass the test by the link, and after the test the bot can continue to flood the multiple link targets again. The bots in this attack can be detected probabilistically when a bot is tested by more than one target link simultaneously since it cannot increase rates for the two tests simultaneously. The detection of the multiplexed link attack can be improved to become *deterministic* when the multiple SPIFFY operations in different ISPs exchange the sender information they are testing (e.g., via standardized channels [111]) and test same bots simultaneously.

4.8.4 Multiple senders sharing a single IP address

When multiple senders in a local network are served by a single NAT gateway, they share the same source IP. If some bots are located in the same local network, they might identified as legitimate senders by

SPIFFY because their flows are mixed with other legitimate flows under the same source IP.

In such cases, we examine whether a particular IP address is shared or not; e.g., via existing mechanisms [27]. Then, we perform the SPIFFY test with *finer* granularity of *flow aggregates*; e.g., per-source-destination, per-source-protocol. This enables SPIFFY to test different smaller sender groups than the entire sender set sharing the same source IP and thus improve the bot-identification accuracy even when senders share a single source IP.

4.8.5 Potential false positives

Despite the mechanisms to minimizes the false positive rates (viz., Section 4.6.2, Section 4.8.4), there are two types of legitimate senders whose behavior still can be misidentified as bots by SPIFFY. Note, however, that these particular types of false positives would *not* be considered as serious collateral damage since the false identification does *not* further penalize legitimate senders who *already* have been experiencing a severe flooding attack when SPIFFY is initiated.

First type of false positives is the legitimate senders that generate large numbers of inherently low-rate legitimate flows. In other words, a sender that runs large numbers of legitimate applications, which do *not* require higher network connections than the attack-degraded rates (e.g., a large SSH server), will likely to be falsely identified since its rate does not respond to the temporary bandwidth expansion. Second type of false positives is the legitimate senders that run large numbers of user applications whose rate control *slowly* adapts to available network bandwidth. Although many application-layer rate control would quickly increase application-layer rates (viz., Section 4.8.2), the senders can be falsely identified whenever any custom application rate control mechanism does not quickly adapt to the increased network bandwidth.

4.9 Related Work

We first summarize link-flooding attacks targeting core network links. We then categorize existing defense approaches, which are insufficient to defend against the link-flooding attacks. Last, we list several other flooding attacks and discuss their relationships with SPIFFY.

Link-flooding attacks. The link-flooding attacks that target the core network links are the main threat model we consider in this chapter. The Coremelt attack [144] utilizes bots to send attack traffic to other bots. This Coremelt attack coordinates large numbers of bot pairs in a way that their communication paths share the links in the Internet core. The Crossfire attack in Chapter 3 coordinates bots to send legitimate-looking low-rate traffic to the attacker-chosen publicly accessible servers (e.g., HTTP servers) in a way that their routes cross the link targets in the core Internet. All attack flows are indistinguishable since they are the connections to the legitimate open services and low-rate protocol-conforming flows.

Profiling-based defense approaches. This type of mechanisms maintain the profiles of legitimate traffic based on their flow rates, source IPs, destination IPs, protocols, etc., and distinguish attack traffic from legitimate one. PSP [38] constructs the profiles of the rate history of origin-destination pairs in a single ISP network. ACC (or Pushback) [99] utilizes the rate/history of flow aggregates at the intermediate routers. Moreover, anomaly detection [91] monitors the unusual changes in the entropy of packet header bits to detect attack traffic. All attack-profiling approaches, however, can be circumvented by an adversary who can freely choose attack sources, destinations, and protocols, and completely conform to network protocols (e.g., TCP congestion control) while successfully flooding a target.

Proof-of-work defense approaches. Proof-of-work mechanisms enable target routers (or servers) to force *both* bots and legitimate senders to submit proofs (e.g., computation resource for solving puzzles [121, 160] or network bandwidth resource [156]) that were performed before allowing access to the target. These systems are fundamentally different from SPIFFY for multiple reasons: (1) proof-of-work systems limit the traffic generation of legitimate senders while SPIFFY limits that of bots only; (2) proof-of-work systems create significant waste of computation/bandwidth at traffic sources, which might be prohibitive in energy/bandwidth-starved devices, whereas SPIFFY does not waste any unnecessary resources; and (3) proof-of-work systems require significant modifications to the current Internet, including senders, routers, end-servers, and protocols between them, while SPIFFY does not require such modifications.

Capacity-provisioning defense approaches. Instead of attempting to distinguish attack traffic or prioritize legitimate traffic, the target network could simply provide more bandwidth either via physical bandwidth addition or traffic engineering for both legitimate and attack traffic [65]. However, capacity provisioning alone cannot be effective because (1) it does not reduce the attack-defense cost asymmetry

(i.e., N-times-provision of bandwidth at the target network requires the same factor (N) of increase of attack bandwidth for successful attacks), and (2), if bandwidth is provisioned via traffic engineering, the additional bandwidth available for the provisioning in typical ISP networks is very small (e.g., 2 – 4 times) as shown in our evaluation in Section 4.7.

