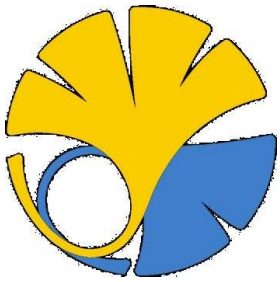
Deployable Mechanisms for Distributed Denial-of-Service (DDoS) Attack Mitigation



Soon Hin Khor

Interdisciplinary Information Studies (III)

University of Tokyo

A thesis submitted for the degree of

Doctor of Philosophy

24 March 2010

I would like to dedicate this thesis to my loving parents and
wonderful friends ...

Acknowledgements

First and foremost, I need to thank my advisor, Dr. Akihiro Nakao, for ensuring that my rice bowl is always filled, bestowing me with the freedom to explore my research potential and offering research guidance along this arduous path. These 3 ingredients have proved critical for me to last the entire distance. I am also thankful for the support in various job and scholarship applications as well as fighting my corner in the graduation process.

I am also extremely grateful to Dr. Nicolas Christin, my Master's course advisor at Carnegie Mellon University (CMU), who is ever-ready to assist me and continue to open up new avenues in my life even today despite the fact that I graduated from CMU, almost a good 3 years ago.

Let me count the ways.
The recommendation letters.
The chance to teach in CMU.
The support in my pursuit of establishing a start-up.
The ramen.

This truly defines the ultimate meaning of education. Many may have gone to universities but only a lucky few will experience education of this kind.

The gamut of emotions that I was immersed in during this academic pursuit, probably encompasses what one can encounter in an entire lifespan, barring, death.

When the nights seem endless, Aiko, Ayumi and Kana have taken turns to amuse me with their antics and ebullience.

When non-academic issues rear their ugly heads, Yabu and Wataru, puts life into perspective.

When I tire of people, the innocence and exuberance of Michi, remind me that there is hope.

When I need cheer and laughter in a language more natural to me, technology brings Julie and Ang Joo virtually closer to me.

When I am hungry at dinner-time or need the adrenaline rush of tumbling down some snowy slopes, there is Iwase.

Without my friends, this journey will be so much poorer and less bearable.

My gratitude also goes out to International Information Systems Security Certification Consortium, Inc., (ISC^2) and Japan Student Services Organization (JASSO) for providing me with scholarships, without which my academic pursuit would not have been possible. Thanks also to University of Tokyo for providing a system to disburse the ISC^2 scholarship monies to me.

I also want to thank the professors and the kind people who have set aside bucket-loads of valuable time to read my research, offered feedback and tried to help me to succeed.

I need to thank all my lab mates for putting up with me and the side-effects of my presence.

Last but not least, words fail me on how to even start thanking my parents for always having the belief but yet accepting me for whatever I was, am and will be. All the gifts and money that I shower them with is just a poor attempt to make up for the time that we are apart.

Abstract

The goal of our research is to make the Internet more resilient against Distributed Denial-of-Service (DDoS). Although there is similarly themed existing research, what sets mine apart is the focus on "deployable resiliency". After more than 10 years of research, despite DDoS mitigation improving significantly, they still lack deployability and/or comprehensive resiliency, thus the term "deployable resiliency"; they require either global-scale cooperation and complex changes to Internet core infrastructure or they incur huge initial outlay to purchase resource for decent yet not comprehensive DDoS resiliency. The former results in DDoS research being un-adopted while the latter restricts deployability to only large wealthy organizations.

To rectify the problem, we examine issues that hamper resiliency and existing proposals that mitigate them. Our strive for deployable resiliency mechanism pushed us to relentlessly question what is holding back adoption of existing proposals from an economic and technological standpoint. The end result consists of 2 economic frameworks and their accompanying defense-in-depth mechanisms.

As a deployability guide, we design a base framework, Burrows, that rectifies existing economic inefficiencies that plague many existing proposals, namely negative externality, misaligned economic incentives and overwhelming infrastructure modification requirements. A second framework, KUMO, extends the first by facilitating the harness of large quantities of Internet resources for DDoS defense through lowering the barrier for cooperation. On top of these frameworks, we construct a multi-prong DDoS defense focusing on deterrence and client/server reaction. Each prong can be implemented independently

to ease deployment concerns but yet ultimately, they complement each other when fully deployed.

The *deterrence* mechanism (Overfort) offers automated malicious packet traceback to the Internet Service Provider (ISP) harboring the originator, thereby forcing/enabling the ISP to act against perpetrators, to avoid black-listing. Path-selection empowers *client reaction* (AI-RON-E) to hotspots by selecting an alternative path while connection establishment urgency signaling empowers *client reaction* to poor service from server by expending more resource to send a stronger signal and receive higher priority. Detection of unsolicited traffic and traffic reception control (Overfort, sPoW) empowers *server reaction* to filter undesirable traffic before it drains server resources and clogs uplinks.

The end result is a high deployable resiliency DDoS mitigation mechanism; it is (1) unilaterally deployable because it is cheap, i.e., based on a pay-as-you-use model, while requiring only minimal changes to existing infrastructure, and (2) resilient against all types of DDoS, with additional deterrent and client empowerment mechanisms.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Scope of Study

A Distributed Denial-of-Service (DDoS)[1] attack can be carried out on anything that offers a service. This can range from cutting off the telephone cables in an office to prevent incoming/outgoing calls, sending a single/bunch of packets to exploit a system/application vulnerability, which terminates the service, hijacking Border Gateway Protocol (BGP) prefix (128)) thus, the reachability of the authentic server, to coordinating a mass web page request/arbitrary packet flood from an army of compromised hosts to bring a server to its knees.

For the last form of attack, depending on the content type of the victim server, which can be static or dynamic, the defense and its effectiveness is different. Static contents are contents whose relevancy lasts for periods longer than a few hours and thus cache-able. This includes news, Frequently Asked Questions (FAQs), weather reports, etc., including video streaming, whose data timeliness is critical but should not be confused with its relevancy period. Dynamic contents, on the other hand, are contents whose relevancy is fleeting thus mandating real-time accessibility for decision-making and processing, e.g., banking transactions, chat messages, online game, etc. As the former can be sufficiently dealt with using existing caching and replication technology such as Akamai (4), CoralCDN (45), and CoDeeN (120), the focus of this study is on DDoS protection of the latter.

[1] We use DoS and DDoS, interchangeably unless the need to distinguish them arises.

1.2 Motivation

Due to the ease (no vulnerabilities and minimal skills required) of perpetrating a DDoS attack compounded by the tedious concerted multi-party effort required to prosecute the perpetrator, mainly due to lack of traceback mechanisms and the lag of legal enforcement behind electronic crimes, the threat of DDoS is incessant. In between 2001 to 2004, 68,700 attacks bombarding 34,700 victims in 5,300 domains were inferred from experiments carried out by Moore et al. (78), while Akamai's State of the Internet (3) report has documented various high-profile DDoS attacks every quarter since its inception in 2008. The numbers reported by the former is also likely to be a lower bound due to the limitation of the "back-scatter analysis" methodology employed; inference is only possible for DDoS attacks that spoof their traffic origins to avoid accountability. Besides network measurements, anecdotal evidence from end-user and infrastructure operator perspectives are equally foreboding. In the CSI Computer Crime and Security Survey 2008 (31), 21% of 522 correspondents from end-users of U.S. corporations, government agencies, financial institutions, medical institutions and universities, have experienced some form of DDoS attack on their infrastructure, while in the Arbor Networks Annual Infrastructure Report 2008 (15), DDoS has remained one of the top security concerns for Internet Service Providers (ISPs) since 2005 and furthermore, 21% of 66 reported that DDoS handling consumed the most amount of their operation resources.

The consequences of the lag in legal enforcement is chilling and none more visible than the brash and bold attempts to overtly sell DDoS tools or organize concerted DDoS on the Internet. The activities of the former are carried out over Internet Relay Chat (IRC) (80), which is a chat room protocol for people of similar interests to gather virtually, or forums, which is the virtual equivalent of classified ads or bulletin boards (44; 110; 114). For the latter, two immediate examples that come to mind are the 2008 DDoS attempt on Channel News Network (CNN) organized by `http://hacksa.cn`, and the current 2009 attempt on a list of Iranian government websites using tools offered on a Google hosted site, `http://sites.google.com/site/nedasites/Home`.

With the number of Internet users booming to 1.7 billion on 30 June 2009 (56) and computer security not getting any easier, the number of compromised hosts and consequently, the scale of coordinated attacks are likely to grow in the future.

The introduction of resource-rich cloud computing platforms, where adopters are charged based on usage of the cloud's server and network resources, known as "pay-as-you-use" or utility computing, may appear to overcome DDoS, i.e., resource bottlenecks are eliminated. However, upon deeper analysis, these clouds merely transform a conventional DDoS attack on server and network resources to a new breed that targets the cloud adopter's economic resource. We term this attack as economic DDoS (eDDoS). In other words, the adopter's economic ability, i.e., the financial means to pay for resources consumed by both legitimate and DDoS traffic, becomes the new bottleneck for attacks.

In summary, the ease of attack, the lack of law enforcement, the growth in Internet complexity with no deployable mitigation in the horizon are factors that motivate this study.

1.3 Problem Statement

The pressing issue of DDoS has galvanized much research. However, two issues plague existing research: (1) they were designed prior to the emergence of cloud computing so they lack awareness of eDDoS thus cannot defend against it effectively, and (2) widespread adoption of DDoS mitigation research remains elusive, most likely due to 3 factors that we briefly posit here:

Negative Externality The effectiveness of a deployment is reliant upon cooperation with many other parties, who are possibly competitors or simply indifferent towards DDoS.

Misaligned Economic Incentive The party in the best position (ISP) to deploy DDoS mitigation mechanisms is not the party afflicted by DDoS (end-user) thus the party in the privileged position lacks deployment incentives, leaving the afflicted party defenseless.

Ossified Infrastructure DDoS mitigation mechanisms require ISPs to abandon or make overwhelming modifications to their existing infrastructure investments, thus diminishing their appeal (116).

1.4 Research Goal

Our goal is to develop *effective* and *deployable* DDoS attack mitigation mechanisms. Our definition of "effective" is that a mechanism or combination of them should be able to mitigate existing DDoS attacks on dynamic contents (Section 1.1) as well as the newly emerged eDDoS attacks on billable resources. Our definition of "deployable" is that a mechanism must not require modifications to ossified core Internet routers and the deployment resources, in terms of time, effort and money, must be within the means of a single small-sized ISP, which we define as having a customer-base, thus Internet access routers (routers installed at customer locations) of at least 15,000.

1.5 Design Principles in a Nutshell

This section presents our design principles for developing effective and deployable DDoS mitigation mechanisms.

1.5.1 Effectiveness Principles

Our proposal has to be effective against DDoS attacks on dynamic contents and eDDoS attacks on billable resources. To defend against DDoS attacks on dynamic contents, we propose two options.

Unfair Arms-race We engage attackers in a resource arms-race but one where the winning odds are in our favor, by designing a mechanism that ease the server task of harnessing DDoS defense resource so that it exceeds attackers' aggregated flooding capacity.

Automatic Traceback and Black-list In a situation, where we cannot tilt the balance of an arms-race, we propose an automatic traceback mechanism to

locate perpetrators with an automatic black-listing scheme that prevents them from further communication with the targeted server. A rapid trace-back and black-listing implementation can shut out all attackers before server resources are exhausted.

eDDoS occurs because of the straight-forward billing mechanism of cloud platforms that indiscriminately charge resource usage whether by legitimate or DDoS traffic to cloud adopters. To tackle eDDoS, we propose:

Do-It-Yourself Adopter-friendly Cloud Billing Mechanism Our research empowers a cloud adopter to modify the cloud platform's billing mechanism on her own, to reduce the cost of resource used for prioritizing every packet destined to her service so that legitimate traffic may be favored over DDoS traffic.

1.5.2 Deployability Principles

To achieve deployability, we design our mitigation mechanisms to abide by the following 3 design principles:

"Semi-closed Model" Internet We refactor the Internet from an "open model" where everybody can freely connect, to a "semi-closed model" where a server is empowered to explicitly declare who it is willing to receive traffic from. All traffic destined to the server is forced through control points where the server can dictate reception and drop unsolicited traffic before they reach it. Thus, DDoS exposure is now limited to those control points, which are fewer and DDoS traffic can be tamed further away from the server, compared to the open model where attacks can come from every conduit and stemming of DDoS traffic occurs near the server, which is ineffective. With fewer exposure/control points, it is also easier to deploy and upgrade DDoS defense.

Ease of Harness of Required DDoS Defense Resource DDoS is, in principle, an arms race; the party with more resources prevails. Therefore, a mitigation mechanism should facilitate the harness of required DDoS defense resource, which may be achieved in 3 ways: (1) facilitate amassing

many limited-resource systems distributed over the Internet, (2) facilitate amassing few rich-resource systems in localized areas or (3) reduce the required resource for successful DDoS mitigation.

Ease of Component Deployment By opting to use Internet edge nodes, which are located further away from the core Internet infrastructure, thus easier to introduce, modify and upgrade by any Internet user, instead of difficult-to-modify core routers, whose access is restricted to ISPs, we increase deployability.

1.6 Research Questions

The following research questions helped us formulate our design principles and guide how we design and develop DDoS mitigation frameworks, architectures and mechanisms to fulfill those principles.

- What research has been done in the last 10 years and why have they not been successful? *DoS mitigation proposals have often overlooked deployability and incentive factors that strongly influence the adoptability of a proposal, especially one that requires cooperation from multiple parties with conflicting interests. See Related Work (Section 2.2).*

- How do we address the factors that hinder DDoS mitigation mechanism adoption—negative externality, misaligned economic incentives and ossified infrastructure? *We introduce a framework that can address those economic issues, which future research can use as a guide. See Burrows (Section 3).*

- Due to the distributed deployment requirement of most DDoS mechanisms in order to amass resources from many locations, acquire multiple viewpoints and stem attacks as early as possible at those deployment points, cooperation between Internet parties becomes mandatory thus negative externality (indifference to deployment) often rear its ugly head. How do we encourage cooperation? *KUMO (Section 6), the resource harness framework, eases the task of amassing the required DDoS defense resource as well*

as offer financial incentives to foster cooperation among distributed participants.

- Is incremental design the best way to encourage participation, i.e., a party will participate solely based on the fact that participation improves its existing defense, regardless of the ineffectiveness of the overall defense if a critical level of cooperation is not achieved? *Incremental approach is a minimal requirement for any cooperative mechanism. However, it is better to introduce economic incentives to hasten cooperation, e.g., in KUMO, economic incentives increases the quantity of available DDoS defense resource quickly.*

- Or is it possible, contrary to conventional wisdom of existing research (75), to reduce the deployment footprint and cooperation while remaining effective? *The use of a few localized and resource-rich resources such as cloud computing allays the requirement for widespread deployment and cooperation thereby achieving effectiveness while increasing deployability but at the expense of losing the ability stem attacks near the sources. See sPoW (Section 7).*

- Parties afflicted by DDoS are not in the most strategic position to take react while ISPs who are, are not incentivized to do so without substantial profit. "Misaligned economic incentives" is an issue that often requires rectification by legal enforcement. Given that it takes many long debates or a disaster of significant proportion for authorities to get their act together, can we develop a technology to re-align incentives; empower parties keen on DDoS protection with a platform for cooperation? *Burrows and Overfort empowers DDoS-afflicted parties with platforms to cooperate. KUMO, sPoW and AI-RON-E goes a step further by enabling unilateral deployment. Besides being a cooperative platform, Overfort can also be deployed by a single entity.*

- Can we reduce the modification required by DDoS mitigation mechanisms on existing ossified infrastructure through better design? *Overfort innovatively utilizes mandatory destination IP field as unspoofable source identity*

while AI-RON-E uses IP spoofing to coax existing routers to create indirect paths and sPoW introduces self-verifying puzzles that enhance a client's ability to compete against DDoS agents (zombies). These features are novel because they avoid making demanding changes to well-accepted protocols and core infrastructure.

- Even if it is impossible to design a deployable mechanism that offers full protection, can we reduce the probability or the impact of a persistent DDoS onslaught? *We adopt a multi-prong approach for DDoS defense, e.g., deterrence (Overfort), prevention (Overfort), client reaction (AI-RON-E, sPoW), server reaction (Overfort, sPoW).*

1.7 Overview and Contributions of Study

Based on those research questions, we design a framework (Burrows) that can address the afore-mentioned three economic issues and further enhance it with a resource harness framework (KUMO). Following that, we design mechanisms based on the Burrows framework to provide multi-prong defense against DDoS.

In the following subsections, we concisely describe the workings of each mechanism and punctuate its contribution to research.

1.7.1 Burrows

Burrows' contribution is a DDoS mitigating framework that be used as a design foundation for aligning economic incentives and minimizing negative externality while preserving existing infrastructure investment. By restricting connectivity to a protected server through mediating control points, referred to as intermediaries, where traffic filtering and prioritization technology can be deployed, the server's security now relies solely on intermediaries within the server owner's control instead of being dependent on various distributed Internet components owned by various third parties, thus minimizing negative externality. Burrows realigns economic incentives by empowering server owners who are keen on DDoS protection to cooperate by pooling their intermediaries to build this protection layer around their servers without assistance from dis-interested parties, e.g., their ISPs. The

tricky problem of negative externality and economic incentive misalignment has now been transformed into a more technological one; how do we differentiate between good and bad traffic or in more general terms, how do we identify traffic that is desired/solicited by the protected servers at those intermediaries. In other words, intermediaries offer traffic reception control, which does not exist in the current Internet. Secondly, the number of intermediaries or their aggregated resources will determine the size of DDoS onslaught that the cooperative defense can withstand.

1.7.2 Overfort

Overfort's contribution is a novel mechanism that uses virtual resources, which are cheaper than physical ones, to automatically locate zombies and black-list the entities, e.g., ISPs, that offer connectivity to those zombies, by rejecting the entities' query about the server location. Due to "fate-sharing", legitimate clients within an ISP harboring zombies, are also punished and their dissatisfaction can exert economic pressure on the ISP, galvanizing it to clean up or monitor its customer base to prevent black-listing, thus preventing DDoS attacks. It is to our knowledge that Overfort is the only automatic traceback and black-list mechanism that is deployable by a single small-sized ISP. The difference between Overfort compared to existing traceback and black-list mechanisms (Section 2.2.5) is that all our traceback and black-list enforcement components are within the control of a single deployer. Other schemes require assistance from third parties to aid in traceback or enforce black-listing, which is deemed economically illogical (24). For example, an ISP is not going to enforce punishment on her own paying clients based on a black-list created by other parties such as the remote ISP of a server under attack. The black-list only needs to be created periodically and it can also be shared with all other ISPs. Overfort's method of segregating zombie-serving entities using virtual resources and barring them from re-discovering a path to the server, instead of engaging in an arms race through over-provisioning implies that Overfort can successfully defend against DDoS even it if has less resources than attacker—a property that is unique.

1.7.3 AI-RON-E

AI-RON-E's (pronounced as 'irony') contribution is a non-centralized, lightweight mechanism, i.e., without requiring extensive network probing or data storage of the entire Internet topology, to discover alternative paths to a desired server dynamically. It empowers clients to bypass congested points, e.g., DDoS, without server assistance, by employing IP spoofing to force "triangular routing" (also known as traffic deflection); it makes a packet detour to a selected point en-route to destination, much like how one takes an alternate road when the most direct path to the destination is congested. Using IP spoofing to create covert channels (95) for data transmission is not new but in AI-RON-E, the scale and purpose that we employ it for—mobilizing all Internet routers as deflection points without requiring changes to them, in order to bypass hotspots is novel.

1.7.4 KUMO

One key question raised in Burrows, is the extent of successful deployment of cooperative DDoS defense relies on the amount of shared resources accumulated from all participants. KUMO's contribution is facilitating any existing system or future one to easily act as a shared DDoS defense resource, e.g., intermediaries, without it being even aware, i.e., no modifications, as well as offer an accounting platform to compensate the system financially for its resource usage. We believe such an incentive system can help seed the growth of shared resources.

1.7.5 sPoW

When the number of connection requests exceeds server capacity, the server needs a way to prioritize requests. Proof-of-Work (PoW) empowers a client to signal its connection urgency through its willingness to expend its own resources to generate a proof that reflects consumed resources. The server utilizes a PoW verifier to confirm the validity of the proof as an indicator level of the client's resource consumption, enabling the server to prioritize requests during DDoS.

In conventional PoW schemes, the verifier role is crucial; it has to process *every single packet* to ensure that the packet is accompanied by a valid puzzle solution

and determine the urgency signal embedded in the solution. The implications are twofold: (1) a PoW verifier needs to be packed with plenty of resources and (2) the usage of a PoW verifier changes the attack focal point from the server to the verifier itself; attackers can just send packets with randomly generated incorrect puzzle solutions that the verifiers still need to expend to process. To deal with (1), we can deploy the PoW verifier in a cloud platform, however because of (2) such deployment will result in eDDoS. This is where sPoW comes in. It is, to our knowledge, the first self-verifying PoW; instead of generating a proof that signals its urgency that needs to be verified, a client solves a PoW puzzle by deciphering it to discover a hidden channel to communicate with the server. In other words, no verification is necessary. To empower clients to signal their urgency, clients request and solve PoW puzzles whose levels reflect their urgency; each puzzle leads to a connection request channel that is served at a priority level reflected by the puzzle's difficulty level. Novelty aside, obviating the need for PoW verifiers that needs to be installed in strategic locations, e.g., in the Internet core, which faces stiff resistance from ISPs due to the disruption and uncertainty of new technology, saves implementation/operational resources and greatly enhances deployability but most importantly it removes the possibility of eDDoS.

1.8 Limitations

1.8.1 Resilient Name Service

Burrows, Overfort, KUMO and sPoW relies on a naming service; a service that converts well-known server hostname to an identity, e.g., an IP address belonging to an intermediary that is aware of how to forward packets to the server or the server can retrieve data from. We have designed the systems in such a way that they only require the naming service to provide small dynamic pieces of data that needs no synchronization, i.e., no need for revocation or systematic replication even if the naming infrastructure is distributed. Moreover, it is possible to make simple modification to existing widely used Domain Name Service (DNS) or the futuristic but backwards compatible peer-to-peer based CoDoNS (89) to serve the

required data. The resiliency of existing DNS through Anycast (2) as well as the scalability of CoDoNS makes the assumption that the naming service is resilient realistic.

1.8.2 Imperfect DDoS Defense

Nothing in our system prevents DDoS completely; instead they improve the chances of good clients reaching a server by empowering them to compete for server resources, e.g., by finding an alternative faster path to the server or sending a stronger urgency signal through resource expenditure to gain higher priority service. This empowerment is not without trade-offs. They result in more resource consumption on the clients themselves, in terms of storage, e.g., AIRON-E cache, or CPU, e.g., sPoW puzzle resolution. Clients with little resources may be disadvantaged. However, this can be rectified through the use of proxy to aid economically-related clients, e.g., low-resource mobile phone clients can utilize their provider proxies. A longer connection establishment and data transfer time is also to be expected during DDoS.

1.8.3 No Client Modification Caveat

Our claim that no client modification is required is based on the assumption that clients are willing to at least make a few mouse clicks to launch and grant zero-installation executables possibly full access to their systems. The danger one normally associates with such an action can be allayed by using signed certificates from top-level certification authority, e.g., Verisign, to vouch for the executables.

1.8.4 No Server Modification Caveat

Our claim that no server modification is necessary holds true because the server/application requires no modification to utilize our research. However, we do need to install a server-side component, which can be co-located on the server or on a separate system. This component hides the details of our technology from the server. Note that a single server-side component can be utilized by multiple servers thus sharing the burden/cost of setup and operations.

1.8.5 Hidden Server Location Enforcement

Since our proposals are based on having intermediaries that act as traffic control points for a protected server to accept/reject traffic, we rely on the need to prevent direct server connectivity. To enforce indirect connectivity, we advocate concealing the reachability of the server, e.g., hiding the IP address from clients, but unfortunately, the server reachability has to be revealed to the intermediaries. Depending on the design for a mechanism, indirect reachability enforcement varies from (1) utilizing technological solutions that requires setup cost and expertise between each intermediary to server, e.g., BGP-MPLS-VPN (Burrows) or plain VPN (sPoW), (2) using only trusted intermediaries, e.g., a single-entity owned intermediaries (Overfort), (3) relying on financial relationships between protected server and intermediary providers that places the onus of protecting hidden server location on intermediary providers (KUMO) and requesting filtering assistance from ISP when intermediaries cannot be trusted, as proposed in Phalanx (38), but also deployable in Overfort, KUMO, Burrows.