Collaboration-based defense approaches. These mechanisms require global collaboration among networks under different ownership. CoDef [93] requires coordination between the attack-target ISP and the ISPs hosting traffic sources to mitigate attack traffic. SENSS [14] assumes the collaboration between the target ISP and the intermediate ISPs in the Internet to control the incoming flooding traffic. SIBRA [26] utilizes the global ISP collaboration to provide botnet-size-independent end-to-end bandwidth guarantees to premium traffic. Although ISP collaboration in general is not readily available in the current Internet whose relationship between ISPs (e.g., [169]) are competitive rather than collaborative, increasing interest in ISP collaboration for DDoS defense (viz., IETF DDoS Open Threat Signaling (DOTS) Working Group [111]) may make these mechanisms feasible in the near future.

SDN-based DDoS defense approaches. Orthogonal to the above defense approaches, recent proposals utilized SDN-based network architectures to handle DDoS attacks; e.g., Bohatei [51]. However, their focus (in particular Bohatei) is on elastic scaling of defense for *legacy* DDoS attacks and they do not tackle the link-flooding attacks we consider here.

Chapter 5

Conclusion and Future Work

In this chapter, we conclude this dissertation and discuss directions for future research.

5.1 Conclusion

In this dissertation, we have investigated a class of link-flooding attacks that exploit the inherent vulnerability of today's Internet and provided a low-cost, first line of defense for handling attacks launched by cost-sensitive adversaries. We summarize our contributions below.

- First, we introduced the notion of the routing bottlenecks, defined it using power-law distributions of route count in network links, and showed that it is a fundamental property of Internet design; i.e., it is a consequence of route-cost minimizations. We identified the key characteristics of these bottlenecks in terms of size, link type, and distance from host destinations, and we measured the degradation of host connectivity caused by attacks that flood bottleneck links, as shown in Section 2.4. We showed that the routing bottlenecks are pervasive in experiments comprising 15 countries and 15 cities around the world and that certain geographic regions are more susceptible than others to link-flooding attacks.
- Second, we presented the Crossfire attack that degrades connections to a variety of selected server targets (e.g., servers of an enterprise, a city, a state, or a small country) by flooding only a few routing-bottleneck links. In Crossfire, a small set of bots directs indistinguishable flows to a large

number of publicly accessible servers to flood bottleneck links. The attack persistence can be extended virtually indefinitely by changing the set of bots, publicly accessible servers, and target links while maintaining the same disconnection targets. We demonstrated the attack feasibility using Internet experiments and attack simulations, and showed its massive effects on a variety of chosen targets in Section 3.5.4.

- Finally, we proposed a system called SPIFFY as a first line of defense mechanism against indistinguishable link-flooding attacks. SPIFFY forces an adversary to choose between two unpleasant alternatives, namely either allow bot detection or accept an increase in attack cost. This cost-detectability tradeoff deters cost-sensitive adversaries who wish to maintain the minimum attack cost while remaining undetected. This way, more complex and expensive collaborative defenses among ASes are required only for the far fewer, cost-insensitive or irrational adversaries. We developed fast traffic-engineering algorithms to achieve effective bandwidth expansion and suggested scalable monitoring algorithms for tracking the change in traffic-source behaviors. We demonstrated the effectiveness of SPIFFY using a software-defined networking testbed and large-scale packet-level and flow-level simulations found in Section 4.7.

5.2 Future Work

Extended routing-bottleneck measurements. In the future, we seek to measure the routing bottlenecks over a longer period of time, such as months or years, to analyze the changes of routing bottlenecks over time. Moreover, we seek to extend the scope of our routing-bottleneck measurements to include servers in a cloud provider, which are often distributed in multiple data centers around the world. With the discovered routing bottlenecks for cloud providers, we can evaluate link-flooding attacks (e.g., Crossfire or new attacks) that degrade the connection between the cloud providers and the rest of the Internet.

Implementation. In this dissertation, we evaluated the effectiveness of the Crossfire attack and the SPIFFY mechanism by simulations and small-scale SDN testbed experiments. In the future, we seek to implement the attacks and defense mechanisms in real ISP networks and evaluate the attack effectiveness and the defense performance in real networks. This would provide a more solid confidence in the

goodness of the attack evaluations and the deployment benefit of the defense mechanism.

Coordinated SPIFFY. In our SPIFFY proposal in chapter 4, we focused on single-AS operations to provide readily available mechanisms in today's Internet. Considering the recent industry efforts in local AS collaboration for DDoS mitigations (e.g., DOTS [111]), in future studies we will seek to extend the current SPIFFY mechanism so that individual SPIFFY instances in different ASes can coordinate themselves to handle link-flooding attacks that target multiple ASes.

Bibliography

[1] Denial of Service Protection: Business protection for business as usual. *http://www.business.att.com/enterprise/Service/network-security/threat-vulnerability-management/ddos-protection/*.

[2] IBM ILOG CPLEX Optimization Studio. *http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html*.

[3] Internet Transit Pricing: Historical and Projected. *http://drpeering.net/white-papers/Internet-Transit-Pricing-Historical-And-Projected.php*.