1.9 Study Presentation Layout

In the next section, we describe the background of our research, specifically, why DDoS occurs and how existing research intends to deal with it. In subsequent sections, we present each of our research in chronological order: Burrows, Overfort, AI-RON-E, KUMO and sPoW. Each research section begins with a look at deployable resiliency; (1) we offer details on how each research is resilient against DDoS and (2) we explain why each research is more deployable compared to existing research. Following that, each section is broken into subsections that follow the familiar flow of assumptions, overview, design goals, architecture, evaluation, limitations, future work and conclusions. Finally, we wrap up with our conclusions of the study.

Chapter 2

Background and Related Work

2.1 Background

Denial-of-Service (DoS) is easy to launch, whether in the electronic or physical world but especially so in the electronic one because of the low cost incurred in generating extreme load on a remote system. Furthermore, two Internet designs, exacerbate DDoS:

Affinity Between an IP Address and Physical Location Discounting an untimely physical relocation, a server is unable to change its IP address to something significantly un-guessable due to the small range of IP address block its owner is assigned. This aids the attacker in quickly rediscovering the server in its attempt to shake off an attack.

Lack of Traffic Control Reception The inability of a server to control who it receives traffic from until the traffic arrives at its doorstep is a another factor that greatly aids the attacker in flooding not only the server but its uplink resulting in collateral damage to other servers sharing the same uplink.

As will become evident later in Section 3.5, an intermediary-based architecture where direct server connectivity is forbidden, instead a pool of distributed intermediaries where each can receive traffic on behalf of a server before forwarding it if requested, makes it difficult for attackers to guess and flood all locations where

the server can be reached. The intermediaries also grant a server the ability to dictate traffic reception to keep itself and its uplink from packet floods. These two important intermediary-based architecture features tackles the two Internet design "flaws".

Unfortunately, the problems do not end there. There are multitudes of ways an attacker can launch a DDoS as documented in Mirkovic's taxonomy of attack mechanisms (74). In our study, the DDoS attacks analyzed are limited to brute-force DDoS on dynamic contents, as defined earlier in the scope of study (Section 1.1) and eDDoS. The former covers the entire taxonomy with the exception of "application targeted attack", "semantic attack" which the application targeted attack is a subset of, and "infrastructure (core Internet router) targeted attack". eDDoS is not reflected in the taxonomy as it emerged after the taxonomy creation. Semantic DoS, which exploits a protocol weakness , e.g., Shrew attack (64) on TCP or an implementation vulnerability, e.g., Slow Loris attack (50) on Apache Web Server while important, are tackled orthogonally by initiatives that ensure more secure software development such as U.S. Homeland Security's Built Security In project (30) and research that can automatically analyze protocols for security weakness. Note that in semantic DoS, a distributed DoS (DDoS) is not necessary since few packets will suffice. Our research does not protect against core Internet router DDoS because we do not want to introduce any technology in them since their owners are understandably resistant to any proposed modification or update, which may affect stability and downtime, and that seriously undermines deployability. For ease of comprehension, we adopt terms used in the taxonomy to refer to attacks we address in this study, namely, *filterable DDoS* to refer to attacks whose packets are distinguishable from legitimate ones thus filterable and *non-filterable DDoS*, which is the complete opposite. It is important to note that filterable DDoS does not only refer to packets with properties that makes it conspicuously identifiable as malicious, e.g., malformed packet headers. It also refers to packets that may look legitimate but can somehow be identified as unsolicited. We would also like to point out that non-filterable DDoS is a misnomer, which we inherit from Mirkovic et al., and used in this paper for the sake of usage consistency with the DDoS research community; Mirkovic's non-filterable DDoS term means the attack packets resemble legitimate

ones thus may be difficult to filter but ultimately *still filterable* through smarter heuristics such as statistical anomaly. To quote a simple example, a thousand legitimate-looking attack packets requesting for the same web page originating from a single source IP, is categorized as non-filterable DDoS, even though it is easily detectable thus filterable. Understanding the nuances of the terms filterable and non-filterable DDoS is necessary for comprehending the ensuing study. Strange as it may seem at first thought, an attacker may prefer a filterable DDoS attack over a non-filterable one for 3 reasons: (1) it is easier to generate thus a heavier onslaught is possible, (2) no understanding of the targeted entity nor its communication protocol is necessary to craft attack packets, and (3) it is not possible to launch non-filterable attacks for entities protected by some authentication service that requires the sender to include some secret tokens. With the emergence of cloud-computing, or more specifically utility-based computing, a "pay-as-you-use" model, which expounds low start up cost and scalability for the masses, a new DDoS—economic DDoS (eDDoS) needs to be reflected in the taxonomy. Unlike conventional DDoS, eDDoS targets the financial constraint of an organization, not its physical network/server constraints. eDDoS occurs when zombies send a large amount of undesired traffic that exploits the cloud elasticity to chalk up an exorbitant amount of cost on a cloud adopter's bill leading to service withdrawal or bankruptcy (63).

Our main gripe with various existing research is that although they have devised clever ideas to patch the two Internet design "flaws", they are often not able to defend against all 3 types of DDoS effectively but worst of all, even the more effective ones, due to the lack on economic insight, remain stuck within the hallowed halls of the academia unable to gain widespread deployment.

In this chapter, we start by categorizing existing DDoS mitigation research based-on the kind of circumvention they offer—deterrence, client/server reaction, or economic framework, and we point out the advantages of our research in each of those areas. Following that, we compare their effectiveness against filterable, non-filterable and eDDoS. Finally, we look at their deployability before concluding the chapter by emphasizing how this study stands out in defense against the 3 types of DDoS analyzed.

2.2 Related Work

We classify DDoS mitigation research based on how they tackle the issue—client reaction, server reaction, prevention, deterrent and economic rectification. Each classification is further sub-divided into more specific classes as necessary.

2.2.1 Economic Framework: Inefficiency Rectification

There is a dearth of analysis addressing the lack of DDoS mitigation mechanism deployability. Although most recognize the importance of deployability, it does not receive enough focus; casually rolling out the "incremental deployment" or peer-to-peer (P2P) based model has become second-nature. The former assumes that the adopter is contented to improve on status quo regardless that the deployment is nonetheless ineffective without achieving critical adoption level. The latter misses the crucial point that the success of file-sharing P2P systems is unlikely to be replicable in security, chiefly because the "gain" perceived by a user in obtaining a tangible file is totally different from the gain perceived in being secure. The security acquired is often offset by the loss of convenience or additional incurred effort, thus there exists only a tenuous link between file-sharing and security collaboration incentives.

Our work, Burrows, is a framework intended to clearly delineate the factors that enhance deployability of a DDoS mechanism; it highlights the *need* as well as the *how* to minimize negative externality, re-align economic incentives and reduce disruption to ossified infrastructure. DDoS mitigation mechanisms that adhere to Burrows framework are those most likely to be adopted and deployed, e.g., (24; 38; 48; 62; 63).

2.2.2 Economic Framework: Resource Harness

Edge node intermediary proposals (Section 2.2.6), hide a protected server and grants the server traffic reception control capability. As a result the intermediaries themselves can come under heavy DDoS. Thus, the resiliency of intermediary-based defense is proportional to the quantity of intermediaries. These proposals mostly rely solely on incremental deployment while some may embed code into

unrelated but popular applications, e.g., P2P clients, which are widely distributed in order to turn them into intermediaries (38). The incentive of incremental deployment seems to be insufficient as evidenced in other security-related technology that advocate such approach, e.g., Secure BGP (58).

Our work, KUMO is an economic inefficiency rectification framework, which complements incremental deployment. It extends Burrows' concept of re-aligning economic incentives by facilitating security collaboration, through lowering the participation "cost", e.g., burden of setup and maintenance, and offering "rewards" for participation, e.g., financial gains. In other words, KUMO enables *any* existing and future edge nodes running any protocol to act as intermediaries by contributing their over-provisioned resource, in return for financial rewards, while not requiring any modification at all. This greatly surpasses the incentives of an incremental deployment model, enabling research that adopts KUMO to quickly attain its critical adoption mass for DDoS defense.

2.2.3 Prevention: Network-Layer Filtering

Network-layer filtering refers to dropping packets based on network and transport-layer (as define by TCP/IP) characteristics. It is easily deployable because it can be supported by existing routers at line speed. There are two types of filtering: egress and ingress. In this study, we disambiguate the two terms by always using them in reference to the Internet, i.e., ingress filtering controls packets entering the Internet while egress does the reverse.

Ingress filtering proposals such as RFC2827 (42) and D-WARD (73) focus on preventing malicious packets from entering the Internet and thereby attacking a victim. The former is employed at each stub network router allowing only packets with source IPs of that stub network to enter the Internet. This prevents spoofed DDoS; DDoS attacks whose packets have falsified origin information to avoid accountability. The latter involves monitoring two-way traffic flows for each source IP at ISP edge routers to detect anomalous traffic and stem them near the sources. Ingress filtering requires widespread deployment points and does not benefit the adopter. This lack of economic incentive usually results in under-deployment; as of 19 Sep 2009, according to MIT's Spoofer project, around 20%

of IP addresses are still spoofable, thus D-WARD, which has a deployment model that mirrors ingress filtering, may likely suffer the same fate.

The two most common egress traffic rules are filtering all traffic to the victim and filtering based on "malicious" sources. The former, completes the DDoS but is nonetheless necessary to spare entities sharing heavily impacted common paths from collateral damage. The latter can be manipulated by attackers spoofing a legitimate entity's source address to get it blacklisted and filtered but can be addressed by Hop-count filtering (HCF) (57), which maintains a map correlating the number of hops, a node takes to reach a server. The starting value of Time-To-Live (TTL) field in the IP layer, which is decremented by each router on the way to the destination, is well-known albeit different for different systems, i.e. it is restricted to values of 30, 32, 60, 64, 128, 255 (43), thus based on the TTL at the destination, and the expected TTL from the HCF table, the victim can determine if the packet is spoofed. Unfortunately the deployment of smart egress filtering near the server cannot mitigate collateral damage.

In order to reduce the deployment points required and address collateral damage issues unmitigated by ingress and egress filtering mechanisms respectively, Distributed Packet Filtering (DPF (82)) proposes a filtering mechanism deeper in the Internet. Each router that has a DPF filter installed, builds an Autonomous System (AS) [1] connectivity map that enables it to deduce from the packets source IP address whether that packet could have arrived at its particular network interface. Due to the power law nature of the Internet, this reduces the deployment points required to only 20% of Autonomous System (AS) sites and its proactive packet dropping at the first DPF-installed router encountered overcomes the collateral damage issue to a great extent. Unfortunately, un-spoofed attack packets will still pass through DPF unmolested.

Our work, sPoW also relies on filtering but a white-list one, which is one of the cornerstones of security (97). When the characteristics of malicious elements, e.g., packets, cannot be clearly defined, can be manipulated by attackers or has large number of possibilities, then white-listing reduces the filter size as well as eliminate false positives, i.e., wrongly disallowing good traffic, since what

[1] A number that is assigned to each ISP by the central Internet registry, IANA.

is allowed/good is always clearly specified in the white-list. One of sPoW's features is to use cryptic and hard-to-guess names for its obscure connection request channels, ensuring only legitimate clients that successfully solve puzzles can find them. A white-list is used to permit only traffic addressed to those cryptic channel names.

2.2.4 Prevention: Overlay

SOS (59), WebSOS (32), Mayday (11), use a small set of ever-changing "secret" overlay nodes that are permitted direct communication with the server, which makes them and the server difficult for the attacker to target. Clients gain entry into the overlay after authentication, e.g., Transport Layer Security (TLS) (36) or simple Turing-test verification, e.g., CAPTCHA (117), at attack-resilient entry points and their data is forwarded to the secret nodes using *overlay routing*, which can withstand routing node failures and hides the location of the secret nodes. The use of overlay routing increases latency and the requirement for ISP assistance to deploy filtering rules to restrict direct communication to secret nodes hinders deployment. Moreover, the economic issue of acquiring many entry points that perform authentication, for DDoS defense has also not been addressed. Our work does not enforce any form of authentication, instead sPoW offers an alternative approach—instead of differentiating attacker and authenticated clients, it empowers a client to signal its connection request urgency in order to receive priority service by expending more of its own resource. This has the advantage of delegating resource consumption to the client-side instead of burdening the entry-points where DDoS can be generated by submitting a multitude of deliberate incorrect authentication information.

2.2.5 Deterrence: Traceback

Single-packet (102), ICMP (20) and probabilistic marking (99) traceback can be deployed to locate zombies. However, given that there are thousands (88) or millions (33) of zombies, traceback is interesting but without the ability to automatically shut out DDoS, is ineffective. Our work, Overfort, does automated

traceback to ISP-level, instead of host-level, which although is less granular, requires less resource, but more importantly also comes with a DDoS shut out mechanism—the ISP will be black-listed and it's entire user base will no longer be able to resolve the server name to its actual IP address. The ISP will remain black-listed until it convinces the server through out-of-band communication that it has cleaned up its user base. Thus Overfort offers a very strong incentive for ISPs to keep their user base clean. Moreover, compared to existing traceback mechanisms, all Overfort traceback and black-list enforcement components are within the jurisdiction of a single deployer, making it easy to deploy.

2.2.6 Server Reaction: Traffic Control Through Middle-boxes

Route-tunnel (48), CenterTrack (105), CAT (24), Phalanx (38), I3 (104), and Burrows (60), prevent direct connectivity, thus mitigating direct DDoS attacks to servers and only allowing indirect connectivity through middle-boxes, which function as server-governed traffic control points. Route-tunnel propose to restrict the dissemination of a server IP to only middle-boxes while the middle-boxes in turn advertise reachability for the server thus effectively sinking all traffic prior to forwarding them to the server. Detection of malicious traffic by the server enables pushback to the middle-box, located in rich bandwidth network core. CenterTrack is an overlay network, consisting of IP tunnels or other connections that is used to selectively reroute interesting datagrams directly from edge routers to special tracking routers. The tracking routers, or associated sniffers, can easily determine the ingress edge router by observing from which tunnel the datagrams arrive, thus dropping malicious packets near ingress points becomes possible. CAT suggests deploying middle-boxes at ISP boundaries that will intercept traffic and perform handshake on behalf of the server to provide secret tokens that grants subsequent packets tagged with them to flow through the middle-boxes to the server. The tokens can timeout or be revoked by the server. Route-tunnel and CAT rely on funneling server traffic through specialized middle-boxes in the core and boundary of Internet, which has abundance of bandwidth. Phalanx and I3 are designed to enable bandwidth accumulation from nodes on Internet edges

for defense by installing software components on them for traffic control purposes. The usage of Internet edge nodes facilitates deployment but unfortunately the server now requires filtering assistance for protection against direct attacks. Phalanx, unlike the previous three and I3, does not rely on DDoS detection to initiate reaction; it classifies traffic using a Proof-of-Work (PoW) scheme. A client solves a cryptographic puzzle and submits the solution together with its packet as proof of the resource it expended, to a PoW verifier, which asserts the solution and prioritizes the packet based on the client resource expended. PoW enables non-filterable DDoS defense; it prioritizes packets solely based on the resources expended by their origins instead of determining whether a packet is malicious, which is not possible in non-filterable DDoS. However, the purpose common to all these middle-boxes proposals is to enable server reaction to drop unwanted traffic before they clog the server and its uplinks. Our work, Overfort, KUMO and sPoW are all edge node intermediary based proposal, thus closest in spirit to Phalanx and I3, with similar traffic control intentions. However, each of them adds an additional defense capability—deterrence (see Section 2.2.5), resource accumulation (see Section 2.2.2) and connection signaling urgency respectively (see Section 2.2.10).

2.2.7 Server Reaction: Packet Scrubbing Infrastructure

Due to the difficulty of determining a packet's intent merely by inspecting its characteristics, the issue of economic incentives and avoiding collateral damage, dFence (69) as well as commercial solutions like Prolexic (86) has proposed or built a packet scrubbing infrastructure. Packet scrubbing is a superset of network-layer filtering as it involves more sophisticated heuristics, such as machine-learning, statistical analysis and anomaly detection to identify undesirable packets. A centralized and shared packet scrubbing infrastructure, has economies of scale to purchase loads of resources for DDoS defense deep in the Internet thus minimizing collateral damage while enabling those who wants acquire DDoS protection the opportunity to do so without the high capital investments.

Besides subscribing to scrubbing centers, a single entity can divert traffic by using a sinkhole (47) to construct a Do-It-Yourself (DIY) scrubbing center

built out of a selection of commercial products from Arbor Networks Threat Management System (TMS) (16), Cisco Anomaly Guard (28) or Intelliguard DPS (55)

Either way, scrubbing is based on anomaly/signature-based detection, which may result in legitimate traffic being cleansed due to false positives. The ability of such technology to handle non-filterable DDoS traffic also has not been well-documented.

Our work currently does not utilize anomaly/signature-based malicious packet detection/filtering because of the false positives/negatives in anomaly detection and the over-reliance of sometimes untimely attack signatures updates in signature-based detection. Such mechanisms are also thrown into disarray when faced with non-filterable traffic. With network bandwidth growing, keeping traffic state to perform resource-intensive anomaly detection may be another area of concern. Therefore instead of determining the malice of each packets, sPoW, utilizes a self-verifying Proof-of-Work (PoW) 2.2.10 to prioritize non-filterable traffic based on the resources expended by their origins.

2.2.8 Server Reaction: Pushback

The term pushback refers to a distributed architecture where a victim is in the best position to detect an attack due to its vantage point of receiving all attack traffic, and it requests for widely-deployed upstream nodes, usually routers, to assist in filtering attack packets way before it clogs the uplinks, to avoid collateral damage to systems sharing those uplinks. The difference between middle-box proposals and pushback is a subtle one. The former uses intermediaries more for traffic control with pushback (often only one-level) as a by-product, while the latter utilizes multi-level iterative communication and cooperation between a large number of widespread nodes for pushback purposes.

The earliest form of pushback deployed, known as "black-hole routing" (47), employs Border Gateway Protocol (BGP) (90) routing announcements to bind a victim's IP address to each upstream ingress router's Null interfaces, which drops packets, during a DDoS attack. Although it completes the DDoS on the victim, it is nonetheless necessary to mitigate collateral damage.

"Pushback" [1] Pushback (68), AITF (18), and StopIt (65) enables a victim to request more granular filtering from the pushback infrastructure instead of dropping all traffic to it. In Pushback, "congestion signature" comprising destination IPs, network and transport layer fields, in AITF, "recorded path" comprising source, destination IPs as well as traversed inter-AS gateways, and in StopIt, the source and destination IP pair, offers granularity in describing traffic that should be blocked as specifically as possible. Pushback, AITF and StopIt can also handle spoofed source IP attack traffic well, with Pushback generating congestion signature without relying on source IPs, while AITF and StopIt is cognizant of AS-level topology thus aware of packets whose embedded path information violates the expected paths derived from AS-level map. All of them serve to push DDoS traffic back to attacking sources through iterative communication; victim servers will solicit their pushback-capable upstream nodes to stem the offending flows and the process is iterated until filters are installed at nodes as near as possible to the attacking sources. Economic-wise, there are 2 problems. The upstream router owners are unlikely to filter sources who are their paying subscribers to aid victims whom they have no economic relationships with (24). Moreover, the requirement for co-operation between competitive ISPs to deploy a standardized piece of technology on their uptime-sensitive core equipments is likely to face stiff resistance. Pushback technologies like filtering technologies rely on the ability to install filters/rules that match DDoS traffic in order to drop them thus they may be also littered with false positives/negatives and are weak in the opposition of non-filterable DDoS traffic. Our work Burrows, Overfort, KUMO and sPoW are all intermediary-based architecture, with the intermediaries offering the server a non-iterative single pushback of undesired traffic to the intermediaries. They trade-off the ability to pushback attack traffic all the way back to distributed attack sources in return for better deployability, i.e., single non-interactive pushback only requires deployment at a deployer's own intermediaries instead of distributed third party components such as core Internet routers or third party ISPs customer access routers.

[1] This research is named "pushback", which coincides with the general term pushback used, thus we capitalize specific references to this technology throughout this literature.

2.2.9 Server Reaction: Network Capabilities

Network capability schemes (13) (called "capability", for short), TVA (126) and SIFF (123) enable a server to control traffic reception by communicating the desirability of a traffic flow to capability-enabled nodes (capability nodes) along a traffic path using a secret token, which is not unlike middle-boxes described in Section 2.2.6. However, for even faster eradication of unwanted traffic, the destination-generated token is installed at capability nodes that the approved traffic has to traverse from source to destination, resulting in traffic without tokens or forged ones, being dropped (or given low priority) as soon as it encounters a capability-enabled node has not been given instruction, i.e., approved tokens, to forward such traffic. With a large number of capability nodes, unwanted traffic will get dropped earlier compared to the middle-box approach.

In short, capability offers an elegant way to provide different levels of protection to initial connection requests and established connections, with the difference being, the latter possessing a server-issued approval token, thus given higher queuing, routing and forwarding priority. An established connection is afforded such priority since it represents a connection that has been accepted by the server and also the server has the authority to downgrade the established connection anytime by revoking its tokens from capability nodes. However, the token acquisition process is hard to protect since it is open to all and most proposals rely on approximate fair queuing at routers based on the area of origin of the token requester.

Phalanx, is slightly different—it does not rely on specialized routers along the client-server path to verify tokens. Instead, it has widely distributed middle-box intermediaries that perform the verification. Unsolicited traffic is thus, not stemmed near the sources but only after they have reached the intermediary. To safeguard the token acquisition process, a PoW scheme is implemented. Our work, sPoW, is similar to Phalanx—the ability to stem attack near sources is trade off for deployability. However, the significant difference is that we utilize only a few resource-rich resource intermediaries, i.e., could platforms, which is easy to obtain and deploy. Moreover, sPoW is a self-verifying PoW while Phalanx's PoW is not and this enables sPoW to defend against the emerging threat of eDDoS.

2.2.10 Client Reaction: Proof-of-Work (PoW)

The introduction of capability (Section 2.2.9) provides servers with the ability to prioritize established connection traffic over connection request traffic even under DDoS and drops excessive lower priority traffic near sources. However, the process of obtaining capability is open to everyone, thus it remains a DDoS target. During this early stage in a connection request, i.e., the capability acquirement process, a server has little information on the legitimacy of clients, thus the best way to differentiate clients is by offering a scheme where a client can create a higher priority signal for connection initiation request by expending more of its own resource, which is known as Proof-of-Work (PoW). Servers can then prioritize clients based on their PoW. Speak-up (118) uses bandwidth as PoW by having clients send more requests during DDoS while OverDoSe (101), Portcullis (83) and Phalanx (38) require clients to solve crypto-puzzles as PoW. Speak-up may give rise to congestion elsewhere while the latter PoWs all require specialized puzzle verifiers that can become new DDoS failure points if they have insufficient capacity. Moreover puzzle re-use, i.e., re-using a solved puzzle solution, permits an attacker to submit a connection request without expending any resource, defeats PoW and is difficult to deal with without timely inter-communication between the puzzle verifiers. Our work, sPoW, avoids issues associated with puzzle verifiers by eliminating the need for them by introducing self-verifying PoW. In sPoW, each crypto-puzzle conceals the information of a connection request channel, i.e., a channel that a server is listening for a connection request. Upon resolution, a client recovers the difficult-to-guess channel name, so that it can send a connection request to the server through the channel.

2.2.11 Client Reaction: Exploiting Alternate Paths

RON (12), SOSR (49) and Detour (98) propose to use intermediary nodes on the Internet to deflect traffic around problematic links/hotspots. This approach is useful for circumventing DDoS when there is an abundance of alternate paths, which is exactly what the Internet has, as pointed out by Teixeira et al. (113). RON, requires a full-mesh connectivity between the intermediaries to exchange routing updates, in order to enable the best intermediary selection given a set

of criteria, e.g., latency or throughput. With connectivity in the order of $O(n^2)$, where n is the number of intermediaries available, RON's scalability is limited to below 100. SOSR, is extremely light-weight in resource consumption since it does not keep state but despite that, by randomly selecting 4 nodes out of 39 available, it can mask 66% of link failures. However, the usage of end-hosts as intermediaries result in indirect paths with more hop-counts since packets has to detour to the Internet edges where end-hosts are before being deflected to their final destinations. The small number of end-host intermediaries also reduces the possibility of masking failures, e.g., 5% of failures can only be masked by 25% of intermediaries, making a case for the availability of a larger set of intermediaries and the importance of better intermediary selection algorithm. Our work, AI-RON-E introduces the possibility of using the large number of existing routers as deflectors, thus reducing indirect path hop-count and increasing possibility of failure masking. It does so by exploiting controlled source IP spoofing, which does not require any change to existing routers, and introducing an lightweight oracle-based intermediary selection algorithm to choose intermediaries that can help bypass link failures with high probability while requiring little resource.

2.3 Deployable Resiliency

We want to measure how our research stacks up against existing work so we created the metric "deployable resiliency", which evaluates the effectiveness and deployability of a DDoS defense from 3 perspectives—resiliency, deployability and incentive. Resiliency is a measure of effectiveness of the research against the 3 types of DDoS and the type of defense it offers (see Figure A.3). Deployability is the ease at which a research can be deployed. This is influenced by its required deployment size, availability of technical skills at deployment locations and ease of installing/modifying new/existing systems to incorporate the research (see Figure A.1). Incentive is defined as the willingness of the owner of a system, e.g., router, server, computer, etc., to deploy the research. This is mainly influenced by the level of affliction DDoS has on a system owner as well as required deployment size (see Figure A.2). We plot Figure 2.1, which shows existing research, and compare it with Figure 2.2, which shows our research. From the figures, it is

clear that our research has higher deployable resiliency. Readers with a need or desire to understand the details of how we derive the measurements for resiliency, deployability and incentive for each research, can read the next two subsections, while others can proceed to Section 2.4.

2.3.1 Resiliency (Effectiveness Against DDoS)

Before proceeding to rate each mechanism's resiliency against DDoS, we would like to provide a more robust definition of effective DDoS defense. Filterable DDoS and non-filterable DDoS are well-documented (74) and both can occur to any server that is connected to the Internet or the DDoS mitigation mechanism protecting the server. Thus effective defense against any of those DDoSes mandates that the mechanism has to successfully mitigate that DDoS at both the server as well as at the mechanism itself. eDDoS, on the other hand, can occur only if a pay-as-you-use system, such as a cloud platform, is utilized. A cloud may be utilized under two circumstances to increase scalability: (a) the DDoS mitigation research itself does not have a mechanism to harness sufficient resource for DDoS defense so it needs to be deployed in the cloud, or (b) the server being protected wants to grow dynamically to meet legitimate traffic demands so it needs to be deployed in the cloud. Thus effective eDDoS defense mandates that a DDoS mitigation mechanism itself has to be resistant against eDDoS (intermediary eDDoS) if it relies on the cloud for scalability and it must certainly reduce undesirable packets heading towards servers that are deployed in the cloud (server eDDoS). In our theoretical evaluation of deployable resiliency, to ensure a fair evaluation of different research, we set the baseline for comparison as follows:

- We assume that servers to be protected are always deployed in cloud platforms thus a mitigation mechanism must always defend against server eDDoS.
- It can be safely assumed that a mitigation mechanism is not susceptible to eDDoS (intermediary eDDoS) unless we specifically indicate that it needs to adopt a cloud platform for scalability reasons.

- Defending against filterable DDoS is a pre-requisite for non-filterable DDoS defense, which is in turn a pre-requisite for eDDoS defense. The resultant corollaries are:
  - Effective defense against a certain DDoS also implies effective defense against the pre-requisite attacks. For example, if a mechanism can defend against non-filterable DDoS, it can also defend against filterable DDoS but not vice-versa.
  - The ability to tackle non-filterable DDoS will determine the extent server eDDoS can be mitigated.

The general approach to address the 3 types of DDoS are as follows. For filterable DDoS defense, a mitigation mechanism has to posses more resources than attackers such that it can drop distinguishable attack packets as they arrive while having spare capacity to handle legitimate packets. For non-filterable DDoS and server eDDoS defense, the principle goal is to handle DDoS packets that slips through filterable DDoS defense by somehow reducing them. The existing approach is to employ anomaly-based detection that requires a lot of resources and susceptible to false positives/negatives, to somehow distinguish non-filterable DDoS packets. A more effective approach is to transform the complex problem of determining the malice of a packet into a more manageable one such as determining a packet's priority that can adjusted by its sender through resource expenditure to generate a mathematical proof embedding a priority level that commensurates with the resource expended, thus enabling the proof accompanying each packet to be verified and the embedded priority level accurately determined by the mitigation mechanism for packet prioritization. This is known as Proof-of-Work (PoW). PoW reduces non-filterable DDoS traffic by empowering legitimate clients to increase their traffic priority and compete against DDoS packets for a server's resources. For intermediary eDDoS defense, we have to ensure that the non-filterable DDoS packet analysis mechanism described above is carried out by non-billable resources otherwise attackers can flood the mechanism with arbitrary packets to drive up billable processing charges.

Table 2.1 and 2.2 summarizes the ability of related work and our research (in *italics*) to defend against the 3 types of DDoS analyzed in this study—filterable, non-filterable and eDDoS (intermediary and server).

Research that exploits alternate Internet paths does not offer a server defense against filterable, non-filterable and eDDoS. However, they can route around congestion created by filterable or non-filterable DDoS as long as there are alternate paths available; a condition that can be created by making intermediaries and servers multi-homed. The intermediaries are not cloud-based thus eDDoS is not applicable here. However, it is defenseless against eDDoS at the server since the primary intention of eDDoS is not congestion, which alternate paths can counter, but rather the financial resource depletion of the target.

Middle-box proposals can defend against filterable DDoS either by pushing back undesirable traffic to a location rich in bandwidth and far away from the bottleneck, e.g., an ISP's core network, or pushing back traffic to a huge number of distributed locations, whose aggregate resources is expansive, e.g., back to traffic origins. Phalanx falls into the first category while Route-Tunnel, CenterTrack and CAT falls into the second. Route-Tunnel, CenterTrack and CAT cannot defend against non-filterable DDoS resulting in attack packets slipping through the defense and causing eDDoS at the server. Phalanx, on the other hand, utilizes PoW thus it can defend against non-filterable DDoS. However, if Phalanx intermediaries that verify all PoW proofs are built from cloud platforms, then even incorrect proofs, incur cloud resource usage making Phalanx itself susceptible to eDDoS. Burrows and KUMO are frameworks that advocate intermediary usage to facilitate traffic pushback thus effective only against filterable DDoS, with KUMO being extremely effective since it facilitates harnessing lots of resource for DDoS defense. If an entity can utilize KUMO intermediary resources without paying then eDDoS is not an issue, however, to encourage growth of intermediaries a pay-as-you-use compensation scheme for intermediaries is required thus exposing a KUMO adopter to eDDoS. Filtering proposals also rely on dropping malicious packets that have particular characteristics identifiable through statistical anomaly, e.g., D-WARD, or signature matching, such as bad source IPs, e.g., egress filtering or violation of expected characteristics, such as TTL-source

IP association with the packet arrival interface, e.g., HCF and DPF. False positives/negatives in D-WARD makes its non-filterable DDoS and server eDDoS defense imperfect, while DPF and HCF is only effective against filterable DDoS. Worse still, localized mechanisms, e.g., egress filtering and HCF, are not capable of pushing back traffic resulting in incomplete defense against filterable DDoS, i.e., collateral damage is incurred on other servers sharing same uplinks.

Packet scrubbing infrastructures rely on the economies of scale of the outsourcing model, i.e., many customers pooling resources to construct a bigger infrastructure than possible if each were to build individual ones, thus have huge resource for defense against filterable DDoS. Pushback proposals, like some middle-box proposals, also push back traffic to distributed locations, whose aggregate resource, is sufficient to stem filterable DDoS traffic preventing bottleneck downstream. Unfortunately scrubbing infrastructures and Pushback mechanisms, utilize detection mechanism, like filtering, thus can be only somewhat successful with non-filterable DDoS due to possible false positives/negatives thus they are only partially effective against eDDoS at servers. Black-holing, which is a local deployment of pushback, by a single entity, to the entry points of the entity's boundary is very crude; it will drop all traffic, regardless good or bad, for a particular destination at those entry points. The end result is that any DDoS on the server always succeed but eDDoS will not happen because the server is taken offline at the first sign of an attack since the primary focus of black-holing is to avoid other servers in the entity network from suffering collateral damage.

Traceback technology is only a partial solution to DDoS; it only enables traceback of attack traffic to each source to facilitate a zombie black-list compilation. It relies on other orthogonal research, such as, attack traffic detection to determine which traffic packets to traceback and an enforcement implementation to actually drop traffic from clients in black-list. Thus by itself, it cannot defend against any DDoS attack. Overfort is different. It is both a traceback black-list enforcement mechanism. An entity can harness sufficient Overfort intermediaries for defense against filterable DDoS either through cooperation or even by itself. Since those intermediaries can come from a non utility computing based pool, e.g., an ISP's access routers, it is not exposed to eDDoS at the intermediaries. However, its reliance on orthogonal malicious traffic detection algorithm exposes it to false

negatives/positives, resulting in only partial defense against non-filterable DDoS and eDDoS at server.

Network capability enables a server to inform widespread intermediaries about approved traffic, enabling non-approved traffic to be dropped as soon as it encounters an intermediary. However, prior to determining approved traffic a server has to accept all initial connection request traffic approval consideration. The large number of widespread intermediaries can provide sufficient resources against filterable DDoS. However, a server has to rely on false positive/negative-ridden anomaly detection algorithm to reject non-filterable DDoS traffic that pretends to be initial connection requests. This reduces its non-filterable DDoS and server eDDoS defense to partial.

Overlay proposals are accompanied by authentication mechanisms that minimize automated, unauthorized traffic from discovering the location of a protected server. As long as the authentication mechanism has sufficient resource, e.g., employing a cloud platform, to sustain the largest of filterable DDoS, then a reverse-Turing authentication mechanism will eliminate non-filterable DDoS and server eDDoS, which are automated. Note that non-automated attacks are rare since they cannot generate sufficient attack traffic to cripple neither the mitigation mechanism nor protected server. However, the usage of a cloud platform as intermediaries exposes it to intermediary eDDoS. Also note that the applications supported by such overlay mechanisms are restricted to those that have graphical user interface and are interactive; both are requirements of a reverse-Turing authentication mechanism.

PoW is described above under Phalanx and the same argument applies here, i.e., it can defend against non-filterable DDoS. The only exception is that Portcullis and OverDoSe intermediaries are non utility-based, while Speak-Up has no intermediaries, thus they are not susceptible to intermediary eDDoS.

sPoW employs a cloud platform to create sufficient intermediaries for defense against filterable DDoS. However, unlike other DDoS mitigation mechanisms that utilize cloud resources for traffic differentiation or prioritization, sPoW, is a carefully devised PoW scheme (a self-verifying PoW) designed to delegate such resource consuming task to the cloud platform's built-in mechanisms, e.g., its firewall, thus circumventing the huge cost of associated with handling non-filterable

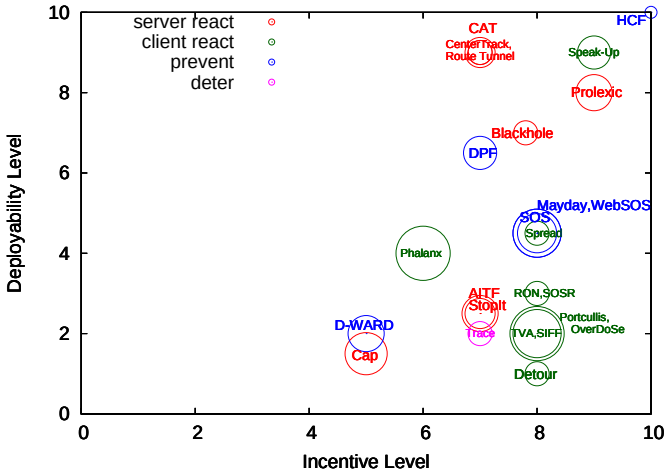


Figure 2.1: Existing deployability, incentive level and effectiveness of significant DDoS mitigation research in from 2000-2009.

DDoS packets (intermediary eDDoS). The power of PoW enables defense against non-filterable and server DDoS.

2.3.2 Deployability

Next we analyze the deployability of existing research, which is made up of incentive and ease of deployment factors. Incentive measures the alignment between the party affected by DDoS and the party that is empowered to deploy the mitigation mechanism. A match between the party afflicted and the party empowered will result in high incentive and thus enhances adoption. Deployment incentive for a research thus depends on where the components needs to be deployed and it is also affected by the quantity of components required for the research to be effective.

The ease of deployment is a measure of the effort involved in deployment and the availability of expertise to deploy. This can be objectively measured by considering the location of deployment and the number of deployment points, with the location itself an indication of the effort required as well as expertise availability. Ease of deployment at *server-side* is high followed by *Internet edge*, *client-side* and *network* in decreasing order of ease, while expertise availability is high at server-side, Internet edge and network but low at clients. The number of

Mechanism Type	Mechanism	Filterable DDoS	Non-filterable DDoS	Intermediary eDDoS	Server eDDoS
Alternate path	RON, SOSR, Detour, *AI-RON-E*	Yes (if alternate path exists)	Yes (if alternate path exists)	Not applicable (non utility-based)	No
Middle box	Route-Tunnel, CAT, Center-Track	Yes	No	Not applicable (non utility-based)	No
Middle box with PoW	Phalanx	Yes (use utility-based intermediary)	Yes (traffic prioritization through competition)	No	Yes
Economic framework	*Burrows*	Yes	Not applicable (dependent on chosen technology)	No	Not applicable (dependent on chosen technology)
Resource harness framework	*KUMO*	Yes	No	No	No
Distributed filter	D-WARD	Yes	Partial (false negatives)	Not applicable (non utility-based)	Partial (false negatives)
Distributed filter	DPF and ingress filter	Partial (only against spoofed DDoS)	No	Not applicable (non utility-based)	Partial (only against spoofed eDDoS)
Localized filter	HCF, egress filter	Partial (neighbor collateral damage and HCF is only effective against spoofed DDoS)	No	No	Partial (HCF is only effective against spoofed eDDoS while egress filter can be bypassed by continuously altering attack packet characteristics)
Packet scrub	Prolexic	Yes	Partial (false negatives)	No (may be charged high monthly premium or utility-based)	Partial (false negatives)

Table 2.1: Resiliency Against DDoS (Part 1)

Mechanism Type	Mechanism	Filterable DDoS	Non-filterable DDoS	Intermediary eDDoS	Server eDDoS
Local push-back	Black-hole	Yes	No	Not applicable	Yes (all traffic dropped)
Pushback	Pushback, AITF, StopIt	Yes	Partial (false negatives)	Not applicable (non utility-based)	Partial (false negatives)
Traceback	Single-packet, ICMP, Probabilistic	No (only traceback while black-list dependent on other mechanism)	Not applicable (only traceback while black-list dependent on other mechanism)	Not applicable (non utility-based)	Not applicable (only traceback while black-list dependent on other mechanism)
Traceback with auto black-list	*Overfort*	Yes	Partial (false negatives)	Deployer billing model-dependent	Possible (with complete black-list)
Network capability	Network Capability, TVA, SIFF	Yes	Partial (established connections are protection but initial connection requests are not)	Not applicable (non entity-based)	Partial (established connection protection)
Overlay	SOS, WebSOS, Mayday	Yes	Yes (use reverse Turing test)	No	Yes
PoW	Speak-Up, Portcullis, OverDoSe	Yes	Yes (traffic prioritization through competition)	Not applicable (non utility-based)	Yes
Self-verifying PoW	*sPoW*	Yes	Yes (traffic prioritization through competition)	Yes	Yes

Table 2.2: Resiliency Against DDoS (Part 2)

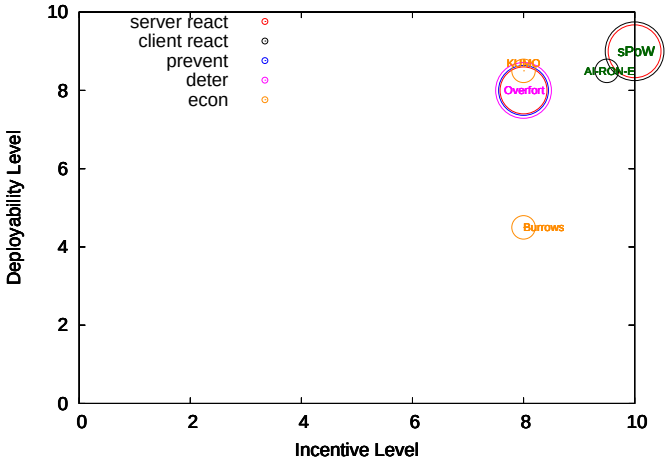


Figure 2.2: Our DDoS research in terms of deployability, incentive level and effectiveness.

deployment point further reduces ease of deployment factor since they increase the effort and expertise required.

2.3.3 DDoS Mitigation Comparison Chart

Based on the 3 criteria of incentive, deployability and effectiveness. We plot Figure 2.1 to represent all existing research. Notice that there are only a few mechanisms that are in the ideal deployable resiliency area—top-right corner. Moreover, those few mechanisms there do not offer effective defense against all the 3 critical DDoS. We plot a similar chart (Figure 2.2) to represent our research. Burrows, which is just a framework, has average deployable resiliency, however the rest of our research have high deployable resiliency.

2.4 The X-Factors

This section elucidates the major factors that distinguish our work from existing research. The combination of KUMO and sPoW is the only work, to our knowledge, that can address all the 3 types of DDoS affecting dynamic contents. Our work is also the first research to address the rising concern of eDDoS by empowering users to virtually alter the billing mechanism of cloud platforms to one that

reduces their exposure to eDDoS, without any assistance from cloud providers but yet not violating any contractual terms and conditions. This empowerment is necessary to transfer some burden of eDDoS from the cloud adopters to the providers whom otherwise will just turn a blind eye to eDDoS due to the lack of incentive to act against it. In addition, as visible from Figure 2.2, most of our work has relatively high deployable resiliency with sPoW offering unprecedented deployability by using cloud computing's enormous resource for DDoS defense. This is possible because our mechanisms take into consideration economic factors that may hinder deployment or reduce deployment incentives. Finally, Overfort, is unique because it offers two features: (1) defense against DDoS even though the mechanism may have less physical resource capacity than the attacker and (2) it performs traceback and enforce black-listing that punishes an entire cluster of users if there is a zombie among them by empowering a server to refuse connectivity to that "bad" cluster. This exerts economic pressure on clusters, e.g., ISPs who are always accountable to their customers, to proactively keep their user base clean in order to avoid disconnectivity. In other words, Overfort delegates security responsibility to many cluster administrators to ease the job of cleaning up and securing the Internet. The added bonuses are Overfort can be deployed unilaterally by a single ISP and the zombie black-list created can be sold to or shared with other ISPs and the Internet community.

Chapter 3

Burrows

This section is an abridged version of my paper co-authored with Nicolas Christin, Tina Wong and Akihiro Nakao entitled "Power to the People: Securing the Internet One-Edge at a Time" published in ACM Large Scale Attack Defense (LSAD) Workshop held in conjunction with ACM SIGCOMM Conference in Kyoto, Japan, Aug 2007 (acceptance rate: 45%).

NOTE: This paper is largely based on my Master's research in Carnegie Mellon Cylab Japan. An excerpt is included in this thesis as the paper was mainly written, published and presented when I was a Ph.D. student at University of Tokyo and more importantly, the concepts have been further refined.

3.1 Deployable Resiliency

3.1.1 Resiliency

Defense Category	Filterable DDoS	Non-filterable DDoS	eDDoS (intermediary)	eDDoS (server)
Economic Framework	Yes	Not applicable[a]	No	Not applicable[a]

[a]Burrows is technology-agnostic. Its defense against non-filterable DDoS is dependent on the technology plugged into the framework.

Table 3.1: Resiliency of Burrows against DDoS

Burrows attempts to rectify the economic inefficiency issues, namely negative externality, mis-aligned economic incentives and ossified infrastructure in order to encourage DDoS mitigation mechanism adoption. It tackles the issue from the following angle:

Economic Framework: Inefficiency Rectification By creating a general framework that future DDoS mitigation research can be based upon, we ensure that future research implicitly takes into consideration these economic factors to enhance adoptability. The Internet is a common public infrastructure, thus, each user is likely to exploit it to her advantage with little regard to its well-being, a phenomenon known as "tragedy of the commons" (92). From a security standpoint, users are disinterested in deploying secure mechanisms because they cost money and inconvenience, thus opting to do nothing, resulting in the burden of an insecure Internet being shared by everyone. The idea put forth by Burrows to create a semi-closed Internet model, where a server is granted the ability to control traffic reception through a group of intermediaries that all traffic destined to the server is forced to pass through. Doing so limits the dependency of a server's DDoS protection solely to the intermediaries' defense capability and capacity thus minimizing negative externality, i.e., dependency on disinterested parties for defense cooperation. Burrows also enable security-minded parties to pool edge node intermediaries to augment defense capacity instead of modifying ISPs ossified infrastructure thus successfully re-aligning economic incentives, i.e., DDoS afflicted server owners are no longer helpless against DDoS, they can deploy Burrows among themselves, and avoid overwhelming infrastructure changes.

With sufficient intermediaries, filterable DDoS can be thwarted. However, defense against non-filterable DDoS and eDDoS is reliant on the technology deployed at the intermediaries to detect and reject them.

3.1.2 Deployability

DDoS research that is based on Burrows framework have a higher probability of widespread deployment because they rectify economic inefficiencies that ex-

ists in the Internet. Even when ISPs, who are in the best position to mitigate DDoS, shun DDoS mitigation mechanism deployment due to poor returns, parties afflicted by DDoS are empowered to defend themselves using Burrows' cooperatively constructed intermediary-based traffic reception control platform. Burrows also reduces the effect non-adopting parties can impose on adopters; instead of relying on *all* parties (dis-interested or otherwise) adopting the same mechanism to effectively deal with DDoS traffic that can originate from anywhere and use many distinct paths to reach a server, in Burrows, all traffic destined for the server is funneled through the traffic control platform obviating the need to equip points along all Internet paths with DDoS mitigation technology therefore increasing deployability. The possibility of funneling traffic for traffic control to edge nodes, which can be easily introduced, upgraded or modified, without requiring changes to the Internet core infrastructure, greatly simplifies DDoS mitigation mechanisms deployment.

3.2 Assumptions

Resilient Naming Service Burrows relies heavily on a naming service to tell clients where to find intermediaries that can forward traffic to a destination server. It can utilize existing Domain Name Service (DNS) for this purpose, i.e., the mapping for a server hostname can be continuously updated to point to available intermediaries that can forward traffic to the server. This implies that Burrows's resiliency is no worse than the existing Internet, which also relies heavily on DNS. For improved name service resiliency, research such as CoDoNS (89) and ConfiDNS (85) can be adopted.

3.3 Overview

Despite a plethora of research in the area, none of the DDoS mitigation mechanisms proposed so far has been widely deployed. We argue that these deployment difficulties are primarily due to economic inefficiency, rather than technical shortcomings. We identify economic phenomena, *negative externality*—the benefit derived from adopting a technology depends on the action of others, *economic*

incentive misalignment—the party who suffers from an economic loss is different from the party who is in the best position to prevent that loss, and *ossified Internet infrastructure*—core Internet infrastructure that is in a strategic position for DDoS mitigation deployment are also the most difficult and resistant to change due to its complexity and the repercussions of its downtime, as the main stumbling blocks of adoption. Our main contribution is to build a DDoS mitigation framework, *Burrows*, with an *economic incentive realignment* property.

At the core of Burrows is a wide-area virtual private network, or secure overlay, carved out of the existing Internet. Entry points into the Burrows overlay are controlled by intermediaries. Together they grant a protected server in Burrows the ability to control traffic reception from any Internet source by employing technologies that can detect and drop unsolicited/attack packets. With the server's security dependent only on the resiliency of intermediaries instead of the entire Internet community, Burrows minimizes negative externality. To rectify the aforementioned economic incentive misalignment, the power to realize Burrows is put into the hands of Internet users, i.e., each server owner can deploy her own intermediaries within her domain of control. Server owners can join forces by sharing intermediaries to create a single well-provisioned overlay with more control points, instead of smaller individual ones with few control points. In addition, Burrows supports incremental deployment; even with as few as two participants, Burrows provides a secure overlay that contains twice as many control points as when there is a single participant. Burrows is also designed such that these intermediaries can be deployed at Internet edges, where modifications and upgrades are easy to implement thus no modification to the ossified core Internet infrastructure is necessary.

3.4 Design Goals

In this section, we discuss the objectives a DDoS mitigation architecture with economic incentive realignment mechanism should strive to fulfill. To that effect, we first identify the *security properties* required to defend against the threat and describe the *economic properties* necessary to stimulate adoption of the design. While some of the properties may have been addressed in past research, their

deployability has not necessarily been fully discussed. Thus, we elect to revisit them and consider issues in their deployment.

3.4.1 Security Properties

We define security properties, in particular *what* needs to be protected and *type* of protection required.

Server Control of Traffic Reception In DDoS attack, zombies simply flood a server with more requests than it can handle. Hence, a server that can dictate how much traffic it should receive will not be susceptible to DDoS. Even though technologies that pushback undesired traffic can be used for such a purpose, they require changes to the existing Internet infrastructure, which is extremely challenging due to the ossification of the Internet (116).

Uplink Protection It may be possible to protect a server from DDoS attacks by constraining direct connectivity to the server, using for instance a Virtual Private Network (VPN). However, even if the server is not directly vulnerable, DDoS can still occur if the Internet uplink of the server is flooded. This situation can notably occur if an attack is targeted at the router that connects the server to its Internet uplink. Thus, it is crucial that the uplink router itself be shielded from attacks, an issue generally ignored by related proposals. Also note that the uplink protection must be implemented in routers, and its wide deployment is likewise hindered.