[4] ns2: The Network Simulator. *http://www.isi.edu/nsnam/ns/*.

[5] Open Flow. *https://www.opennetworking.org*.

[6] Open vSwitch. *http://openvswitch.org*.

[7] POX, Python-based OpenFlow Controller. *http://www.noxrepo.org/pox/about-pox/*.

[8] ACM Code of ethics and professional conduct. *Commun. ACM*, 35(5):94–99, May 1992. ISSN 0001-0782. doi: 10.1145/129875.129885.

[9] Alexander Afanasyev, Neil Tilley, Peter Reiher, and Leonard Kleinrock. Host-to-host congestion control for TCP. *IEEE Communications Surveys & Tutorials*, 2010.

[10] Akamai. The State of the Internet 2nd Quarter. *Report*, 2012.

[11] Aditya Akella, Srinivasan Seshan, and Anees Shaikh. An empirical evaluation of wide-area Internet

bottlenecks. In *Proc. IMC*, 2003.

[12] Saamer Akhshabi, Ali C Begen, and Constantine Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In *Proc. ACM MMSys*, 2011.

[13] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and attack tolerance of complex networks. *Nature*, 406(6794), 2000.

[14] Abdulla Alwabel, Minlan Yu, Ying Zhang, and Jelena Mirkovic. SENSS: observe and control your own traffic in the Internet. In *Proc. ACM SIGCOMM*, 2014.

[15] Yair Amir and Claudiu Danilov. Reliable communication in overlay networks. *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*, 0:511, 2003. doi: http://doi.ieeecomputersociety.org/10.1109/DSN.2003.1209961.

[16] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient Overlay Networks. In *Proc. ACM SOSP*, 2001.

[17] David G. Andersen, Hari Balakrishnan, Nick Feamster, Teemu Koponen, Daekyeong Moon, and Scott Shenker. Accountable Internet Protocol (AIP). In *Proc. of ACM SIGCOMM*, 2008.

[18] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. *Sizing router buffers*. Proc. ACM SIGCOMM, 2004.

[19] Arbor Networks. Worldwide Infrastructure Security Report: Volume IX. *Arbor Special Report*, 2014.

[20] Arbor Networks. Worldwide Infrastructure Security Report: Volume XI. *Arbor Special Report*, 2016.

[21] Brice Augustin, Timur Friedman, and Renata Teixeira. Measuring load-balanced paths in the Internet. In *Proc. IMC*, 2007.

[22] Brice Augustin, Balachander Krishnamurthy, and Walter Willinger. IXPs: mapped? In *Proceedings of IMC '09*, pages 336–349, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-771-4. doi: 10.1145/1644893.1644934.

[23] Daniel Awduche, Angela Chiu, Anwar Elwalid, Indra Widjaja, and XiPeng Xiao. Overview and

principles of Internet traffic engineering. Technical report, RFC 3272, may, 2002.

[24] Baruch Awerbuch and Rohit Khandekar. Greedy distributed optimization of multi-commodity flows. In *Proc. ACM PODC*, 2007.

[25] François Baccelli and Dohy Hong. Flow level simulation of large IP networks. In *Proc. IEEE INFOCOM*, 2003.

[26] Cristina Basescu, Raphael M Reischuk, Pawel Szalachowski, Adrian Perrig, Yao Zhang, Hsu-Chun Hsiao, Ayumu Kubota, and Jumpei Urakawa. SIBRA: Scalable Internet Bandwidth Reservation Architecture. In *Proc. NDSS*, 2016.

[27] Steven M Bellovin. A technique for counting NATted hosts. In *Proc. ACM SIGCOMM Workshop on Internet Measurement*, 2002.

[28] Steven M. Bellovin and Emden R. Gansner. Using link cuts to attack internet routing. *Tech. Rep., ATT Research, 2004, Work in Progress 2003 USENIX*, 2003.

[29] Marc Bourreau and Pinar Dogan. Unbundling the local loop. *European Economic Review*, 49(1), 2005.

[30] Lawrence S. Brakmo and Larry L Peterson. TCP Vegas: End to end congestion avoidance on a global Internet. *IEEE Journal on Selected Areas in Communications*, 1995.

[31] Peter Bright. Can a DDoS break the Internet? Sure... just not all of it. In *Ars Technica*, April 2, 2013. `http://arstechnica.com/security/2013/04/can-a-ddos-break-the-internet-sure-just-not-all-of-it/`.

[32] Jon Brodkin. Study: Comcast and Verizon connections to Cogent dropped below 0.5 Mbps. In *Arstechnica*, October 28, 2014.

[33] Hal Burch and Bill Cheswick. Tracing anonymous packets to their approximate source. In *Proceedings of the 2000 USENIX LISA Conference*, pages 319–327, 2000.

[34] Juan Caballero, Chris Grier, Christian Kreibich, and Vern Paxson. Measuring Pay-per-Install: The Commoditization of Malware Distribution. In *Proc. USENIX Security*, 2011.

[35] CAIDA Monitors. The Archipelago Measurement Infrastructure, 2002.

[36] Jin Cao, William S Cleveland, Yuan Gao, Kevin Jeffay, F Donelson Smith, and Michele Weigle. Stochastic models for generating synthetic HTTP source traffic. In *Proc. IEEE INFOCOM*, 2004.

[37] Sinead Carew. Hurricane Sandy disrupts Northeast U.S. telecom networks. In *Reuters*, October 30, 2012.

[38] Jerry Chi-Yuan Chou, Bill Lin, Subhabrata Sen, and Oliver Spatscheck. Proactive Surge Protection: a Defense Mechanism for Bandwidth-based Attacks. *IEEE/ACM Transactions on Networking*, 2009.

[39] CISCO. How To Calculate Bandwidth Utilization Using SNMP. 2005. `http://www.cisco.com/c/en/us/support/docs/ip/simple-network-management-protocol-snmp/8141-calculate-bandwidth-snmp.html`.

[40] David Clark, Robert Braden, Aaron Falk, and Venkata Pingali. FARA: Reorganizing the addressing architecture. In *ACM SIGCOMM Computer Communication Review*, 2003.

[41] Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4), 2009.

[42] Reuven Cohen and Liran Katzir. The generalized maximum coverage problem. *Information Processing Letters*, 108(1):15 – 22, 2008. ISSN 0020-0190. doi: 10.1016/j.ipl.2008.03.017.

[43] Ítalo Cunha, Renata Teixeira, and Christophe Diot. Measuring and characterizing end-to-end route dynamics in the presence of load balancing. In *Proceedings of PAM'11*, pages 235–244, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-19259-3.

[44] David Dagon, Cliff Zou, and Wenke Lee. Modeling botnet propagation using time zones. In *In Proceedings of the 13 th Network and Distributed System Security Symposium NDSS*, 2006.

[45] Dancho Danchev. Coordinated Russia vs Georgia cyber attack in progress. In *ZDNet*, August 11, 2008.

[46] B.S. Davie and A. Farrel. *MPLS: Next Steps*. Morgan Kaufmann Series in Networking. Elsevier/Morgan Kaufmann Publishers, 2008. ISBN 9780123744005.

[47] DHS. Project Shine. *ICS-CERT Monitor, Quarterly Newsletter, Oct-Dec.*, 2012.

[48] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1 (1), 1959.

[49] Benoit Donnet, Matthew Luckie, Pascal Mérindol, and Jean-Jacques Pansiot. Revealing MPLS tunnels obscured from traceroute. *ACM SIGCOMM CCR*, 42(2):87–93, 2012.

[50] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the Internet topology. *ACM SIGCOMM CCR*, 29(4), 1999.

[51] Seyed K Fayaz, Yoshiaki Tobioka, Vyas Sekar, and Michael Bailey. Bohatei: Flexible and Elastic DDoS Defense. In *Proc. USENIX Security*, 2015.

[52] Seyed Kaveh Fayazbakhsh, Luis Chiang, Vyas Sekar, Minlan Yu, and Jeffrey C Mogul. Enforcing network-wide policies in the presence of dynamic middlebox actions using FlowTags. In *Proc. USENIX NSDI*, 2014.

[53] FBI National Press Office. Over one million potential victims of botnet cyber crime. `http://www.fbi.gov/news/pressrel/press-releases/over-1-million-potential-victims-of-botnet-cyber-crime`, June 13, 2007.

[54] Aiguo Fei, Guangyu Pei, Roy Liu, and Lixia Zhang. Measurements on delay and hop-count of the Internet. In *IEEE GLOBECOM'98-Internet Mini-Conference*, 1998.

[55] Anja Feldmann, Albert Greenberg, Carsten Lund, Nick Reingold, Jennifer Rexford, and Fred True. Deriving traffic demands for operational IP networks: methodology and experience. *ACM SIGCOMM Computer Communication Review*, 30(4), 2000.

[56] P. Ferguson. Network Ingress Filtering:Defeating Denial of Service Attacks which employ IP Source Address Spoofing. *RFC 2827*, 2000.

[57] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine*, 2002.

[58] Pierre Francois, Clarence Filsfils, John Evans, and Olivier Bonaventure. Achieving sub-second igp

convergence in large ip networks. *SIGCOMM Comput. Commun. Rev.*, 35(3):35–44, July 2005. ISSN 0146-4833. doi: 10.1145/1070873.1070877.

[59] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 1977.

[60] Brian Fung. What Europe can teach us about keeping the Internet open and free. In *The Washington Post*, September 20, 2013.

[61] Lixin Gao. On inferring autonomous system relationships in the Internet. *IEEE/ACM Transactions on Networking (ToN)*, 9(6), 2001.

[62] M. Geva, A. Herzberg, and Y. Gev. Bandwidth Distributed Denial of Service: Attacks and Defenses. *IEEE Security & Privacy*, 12(1), January 2014.

[63] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. Understanding network failures in data centers: measurement, analysis, and implications. In *ACM SIGCOMM Computer Communication Review*, 2011.

[64] Phillipa Gill, Michael Schapira, and Sharon Goldberg. A survey of interdomain routing policies. *ACM SIGCOMM Computer Communication Review*, 44(1), 2014.