Traffic Protection The third key component to protect against DDoS attacks is the traffic itself. Non-filterable DDoS traffic masquerades as "legitimate" traffic and as such cannot be easily filtered, e.g. a few million valid HTTP requests targeted at a web server, may prevent valid traffic from reaching its destination, and must be filtered. CAPTCHA (117), "fight fire with fire" (118) and Proof-of-Work (PoW) (19; 34; 121) are some existing technologies that address this issue.

Protection Independent of Offered Service Some DDoS-protection architectures are based on content replication (e.g., as in Akamai). Replicating

content is (thus far) mostly limited to HTTP and streaming traffic. Conversely, we strive for an architecture that can be resilient to DDoS attacks regardless of the contents being served.

3.4.2 Economic Properties

In addition to the security properties outlined above, a DDoS-resilient architecture must adhere to certain economic properties to be deployable.

Minimizing Negative Externality In designing an architecture for adoption on the Internet, we have to ensure that the effectiveness of the architecture is minimally affected by people who choose not to adopt the architecture. Failing to do so will result in the architecture being overlooked by even its most ardent advocates.

Realigning Economic Incentives ISPs may not experience direct economic losses due to DDoS, while end-users with web presence do. Since end-users are more likely to have incentives to deploy DDoS mitigation mechanisms, we need an architecture that empowers end-users with the ability to secure their servers without requiring aid from ISPs.

Small "Critical Mass" Effectiveness Most DDoS mitigation mechanisms require substantial deployment size, or "critical mass," to be effective. Building critical mass requires time and effort. Instead, we strive to design for incremental deployment that ensures that even with as few as two participants, both participants extract tangible benefits from their involvement.

Backwards Compatibility We need existing servers to be able to utilize our DDoS mitigation mechanism with little or no modification. We also need the mechanism to be completely transparent to existing clients as well as mandating little changes to ossified Internet infrastructure. Indeed, unless the architecture has an extremely compelling property, e.g. total DDoS elimination, a backwards compatible (evolutionary) solution is more likely to gain acceptance than one that is not (revolutionary) (100).

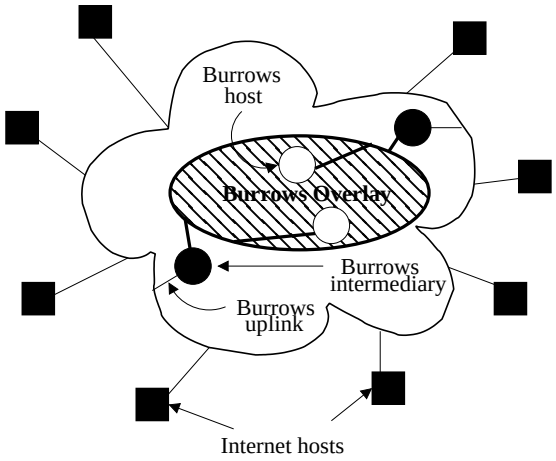


Figure 3.1: Protected servers in Burrows only connect with the rest of the Internet through Burrows intermediaries.

3.5 Architecture

In this section, we describe the architecture of Burrows and discuss how we fulfill the objectives described in Section 3.4.

3.5.1 Burrows Architecture

Throughout the design of Burrows, we ensure that our architecture requires minimal changes to the existing servers, clients and Internet infrastructure. Designing Burrows as an overlay network makes it possible to embed various DDoS mitigation mechanisms (see following subsections) to achieve the properties identified in Section 3.4 without modification to the current Internet.

Server and Uplink Protection As depicted in Figure 3.1, at the core of Burrows is an overlay network, Burrows overlay. The overlay is formed by a collection of Burrows intermediaries distributed all over the Internet. Burrows servers, i.e., servers that need DDoS protection, are connected only to the Burrows overlay. Burrows is technology-agnostic so it does not dictate the technology used for creating this overlay and connecting the Burrows server to the overlay. However, usage of Border Gateway Protocol Multi-Protocol Label Switching Virtual Private Network (BGP-MPLS-VPN) (93)

for server-overlay connectivity is advocated since it can protect both the server and its uplink against direct DDoS attacks. This is possible since a server utilizing BGP-MPLS-VPN does not expose its IP address to the Internet but instead uses the upstream router's IP address for connectivity purposes. Even though this merely shifts DDoS attacks to an upstream component, that ISP-owned component is usually better provisioned thus offers stronger defense against DDoS and its upstream location can relieve an uplink from a deluge of unwanted traffic.

Since the overlay protects Burrows servers from direct reachability, connectivity from an Internet host (i.e., a system outside Burrows) to a Burrows server is achieved through Burrows intermediaries. Indeed, each Burrows intermediary added, provides an additional entry point into the overlay leading to the server. This intermediary-based architecture addresses the two Internet design "flaws" described in Section 2.1 by (1) enabling a server to be reachable from a wider range of IP addresses, i.e., the distributed intermediaries' IP addresses, thus making it difficult for attackers to find and clog all access points and (2) enabling a server to control traffic reception at intermediaries in order to prevent attack traffic from overwhelming its uplink and itself.

Traffic Protection Since it is difficult or error-prone to determine a packet's malice by just examining its contents or detect anomaly in statistics of packets received, proposals to prioritize packets based on a client-controllable packet "property" that is "visible" to a server (thus prioritizable) and can reflect its connection request urgency has been put forth (Section 2.2.10). Such schemes require resources to receive all traffic prior to analysis or prioritization. Using multiple intermediaries provides a possible way to construct a front-end platform powerful enough to receive all traffic. By incorporating the best of such non-filterable DDoS protection with this platform, it is possible for Burrows to defend against non-filterable DDoS.

Small Critical Mass Effectiveness Even with a minimum of two intermediaries, a Burrows server acquires the benefit of having twice the capacity

(two Burrows uplinks) to resist DDoS than it had before (its sole Internet uplink). We use VPN to prevent direct server connectivity to Burrows servers thereby forcing all Internet traffic to go through intermediaries and then use multiple intermediaries to achieve the effect of uplink replication and small critical mass effectiveness.

Incentive Realignment and Minimal Negative Externality We adopt a *self-scaling* model (as seen in peer-to-peer applications) to empower end-users to build Burrows thereby realigning economic incentives. Each participant is entitled to harbor her server in Burrows only if she contributes at least one Burrows intermediary. Through this model, it is likely that Burrows comprises only security-conscious participants. We accordingly decrease the probability that insecure systems exist within Burrows, further reducing negative externality and the effect of tragedy of commons. PlanetLab (84) is an extremely successful example of an infrastructure build incrementally using a self-scaling model with a similar incentive alignment mechanism.

Edge Node Technology Deployment By empowering participants to build Burrows platform using edge nodes, which are easy to introduce, upgrade or modify instead of mandating changes to ossified core Internet infrastructure, we increase the deployability.

3.5.2 Miscellaneous Components

On top of the components required for DDoS defense, we introduce 3 peripheral components here that are necessary to ensure the functionality and integrity of Burrows.

Name Resolution Service An Internet host that wants to initiate a connection to a Burrows server needs to resolve the server's hostname to the IP address of an intermediary that is willing to handle the server's traffic. A Burrows server can continuously authenticate to a dynamic DNS system to update its hostname record to map to a fleeting set of intermediaries that will handle its traffic at a given period of time.

Accounting Database Any peer-to-peer overlay network is always faced with the problem of free-riding participants, i.e., participants who benefit from the overlay network without contributing to it. In the case of Burrows, a given end-user's server can consume more traffic than what her intermediary routes. To alleviate free-riding, we propose a traffic accounting mechanism consisting of a database that keeps track of how much traffic each Burrows intermediary has routed and for which Burrows server the traffic routed was for. We can adopt an existing wide-area resilient database such as a public Distributed Hash Table (DHT) (91), to store traffic accounting information. In order to prevent the traffic accounting system from being tainted with bogus information, we assign public/private key pairs to all the Burrow servers for accounting information signing and verification.

Whenever an intermediary routes traffic for a server, it keeps track of the traffic. At different preset intervals, each intermediary will request each server that it routed traffic for to digitally sign on accounting information indicating its resource consumption on that intermediary. A server has the right not to sign if the information in deemed inaccurate, similarly, an intermediary can refuse to forward traffic for the server should the server refuse to sign. Each intermediary then updates the accounting database with the digitally signed accounting information.

At preset intervals, before an intermediary routes packets for a server, it checks the traffic accounting system to see if that server's corresponding intermediary has contributed enough to the overlay. If not, the intermediary will notify the owner of that server about its free-riding violation and discard the packets. The owner can react to the violation notice by increasing the uplink and capacity of her intermediary. If there is no free-riding violation, the intermediary routes for the destination server.

Hidden Action Monitor By routing traffic between Internet hosts and Burrows servers, malicious intermediaries can perform subtle attacks known as hidden actions, i.e., they can eavesdrop, modify and discard packets, which transit through them without the communicating parties being aware of such misdeeds.

The simplest way to prevent hidden actions is to use end-to-end encryption such as Secure Sockets Layer (SSL) to encrypt traffic between client and server. Without end-to-end encryption, for hidden attack detection purpose, we need to distinguish between two traffic directions: a malicious intermediary can drop (or tamper with) packets originating from Burrows to the Internet and vice versa.

For traffic exiting Burrows, the Burrows server may occasionally send "predictable reply test packets," i.e., packets that elicit predictable responses. For example, while communicating with an email server at *abc.com*, the test packet can be crafted to request that the email recipient is *kokoro@hotmail.com*. If the reply to the test packet differs from the expected "no forwarding permitted"[1] error message, one can conclude that the intermediary has dropped/modified the packet or its corresponding reply.

To examine hidden actions when packets are originating from the Internet and head towards the Burrows servers, we use the same mechanism but a predictable reply test packet has to be sent out by some designated trusted nodes periodically.

3.6 Limitations

Heavy Resource Consumption Forwarding The need for intermediaries to keep accounting information, interact with servers for digital signing and update/refer accounting data from a distributed accounting database, can consume substantial intermediary resource and increase traffic forwarding delay. Whether an attacker can introduce traffic load that exploits this resource consumption to deplete intermediary resources resulting in a different type of DDoS or increase forwarding delay significantly is a subject for future work.

Hidden Action Monitor Deficiency The hidden action monitor relies on sending requests with predictable replies. Malicious intermediaries can introduce

[1]This error message is expected since typical email servers will not receive emails for a domain which they are not configured for.

heuristics or signatures to detect request with predictable replies, much like how anti-viruses detect malicious traffic, so that they avoid tainting those test packets. This will result in an arms-race between hidden action monitor and malicious intermediaries. Although imperfect, this defense can help reduce the possibility of malicious intermediaries. An alternative approach is to use only intermediaries of trusted entities and/or limit the number of intermediaries to a few but resource-rich ones. Although using only trusted intermediaries sounds unreasonable, we point the reader to the Internet, where malicious routers can be difficult to combat but their existence is very low due to the fact that routers only interconnect with other routers that their owners have trust or business relationships with.

3.7 Future Work

We would like to use Burrows as a framework for designing future DDoS research and encourage other DDoS researchers to do likewise. Two ideas that we intend to further explore are: (1) using an ISP's access routers, which are distributed to its Internet subscribers as intermediaries, and (Section 4) (2) using resource-rich cloud computing platforms as intermediaries (Section 7). In both cases, the intermediaries belong to trusted entities, thus, the need to implement complex hidden action monitor and traffic accounting, is relaxed. In the former, the access router intermediaries can be trusted to keep accurate accounting, while in the latter, the cloud computing accounting is kept by the cloud provider itself, which can to be trusted.

3.8 Conclusion

We postulate that the main impediment to large-scale deployment of existing DDoS mitigation infrastructure lies in the misalignment of economic incentives among Internet parties. Our main contribution in this paper is to introduce a DDoS mitigation framework that intrinsically incorporate economic incentive realignment mechanisms. We identify minimizing negative externality, empowerment of end-users with the ability to protect themselves and ease of deployment

as the three key incentive realignments necessary. We realize these realignments with a secure overlay, Burrows, which employs easy-to-modify edge node intermediaries to act as entry points into the overlay leading to protected servers. Forcing all traffic through these intermediaries limits a protected server's security exposure to those intermediaries, which we can easily implement strong security mechanisms, such as attack traffic filtering and traffic control reception. We also adopt a peer-to-peer model to empower end-users to build Burrows without requiring aid from infrastructure providers, by creating a platform for pooling intermediaries to increase defense against DDoS.

Chapter 4

Overfort

This section is largely adapted from my paper co-authored with Akihiro Nakao entitled "Overfort: Combating DDoS with Peer-to-Peer Puzzle" published in Secure Systems and Network (SSN) Workshop, held in conjunction with IEEE International Parallel and Distributed Processing Systems (IPDPS) Conference in Miami, USA, May 2008.

4.1 Deployable Resiliency

4.1.1 Resiliency

Defense Category	Filterable DDoS	Non-filterable DDoS	eDDoS (intermediary)	eDDoS (server)
Server React, Deter/Prevent	Yes	Not applicable[a]/Fair-share[b]	Model-dependent[c]	Possible[d]

[a]Its defense against non-filterable DDoS is dependent upon technology that a deployer is free to chose.

[b]Overfort can allocate server resources "fairly" to all clusters when non-filterable DDoS detection cannot be handled.

[c]A common pool of intermediaries may be shared at no cost, provided at a fixed cost or utility-based pricing

[d]If zombies can be completely segregated and black-listed, which is dependent on the deployer-chosen non-filterable DDoS detection technology, before aggregated intermediary resource is depleted then server eDDoS can be prevented.

Table 4.1: Resiliency of Overfort against DDoS

Overfort offers protection against DDoS from two angles: deterrence/prevention and server reaction.

Deter/Prevent Upon attack detection, a server uses Overfort to traceback the attack to a client "cluster", i.e., clients that are associated together because they utilize the same local Domain Name Service (LDNS) server to translate the server's domain name to one of the many IP addresses that leads to the server. An example of a cluster is the entire user base of an ISP since they share the ISP's LDNS. The entire cluster can then be refused connectivity to the server as punishment for an infringement by one of them. The LDNS/cluster administrator is forced to quickly locate and remove zombies for connectivity to that server to resume. She also has to continually keep the cluster zombie-free to avoid future disconnectivity that may antagonize her clients.

Server Reaction: Traceback and Black-list of Bad Clusters To defend against filterable DDoS with less physical resources than attackers, Overfort enables a server to quickly traceback and black-list all bad clusters by using cheap virtual resources, before server resources are depleted. Overfort can enforce black-listing, without relying on other parties, by not divulging an IP that can lead to the server, to clusters in the black-list. This preserves the remaining server resources for legitimate client use.

As shown in Table 4.1, Overfort can withstand filterable DDoS if various parties cooperate by pooling resources to deploy Overfort components (intermediaries) such that the aggregated intermediary resource exceeds the attacker capacity. Filterable DDoS, like non-filterable DDoS, and server eDDoS can also be successfully defended against, if all attack traffic including non-filterable DDoS traffic can be detected so that all bad clusters can be traceback and black-listed before the aggregated intermediary resource is depleted. Since non-filterable DDoS detection has some inaccuracy, defense against non-filterable DDoS and server eDDoS is possible but rated as partial in the case where false positives occur. In the worst case that no segregation is possible, Overfort gracefully degenerates into a fair-sharing system; all clusters are allocated equal server resources with clients

in clusters with zombies receiving the short-end of the stick. Intermediary eDDoS occurs if Overfort intermediaries bill servers based on resource used to handle the server-destined traffic. A cooperative model where intermediaries are pooled for common usage or a fixed monthly subscription is charged, intermediary eDDoS will not happen.

4.1.2 Deployability

Overfort contributes to DDoS mitigation mechanism deployability in two distinct ways: (1) empowers afflicted end-users to cooperate incrementally or/and (2) empowers ISPs with unilateral deployment.

End-user Empowerment Each end-user can install an Overfort component, known as Overfort Gateway (OFG), on her system to turn it into an Overfort intermediary; each intermediary can pool their resources for stronger DDoS defense. To hasten critical mass resource accumulation, i.e., the aggregate resource required for effective DDoS defense, Overfort is designed to rely on virtual resources instead of physical ones, i.e., each end-user's access link (physical resource) can be arbitrarily divided up into multiple virtual links (virtual resources). The number of virtual links required to completely traceback and black-list all bad clusters depend on each DDoS attack's properties, e.g., number of bad clusters, total number of clusters in the Internet, etc. Given that with an OFG, each end-user can split a physical uplink into arbitrary number of virtual links, Overfort makes it easier to achieve the required critical mass. However, there are limitations and negative impacts of splitting a physical link into too many virtual links (see Section 4.8).

ISP Empowerment The OFG is a piece of non-complex code that can easily be incorporated into an ISP customer's access router. Thus, a *single* ISP with a user base of a few thousands can unilaterally deploy Overfort to overcome even the most powerful of DDoS (Section 4.6.4).

4.2 Assumptions

Malicious Traffic Detection Overfort needs to distinguish between legitimate and attack traffic to instantiate traceback and punishment. Research in attack traffic detection abounds, thus Overfort is designed to plug-in any type of detection mechanism thereby inheriting the strengths and weaknesses of that mechanism.

Resilient Naming Service Overfort relies heavily on a naming service to enable clients to find Overfort intermediaries that can lead to their destination servers. For this purpose, Overfort can utilize existing DNS by just making some changes to the authoritative server—a protected server continuously update the dynamic DNS to map its hostname to available intermediaries. This implies that Overfort's resiliency is no worse than the existing Internet, which also relies heavily on DNS. For improved resiliency, existing research such as CoDoNS (89) or ConfiDNS (85) can be adopted.

Line-speed Rate-Limiting/Throttling OFG creates virtual links by splitting the physical link network bandwidth of the hardware it is installed on. It does so by assigning an IP (or any form of ID that can be used for routing packets in the future Internet) to each virtual link and realizes the virtualized link concept by limiting the bandwidth consumption for each IP; each OFG relies on the hardware on to drop packets when the bandwidth usage quota of a virtual link IP has been exceeded, at line speed. Such a rate-limiting requirement is reasonable since it is done based solely on the destination IP field of a packet whose location within a packet is fixed, which facilitates processing, and state-keeping of bandwidth usage is limited to per-virtual link IP of an OFG. If an OFG is installed on a generic hardware, such as a PC or server, instead of a router, as in the case of end-users, it may be better for the OFG to communicate with the access router to dynamically control rate-limiting at the access router for superior filtering performance (See Figure 4.1).

Hidden Server Location Enforcement Overfort's functionality relies on the enforcement of indirect connectivity, which prevents direct attacks. Thus it

is critical that each OFG that relays traffic to the protected hidden server does not divulge the server's location. In a unilateral or cooperative deployment between ISPs, all OFGs are installed in access routers that belong to a single or multiple ISPs. Since ISPs are naturally trusted, we assume that the hidden location of a server can be well preserved. When OFGs are installed by a motley crew of end-users on their own systems, whose trustworthiness is questionable, preservation of a server's hidden location mandates additional effort of setting up a VPN tunnel from the intermediaries to the protected server's upstream ISP router, e.g., BGP-MPLS-VPN (93). In such setup, intermediaries are only aware of the protected server's upstream ISP router IP instead of the server's, which is completely private. With the upstream router better provisioned for DDoS attacks compared to the server itself, its IP leak has a less severe impact.

4.3 Overview

There are many existing attack traceback mechanisms (20; 99; 102). However, they have not successfully contend with two issues: (1) deployability and (2) enforcement of punishment. The former arises because thus far, traceback mechanisms require installation of traceback technology-specific components on a large number of deployment points, which is both time and resource-consuming in terms of the actual deployment effort and the negotiating process among all deployers to agree on a technology, respective responsibilities and benefits. Such cooperative mechanisms also incur a lot of message exchanges during traceback that may flood the Internet. The latter is often an un-answered question. "What happens when you have identified large numbers of distributed zombies?". Shutting them down requires gargantuan effort, which is also untimely and dependent on other parties. Overfort is an attempt to design a unilaterally deployable traceback mechanism that is also capable of automated unilateral zombie shutdown enforcement. The key idea is to create multiple obscure channels that can lead to a protected server. In the simplest case, where each client/zombie can be assigned to a unique channel, an attack can easily be traced back to its origins by the mere observation of the channel under attack. After a zombie has been

black-listed, its subsequent request for a channel will be rejected leaving it unable to find and attack the server. The obscurity of channels reduces the likelihood that zombies can guess their whereabouts. Our research also attempts to deal with the realistic scenario where a one-to-one client/zombie-to-channel mapping is not possible through "clustering".

The key design is to construct an overlay using ingress gateways (intermediaries) with multiple access channels (virtual links) with different bandwidth that lead to the server within it, thus projecting an illusion of multiple servers *sans* the intensive resource requirements. This overlay is constructed on-demand, i.e., only enabled under attack, using over-provisioned resources such as IP blocks and CPU cycles/bandwidth at routers and servers that are leased from or donated by the participants in Overfort.

By provisioning these multiple access channels with different bandwidth in the overlay, we transform the act of perpetrating a DDoS attack from a brute force automated attack targeted at a single IP into a combinatorial puzzle of finding all the access channels and determining sufficient DDoS traffic required to clog every channel. Furthermore, with a segregation mechanism that clusters clients based on LDNSes they utilize to perform DNS resolution, and iteratively assigns the clusters to different access channels over time for attack observation, we can identify and segregate clusters containing zombies to a limited number of access channels enabling the larger pool of remaining channels to serve productive traffic. Both of these features are the keys to enabling Overfort to defend against DDoS with significantly less resources.

4.4 Design Goals

A DDoS attack usually directs all the DDoS traffic in an automated brute force fashion to a server's single IP (or an upstream router) from multiple zombies distributed across the Internet, in an attempt to clog a bottle-neck. Another important observation is that although an attacker may spoof the source IPs of attack packets, she needs legitimate packets to consult the DNS for an IP of the server before launching an attack, especially when the DNS mapping changes frequently.

Based on these observations, we define the following five strategies to raise the barrier against DDoS attacks.

Proliferating Access Channels On-Demand This strategy proliferates the number of IPs a server is accessible from. Increasing the number of access channels obviously evades a single point of failure as well as installs multiple targets that an attacker needs to flood, making DDoS coordination hard. Moreover, we enable these channels to be switched "on" only under a DDoS attack; otherwise, clients connect directly to the server. An attacker will find it hard to harvest channels in preparation for a planned DDoS attack.

Segregating and Penalizing Malicious Access Attackers have to continuously query LDNSes to find access channels that lead to a server. We exploit this as a mean to track which access channels we have handed out to whom so that if an attack occurs at a certain access channel, we can identify the perpetrator and prevent her from obtaining new access channels by rejecting her subsequent name resolution queries.

Obfuscating Channel Location and Bandwidth A DDoS attack will not succeed without identifying where all the access channels are located and how much traffic is required to clog them. This strategy aims to make it difficult to find all the IPs of the access channels, i.e., the discovery of one access channel IP will not make it any easier to find the remaining ones. In addition, we make it pain-staking to discover the bandwidth that each access channel can handle. To do so, we craft responses to requests that are received through the access channels, so that they have inconclusive significance, e.g., its non-responsiveness or unresponsiveness at higher traffic volume does not necessarily infer that its bandwidth is completely exhausted.

Providing Fair Service Under Non-Filterable DDoS Attack/Flash Crowd Most DDoS mitigation mechanism may prosper only when DDoS traffic is anomalous and thus detectable. Non-filterable DDoS traffic, however, mimics legitimate traffic and resembles flash crowd; a surge in legitimate client requests due to popularity of content. When faced with such attacks or

flash crowd, we gracefully degrade network services provided to clients in a fair manner.

Empowering Afflicted Parties To Defend Against DDoS We elect to achieve the above goals using a self-scaling scheme such that one can utilize unused resources, e.g., IP blocks and CPU cycles/bandwidth at routers and servers belonging to oneself or contributed by multiple parties, as Overfort's access channels. Such a unilateral or co-operative coordinated peer-to-peer system empowers DDoS afflicted parties to construct Overfort themselves without relying on dis-interested third parties (60).

4.5 Architecture

This section describes the architecture of Overfort that fulfills our five design goals specified in Section 4.4.

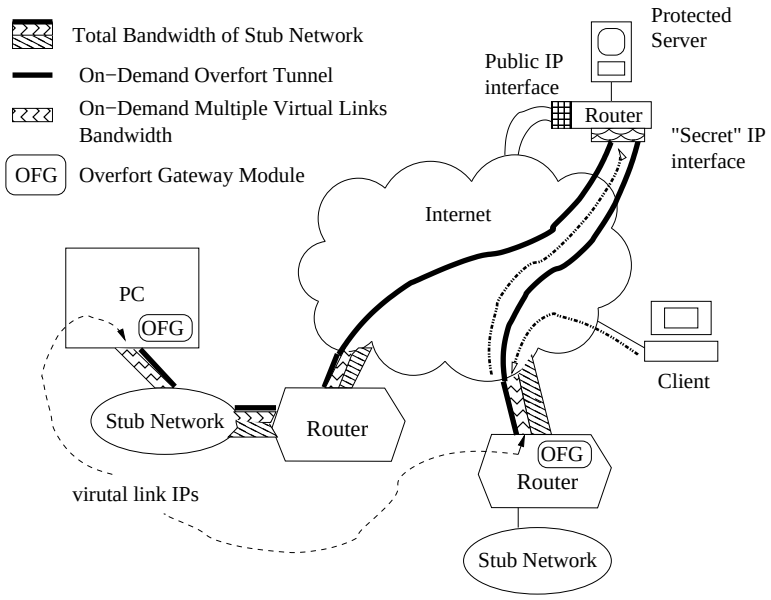


Figure 4.1: Overfort architectural overview

On-Demand DDoS Mitigation Overlay To introduce numerous access channels, we utilize OFGs scattered over the Internet in different IP subnets,

e.g., in stub networks as shown in Figure 4.1. Each OFG acts as an intermediary that listens on a set of IPs. Each IP is termed as a *virtual link IP* and they can accept traffic destined for the server being protected. Name resolution for the server will resolve into any of virtual link IPs at the intermediaries. Therefore, clients can send packets destined for the server to any virtual link IP, which, in turn, will forward these packets through its Overfort tunnel to the server and vice versa. Indeed, the intermediaries also serve as strategic points for implementing various DDoS detection and mitigation technology such as Pushback (68) and low-rate DDoS attack defense mechanisms (107).

For Overfort to be enabled on demand, the protected server is multi-homed to at least 2 IPs; the public IP that connects the server to the Internet directly and the "secret" IP that is only known to intermediaries. Under normal traffic conditions, name resolution for the server will return the public IP, so clients may connect directly to the server and do not suffer performance penalty with using the intermediaries. Under a DDoS attack, the usage of the public IP will cease and each LDNS that requests for name resolution will receive different virtual link IP instead as reply. The implications of the secret IP being divulged or discovered can be alleviated if it is made a private IP and intermediaries use virtual private tunnels that require both ends, i.e., the intermediary and the protected server, to be pre-configured.

Traffic Segregation with Dynamic DNS Clients can be naturally clustered based on the LDNSes that they utilize. As name resolution queries arrive at our authoritative DNS server, we check if the server has requested for Overfort to be switched on. With Overfort enabled, name resolution queries from different LDNSes will be assigned to different virtual link IPs in a round-robin fashion. After the last virtual link IP available has been assigned, we will re-use them starting from the first one.

When an attack occurs on a certain virtual link IP, we mark all the LDNSes assigned to that virtual link IP as "suspicious" and they will be dis-associated from that virtual link IP. When a subsequent name resolution query arrives

from a dis-associated LDNS, it will be reassigned to a new virtual link IP. On the other hand, if there is no DDoS attack on a virtual link IP, the associated LDNSes will remain assigned to the same virtual link IP. Note that these immediate association and dis-association can be done through setting the TTL of name resolution relatively short. By this suspicious marking scheme, we can accurately identify a bad LDNS—the LDNS is serving zombies—by locating LDNSes with more markings than the average.

Virtual Link IP and Bandwidth Obfuscation To make guessing of virtual IPs non-trivial, we scatter intermediaries all over the Internet so that they are likely to be in disjoint IP subnets. However, in practice, we utilize multiple IPs within the same IP subnet for virtual links at each OFG, which may make it easy for an attacker to discover another virtual link IP once she finds a single virtual link IP, e.g., by periodically sending protected server-bound probes to the entire IP subnet range.

To deal with this, we restrict virtual link IPs from responding to clients that are not associated with the LDNSes assigned to them by introducing *client-LDNS association* mechanism. Each virtual link tracks which client IPs are within k Autonomous Systems (AS) away from IPs of LDNS assigned to it, i.e., it assumes that each LDNS only serve clients within k AS distance (k should be a low integer, e.g., 1) This mechanism also serves to obfuscate virtual link bandwidth probing; a non-response could be interpreted as the virtual link is not in service, has no bandwidth available or the client is not entitled to utilize the virtual link because its LDNS is not assigned to the virtual link. To defeat the mechanism, an attacker can make all her zombies send probes to all the virtual links. However, intermediaries that are assigned with LDNSes, which the attacker does not have zombies k hops from cannot be subjected to such brute force probing method.

Fair Service Under Indistinguishable DDoS Attack Since LDNS segregation described above enables us to keep track of which LDNSes are associated with which virtual link IPs, it is possible to provide a fair network

service during indistinguishable DDoS attack; we assign all the available virtual links fairly among all the LDNSes. Clients that share the same LDNSes with zombies, however, will be significantly marginalized. Section 4.7 discusses why even under such circumstances, this is the best way to provide fair service.

Peer-to-Peer and Virtualization Mechanism We use the term "peer" to loosely encompass the entire gamut from end-user peers to ISP peers – in order of increasing level of trust among peers. We expect end-user peers to install OFG software on their servers while ISP peers can opt to modify stub routers to incorporate intermediary functionality (see Figure 4.1). The OFG virtualizes each peer's physical link into multiple virtual ones.

The incentive to co-operate can vary; end-user peers benefit from the synergy of shared resources in mitigating DDoS, while with a distributed packet accounting service to keep track of traffic routed by one's intermediary for a protected server, not unlike the one described in Burrows (60), ISP peers will be motivated to provide gateways for a premium. In scenarios where end-user peers exist, the level of trust of these peers can be elevated using mechanisms to deter malicious hidden actions as in Burrows (60).

4.6 Evaluation

4.6.1 Overfort Segregation Algorithm Approach

Among the architectural components introduced in Section 4.5, the most crucial one is the bad LDNS detection and segregation mechanism using multiple virtual link IPs. If every LDNS can be mapped to a single virtual link IP then identification of bad LDNSes in an attack becomes straight-forward; through LDNS-to-virtual link IP assignment records, we can trace an attack at an IP back to a LDNS. Unfortunately, there are already approximately 800,000 LDNSes (88) in Internet in 2006. Ideally, we can acquire a similar number of systems/access routers to install OFGs so that the straight-forward bad LDNS detection and segregation algorithm suffices; given that there are 1.7 billion users (56), even if we

make the unreasonable assumption that among every 2000 users, only one possesses an access router, we can acquire 800,000 routers for OFG installation, which is sufficient make one-to-one LDNS-virtual link IP mapping possible. However, to minimize negative externality and increase deployability we devise a segregation algorithm that greatly reduces the number of virtual link IPs required so that even a single entity's resource, e.g., a single ISPs access routers, is sufficient to meet the virtual link IP requirement. We trade-off algorithm simplicity for a less resource-intensive algorithm by mapping more than one LDNS to a single virtual link IP and during an attack, re-assign each of the LDNSes to different virtual link IPs over time so that through observations of which virtual link IPs are under attack over time, we can deduce the bad LDNSes. This section presents the feasibility study focused on evaluating the number of virtual link IPs required to complete LDNS segregation and the accuracy of the segregation mechanism under different operating conditions as well as under a variety of DDoS attacks.

4.6.2 Overfort Simulation Model

This section introduces the Overfort simulation model that consists of Overfort configuration parameters, parameters adjustable to represent different operating conditions, and outputs that measure the effectiveness of LDNS segregation.

4.6.2.1 Overfort Configuration Parameters

Deviation from average markings: Y As described in Section 4.5, an attack on a virtual link IP will result in all the LDNSes associated with that IP being marked as suspicious. Eventually, LDNSes with Y more markings than the average are classified as bad. Y must be carefully chosen; a high Y will result in less false positives (good LDNSes wrongly identified as bad), but utilizes more virtual link IPs with segregation taking longer and vice versa.

Number of virtual link IPs in used at any time: N_{use} Although Overfort may have many virtual link IPs, at any given time, only N_{use} are in use,

i.e., LDNSes are assigned to only these N_{use} virtual link IPs in a round-robin fashion. Since the number of LDNSes usually exceeds N_{use}, multiple LDNSes could be *multiplexed* onto a single virtual link IP. When a virtual link IP is assigned to a bad LDNS, the attacker will relentlessly flood it. Overfort must allocate another virtual link IP from its unused set to replace the flooded one to ensure that there is always N_{use} virtual link IPs available for assignments.

4.6.2.2 Overfort Operating Condition Parameters

Total number of LDNSes requesting name resolution: N_{LDNS} This represents the total number of LDNSes that will request name resolution for a server under attack. It comprises the number of LDNSes requesting name resolution during the same period of time under normal conditions plus the number of bad LDNSes that perform name resolution during attack.

Ratio of good vs. bad LDNSes: R This is the ratio of good LDNSes versus bad LDNSes out of N_{LDNS} LDNSes that request name resolution for the server throughout the duration of attack.

Expected bad LDNS requests arrival rate: arr_{bad} arrivals per second This represents the uniformly distributed expected arrival rate of name resolution requests for the server at a bad LDNS—the LDNS that is serving zombies for name resolution. A straight-forward DDoS attack will most likely aim to find all the virtual link IPs as fast as they can through continuous query of DNS, i.e., arr_{bad} will have a high value. This is, however, upper-bounded by the DNS TTL value that we assign to the virtual link IP returned as the name resolution reply. Attempts to request name resolution at a faster rate will just result in the same virtual link IP cached at the LDNS to be returned.

Expected good LDNS requests arrival rate: arr_{good} arrivals per second This represents the Poisson distributed expected arrival rate of name resolution request for the server at each good LDNS (27). We believe that

arr_{good} varies for each good LDNS depending on the number of the clients that LDNS is serving, making it hard to assign a single representative value.

Attack detection time: t_d seconds t_d sec after a bad LDNS has been assigned to a virtual link IP, the model determines whether an attack has occurred [1] and if so, all LDNSes associated with the virtual link IP will be marked as suspicious.

A lower value of t_d is desirable, since it implies speedy attack detection, which results in less LDNSes being multiplexed to a single virtual link IP, thus reducing the likelihood of a good LDNS sharing the same virtual link IP as a bad LDNS and being implicated when the virtual link IP is attacked. A good DDoS detection technology will have a low t_d.

4.6.2.3 Output: Overfort Effectiveness Measurement

Accuracy percentage: *acc* % This shows the percentage of bad LDNSes correctly identified as bad. A value of 100% indicates that Overfort identifies all bad LDNS correctly without any false positives.

Additional number of virtual link IPs required: N_{add} As soon as a virtual link IP is assigned to a bad LDNS is identified, the virtual link will be incessantly attacked, so a new virtual link IP must be introduced to maintain N_{use} number of IPs in service. N_{add} represents the number of additional virtual link IPs introduced in total before Overfort completes LDNS segregation.

4.6.3 Overfort Simulation Algorithm

All the LDNSes start in the "unassigned" pool, i.e., they are not assigned to any virtual link IP since no client has requested any of the LDNSes to perform name resolution. We label each LDNS as good or bad so that the ratio of good over bad LDNSes corresponds to the configured parameter R. At a prefixed interval, e.g., 1 sec, for each unassigned LDNS, the expected arrival rate (either one of

[1] As in Section 4.6.4, in *on-attack* mode, an attack will always occur while in *flip-flop-attack* mode, an attack will occur with probability of 0.5.

arr_{good} or arr_{bad} depending on whether the LDNS is good or bad) determines if a name resolution is requested. The requesting LDNS will be assigned to the next available virtual link IP from the pool of N_{use} IPs in a round-robin fashion. We keep track of each LDNS's arrival time and its virtual link IP assignment. For each LDNS, after t_d has elapsed since its assignment, we simulate if the LDNS attacks its virtual link IP. If so, we increment the suspicious marks on all the LDNSes that has been assigned to that virtual link IP. LDNSes with Y more marks than average will be classified as bad and they retain their virtual link IP assignments indefinitely, which effectively prevents bad LDNSes from capturing more virtual link IPs with subsequent name resolutions, while other LDNSes are dis-associated from the virtual link IP and dumped back into the unassigned pool. They will be re-assigned to other virtual link IPs at their next name resolution query arrival. We have to allocate a new virtual link IP to replace the one that has been discovered and increment N_{add} by one. The entire process is repeated at the prefixed interval until all bad LDNSes has been completely segregated or we stop the simulation manually.

4.6.4 Overfort Simulation Results

Through some experimentations, we settled on the value of N_{use}=30, Y=4 and t_d=80 [1]. Due to the lack of information about R and arr_{bad}, we explore how Overfort performs under different values. All the experiments use N_{LDNS}=100 and arr_{good}=0.02. We show that for other values of N_{LDNS}, the results of the experiments can be extrapolated linearly in Figure 4.4. For all experiments except otherwise stated, we assume that attackers employ the *on-attack* mode, i.e., a virtual link IP returned to an bad LDNS as a response to a zombie's DNS request is attacked immediately. We simulate two other attack modes, *on-off attack* and *random attack*, that attackers will employ in their attempts to defeat Overfort's defense mechanism and show their impact in Figure 4.3. We run each experiment 5 times and take the averaged results.

[1] In the future, the values of N_{use} and Y can be chosen adaptively based on operating conditions. We use a conservative t_d=80 since a lower value t_d will intuitively give better results.

Figure 4.2 (right-axis) shows how Overfort performs with various R and arr_{bad}. Even under severe DDoS conditions, i.e., R=0.5, N_{add} stayed below 510. With the initial virtual link IPs, N_{use}=30, we require at most 510+30=540 virtual link IPs to completely perform LDNS segregation. In other words, after re-assigning LDNSes to 540 different virtual link IPs, we have identified the bad LDNSes and stop providing available virtual link IPs to them. Assuming that each intermediary has an average of 5 virtual link IPs, we only need a total of 540/5=108 intermediaries. Under less averse condition of R=0.97, less than 50 total virtual link IPs, i.e., less than 10 intermediaries are required. In other words, N_{add}, virtual link IP requirements for complete segregation, decreases monotonically with R. Also note that under this operating condition with N_{LDNS}=100, when R is lower than 0.85, using a one-to-one mapping, which requires 100 virtual links, is more efficient than the Overfort segregation algorithm, which will require more than 100 virtual links.

In Figure 4.2 (left-axis), we show the accuracy of the LDNS segregation algorithm. When arr_{bad}=0.1, 0.05 and 0.02, the accuracy is sustained at 100%. When arr_{bad}=0.002, however, the accuracy drops drastically to 80%. This scenario occurs if attackers slow down their name resolution intentionally. With more good LDNS arrivals than bad ones, there will be few attacks, which causes the average suspicious marking to be extremely low. As a result, good LDNSes that are wrongly marked a few times will have Y more markings than the average resulting in false positives. Fortunately, this strategy cannot result in a DDoS flood because zombies only make 1 name resolution request every 1/0.002=500 seconds, which does not prevent legitimate clients from finding an available channel to communicate with server. An attacker can exploit this to generate false positives—some innocent LDNSes will be black-listed but the attacker cannot target a specific LDNS to be black-listed since she does not know where which virtual link IP a targeted LDNS is assigned to. One possible fix for this is to adapt Y dynamically based on overall name resolution inter-arrival rates.

We also evaluated how Overfort will perform when an attacker tries various forms of DDoS attacks to subvert it. Three types of attacks are simulated, (1) *on-attack*, zombies will always attack once they obtain a virtual link IP (2) *flip-flop-attack*, zombies will attack a virtual link IP with 0.5 probability, i.e., they

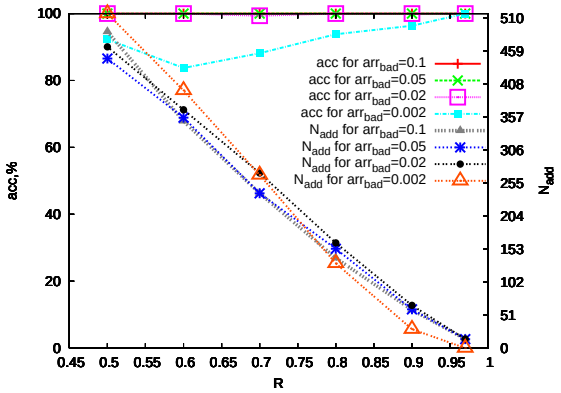


Figure 4.2: *acc* (left y-axis) and N_{add} (right y-axis) under varying R and arr_{bad} with Y=4, N_{use}=30, N_{LDNS}=100, arr_{good}=0.02 and t_d=80

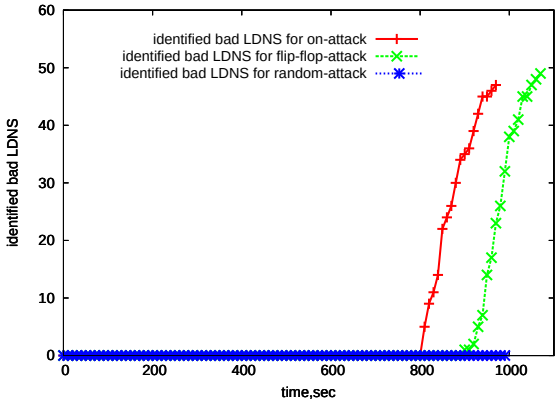


Figure 4.3: Number of LDNSes that are identified as bad under different types of attack with Y=4, N_{use}=30, N_{LDNS}=100, R=0.5, arr_{bad}=0.05 and t_d=80

attempt to evade segregation by not always attacking, and (3) *random-attack*, zombies randomly attack any virtual link IP, i.e., they try to frame good LDNSes. We simulate the scenario with N_{LDNS}=100 and R=0.5, i.e., there are 50 bad LDNSes. As shown in Figure 4.3, Overfort correctly identified all the 50 bad LDNSes regardless of *on-attack* or *flip-flop-attack*. In case of *flip-flop-attack*, however, it takes longer because the lower frequency of attacks leads to bad LDNSes taking longer time to exceed the Y threshold. Note that for random attacks, no bad LDNS was identified since, by randomly attacking virtual link IPs, it is likely that all the LDNSes are uniformly marked, resulting in none of them having Y more markings than the average. In short, bad LDNSes cannot escape segregation nor can zombies successfully frame good LDNSes through different attack modes.

Throughout our simulations, we choose N_{LDNS}=100 since a larger value will result in simulations taking very much longer. Figure 4.4 shows linearity holds for N_{add} with respect to N_{LDNS} even for various R. Therefore, even though we simulate for N_{LDNS}=100, we can extrapolate our results linearly to estimate the N_{add} required for larger values of N_{LDNS}.

In the current Internet, it has been estimated that there are approximately 800,000 LDNSes with approximately 10% bad ones. Extrapolating the results in

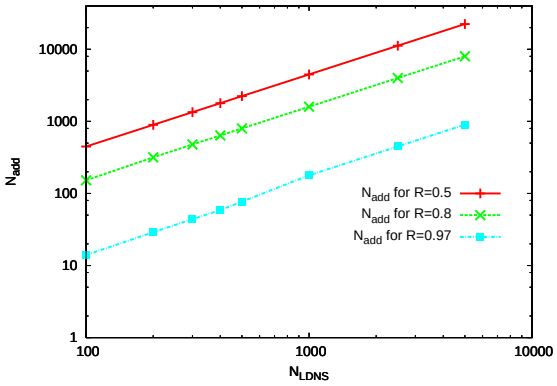


Figure 4.4: N_{add} required for LDNS segregation as N_{LDNS} varies under 3 different R values. It is faster for the simulation to run with smaller N_{LDNS} values thus we have more points where N_{LDNS} <1000 and the use of log-scale.

Figure 4.4, we require approximately 120,000 virtual links to completely segregate all bad LDNSes. This amounts to only 2 class B IP addresses.

In summary, note how the Overfort algorithm significantly reduces the number of virtual links required for segregation, compared to a straight forward one-to-one mapping, i.e., from 800,000 to 120,000 virtual links. Assuming that each physical OFG has 5 virtual links, then only 120,000/5 = 24,000 physical OFGs are required. A small/medium size ISP thus can unilaterally install the OFG code into each customer's access router and successfully perform this segregation and share/sell this blacklist with/to the Internet community.

4.7 Discussion

4.7.1 LDNS Granularity Segregation

It is true that LDNS segregation may lead to false positives that aggrieve innocent clients while bad LDNSes that are accurately identified incurs collateral damage to innocent clients sharing the same LDNSes as zombies, but we postulate that penalizing bad LDNSes is the best mechanism due to the following rationale:

Infeasibility and Scalability Overfort can only segregate clients at LDNS granularity because it cannot associate each client with a virtual link IP since

clients interact with Overfort indirectly through their LDNSes. Even if we could, it will imply that even more virtual link IPs are required to track and identify individual zombies. This clearly does not scale.

Security Responsibility Delegation By penalizing bad LDNSes, we delegate the task of shutting down zombies to the vast army of LDNS administrators on hand. Indeed, such security responsibility delegation will encourage LDNS administrators to be more prudent about the integrity of their client base.

4.7.2 Multi-server Protection

Thus far, we described how Overfort can protect a single server; all virtual links tunnel back to a single protected server thus forming a virtualized network. This concept can be extended to protect multiple servers; different sets of virtual links will tunnel back to different protected servers thus each protected server and their corresponding virtual links will form different virtualized networks (116). At first glance, increasing the number of protected servers, will increase the number of virtual resources required linearly giving rise to scalability concerns. However, since the bad LDNS blacklist of a protected server can be shared with other protected servers, they can choose to snub a bad LDNS in the shared blacklist without assigning it any virtual link. In other words, the number of virtual links required is not proportional to the number of protected servers protected by Overfort, but rather is fixed by the number of zombies in the Internet.

4.8 Limitations

Client-LDNS Association Verification In Overfort, virtual link IPs are numerous with the active ones unknown to a client unless it performs a name resolution through its LDNS. An attacker may not want to perform name resolution to avoid traceback so it has to perform DDoS blindly; it floods the entire virtual link IP space. Overfort's large virtual link IP space thins out a brute-force attack that arrives at a protected server's doorstep, However, because virtual link IP addresses may be adjacent due to the block assignment

method adopted by the Internet central authority IANA, virtual link IPs may be guessable, which facilitate attackers discovering supposedly obscure virtual links. To alleviate this issue, we try to restrict virtual link usage only to clients that has performed name resolution through enforcement of client-LDNS mapping. The enforcement attempts to determine a client has performed name resolution by checking if the virtual link it attempts to use is the one assigned to its "associated" LDNS. A client is deemed associated to an LDNS if their distance in AS hops is within k, where k is configurable by the Overfort deployer. Enforcing the mapping requires 3 pieces of information: client AS, LDNS AS and AS distance between the client-LDNS. Existing IP-to-AS resolution service, e.g., CYMRU (112), provides client and LDNS AS, while, an Internet AS map build from RouteViews (94) provides the AS distance. Building the AS map is resource-consuming only at initial stage. With AS relationship relatively static incremental AS map modification is fairly straightforward. However, the IP-to-AS resolution consumes resources as it needs to be performed for each packet that arrives at the virtual link, thus opening up an avenue for resource depletion attack. Caching IP-to-AS mapping will alleviate this to some extent, but this remains an valid area of concern.

Cheap Virtual Resources Overfort relies on the quantity of virtual links to perform complete segregation. Moreover, some form of ID, e.g., an IP address, is required to be associated with each new virtual link. Currently, the Internet uses IPv4 for routing, and IPv4 is running out (10) thus the usage of IPv4 as ID is not considered "cheap". However, this problem can be mitigated once the large address space IPv6 is adopted.

Restricted Virtual Link Bandwidth With each additional virtual link creation, the bandwidth of every existing virtual link at the OFG where the virtual link is created, is equally reduced to free up resources for the new virtual link. Having more virtual links aid complete segregation and reduces the requirement for large number of physical OFGs, e.g., access routers, but at the expense of reduced virtual link bandwidth. This can be alleviated by balancing number of physical OFGs and virtual links, e.g., participation

from a few more ISPs increases the physical OFG locations thus relaxing the virtual link quantity requirement.

Intermittent Connection During Segregation Time As shown in Figure 4.3, when there are 100 LDNSes where 50% are bad, it takes at least 1000 seconds or 17 minutes to completely perform segregation. However, in the meantime, a legitimate client using a virtual link which is also assigned to zombies will experience intermittent connectivity because it will be reassigned a new virtual link when an attack occurs at its current virtual link. However, once segregation is completed, the occurrence of choppy connectivity due to constant virtual link reassignment is minimized.

Reliant on Non-filterable DDoS Attack Detection Effectiveness In order to perform traceback and black-list, Overfort needs to detect an attack. Thus ultimately, Overfort's bad LDNS segregation success relies on the effectiveness of the non-filterable DDoS detection technology employed. Overfort is designed to be detection technology-agnostic; it can be upgraded by plugging in the latest non-filterable DDoS attack detection technology.