[65] Dimitrios Gkounis, Vasileios Kotronis, and Xenofontas Dimitropoulos. Towards Defeating the Crossfire Attack using SDN. In *arXiv:1412.2013*, 2014.

[66] Virgil D Gligor. A Note on the Denial-of-Service Problem. In *Proc. IEEE S&P*, 1983.

[67] Virgil D. Gligor. Guaranteeing access in spite of distributed service-flooding attacks. In *Security Protocols Workshop*, pages 80–96, 2003.

[68] Dan Goodin. How extorted e-mail provider got back online after crippling DDoS attack. In *Ars Technica*, November 10, 2015. `http://arstechnica.com/security/2015/11/how-extorted-e-mail-provider-got-back-online-after-crippling-ddos-attack/`.

[69] Dirk Grunwald and Douglas Sicker. Measuring the Network-Service Level Agreements, Service Level, Monitoring, Network Architecture and Network Neutrality. *International Journal of Communication*, 2007.

[70] Krishna P Gummadi, Richard J Dunn, Stefan Saroiu, Steven D Gribble, Henry M Levy, and John

Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. *ACM SIGOPS Operating Systems Review*, 37(5), 2003.

[71] Arpit Gupta, Laurent Vanbever, Muhammad Shahbaz, Sean P Donovan, Brandon Schlinker, Nick Feamster, Jennifer Rexford, Scott Shenker, Russ Clark, and Ethan Katz-Bassett. SDX: A software defined Internet exchange. In *Proc. ACM SIGCOMM*, 2014.

[72] Dongsu Han, Ashok Anand, Fahad Dogar, Boyan Li, Hyeontaek Lim, Michel Machado, Arvind Mukundan, Wenfei Wu, Aditya Akella, David G Andersen, John W. Byers, Srinivasan Seshan, and Peter Steenkiste. XIA: Efficient support for evolvable internetworking. In *Proc USENIX NSDI*, 2012.

[73] Yihua He, Michalis Faloutsos, Srikanth Krishnamurthy, and Bradley Huffaker. On routing asymmetry in the Internet. In *IEEE GLOBECOM*, 2005.

[74] Dorit S Hochbaum. Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems. In *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 1996.

[75] Christian E Hopps. Analysis of an equal-cost multi-path algorithm. *RFC 2992*, 2000.

[76] Ningning Hu, Li (Erran) Li, Zhuoqing Morley Mao, Peter Steenkiste, and Jia Wang. Locating Internet bottlenecks: algorithms, measurements, and implications. In *Proc. SIGCOMM*, 2004.

[77] Kai-Lung Hui, Seung Hyun Kim, and Qiu-Hong Wang. Marginal deterrence in the enforcement of law: Evidence from distributed denial of service attack. *Unpublished manuscript*, 2013.

[78] G. Iannaccone, Chen-Nee Chuah, S. Bhattacharyya, and C. Diot. Feasibility of ip restoration in a tier 1 backbone. *Network, IEEE*, 18(2):13 – 19, mar-apr 2004. ISSN 0890-8044. doi: 10.1109/MNET.2004.1276606.

[79] IEEE. *IEEE Policies*, chapter 7.8 IEEE Code of Ethics. IEEE, February 2012.

[80] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *Proc. IEEE FOCS*, 2000.

[81] Internet2 Network – Layer3 / IP Connectors Map, 2013.

[82] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, and Rebecca L Braynard. Networking named content. In *Proc. ACM CoNEXT*, 2009.

[83] Jaeyeon Jung, Balachander Krishnamurthy, and Michael Rabinovich. Flash crowds and denial of service attacks: characterization and implications for cdns and web sites. In *Proceedings of WWW '02*, pages 293–304, New York, NY, USA, 2002. ACM. ISBN 1-58113-449-5. doi: 10.1145/511446.511485.

[84] Kaspersky Lab. Denial of service: how businesses evaluate the threat of DDoS Attacks. *Kaspersky: IT SECURITY RISKS SPECIAL REPORT SERIES*, 2015.

[85] Dave Katz and Dave Ward. Bidirectional forwarding detection (BFD). 2010.

[86] Angelos D. Keromytis, Vishal Misra, and Dan Rubenstein. SOS: secure overlay services. In *Proc. of ACM SIGCOMM*, 2002.

[87] Simon Knight, Hung X Nguyen, Nick Falkner, Rhys Bowden, and Matthew Roughan. The Internet Topology Zoo. *IEEE JSAC*, 2011.

[88] Balachander Krishnamurthy, Subhabrata Sen, Yin Zhang, and Yan Chen. Sketch-based change detection: methods, evaluation, and applications. In *Proc. ACM IMC*, 2003.

[89] Ram Krishnan, Muhammad Durrani, and Peter Phaal. Real-time SDN Analytics for DDoS mitigation. *Open Networking Summit*, 2014.

[90] Anukool Lakhina, John W Byers, Mark Crovella, and Peng Xie. Sampling biases in ip topology measurements. In *Proceedings of IEEE INFOCOM 2003*, volume 1, pages 332–341. IEEE, 2003.