4.9 Future Work

In future, we would like to implement Overfort over PlanetLab (84) to further study the relationships between the configurable parameters. The ideal implementation will be able to dynamically adapt Y and N_{use} to increase its effectiveness against more powerful foes with efficient use of the virtual link IPs. Another important issue is to balance the remote IP-to-AS queries and the quantity of information cached locally on each OFG, which is necessary to determine the legitimacy of a packet sent to a virtual link in order to avoid attacker from probing for active links.

4.10 Conclusion

We propose Overfort, an on-demand overlay architecture that makes perpetrating DDoS difficult by transforming it from an automated attack on a single IP address

into a combinatorial puzzle of discovering all access channels and the bandwidth allocated to each. By reassigning each LDNS to a different virtual link each time its current virtual link is attacked, eventually, through careful reassignment-and-observation process, even though multiple LDNSes are assigned to a single virtual link, we can determine the bad LDNS(es) among the set of LDNSes. This enables Overfort to tame DDoS if we have sufficient virtual links to segregate all LDNSes utilized by zombies. Failing which, at least Overfort increases the work factor of DDoS perpetration through additional name resolutions incurred, by a factor equivalent to the number of virtual links at its disposal. Compared to existing traceback mechanisms, Overfort excels by offering a unilateral, instead of co-operative, scheme, to simplify traceback as well as a unilateral auto punishment system, which is lacking in today's systems. Our preliminary study shows that Overfort approach is promising for defending against DDoS attacks even when attackers have more resources.

Chapter 5

AI-RON-E

This section is largely adapted from my paper co-authored with Akihiro Nakao entitled "AI-RON-E: Prophecy of One-hop Source Routers" published in IEEE Globecom Next Generation Networks Symposium in New Orleans, USA, Dec 2008 (acceptance rate: 36.8%[1].).

5.1 Deployable Resiliency

5.1.1 Resiliency

Defense Category	Filterable DDoS	Non-filterable DDoS	eDDoS (intermediary)	eDDoS (server)
Client React	Yes[a]	Yes[a]	Not applicable	No

[a]If alternate paths exist.

Table 5.1: Resiliency of AI-RON-E against DDoS

AI-RON-E offers protection against DDoS from the angle of client reaction.

Client Reaction: Exploit Internet Alternate Paths Clients that deploy AI-RON-E are empowered to route around congestion points, resulting from operational failures or DDoS attacks, without resource-intensive requirements, e.g., keeping the entire map of the Internet.

[1]This is unofficially obtained from http://www.cs.ucsb.edu/ãlmeroth/conf/stats/#globecom

As shown in Table 5.1, AI-RON-E can bypass filterable DDoS; it empowers clients to find alternative routes that are not clogged with a high probability of 0.69 (see Section 5.6.2.2). It has no reliance on distinguishing good/bad traffic thus it is useful even in the event of non-filterable DDoS. However, eDDoS's intention is not to clog paths to the server but rather increase the server utilization cost in a utility-based environment thus AI-RON-E is ineffective against server eDDoS. It is not affected by intermediary eDDoS because the intermediaries are routers, which in today's Internet do not impose utility-based charges.

5.1.2 Deployability

Existing hotspot-bypass mechanisms have resource-intensive requirements; either they require construction of an Internet map whose path information needs to be updated constantly (67), continuous probing of multiple network paths to discover congested paths (12; 98) or taking opportunistic measurements from a large variety of sent packets, at their destinations, which requires a wide deployment of receptors at those destinations (127). AI-RON-E aims to reduce stringent resource requirements without overly-compromising on the congestion bypass effectiveness; an AI-RON-E-enabled client consults specialized nodes known as "oracles" to obtain their partial views of the Internet, specifically, the paths those oracles take to reach the client's desired destination. Oracles have only that simple single function implemented by returning traceroute (115) results of a specified destination. A partial view from a random oracle is sufficient to enable a client to determine which possible detour points can help bypass any broken links with high probability. As shown in Section 5.6, having approximate 100 nodes is sufficient to give diverse partial views. No state is ever required on oracles and their system requirements can be met with minimum PC specifications. The loss of one oracle does not affect the AI-RON-E system and can easily be replaced by another system elsewhere in the Internet. For maximum diversity of views, preferred oracle locations are at the Internet edges, thus deployable by anyone with an Internet connectivity. Clients that wants to enjoy the benefit of AI-RON-E can either install the AI-RON-E network stack or download AI-RON-E-powered zero-installation clients, deployed on Java Web Start (JWS) technology,, which

empowers them to bypass congestion without relying on the destination server or network providers. In the short term, for high deployability purposes, AI-RON-E client can utilize source IP spoofing (see Section 5.7.2) to mobilize all existing routers as detour points to bypass hotspots. In the longer term, we make recommendations to router vendors on appropriate changes to existing loose/strict source routing code to enable this detour feature without resorting to source spoof, which can be released as patch and incrementally rolled out to production routers. All the above properties ensure that oracles and AI-RON-E clients have very low deployment barriers while our dual time-scale and incremental approach makes deployability on routers feasible compared to most research.

5.2 Assumptions

Oracle Deployment Besides changes to routers, which we have a short-term alternative (See Section 5.1.2), the component that parties have lowest incentive to deploy are oracles. However, due to the low specifications, minimal deployment effort and insignificance of a single oracle unreliability in an entire system, we believe that oracle deployment is not an issue. For example, we can utilize the hundreds of Planet Lab (84) nodes as oracles.

5.3 Overview

In One-hop Source Routing (OSR), we select a detour point (intermediary) to create an indirect path consisting of source-to-intermediary and intermediary-to-destination. The careful selection of an intermediary will ensure that the indirect path can bypass a hotspot interfering with a direct client-server communication. OSR research is not new but existing work, either uses edge nodes as intermediaries, e.g., RON (12), SOSR (49), RON-DG (87), Fei et al. (41) or introduce OSR code in routers, e.g., Yang et al. (125). The former increases deployability while requiring fewer OSR nodes since by just deploying them at different country ISPs, they are likely to have maximum divergence—relatively disjointed paths that lead to the same destination, as shown by the success of random intermediary selection in failure masking (49). By disjointed paths, we mean paths

that have few overlaps. However, since packets need to detour out to these edge nodes before being deflected to their destinations, the hop-count of these indirect paths are long. The latter have shorter indirect paths but widespread deployment faces resistance from infrastructure owners due to the effort involved, the risk of disruption and lack of financial incentives. Requiring each client to keep a list of thousands of intermediaries wastes resources and the selection of a suitable intermediary to bypass failure among the thousands in real-time also poses a tricky issue. Parsing down the thousands of nodes to those in strategic locations may allay the issue. However, the parsing algorithm is resource-intensive (26) even when considering just a localized area, e.g., within an AS. AI-RON-E bridges the space between edge node and router-based OSR. AI-RON-E reduces deployability headache in the short-term by utilizing source IP spoofing to mobilize all existing routers on the Internet as intermediaries without requiring any modification on the routers. In the longer term, OSR-capable routers can be incrementally introduced; given that the OSR code on the router simply requires processing an additional header field to determine the deflection target (intermediary), which is a subset of existing code that performs loose/strict source routing, this is a reasonable long-term assumption.

AI-RON-E refines the use of edge nodes; instead of using edge nodes as intermediaries, AI-RON-E clients use them as oracles; the oracles' diverse partial views of the Internet is utilized to aid the selection of a suitable intermediary among the thousands of routers. As a measure of the light-weightedness and effectiveness in failure masking, we designed AI-RON-E to match SOSR's light-weightedness of randomly selecting 4 edge nodes to attempt to bypass failure. The reason why we want AI-RON-E to outperform SOSR (described in Section 2.2.11), not only in terms of light-weightedness but also failure masking rate, is because SOSR is the most cost-effective alternative path selection mechanism to-date—it has the best ratio of failure masking rate over resource consumption, e.g., network probing and Internet information caching.

5.4 Design Goals

AI-RON-E aims to provide the following:

Reduced hop-count indirect paths With only edge nodes available as intermediaries, traffic detouring to them, en-route to destinations, has to travel from one edge of the Internet (source) to another edge (intermediary) before arriving at the final edge (destination). This network edge-to-edge-to-edge routing is superfluous and can be reduced.

Better failure-masking ability than SOSR AI-RON-E strives to improve failure-masking ability from two angles—mask more link failures and find failure-masking indirect paths quicker compared to SOSR.

Light-weight intermediary selection algorithm From a large pool of candidates, the intermediary selection algorithm has to, within a few attempts, successfully find an intermediary that can mask a given link failure using minimal resources such as network probings for path discovery.

Internet-scale infrastructure and Internet path diversity utilization We want AI-RON-E to deliver an infrastructure that can support a large Internet community and also enable Internet path diversity to be better utilized than it currently is.

Incremental deployment Although the AI-RON-E prophecy foretells a future where all Internet routers are OSR-capable, a partial deployment is definitely useful thus an incremental deployment makes sense. Moreover, AI-RON-E requires making modifications to the Internet routers and this has traditionally faced steep resistance. Besides progressive changes, incremental deployment enables users to opt-in with resource investment that suits their technology-risk appetite with the option to scale up the infrastructure that does not require major re-work when the need arises. The appeal of incremental deployment has been shown to work in practice, e.g., in Planet Lab (84) and peer-to-peer networks such as BitTorrent (22).

5.5 Architecture

In the AI-RON-E infrastructure (see Figure 5.1), any Internet system, e.g., servers, routers, etc., may play any/all combination of the 3 roles: *oracles* that provide

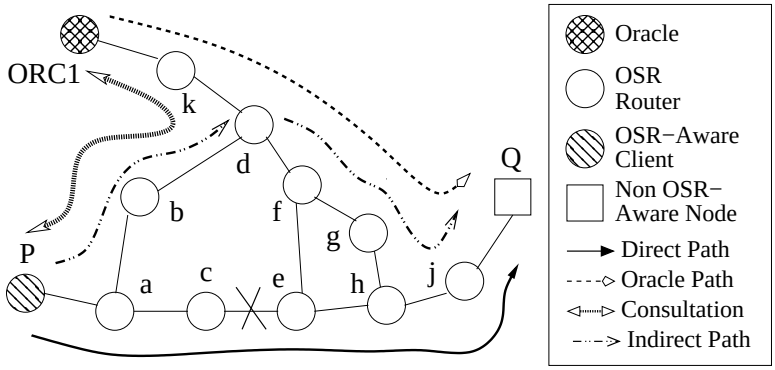


Figure 5.1: AI-RON-E architecture consists of OSR routers (non edge node intermediaries), oracles and clients embedded with logic to consult oracles to aid indirect path construction.

their partial viewpoints of the Internet (similar to existing traceroute servers (66)), *intermediaries* (OSR routers) that redirect traffic or/and *clients* that utilize the infrastructure to mask link failures.

Once an AI-RON-E client (P) detects a failure (c-e) in its direct path to a destination (Q) using a mechanism similar to traceroute, it will choose an intermediary to bypass the failure by consulting a random oracle (ORC1) to obtain its "oracle path"—the path the oracle takes to reach the same destination, e.g., ORC1's oracle path is ORC1-k-d-f-g-h-j-Q. Nodes along the oracle path, excluding the destination (ORC1-k-d-f-g-h-j) are candidate intermediaries that P can choose from since they are OSR routers that are capable of re-directing packets. For example, with *d* as an intermediary, P can bypass *c*-*e* using the indirect path constructed by concatenation of client-intermediary path P-a-b-d and intermediary-destination path d-f-g-h-j-Q.

The architecture is designed with the following properties to fulfill the design goals described in Section 5.4 and in Section 5.6 we show that AI-RON-E can indeed achieve them.

OSR routers far from the Internet edges offer shorter indirect paths With only edge node intermediaries as in SOSR, e.g., ORC1, in Figure 5.1, client P has to use ORC1 as an intermediary and the indirect path formed would have 12 hops (P-a-b-d-k-ORC1-k-d-f-g-h-j-Q). In AI-RON-E, an OSR router

that is further from the Internet edge, e.g., *d*, can be used as an intermediary thereby creating a shorter indirect path of 8 hops (P-a-b-d-f-g-h-j-Q).

Oracle-based intermediary selection for better failure-masking To formulate an intermediary selection algorithm that rivals the failure-masking rate of SOSR, we borrow SOSR's random intermediary selection algorithm and add a twist to it—since random edge node intermediaries can provide good failure-masking performance by virtue of their location at the Internet edges that gives rise to indirect path disjointedness from the direct path, our hunch is that by selecting a random node along an *edge node intermediary-destination* path as an intermediary, AI-RON-E can delivery similar effectiveness.

To compensate for the possible reduction of path disjointedness resulting from selection of an intermediary further away from the Internet edges, AI-RON-E has heuristics to remove candidates that cannot contribute to failure-masking; candidates in an oracle path that also appear in the direct path (source-destination) are filtered out. For example, in Figure 5.1, ORC1-k-d-f-g-h-j are in ORC1's oracle path but we exclude *h* and *j* as candidates. Since they lie along the direct path (P-a-c-e-h-j-Q), if they are used as intermediaries, P will most likely reach them via the direct path that includes the assumed broken link *c*-*e* resulting in bypass failure.

From the remaining candidates, ORC1-k-d-f-g, we choose only 2 for use– one between the first node and mid-point of optimized oracle path (*k*) and another between the mid-point and the last node in the optimized oracle path (*f*).

This avoids using adjacent candidates that may have similar client-intermediary paths resulting in "fate-sharing"; both candidates succeeding or failing. In Figure 5.1, P reaches adjacent nodes *f* and *g* through the paths P-a-c-e and P-a-c-e-f respectively, resulting in both candidates failing to mask the failure at c-e. Through "bad" candidates filtering, AI-RON-E naturally seeks out intermediaries that has a *higher* chance of masking failure *quicker* than otherwise. However, note that filtering may result in some oracle paths producing only 1 or no candidate intermediary.

Caching oracle paths to retain light-weightedness We use SOSR's light-weightedness as a benchmark for AI-RON-E. SOSR sends N packets and waits for at most N round-trip-times (RTT), before failing or succeeding in masking failure, where N is the number of intermediaries selected. A higher N instinctively provides a better masking rate, but it also incurs more network overhead. With the law of diminishing returns applying, for SOSR, N=4 is ideal (49).

Oracle paths form the basis for AI-RON-E to select better intermediaries but is expensive in terms of network probes required to obtain them. Thus, AI-RON-E retains SOSR's light-weightedness by trading-off some storage, i.e., requiring each AI-RON-E client to maintain previously acquired oracle paths in a cache of size $0.5N$ to $1.5N$. Clients always use their own cached oracle paths to find candidate intermediaries and expensive oracle consultations are only triggered when N intermediaries cannot be provided after exhausting all cache entries. As shown in Section 5.6.2.2, despite using the same few cached oracle paths, there are enough intermediaries along those paths to produce diverse indirect paths to mask various link failures on a path.

Distributed OSR routers aid scalability and path diversity utilization The huge pool of OSR routers enable AI-RON-E to support a large user community. Moreover, since the pool comprises widely-dispersed OSR routers located at different depths from the Internet edges, AI-RON-E clients that obtain random oracle paths and then select random intermediaries within the oracle paths, will most likely utilize different intermediaries resulting in traffic being distributed evenly over the Internet, thereby utilizing path diversity better and reduce congested "hotspots".

Loose-coupling enables incremental deployment Incremental deployment offers incremental benefits; with each OSR router introduced, it creates an extra opportunity for AI-RON-E clients (1) to mask link failures or divert traffic from current congested "hotspots" and (2) better utilize Internet path diversity. Incremental deployment is possible by making AI-RON-E

components—client, oracle and intermediary—loosely-coupled, i.e., a client can use any AI-RON-E node as an oracle or intermediary making bootstrapping into the AI-RON-E infrastructure extremely effortless and relatively de-centralized since existing tools, e.g., traceroute from remote systems (66) or from the client itself can be used to discover routers that can be used as AI-RON-E oracles/intermediaries. Moreover, destinations do not need to be AI-RON-E-aware. Thus, new AI-RON-E components can be "dropped" in any time and any place to scale up the existing infrastructure. As part of our goal to hasten deployability, we propose a shorter-term solution of utilizing source-spoofing to mobilize all existing routers as OSR points (see Section 5.7.2).

5.6 Evaluation

This section evaluates AI-RON-E's link failure masking approach. Using our hypothetical link failure evaluation methodology, we can assess more link failures than a conventional empirical link failure monitoring approach can within a given period of time but with the assumption that only a single link within a path fails at a time. Through it, we can foretell the effectiveness of AI-RON-E even though the infrastructure has yet to exist. More specifically, we want to find out if AI-RON-E can achieve its design goals: (1) Does AI-RON-E provide shorter hop-count indirect paths than SOSR? (2) Can AI-RON-E mask link failures as well and as fast as SOSR even though the intermediaries have to be selected from a larger pool of candidates that are mostly further from the Internet edges? and (3) How light-weight is AI-RON-E's resource utilization, specifically, cache storage?

5.6.1 Evaluation Methodology

We first describe the general architecture setup followed by a description of our hypothetical link failure evaluation approach and finally how we can simulate SOSR and AI-RON-E on this same setup for comparison purposes.

We use Planet Lab (PL) nodes at 100 distinct locations as traceroute servers. When consulted, each can provide the traceroute from itself to a given destination.

Next, we select 15 PL nodes (Asia - 3, US - 3, East Europe - 3, West Europe - 3, South America - 2 and Canada - 1) as traffic source and each source performs traceroute to the same set of 25 randomly-selected distinct PL locations to obtain direct paths. In our hypothetical link failure evaluation approach, for *each link* along each direct path, we check if there is an indirect path that can bypass it in the event of failure.

To simulate SOSR, any one of the 100 PL traceroute servers can be used as intermediaries and the indirect path is constructed by concatenating the source-PL path (obtained by traceroute from source) with the PL-destination path (obtained by consulting the selected PL traceroute server).

For AI-RON-E, the 100 PL nodes act as oracles and intermediaries can be any node along a randomly selected oracle's oracle path, excluding bad candidates that are heuristically filtered out (see Section 5.5). An indirect path can be constructed by concatenating the source-intermediary path (obtained by traceroute from source) with the intermediary-destination path (the sub-path of the oracle path obtained by consulting the selected oracle/PL node, starting from the intermediary).

For both SOSR and AI-RON-E, we conclude that an indirect path can bypass a failed link if it does not appear in the indirect path.

5.6.2 Results

In total, we evaluated approximately 15 x 25 = 375 paths comprising about 4500 links. For each link we theoretically assess if a bypass is possible. This is more than 3 times the amount of link failures analyzed by SOSR. From hereforth, "SOSR-N" and "AI-RON-E-N" are used to refer to SOSR and AI-RON-E algorithms with N-selected intermediaries respectively.

5.6.2.1 Hop-count of indirect paths

As shown in Figure 5.2, AI-RON-E indirect paths always have shorter hop-counts than SOSR ones because traffic does not need to detour to distant Internet edges en-route to destinations; the AI-RON-E plots are always further to the left of SOSR ones. At the 80th percentile, regardless of N, for AI-RON-E, the ratio of

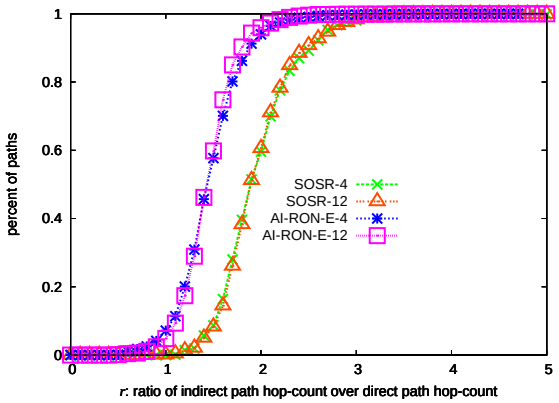


Figure 5.2: A CDF showing that AI-RON-E indirect paths have shorter hop-counts than SOSR ones regardless of the number of intermediaries selected for use.

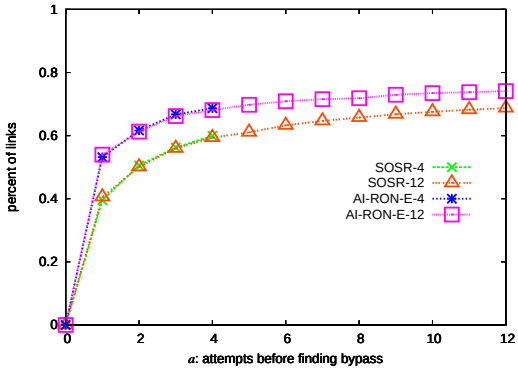


Figure 5.3: A CDF showing the number of intermediaries used, before finding one that can mask a given failure.

an indirect path hop-count length over its direct path length, r, is roughly 1.6 whereas for SOSR, r is approximately 2.3. This represents 30% improvement in path length reduction. It is also interesting to note that in some cases, AI-RON-E indirect paths are shorter than their direct paths, i.e., *percent of paths*>0 for r<1. There are two possible explanations: (1) Internet routing policies determined by ISP relationships (54) may dictate that Internet routing choose a path based on other characteristics rather than the shortest one and (2) Internet routing relies on Border Gateway Protocol (BGP) (90), which selects the shortest path to destination based on Autonomous System (AS) hop-counts but huge ASes have many router hop-counts within therefore, a shortest AS hop-count path (chosen by BGP) does not necessarily mean a shortest router hop-count path (as discovered by AI-RON-E).

5.6.2.2 Speed and ability of link failure masking capability

Figure 5.3 shows which intermediary out of N selected ones, a, successfully masks a failed link. Note that the plots for SOSR-4 and AI-RON-E-4 are discontinued after a=4 while for SOSR-12 and AI-RON-E-12, they end after a=12 since the algorithms themselves limit the number of intermediaries (N) being selected. These end-points, a=4 and a=12, also represent the number of links that can be

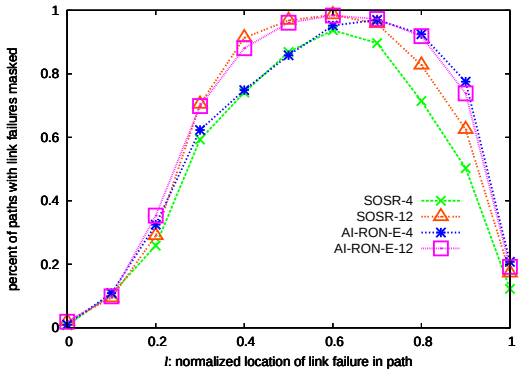


Figure 5.4: Failure-masking rate of normalized link locations, l, for SOSR and AI-RON-E variations. Failure-masking rate is highest near Internet core, l∼0.5.

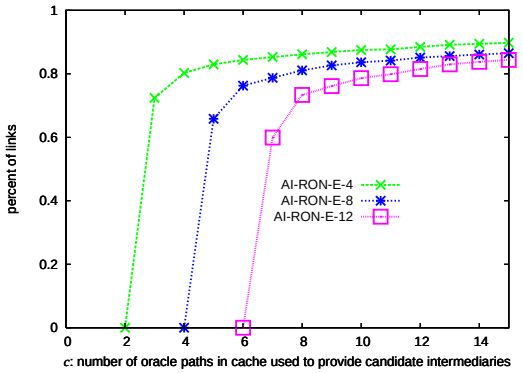


Figure 5.5: The number of entries in the cache used to provide enough intermediaries as required by the different AI-RON-E variations.

masked by each algorithm variation after trying all N intermediaries. For N=4, AI-RON-E masks 69% of failures compared to SOSR's 60% while for N=12, AI-RON-E masks 74% of failures versus SOSR's 69%. In both cases, AI-RON-E masks more link failures than its SOSR counterpart.

Figure 5.4 also shows the higher failure-masking rate of AI-RON-E but from a different angle; it shows the location of link failures that can be masked by indirect paths. Link failure location, l, are normalized within the path, i.e., l vary from 0 to 1 with l∼0 indicating links near the source of a path and l∼1 indicating links near the destination. As expected, links nearer the Internet core, l∼0.5, has better path diversity resulting in higher failure-masking rate, i.e. close to 100% for l=0.5 to 0.7.

Besides showing the masking rate of the algorithms, Figure 5.3 also shows that AI-RON-E can seek out indirect paths faster than SOSR between 6-12% of the time. For instance, AI-RON-E-4 can find 52% of failure-masking indirect paths by just using the 1st intermediary while SOSR-4 can only manage to find only 40%. The filtering of bad candidates by AI-RON-E obviously aids finding an intermediary that can mask a given failure quicker.

5.6.2.3 Intermediary selection algorithm resource consumption

Figure 5.5 allays the fear one may have about the size of the oracle path cache required by each AI-RON-E client. Since we select at most 2 candidates from each oracle path, for AI-RON-E-N, the minimum number of oracle paths needed to find N intermediaries is $N/2$. This explains why the CDFs for AI-RON-E-4, 8 and 12 start at 2, 4, and 6 respectively on the x-axis. However, after filtering out bad candidates, some oracle paths are left with no candidates. Under those circumstances, more than $N/2$ oracles paths will be required to produce the number of intermediaries specified by the algorithm.

As N increases, the number of cached oracle paths, thus the cache size, required to provide those intermediaries increases. However, the plots show that to provide N required intermediaries to 70-80% of links, the upper-limit of the cache size is bounded ("knee" of the plots); a meager cache size of $0.5N$ to $1.5N$ is sufficient. For the remaining 20-30% of links, it is extremely challenging to provide the required number of intermediaries because those links have low path diversity thus more resources, i.e., cache storage or network probes, has to be sacrificed to find candidate intermediaries.

5.7 Discussion

5.7.1 Minimizing link failure effects

The hypothetical approach to analyzing link failures without them actually occurring gives rise to AI-RON-E's ability to minimize link failure effects *prior to link failures*, i.e., once a direct path is established, AI-RON-E immediately attempts to find indirect paths to bypass each link along the direct paths, in the background in preparation for link failures. Moreover, these indirect paths may also be used concurrently to enable multi-paths and increase transmitted bandwidth (14), (53). However, minimizing link failure effect only makes sense for long-term connections, e.g., daily critical back up of data to remote data center.

5.7.2 Deployment issues and workaround

Using PL nodes in the existing Internet as AI-RON-E intermediaries is unsatisfactory since PL nodes are not *non-edge* intermediaries. Therefore, we have to exploit a peculiarity in IP routing—source IP spoofing, to "coax" existing routers to function as non-edge intermediaries. Upon encountering a link failure, through oracle consultation, an AI-RON-E client selects an "non-edge intermediary" (this can be any conventional router) to mask the failure. To coax the conventional router to behave as an intermediary, the client crafts a packet by spoofing the source IP header with the intended destination IP and insert into the destination IP header, the IP of the router (intermediary) with TCP port number set to 0, which triggers an error condition at the packet recipient. Note that any other invalid settings on IP or TCP/UDP fields that can result in an error condition will also suffice. When the packet reaches the router, it gets rejected with "ICMP port unreachable" and the router attempts to send it back to its "originator" which in this case, has been spoofed with the intended destination IP. We are aware that IP filters reject spoofed IP packets so this mechanism may not work for all nodes. However, we foresee that nodes with specialized need for resilient connections to selected targets, e.g., daily inter-bank data transfer, can be permitted "controlled spoofing", i.e., the node can spoof its source IP to be one of those permitted range of target IPs and this can be controlled at the ingress firewall belonging to either/both the end-user or/and ISP.

5.7.3 AI-RON-E Client Code

Clients that want empowerment will install AI-RON-E code in one of the 3 forms: (1) AI-RON-E network libraries that applications can be recompiled with, (2) application-specific AI-RON-E clients in the form of zero-installation code, e.g., Java Web Start(JWS) or Java Applets, and (3) operating system (OS) with AI-RON-E-aware network stack. The first is ideal for developers to recompile their applications without changes, in order to acquire AI-RON-E features, and distributed the new AI-RON-E-aware application binaries to users. The second is ideal for application service providers, e.g., a retail web site, since by writing a JWS AI-RON-E client for the site, all its consumers can transparently download

this JWS client from the homepage and use it through their browsers to acquire AI-RON-E features. The third option of modifying the OS network stack requires meticulous OS recompilation but is ideal since all applications can acquire AI-RON-E capabilities without changes.

5.8 Limitations

Cache size per destination Each cached oracle path provides information for bypassing link failures only to a particular destination. As the number of destinations increase, the cache entries increase linearly. This seems to detract some value from our proposition at first glance, however, the number of resilient connections that a client actually require may be smaller than thought; a lot of connections are for casual browsing, with only the occasional critical transactional connection or other protocols, e.g., SSH (111), requiring resiliency.

Source Spoofing Exploitability Source spoofing may be blocked by access routers manually configured (42) or enabled automatically by uRPF-capable (29) routers. Therefore, AI-RON-E may not be available to everyone. However, as reported by the MIT Spoofer Project (71), the number of networks not protected against source IP spoofable is still substantial. Moreover, we foresee that AI-RON-E is utilized for critical applications such as, critical data transfer between two banks, more than for casual use. In such cases, the initiator of the connection can easily permit "controlled" source spoofing, e.g., allow only source IPs to be spoofed as a restricted range of permitted destination IPs.

Uncertainty in Traceroute Prior to failure, a client determines its traceroute to its destination. When its connection is disrupted, a second traceroute is performed to determine the failed link; a link is identified by its two adjacent routers. By determining if the failed link is contained within an analytically constructed indirect path, we can determine whether bypass is possible using that indirect path. However, due to security reasons or mis-configurations, hops in traceroutes sometimes cannot be ascertained

and they are marked with asterisks/stars "*". This makes determining whether an indirect path can bypass a failed link without actually sending the packet, impossible. To obtain a lower bound for our evaluation, in other words, to under-estimate our bypass success rate, we apply strict rules for determining bypass; when a failed link has "*" as one of its adjacent routers, we reject the possibility that an indirect path can bypass the link failure as long as the other non-"*" adjacent router appears in the oracle path used to construct the indirect path. For example, consider that an oracle path is ORC1-k-d-f-g-h-j-Q, while the failed link in the direct path of P-a-c-e-h-j-Q is e-h but the client traceroute only returns P-a-c-*-h-j-Q as direct path, with *-h being the failed link. Even though the oracle path does not contain the broken link e-h, the "*" in makes it inconclusive and based on our strict rule, we deduce that the oracle path cannot be used to bypass the broken link since the "h" in *-h appears in the oracle path, leading to under-estimation of success rate.

Single Link Failures In our evaluation, we considered only single link failures in the path since single link failures account for 70% of failures in the IP backbone (70). Multiple link failures is part of our future work.

Router Alias A router has many interfaces, thus many IPs. Therefore, even when two paths appear disjointed based on the IPs of its router hops, they may share a router or even a link (share two adjacent routers). In other words, an indirect path may appear to bypass a failed link when in reality it cannot. Therefore, we need to utilize alias resolution tools like Ally (103), which employs multiple heuristics, to determine if different IPs actually belong to the same router. The evaluation is thus reliant on the accuracy of these alias resolution tools.

Effectiveness Reliant on Path Diversity The effectiveness of AI-RON-E ultimately relies on the availability of path diversity. As shown in Figure 5.4, near the Internet edge (near source and destination) failure masking rate is low due to the low path diversity in such areas. To improve path diversity, the source or destination can be made multi-homed.

5.9 Future Work

Before the long-term goal of introducing AI-RON-E code into routers, we would like to further study the implications of OSR. Although OSR is somewhat similar to source-routing in IPv4 and IPv6, with the IPv4 version already outlawed through best practice and the IPv6 version likely to follow suit after the dangers were brought into light (21), we believe OSR, which is a very restricted form of source routing that permits the sender to control the path taken by specifying one-hop instead of multiple hops, poses less security issues. Through the security study, we can make recommendations to router vendors to enable OSR by slight modification to already existing loose/strict source routing router code. Another area that we can improve on is adapting AI-RON-E to bypass multiple link failures that account for 30% of failures in the IP backbone (70).

5.10 Conclusion

We described the AI-RON-E prophecy that involves equipping every router with OSR capability to turn them into AI-RON-E intermediaries and clients with logic to select failure-masking intermediaries through oracle consultation. The proximity of router intermediaries to the Internet core offers shorter hop-count indirect paths of up to 30% while their sheer number enables AI-RON-E to exploit path diversity to find failure-masking indirect paths. AI-RON-E can improve SOSR failure masking rate by 6-8% and seeks out the indirect paths faster 6-12% of the time. All this come at the expense of a slightly more involved algorithm, i.e., filtering of bad candidates and the maintenance of an oracle path cache with $0.5N$ to $1.5N$ entries for each destination, at each client.

Chapter 6

KUMO

6.1 Deployable Resiliency

6.1.1 Resiliency

Defense Category	Filterable DDoS	Non-filterable DDoS	eDDoS (intermediary	eDDoS (server)
Economic Framework	Yes	No	No	No

Table 6.1: Resiliency of KUMO against DDoS

KUMO enforces DDoS defense from the following angle:

Economic Framework: Resource Harness KUMO is an intermediary-based framework; it conceals servers from direct client connectivity thus offering traffic reception control ability to servers. This shifts the brunt of DDoS attack to the intermediaries. Thus, a large number of intermediaries is necessary for defense against filterable DDoS attacks. Existing intermediary-based research has often incorporated incremental deployment mechanism to encourage users to adopt the research with the hopeful eventuality that everyone does likewise resulting in sufficient intermediaries for DDoS defense. However, we would like warn against such wishful thinking by pointing to the lack of success of other incrementally deployable security standards namely Secure BGP (S-BGP) (58) and secure DNS (DNSSEC) (17).

The lack of adoption in the former is due to concerns about the resource consumption on routers when cryptography is used and the issues with Public Key Infrastructure (PKI), both of which are critical S-BGP components (23). The problems of the latter is due to the perception of a lack of need, the concern about the operational and technical issues, as well as the financial constraints as reported by 65 DNS registry participants of a DNSSEC survey (25). KUMO represents the first DDoS research that attempts to tackle the issue of harnessing intermediaries head-on with incremental deployability only as a sub-feature. It does so by facilitating the usage of *any* existing or future Internet systems as intermediaries and offering financial compensation for intermediary owners.

KUMO is superior in harnessing intermediaries for filterable DDoS defense. Unfortunately, it offers no defense against non-filterable DDoS and server eDDoS because the intermediaries that shield a server are normal systems (and not specially constructed defense systems) belonging to other organizations, thus no DDoS detection and filtering mechanism can be built onto them. The proposed KUMO pay-as-you-use (utility-based) billing system for intermediary resource usage further exposes a KUMO adopter to intermediary eDDoS; when attackers flood intermediaries with requests to forward packets to the adopter's server, this rakes up the server's intermediary usage charges.

6.1.2 Deployability

The ease of recruiting any existing and new system as an intermediary and the remuneration offered in return for an intermediary's resources, facilitates resource harness for DDoS defense from both a technological and economic standpoint as never before. We foresee the following as rich sources of intermediaries: (1) the thousands of existing Internet servers, e.g., Internet Relay Chat (IRC) (80), forums[1], Software-as-a-Service (SaaS) such as OnlineMQ (81)—a message queuing service, Amazon Simple Storage Services (S3) (7), etc., (2) the smaller number but equally significant amount of next generation prototypes, e.g., I3 (104)—Berkeley's Internet indirection architecture, proxy networks (119), Tor (37)—an

[1]Forums are also known as message boards or discussion groups.

anonymous routing system, AI-RON-E (61), etc., and (3) the possible emergence of an *intermediary marketplace* consisting of suppliers, brokers and consumers to reap the incentives at stake. We do not offer any economic proof substantiating a marketplace emergence but merely envision that KUMO can offer a foundation for it better than existing DDoS mitigation schemes due to the hassle-free (no component installation and maintenance) nature of participation (see Section 6.8.1).

6.2 Assumptions

Resilient Naming Service KUMO relies heavily on a naming service to tell clients which intermediaries can forward their connection requests to the desired destinations. It can utilize existing DNS for this purpose by just making some changes to the authoritative server, i.e., the mapping for a KUMO protected server hostname can be continuously updated to point to available intermediaries. This implies that KUMO's resiliency is no worse than the existing Internet. For improved name service resiliency, existing research such as CoDoNS (89) and ConfiDNS (85), can be adopted.

Aggregate Intermediary Resource The aggregate resource of intermediaries employed must be greater than the capacity of zombies used to launch a filterable DDoS in order for defense to be effective. This is achievable because of the ease KUMO can recruit intermediaries to exceed the size of zombies.

Hidden Server Location Enforcement Intermediaries play an important role in concealing a server behind to avoid direct attacks. Servers have to utilize only trusted intermediaries that will not divulge their locations thus exposing them to direct attacks. This reduces the number of intermediaries available for use thus possibly affecting the aggregated resource capacity, which is critical for DDoS defense. We assume that there are sufficient intermediaries that a server can trust or enforce trust upon, e.g., through real-world business relationships or mutually beneficial DDoS cooperative initiatives, to acquire sufficient intermediaries for DDoS defense.

6.3 Overview

DDoS mitigation proposals have gained little traction in deployment to date because of their two misconceptions: (1) taking for granted that a large number of DDoS mitigation intermediaries are easily available or (2) requiring modifications to critical and proprietary existing core systems spanning multiple parties, e.g., core routers, to incorporate DDoS mitigation features in order to use them as intermediaries. KUMO is the first work, to the best of our knowledge, that attempts to address these critical misconceptions through its framework design; it facilitates using *any* existing or future Internet systems, instead of core systems, *without modification*, as intermediaries; by making *non-disruptive modifications* on the client-side through zero-installation technology such as Java Web Start (JWS) (109) or Java Applets (108), and the server-side through a server component, we enable tunneling of client-server communication through any intermediary's application layer protocol. KUMO also offers an accounting platform to establish an *incentive scheme*, which compensates intermediaries for resource utilization. KUMO's protection principle of "a degraded service is better than no service" focuses on service *availability* rather than *performance* guarantees since the latter is always going to be difficult under adverse conditions. In other words, KUMO's strength is ensuring the availability of critical services albeit with degraded performance.

6.4 Design Goals

Swarm DDoS Protection The communication between client-server must be resistant against filterable DDoS that may flood any point in the data path. We consider four possible locations of congestion: (1) client-intermediary path, (2) intermediary, (3) intermediary-server path and (4) protected server. We adopt the "swarm" approach of harnessing intermediaries, whose aggregate resource is greater than zombies perpetrating DDoS, thus mitigating (2) and shielding protected servers against filterable DDoS. The distributed locations of intermediaries increase path diversity thereby reducing the occurrence of hotspots in the path from distributed clients to intermediaries

(1). The server (4) and the path from intermediaries to a server (3) is not susceptible to congestion because a server is capable of controlling traffic reception by either retrieving data at its own pace from the intermediaries or terminating connections to intermediaries to stop packet arriving from them.

Intermediary Recruitment Ease To harness a large swarm of intermediaries we must (1) avoid mandating complex changes to existing infrastructure, (2) make it easy to adopt the technology and (3) allay concerns about disruption to existing operation.

Economic Incentive Another important factor in encouraging intermediary participation to increase swarm size is to offer economic incentives.

Utility-Based DDoS Mitigation Service Although DDoS mitigation services currently exist, the monthly subscription charges are costly because service providers need to recoup the cost of building the large-scale infrastructure. This hampers service adoption especially by smaller organizations. We want to build a "pay-as-you-use" service (utility-based service) to enable more organizations to acquire DDoS protection.

Highly Reliable *Eventual* Communication KUMO's focus on availability rather than performance guarantees, is crucial to mitigate havocs wrecked by worm epidemics (77), e.g., disrupted Auto Teller Machines and airline flights, etc. Therefore, "exclusive" web sites— in terms of information stored, e.g., an airline reservation system or a bank that hold exclusive user information, and in terms of products/services on offer, e.g., a sole product distributor, whose unavailability rather than poor response will cause major inconveniences, are the most likely benefactors of KUMO.

Resistant Against Complete Persistent Disruption It should be difficult for an attacker to persistently disrupt the communication between a targeted client-server pair.

6.5 Architecture

In this section, we describe the features used to fulfill the design goals in Section 6.4 followed by full details about the KUMO architecture.

Intermediary-based Architecture To protect against filterable DDoS, an intermediary-based architecture shields a server from direct client connectivity and grants it the ability to control traffic reception. This shifts the brunt of the attacks to the intermediaries. Such an approach makes sense since it is easier to replicate intermediaries compared to a server; intermediaries have functionality limited to just traffic forwarding or temporarily storing client-server traffic. The complex issue of replicating a server has now been transformed to a more tractable issue of harnessing sufficient intermediaries to absorb DDoS attacks with the remaining ones available to relay client-server traffic.

Utilization of Any System as Intermediary To increase a defense's swarm size quickly, we facilitate recruitment of any existing or new Internet system as a swarm member. A dynamic client component, such as Java Applet, and a server-side component can enhance both a client and server with the ability to tunnel their traffic through any intermediary's protocol. Besides, easing deployment, the minimal modification required on client, server and intermediary serves to protect various parties's existing IT investments thus encouraging adoption.

Technology Transparency To enable usage of any existing or future Internet systems as intermediaries, we design KUMO such that a KUMO client can communicate with its desired server by tunneling the communication through an intermediary's protocol. An intermediary does not even know that it is being utilized by KUMO. This technology transparency implies 2 important things: (1) a system can become an intermediary without making any technological changes and (2) a system owner does not have to exceedingly worry about possible disruption to existing operation when the system doubles-up as an intermediary since no new technology is introduced and KUMO merely uses the system like a conforming client.

Traffic Accounting System By incorporating an accounting system to keep track of how much traffic each intermediary has handled for each protected server, KUMO facilitates payment from the server owner to the intermediary owners. This economic incentive spurs existing and future system owners to offer their over-provisioned system resources for intermediary purposes, which facilitates the KUMO swarm to achieve large sizes.

Idle Resource Loan A lot of Internet systems are over-provisioned for peak load, which only occurs occasionally, while at other times, the over-provisioned resources remain idle. To increase the efficiency of over-provisioned resources, these systems can "loan" them out for intermediary purposes in return for financial compensation, while retaining the control of how much resources to loan out and the ability to pre-empt the loan when peak load occurs. KUMO makes use of "idle resource loan" concept to build a utility-based DDoS mitigation service; the already paid-for but idle over-provisioned resources can be activated for use against DDoS defense by KUMO subscribers, which the subscribers only have to pay for based on usage.

Capabilities and Multipath Communication To mitigate against complete persistent disruption on a certain client-server pair's communication, KUMO differentiates between initial connection (IC) requests and established connections (EC), and keep the intermediary channels used for EC privy only between the client-server pair. In other words, KUMO has a two-stage connection establishment process. In stage 1, the IC channels for any server are known and open to every client. Upon successfully competing among themselves and possibly against zombies, clients whose IC request is processed by the server will enter stage 2, where each client and server pair will agree upon multiple EC channels for subsequent communication. The secrecy of the EC channel names enhanced by their multiplicity, ensure that complete and persistent disruption of a client-server pair communication is extremely difficult for attackers.

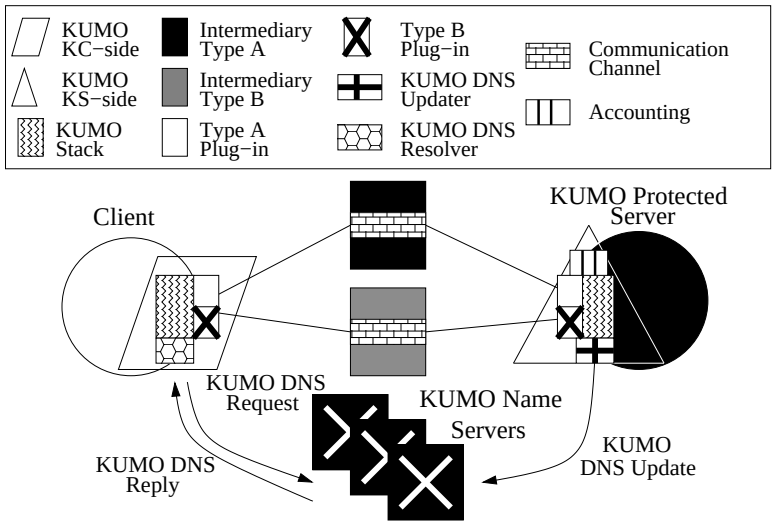


Figure 6.1: KUMO framework is implemented as KS-side and KC-side. Both hide KUMO details from client, protected server and intermediaries. In addition, KUMO provides multipath and accounting facility.

6.6 Implementation

KUMO consists of a framework and protocol implemented in the form of a name server, a protocol stack, a client-side (KC-side) and server-side (KS-side) component. We start with the description of the framework and protocol, followed by the other components classified based on their functionality.

6.6.1 Framework

The framework seeks to: (a) hide details of KUMO, enabling clients and servers to utilize KUMO without requiring any modification, (b) enable any Internet system to be used as an intermediary unmodified, (c) grant traffic control reception to a protected server and (d) incorporate an accounting platform for billing purposes. For (a), the framework provides a *multipath facility* that is manifested as the KUMO stack embedded in the KC-side and KS-side (see Figure 6.1). Both components work hand-in-hand to handle data fragmentation/assembly, data fragment transmission/reception over multiple intermediary channels, lost data fragment request/retransmission and "unreliable" intermediary channel tracking. To

achieve (b), the multipath facility is designed only to require a simple code plug-in, *written once* for each intermediary type and dynamically pluggable into the stack thus enabling KUMO to understand how to send/receive data to/from that intermediary type using that intermediary's protocol. As long as KUMO knows how to send/receive data to/from an intermediary type, the multipath facility will be able to perform various packet manipulation, e.g., fragmentation/assembly and data tunneling, over those intermediaries. For protected server traffic control reception (c), KUMO offers two ways. First, if a protected server has capacity to handle new clients, it creates a new listening channel at the intermediary and updates the KUMO name server with a new server hostname mapping to the new channel. A client that consults the name server will discover the available connection request channel. Second, if the protected server is overwhelmed by traffic from a certain channel, it can drop the "synchronous" channel, i.e., a channel created on a *traffic forwarding* intermediary, or the protected server can retrieve traffic at its own pace from an "asynchronous" channel, i.e., a channel created on intermediaries that act as message a pick-up point. An *accounting facility* (d), can be naturally implemented at the KS-side where all traffic is accumulated before transmission to the protected server.

6.6.2 Protocol

The KUMO protocol is message-based as oppose to stream-based; it receives transmission data from application layer as a piece of message from which the size can be determined and acknowledgments, request for retransmissions and retransmissions themselves are all done in relation to that message, which is uniquely identified by its message ID. Each message can be fragmented into different sizes corresponding to the maximum transmission unit (MTU) of the various intermediaries in use and sent simultaneously. To support such multipath transmission, the protocol headers include message ID, fragment offset (location of the fragment within the message), fragment length and total message length, which are necessary for fragmentation and re-assembly. For retransmission request, the receiver indicates the fragment offset and fragment length missing. If there are multiple consecutive fragments missing, the fragment length missing will be the sum of

the lengths of all the consecutive fragments. Thus a single retransmission request can solicit multiple fragment retransmissions. Retransmission of a fragment can occur over another channel type that is different from the one it was sent over initially thus possibly incurring further fragmentation.

6.6.3 Multipath Facility Components

The KUMO framework and protocol are implemented as a protocol stack using Ruby because Ruby facilitates fast prototyping. However, there is a performance trade-off since it is an interpreted language. As part of our future work, we will port the entire protocol stack into C for performance reasons and Java, in order to support zero-install dynamic client—Java Web Start (JWS) or Java Applet.

The multipath capability is mainly implemented in the KUMO stack, which is utilized by KC-side and KS-side. The KUMO stack can be installed as a *library* that exposes an application programming interface (API) to all of KUMO's features. Applications can be written or ported to take advantage of KUMO multipath facility through the API, with minimal knowledge about KUMO's internals. Listing 6.1 shows the sample Ruby server code that listens for a initial connection request.

Listing 6.1: Ruby code for creating a server socket to listen for connection requests using KUMO API

```
1  # NOTE: Lines beginning with # are comments
2  @kumo_controller = KUMOCompositeChannelController.new
3  @ksside = KSside.new(@kumo_controller)
4  @ksside.start
5  # Parameter descriptions:
6  # domain: The domain name the listening socket is for
7  # send_domain: The name of destination where reply should
8  #              be sent. Usually this is just the listening "domain"
9  #              with some appended with random number to create a
10 #              unique destination.
11 # output: The contents of the connection request
12 # sockid: The socket ID used to send reply
13 @ksside.register_ic_handler(domain) { |domain, send_domain, output,
14                                        sockid |
15 ...
```

```
16 # insert code to perform task when a connection request is received
17 ...
18 }
```

Lines 2-4 are standard server-side initiation code. The developer just need to insert application specific code to handle connection request, in between the parentheses in lines 13 and 18. For example, the code to handle the 2nd stage of connection phase, i.e., negotiating the intermediaries that is privy to only a client-server pair for further communication, can be done here. Using KUMO API on the client-side is equally easy as shown in Listing 6.2.

Listing 6.2: Sample listing for creating a client code to send data using KUMO API

```
1  # NOTE: Lines beginning with # are comments
2  # Parameter descriptions:
3  # send_domain: the destination domain name of the data to be sent
4  @kumo_controller.start_all_channel_sets_for_domains([send_domain])
5  # The values of the parameters are:
6  # send_msg: The data to be sent
7  # reply: The reply from the server
8  # reply_sockid: The socket ID through which we can respond
9  #               to the reply we receive
10 # processed: True if we decide to handle reply.  Advance use only.
11 @kumo_controller.send(send_domain, send_msg) { |reply, reply_sockid,
12                                                 processed|
13 ...
14 # insert code to perform task when reply to data sent is received
15 ...
16 }
```

The developer just need to insert application specific code to handle reply received for the data sent, in between the parentheses in lines 11 and 16. For example, the client code, used to negotiate intermediaries for 2nd stage of connectivity with the server, can be placed here.