[91] Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining anomalies using traffic feature distributions. In *ACM SIGCOMM CCR*, 2005.

[92] Soo Bum Lee and Virgil D Gligor. Floc: Dependable link access for legitimate traffic in flooding attacks. In *Proc. IEEE ICDCS*, 2010.

[93] Soo Bum Lee, Min Suk Kang, and Virgil D Gligor. CoDef: collaborative defense against large-scale link-flooding attacks. In *Proc. ACM CoNEXT*, 2013.

[94] Kyriaki Levanti. *Routing Management in Network Operations*. PhD thesis, CARNEGIE MELLON

UNIVERSITY, 2012.

[95] Lun Li, David Alderson, Walter Willinger, and John Doyle. A first-principles approach to understanding the Internet's router-level topology. *ACM SIGCOMM Computer Communication Review*, 34(4), 2004.

[96] Matthew Luckie, Bradley Huffaker, Amogh Dhamdhere, Vasileios Giotsas, et al. AS relationships, customer cones, and validation. In *Proc. IMC*, 2013.

[97] Reiner Ludwig and Randy H Katz. The Eifel algorithm: making TCP robust against spurious retransmissions. *ACM SIGCOMM Computer Communication Review*, 2000.

[98] Damien Magoni. Tearing down the Internet. *IEEE Journal on Selected Areas in Communications*, 21(6), 2003.

[99] Ratul Mahajan, Steven M Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker. Controlling high bandwidth aggregates in the network. *ACM SIGCOMM CCR*, 2002.

[100] Ratul Mahajan, Neil Spring, David Wetherall, and Tom Anderson. Inferring link weights using end-to-end measurements. In *Proc. ACM SIGCOMM Workshop on Internet measurement*, 2002.

[101] BB Mandelbrot. Information theory and psycholinguistics. *BB Wolman and E*, 1965.

[102] Zhuoqing Morley Mao, Ramesh Govindan, George Varghese, and Randy H Katz. Route flap damping exacerbates Internet routing convergence. *ACM SIGCOMM Computer Communication Review*, 32(4), 2002.

[103] Zhuoqing Morley Mao, Jennifer Rexford, Jia Wang, and Randy H Katz. Towards an accurate AS-level traceroute tool. In *Proc. of ACM SIGCOMM*, 2003.

[104] Pietro Marchetta, Valerio Persico, Ethan Katz-Bassett, and Antonio Pescapé. Don't trust traceroute (completely). In *Proc. CoNEXT Student workshop*, 2013.

[105] Pietro Marchetta, Valerio Persico, and Antonio Pescapé. Pythia: yet another active probing technique for alias resolution. In *Proc. CoNEXT*, 2013.

[106] Matthew Prince. The DDoS That Almost Broke the Internet. *CloudFlare Blog*, March 27, 2013.

[107] Declan McCullagh. How Pakistan knocked YouTube offline (and how to make sure it never happens

again). In *CNET*, February 25, 2012.

[108] Machael Mimoso. 400 Gbps NTP Amplification attack alarmingly simple. In *Threatpost*, Feb. 13, 2014.

[109] Jelena Mirkovic and Peter Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2):39–53, April 2004. ISSN 0146-4833. doi: 10.1145/997150.997156.

[110] David Moore, Geoffrey Voelker, and Stefan Savage. Inferring Internet Denial-of-Service Activity. In *Proc. of Usenix Security*, 2001.

[111] A. Mortensen. DDoS Open Threat Signaling Requirements. *IETF draft-mortensen-threat-signaling-requirements-00*, 2015.

[112] R. Moskowitz and P. Nikander. Host Identity Protocol (HIP) Architecture. *RFC 4423*, 2006.

[113] MPLS Traffic Engineering (TE)–Automatic Bandwidth Adjustment for TE Tunnels. `http://www.cisco.com/c/en/us/td/docs/ios/12_0s/feature/guide/fsteaut.html\#wp1015347`.

[114] T.D. Nadeau. *MPLS Network Management: MIBs, Tools, and Techniques*. Morgan Kaufmann Series in Networking. Elsevier Science, 2002. ISBN 9781558607514.

[115] Onuttom Narayan and Iraj Saniee. Scaling of load in communications networks. *Phys. Rev. E*, 82: 036102, Sep 2010. doi: 10.1103/PhysRevE.82.036102.

[116] Jose Nazario. DDoS attack trends through 2009-2011. *NANOG 54*, February 2012.

[117] Mark EJ Newman. A measure of betweenness centrality based on random walks. *Social networks*, 2005.

[118] Erik Nordström, David Shue, Prem Gopalan, Robert Kiefer, Matvey Arye, Steven Y Ko, Jennifer Rexford, and Michael J Freedman. Serval: An end-host stack for service-centric networking. In *Proc USENIX NSDI*, 2012.

[119] Erik Nygren, Ramesh K Sitaraman, and Jennifer Sun. The Akamai network: a platform for high-performance Internet applications. *ACM SIGOPS Operating Systems Review*, 44(3), 2010.

[120] Christos Pappas, Raphael M Reischuk, and Adrian Perrig. FAIR: Forwarding Accountability for Internet Reputability.

[121] Bryan Parno, Dan Wendlandt, Elaine Shi, Adrian Perrig, Bruce Maggs, and Yih-Chun Hu. Portcullis: protecting connection setup from denial-of-capability attacks. In *Proc. ACM SIGCOMM*, 2007.

[122] Parveen Patel, Deepak Bansal, Lihua Yuan, Ashwin Murthy, Albert Greenberg, David A. Maltz, Randy Kern, Hemant Kumar, Marios Zikos, Hongyu Wu, Changhoon Kim, and Naveen Karri. Ananta: Cloud Scale Load Balancing. In *Proc. ACM SIGCOMM*, 2013.

[123] Vern Paxson. End-to-end routing behavior in the internet. In *Conference proceedings on SIGCOMM '96*, pages 25–38, New York, NY, USA, 1996. ACM. ISBN 0-89791-790-1. doi: 10.1145/248156.248160.

[124] Vern Paxson. Strategies for sound Internet measurement. In *Proc. IMC*, 2004.

[125] PlanetLab. `http://www.planet-lab.org/`.

[126] Ivan PL Png, Chen-Yu Wang, and Qiu-Hong Wang. The deterrent and displacement effects of information security enforcement: International evidence. *Journal of Management Information Systems*, 2008.

[127] Ponemon Institute. Safeguarding Brand And Revenue This Holiday Season. *The 2013 eCommerce Cyber Crime Report*, 2013.

[128] Zafar Ayyub Qazi, Cheng-Chun Tu, Luis Chiang, Rui Miao, Vyas Sekar, and Minlan Yu. SIMPLE-fying middlebox policy enforcement using SDN. In *Proc. ACM SIGCOMM*, 2013.

[129] Dipankar Raychaudhuri, Kiran Nagaraja, and Arun Venkataramani. MobilityFirst: a robust and trustworthy mobility-centric architecture for the future Internet. *ACM SIGMOBILE Mobile Computing and Communications Review*, 2012.

[130] S.M. Ross. *Introduction to Probability and Statistics for Engineers and Scientists*. Wiley series in probability and mathematical statistics. Academic Press/Elsevier, 2009. ISBN 9780123704832.

[131] Christian Rossow. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *Proc.*

NDSS, 2014.

[132] Jerome H Saltzer. Slammer: An urgent wake-up call. In *Springer, Computer Systems*. 2004.

[133] Max Schuchard, Abedelaziz Mohaisen, Denis Foo Kune, Nicholas Hopper, Yongdae Kim, and Eugene Y. Vasserman. Losing control of the Internet: using the data plane to attack the control plane. In *Proc. NDSS*, 2010.

[134] Siddhartha Sen, David Shue, Sunghwan Ihm, and Michael J Freedman. Scalable, optimal flow routing in datacenters via local link balancing. In *Proc. ACM CoNEXT*, 2013.

[135] Aman Shaikh, Anujan Varma, Lampros Kalampoukas, and Rohit Dube. Routing stability in congested networks: Experimentation and analysis. In *Proc. of ACM SIGCOMM*, pages 163–174, 2000.

[136] Ali Reza Sharafat, Saurav Das, Guru Parulkar, and Nick McKeown. MPLS-TE and MPLS VPNs with OpenFlow. In *ACM SIGCOMM CCR*, 2011.

[137] Yuval Shavitt et al. The DIMES Project, 2008.

[138] Justine Sherry, Ethan Katz-Bassett, Mary Pimenova, Harsha V. Madhyastha, Thomas Anderson, and Arvind Krishnamurthy. Resolving ip aliases with prespecified timestamps. In *Proceedings of IMC '10*, pages 172–178, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0483-2. doi: 10.1145/1879141.1879163.

[139] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP topologies with Rocketfuel. *ACM SIGCOMM CCR*, 32(4), 2002.

[140] Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to own the internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium*, pages 149–167, Berkeley, CA, USA, 2002. USENIX Association. ISBN 1-931971-00-5.

[141] RA Steenbergen. A practical guide to (correctly) troubleshooting with traceroute. *North American Network Operators Group*, pages 1–49, 2009.

[142] Richard Steenbergen. MPLS RSVP-TE Auto-Bandwidth - Lessons Learned. *NANOG 58*, 2013.

[143] Chris Strohm and Engleman Eric. Cyber Attacks on U.S. Banks Expose Computer Vulnerability.

In *Bloomberg*, Sept. 27, 2012.

[144] Ahren Studer and Adrian Perrig. The Coremelt attack. In *Proc. ESORICS*, 2009.

[145] Michael P. H. Stumpf and Mason A. Porter. Critical truths about power laws. *Science*, 335(6069): 665–666, 2012. doi: 10.1126/science.1216142.

[146] L. Subramanian, S. Agarwal, J. Rexford, and R.H. Katz. Characterizing the internet hierarchy from multiple vantage points. In *Proc. of IEEE INFOCOM*, 2002.

[147] Kun Tan, Jingmin Song, Qian Zhang, and Murari Sridharan. A compound TCP approach for high-speed and long distance networks. In *Proc. IEEE INFOCOM*, 2006.

[148] Renata Teixeira, Aman Shaikh, Tim Griffin, and Jennifer Rexford. Dynamics of hot-potato routing in IP networks. *ACM SIGMETRICS Performance Evaluation Review*, 32(1), 2004.

[149] H Thomas et al. *Introduction to algorithms*. MIT press, 2009.

[150] Traceroute.org. Public route server and looking glass site list. `http://www.traceroute.org/`.

[151] Patrick Traynor, William Enck, Patrick Mcdaniel, and Thomas La Porta. Exploiting open functionality in sms-capable cellular networks. *Journal of Computer Security*, 16(6):713–742, 2008.

[152] Vytautas Valancius, Cristian Lumezanu, Nick Feamster, Ramesh Johari, and Vijay V Vazirani. How many tiers?: pricing in the Internet transit market. In *ACM SIGCOMM CCR*, 2011.

[153] Iljitsch van Beijnum. Playing chicken: ISP depeering a high-school lovers quarrel. In *Arstechnica*, March 21, 2008.

[154] Patrick Verkaik, Dan Pei, Tom Scholl, Aman Shaikh, Alex C Snoeren, and Jacobus E Van Der Merwe. Wresting Control from BGP: Scalable Fine-Grained Route Control. In *USENIX ATC*, 2007.

[155] Paul Vogt. Minimum cost and the emergence of the Zipf-Mandelbrot law. In *Proc. Artificial Life IX*, 2004.

[156] Michael Walfish, Mythili Vutukuru, Hari Balakrishnan, David Karger, and Scott Shenker. DDoS defense by offense. In *Proc. ACM SIGCOMM*, 2006.

[157] Feng Wang and Lixin Gao. On inferring and characterizing internet routing policies. In *Proceedings of IMC '03*, pages 15–26, New York, NY, USA, 2003. ACM. ISBN 1-58113-773-7. doi: 10.1145/948205.948208.

[158] Jessie Hui Wang, Dah Ming Chiu, John CS Lui, and Rocky KC Chang. Inter-AS inbound traffic engineering via ASPP. *IEEE Transactions on Network and Service Management*, 4(1), 2007.

[159] Ning Wang, Kin Ho, G. Pavlou, and M. Howarth. An overview of routing optimization for Internet traffic engineering. *IEEE Communications Surveys Tutorials*, 10(1), 2008.

[160] XiaoFeng Wang and Michael K Reiter. Defending against denial-of-service attacks with puzzle auctions. In *Proc. IEEE S&P*, 2003.

[161] XiaoFeng Wang and Michael K Reiter. Mitigating bandwidth-exhaustion attacks using congestion puzzles. In *Proc ACM CCS*, 2004.

[162] Yubo Wang, Shi Xiao, Gaoxi Xiao, Xiuju Fu, and Tee Hiang Cheng. Robustness of complex communication networks under link attacks. In *Proc. ICAIT*, 2008.

[163] Martyn Williams. DDoS attack in March likely N.Korean work, says McAfee. In *PCWorld*, July 6, 2011.

[164] Walter Willinger, David Alderson, and John C Doyle. Mathematics and the Internet: A Source of Enormous Confusion and Great Potential. *Notices of the AMS*, 56(5), 2009.

[165] Xiaowei Yang and David Wetherall and Thomas Anderson. A DoS-limiting network architecture. In *SIGCOMM '05*, 2005. ISBN 1-59593-009-4. doi: http://doi.acm.org/10.1145/1080091.1080120.

[166] Wen Xu and Jennifer Rexford. MIRO: Multi-path Interdomain ROuting. In *Proc. of ACM SIGCOMM*, 2006.

[167] Abraham Yaar, Adrian Perrig, and Dawn Song. Pi: A path identification mechanism to defend against DDoS attacks. In *Proc. IEEE S&P*, 2003.

[168] Abraham Yaar, A. Perrig, and D. Song. SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks. In *Proc. IEEE S&P*, 2004.

[169] Serdar Yegulalp. Level 3 accuses Comcast, other ISPs of 'deliberately harming' broadband service.

In *InfoWorld*, May 6, 2014.

[170] Che-Fn Yu and Virgil D Gligor. A Formal Specification and Verification Method for the Prevention of Denial of Service. In *Proc. IEEE S&P*, 1988.

[171] Minlan Yu, Lavanya Jose, and Rui Miao. Software Defined Traffic Measurement with OpenSketch. In *Proc. USENIX NSDI*, 2013.

[172] Min Zhang, Maurizio Dusi, Wolfgang John, and Changjia Chen. Analysis of UDP traffic usage on Internet backbone links. In *Proc. IEEE SAINT*, 2009.

[173] Xin Zhang, Hsu-Chun Hsiao, Geoffrey Hasker, Haowen Chan, Adrian Perrig, and David G Andersen. SCION: Scalability, control, and isolation on next-generation networks. In *Proc. IEEE S&P*, 2011.

[174] Ying Zhang, Z. Morley Mao, and Jia Wang. Low-rate TCP-targeted DoS attack disrupts Internet routing. In *Proc. of NDSS*, 2007.