Besides creating standard client-server application code, we can use the KUMO API to create a KUMO-based on-demand zero-install KC-side, e.g., through JWS [1], which is automatically packaged with the KUMO API libraries, to of-

[1]This will be possible once we port the KUMO stack to Java.

fer out-of-box KUMO-aware applications, e.g., a KUMO-based web browser.

In either case, the KC-side is protocol-neutral, i.e., regardless of the type of message sent through it, e.g., HTTP (web), SMTP (email), etc., its role is just to encapsulate the message with KUMO protocol, utilize the multipath facility to fragment and tunnel the KUMO encapsulated message over multiple intermediaries channels using those intermediaries protocol. *Intermediary plug-ins* that can dynamically loaded by the KUMO stack enable the messages to be sent/received to/from various intermediaries. The usage of intermediaries is controlled by the KS-side and shielded from developers. Intermediaries are stored in a pool indexed based on the domain name the intermediaries will serve on the KS-side stack. Upon initialization of a listening socket, using the API *register_ic_handler(domain)* on a *ksside* object (Listing 6.1), the KS-side stack will automatically select an intermediary from the pool serving the *domain*, create a channel on the intermediary and update the dynamic DNS entry for the domain with a mapping to the new channel. Note that the KS-side stack can maintain multiple intermediary pools for different domains, thus a single KS-side can be shared by multiple servers from different domain. Once an IC connection request of a client is accepted by the server on the listening socket, the client-server pair will agree upon multiple intermediary EC channels for subsequent communication. The role of the KS-side includes retrieving tunneled KUMO traffic from intermediary protocol, stripping the KUMO headers, reassembling the fragmented data and determining what to do with the data, i.e., data from IC connections will be used to establish EC connections, while data from EC connections will be forwarded to the destination server. Replies from the destination server will be tunneled back over multiple intermediary channels to the client by the KS-side through the KC-side.

The KUMO stack also handles retransmission and tracks intermediary channels used for each fragment in order to identify "unreliable" intermediary channels. KUMO uses distinct counters to record fragments dropped by each channel and if a counter exceeds a configurable threshold, its channel is marked "unreliable" and excluded for use with the possibility of a new channel being created by the server to replace it. It is necessary to quickly discontinue the use of unreliable

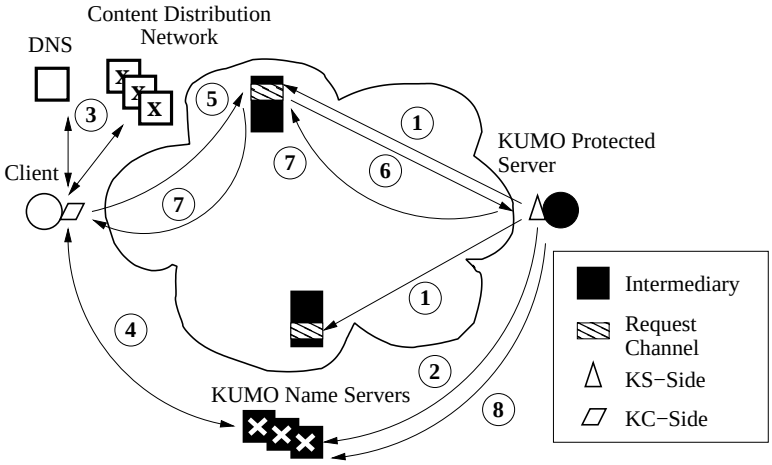


Figure 6.2: The KUMO communication process from IC channel creation, IC request to EC establishment as described in Section 6.6.5.

channels, because they result in many retransmission requests, which slows down data transfer.

6.6.4 Accounting Facility

A KS-side has an accounting facility to keep track of the utilization of each intermediary by each protected server. Through accounting, a KS-side can: (1) abide by its agreement with each system owner not to exceed the capacity committed for intermediary functionality; an intermediary that no longer has excess capacity will not be selected for use until the excess capacity is available again and (2) bill a protected server to compensate the intermediaries used for data relay. It is interesting to note that if intermediaries have built-in billing mechanisms, e.g., SaaS platforms such as Amazon's S3, then a protected server can be billed using the intermediary's billing system with KUMO's accounting system used to validate the billing. This can foster trust among intermediary and protected server owners in KUMO thereby encouraging participation.

6.6.5 Walkthrough

To tie-up loose ends and for ease of reference, we provide a walkthrough of KUMO here. The example protected server we are using is a fictional web pet store, AiKoPon. To utilize KUMO, AiKoPon develops a KUMO-based Java Applet to deliver a zero-installation client to web browsers. The applet can be embedded in a static web page stored in a highly-resilient Content Distribution Network (CDN) (4; 45; 120). It is transparently downloaded when a user hits on the static web page. In this scenario, we show the KS-side being installed on a protected server but it can be installed on separate system. The parenthesized numbers in our description correspond to the numbered steps in Figure 6.2. We do not show the process of intermediaries registering with the KS-side and indicating the domain that they would like to restrict their service to. Rather, we assume that this has been completed. The KS-side selects 2 intermediaries where it will listen for connection requests and creates IC request channels on them (1). It updates its dynamic DNS entries on the KUMO name servers to point to those new channels (2). A user that wants to visit AiKoPon's website consults the DNS server and is directed to load the homepage containing the applet from the CDN (3). The applet consults the KUMO name servers to obtain an available IC request channel for AiKoPon (4) and sends a connection request to the channel on the intermediary (5). If the request times out, e.g., gets dropped by the intermediary due to congestion, or is already occupied by another client, thus receiving no reply from the protected server, the applet will retry steps (4) and (5) until it eventually succeeds. The intermediary, if it is a synchronous one, forwards the connection request to the KS-side (or KS-side will pull the request if the intermediary is asynchronous), and retrieves the connection details (6). The KS-side processes the connection request using application specific code (Listing 6.1) and sends its reply back to the applet through the intermediary (7), e.g., negotiating multiple EC channel intermediaries privy only to a client-server pair for further communication. At the same time the KS-side will inform the KUMO name servers to remove the hostname mapping to the channel that is now occupied (8). When the KS-side has resources available to serve clients, it will start over from

step (1). A client can now proceed to communicate with the server by tunneling communication over the negotiated EC channels.

6.7 Evaluation

To prove KUMO's much vaunted deployability, its usability and ability to mitigate DDoS; we demonstrate: (1) KUMO's ease of use by plugging-in as intermediaries, 5 types of live Internet systems, 1 experimental one as well as 4 hybrids, which are a combination of synchronous and asynchronous live Internet systems, (2) data transmission occurs within acceptable times through these intermediaries and (3) data transmission is still possible under data loss of up to 60% (and possibly higher) by using multipath as a case study. In addition, we define a metric for path availability and show how distributed intermediaries enhance path availability by using IRC intermediaries as a sample set.

Our setup consists of a single Planet-Lab (PL) node in the US with the KS-side installed, acting as a KUMO protected web server and 20 PL nodes (5 in Asia, 7 in Europe and 8 in US) with the KC-side installed, acting as KUMO clients. We perform data transfer between each KUMO client and protected server pair (20 pairs) in turn using different intermediary types.

6.7.1 Flexibility

We plug-in live application systems as intermediaries and perform file transfer of a 51kB (25 A4-size pages) text file over them using HyperText Transfer Protocol (HTTP) between each KUMO client-server pair. KUMO is protocol-agnostic so the usage of HTTP serves as only one of the possibilities. We tested the following as intermediaries: 3 live application systems (forums, IRC and OpenDHT (91)—a distributed peer-to-peer database system), 1 Web 2.0 application (Flickr—a social network photo site), 1 SaaS platform (Amazon's S3) and 1 next generation server (I3). These intermediaries can be categorized into synchronous and asynchronous ones. Synchronous intermediaries (IRC and I3) can immediately forward data it receives from a sender to the receiver while for asynchronous ones (forum, OpenDHT, Flickr, S3), a receiver has to pull the data stored by the sender from

them. Receivers using asynchronous intermediaries need to periodically poll for data availability, which involves a trade-off between timeliness of data reception and polling resource consumption. Since the resource consumption taxes both receiver and intermediary, we introduce hybrid intermediaries, i.e., a combination of synchronous intermediaries as control channels to inform the remote end when data transfer is completed and asynchronous intermediaries for the data transfer. The 4 hybrid intermediaries used are: hybrid-forum, hybrid-opendht, hybrid-flickr and hybrid-s3.

In all cases, the file transfers were successfully tunneled through the various mix of intermediaries using multipaths, validating our flexibility and *eventual communication* goal claims.

6.7.2 Data Transfer Time

To tunnel data through the intermediaries, data has to be wrapped in KUMO protocol headers and then further encapsulated using those intermediaries' protocol headers. The increased header to data size ratio and additional processing overhead during encapsulation/decapsulation prolong data transfer time. Thus besides demonstrating flexibility, we need to measure the file transfer time over the intermediaries as usability proof. In the setup, we store 5 text files on the PL protected server (in US): RFC 1412 (4 text pages/7kB), RFC 1918 (9 pages/22kB), RFC 1738 (25 pages/51kB), RFC 1044 (43 pages/101kB) and RFC 1700 (230 pages/449kB). In our evaluation, we transferred these files over the 9 intermediary types (IRC, forum, OpenDHT, Flickr, hybrid-forum (IRC-forum), hybrid-opendht (IRC-OpenDHT), hybrid-flickr (IRC-Flickr), hybrid-s3 (IRC-S3), I3). The limited deployment of certain intermediaries, e.g., Flickr and forums, forces us to vary the locations of the KUMO clients instead of the intermediaries, in order to test under various network conditions. In total, we ran the file transfers: 5 (different file sizes) * 9 (intermediary types) * 20 (different KUMO client locations) = 900 times.

The plot shows the average transfer time of each file size over each intermediary type from 20 different client locations. To preserve visibility details of the transfer time of the rest of the intermediary types, we do not show the transfer

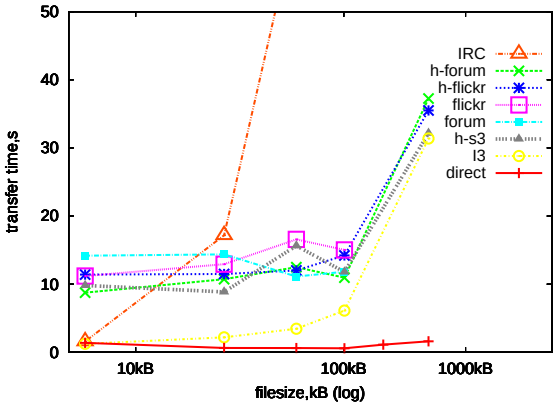


Figure 6.3: The average transmission time of various files sizes through different intermediary types.

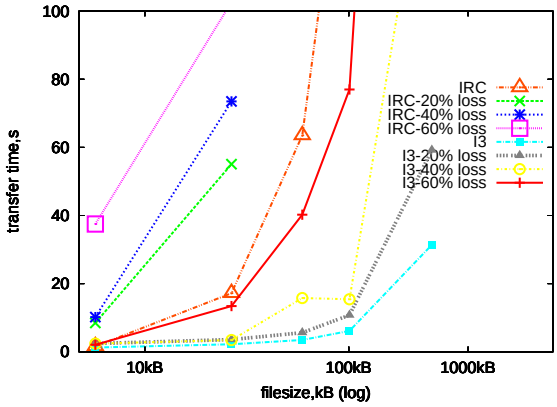


Figure 6.4: The average transmission time of various files sizes through I3 and IRC when different percentages of multi-paths fail due to congestion.

times using OpenDHT because they are a magnitude slower than the rest. As shown in Figure 6.3, the average transfer time using I3 is *timely* ($<$ 5s), i.e., comparable to direct transfer except at file size 449kB. With a 449kB file, a large portion of the time is spent in the still unoptimized KUMO stack *fragment reassembly* code. We verified this in a separate experiment where we measured the data transfer time without reassembly and the transfer completed within 13 seconds. IRC, hybrid-forum, hybrid-flickr, hybrid-s3 can be used to achieve *eventual* data communication within 10 seconds for file sizes up to around 101kB.

In conclusion, for fast data transmission of small pieces of data around 10kB, e.g., incremental data retrieval in AJAX communication, command/control data transmission in SOAP messages, etc., I3 and IRC are ideal intermediaries while for fast transmission of data around 101kB, only I3 is appropriate. For *eventual* communication of data around 101kB, such as airline reservation or personal banking information, hybrid-forum, hybrid-flickr and hybrid-S3 can satisfy the requirements. Even with the few types of intermediaries tested, we show that KUMO can satisfy a range of data communication requirements. As future work, we strongly believe that by porting KUMO from Ruby to C, optimizing the code, and fine-tuning the various timers in the KUMO protocol stack, e.g., retransmission timers, KUMO will be more efficient in catering to the various data

communication needs.

6.7.3 Multipath Data Transfer Under DDoS

Under the duress of DDoS, a portion of the multipaths utilized by a KUMO client-server pair may drop packets. In this experiment, we explore the scenario where 20%, 40% and 60% of the multipaths become completely unusable by completely dropping all packets traversing them. We only tested IRC and I3 intermediaries because only these intermediary types have significant transfer time performance gain through multipath utilization due to their small packet payload size. For other intermediaries, sending the entire data through a single connection while relying on the intermediary's protocol or TCP is more efficient.

As expected, the increase in packet loss and prolongs the data transfer time (see Figure 6.4). However, for small data sizes (up to 7kB), both IRC and I3, exhibits decent performances (below 10 seconds) for losses of up to 40%. Moreover, for I3, the increase in transfer time for up to 101kB data size, is well within 10 seconds. Due to the time taken and data retransmission volume required over IRC, we do not carry out data transfer beyond 22kB. For the same reason, we totally avoid the experiment with 80% of the multipaths impaired. We can safely conclude that KUMO offers *eventual* communication even in the extreme cases of 60% data loss.

6.7.4 IRC intermediary stability

Stability of intermediaries is crucial for KUMO's functionality. Of the intermediaries that we tested, all but IRC and I3 are expected to have stable existence due to their commercial nature. I3 nodes reside on Planet Lab so they inherit its stability. Therefore, in this experiment, we only measure the stability of IRC intermediaries. We cull the IRC server hostnames from the list at *netsplit.de*, which contains a comprehensive list of IRC servers under all domains and perform domain name resolution for the hostnames to eliminate non-existent servers. We scripted a Ruby program based on SIL (96), which is a generic banner grabber, to connect to each IP address and check for IRC *NOTICE* messages that are part of IRC connection establishment protocol. This script is run only once on each

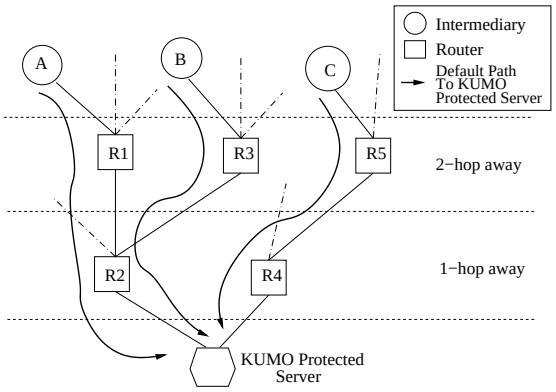


Figure 6.5: Path diversity measurement in relation to number hops away from KUMO protected server.

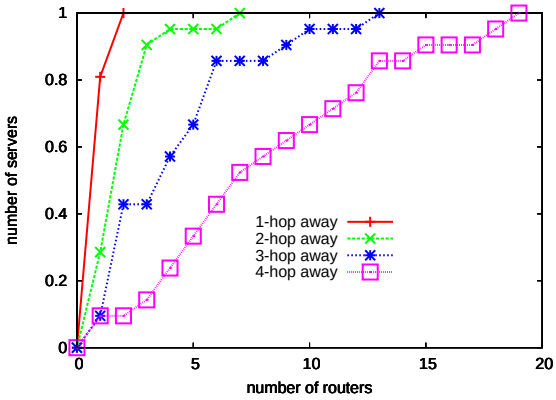


Figure 6.6: Improvement in path availability when intermediaries are utilized from the perspective of the server at various hops away.

of the 20 PL nodes with at least a 24-hour gap between the runs. This enables us to measure the *stability* of live IRC intermediaries over 14 days (1 Oct to 14 Oct 2008). We ran the script from different nodes so that IRC servers that are very restrictive on locality of clients they serve can be reflected since they may not be good intermediary candidates. The availability of IRC servers over the 2-week period is relatively consistent, around 1500 servers.

6.7.5 IRC intermediary path diversity

We are interested to show that having widely distributed intermediaries available to choose from increases the path diversity to a protected server. Therefore we attempt to measure the path diversity that is offered by IRC intermediaries since among the intermediary types we used, IRC has the largest base, so it is a good indicator of future path diversity should intermediaries become easily available. With direct connectivity, at any point in time, a KUMO client can reach a protected server only through a single Internet default path. However, with intermediaries available to bounce traffic, the path diversity/availability to reach the protected server increases. As a metric for comparing path availability, we analyze the number of distinct routers, d, through which the intermediaries can offer a path to the protected server n hops away from the protected server

where n=1 to 4. In Figure 6.5, if A and B are our intermediaries, even if can use either of them to bounce traffic, at n = 1-hop away, the path diversity to protected server is still only 1, i.e., only one router (R2) that can lead to the protected server. If our intermediaries are B and C, now at n = 1-hop away, the path diversity has increased to 2 (R2 and R4). Thus, d is a good measure for path diversity with reference to n. Without access to the intermediaries, we cannot obtain the traceroute of the default path from them to the protected server. To "predict" the path, we carry out the following steps: (1) We traceroute from the protected server to a selected intermediary with undetermined hops (hops marked by "*") removed. (2) Assuming that step (1) produces a traceroute of k hops, with the intermediary itself being the k-th hop, we utilize *iPlane* (67) to predict the reverse path starting at the k-th hop to the protected server. (3) If prediction fails, we make iPlane predict a shorter reverse path by starting from the $(k-1)$-th hop. (4) We repeat (3) until iPlane successfully or fails to predict the reverse path. It is unlikely that iPlane can predict the complete reverse path from an intermediary to the protected server but we always end up with a reverse path starting from the intermediary's ISP point-of-presence to the protected server. Given that the intermediary is likely single-homed to its ISP, the last few hops of our traceroute from a protected server to an intermediary is probably the same as first few hops of the reverse path from intermediary to protected server so we can safely conclude the reverse path starting from an intermediary's ISP predicted by the iPlane reflects the path taken by traffic originating from the intermediary.

In this experiment, the 20 PL nodes are used as protected servers. From Figure 6.6, by observing the x-axis where *number of routers* = 1, i.e., the number of routers that can lead to a PL protected servers when no intermediaries are used, as a reference point, we can see that at 1 hop away, employing intermediary increases the availability of 20% of the PL protected servers by twice while at 2 hops away, the availability is increased for 70% of PL protected servers by $2-7$ times. Overall, the presence of intermediaries does indeed increase path availability to a protected server significantly.

6.8 Discussion

6.8.1 Marketplace

With incentives at stake, parties may be interested to act as intermediary providers. We foresee 3 types of providers: (1) those offering their existing operational servers to defray the cost of their over-provisioned peak load capacity, (2) non-profit organizations whose sustenance has been based on charity and (3) those who feel that offering high-bandwidth and low-latency intermediaries is a profitable business model. The last group will be keen to implement next generation intermediaries, e.g., I3, AI-RON-E (61) (where routers are capable of bouncing traffic) or SaaS-type intermediaries that are specifically designed to possess superior traffic forwarding capabilities.

Brokers will enter the eco-system to harness the islands of intermediaries to form KUMO clouds with sufficient mass for filterable DDoS defense and offer it as a service to consumers. Protected servers can choose from the array of brokers who may differ in terms of KUMO cloud capacity, intermediary types, pricing scheme and value-added services, e.g., availability of a large base of KS-side servers, which enables protected servers to attain KUMO protection without even having to worry about KS-side installation.

Even though some other DDoS schemes, e.g., Phalanx, may arguably offer similar incentives and spawn similar marketplaces, they require additional components to be introduced onto systems that wish to act as intermediaries giving rise to 2 sticking points with adopters that KUMO does not: (1) security policies and IT governance forbid installation of additional components on corporate systems that do not contribute directly to business objectives and (2) installation of additional DDoS mitigation components requires additional expertise and cost in installation, maintenance and troubleshooting.

6.9 Limitations

Defenseless Against Non-filterable DDoS As noted, KUMO enables a protected server to control traffic reception to prevent filterable DDoS. How-

ever, during a non-filterable DDoS attack, the inability to differentiate traffic from different origins leads to a server offering services based on a first-come-first-serve basis. Such a scheme marginalizes legitimate clients since the large number of zombies will reduce the probability that a particular client can obtain an available channel from the KUMO name server before the zombies do, to almost negligible. Having the ability to somehow differentiate between traffic sources is critical for defense against non-filterable DDoS.

6.10 Future Work

We intend to tune the KUMO stack so that most performance issues can be ironed out. On top of that, porting the stack to Java is a priority since it enables developers to create zero-install JWS clients or Java Applets that their customers can use and gain protection against DDoS disruption transparently. It is also a necessity to incorporate a mechanism for servers to distinguish between traffic sources to defend against non-filterable and eDDoS.

6.11 Conclusion

We present KUMO, a DDoS mitigation framework that facilitates adopting any existing or future Internet system as an intermediary to achieve an intermediary pool size sufficient to withstand a filterable DDoS attack. We envision that KUMO's unique ability to utilize servers without modification and its incentive scheme can attract more participants than previous DDoS mitigation schemes that require component installation, therefore entailing maintenance effort and possibly violating a system's security policies. With the emergence of faster and more varied SaaS platforms, we believe that KUMO can play an important role to unite all these over-provisioned resources for defense against DDoS.

Chapter 7

sPoW

This section is largely adapted from my paper co-authored with Akihiro Nakao entitled "sPoW: On-Demand Cloud-based eDDoS Mitigation Mechanism" published in IEEE/IFIP Hot Topics on Dependable Systems Workshop held in conjunction with Dependable Systems and Network (DSN) in Estoril, Portugal, Jul 2009 (acceptance rate 37%).

7.1 Deployable Resiliency

7.1.1 Resiliency

Defense Category	Filterable DDoS	Non-filterable DDoS	eDDoS (intermediary)	eDDoS (server)
Client/Server React	Yes	Yes[a]	Yes[a]	Yes[a]

[a]Prioritize traffic base on resource expended by client instead of distinguishing good/bad traffic.

Table 7.1: Resiliency of sPoW against DDoS

sPoW offers protection against DDoS from the angle of client and server reaction collaboration.

Client/Server Reaction: Request Urgency Signal and Prioritization On

the server-side [1], we actively maintain a set of obscure channels to handle connection requests with varying priorities based on the protected server's request load. A puzzle is created to conceal a channel's property information, such as its location, with a difficulty proportional to the channel's priority. A client requests for a server-side generated puzzle by specifying the destination server and a puzzle difficulty level, which reflects its connectivity establishment urgency. The puzzle difficulty determines the amount of resource a client has to expend to solve it, in order to recover the obscure channel location from the puzzle. This mechanism enables a server to prioritize traffic based on resources expended by a client, instead of distinguishing bad/good traffic using resource intensive and inaccurate bad traffic classification technology.

As shown in Table 7.1, sPoW can bypass filterable DDoS because it adopts KUMO framework (see Section 6) to enlist resource-rich cloud infrastructures as its intermediaries, i.e., traffic reception control point. With just a few cloud intermediaries, we can harness sufficient resource for filterable DDoS defense. In order to circumvent the thorny issue of distinguishing good/bad traffic with minimal false positives, which becomes almost impossible in the case of non-filterable DDoS, sPoW proposes to utilize PoW (Proof-of-Work) to differentiate traffic based on the traffic originators' willingness to expend its own resources for the purpose of connection establishment. With PoW, due to the huge but yet finite resources of attackers, legitimate clients can establish connectivity within bounded albeit prolonged time (see Figure 7.4). Although enlisting cloud intermediaries for DDoS defense is an appealing proposition, it exposes a server owner to excessive cloud utilization triggered by attackers sending arbitrary traffic into the cloud, i.e., intermediary eDDoS, which gets forwarded to the server, i.e., server eDDoS. sPoW is, to the best of our knowledge, the first research to address both eDDoS through the use of self-verifying PoW; only clients that expend sufficient resource to recover obscure channels can send traffic through those channels in the cloud. Even so, there is no guarantee that all traffic from originators that expend resources

[1] Server-side refers to part of the server infrastructure not necessarily the server and it may not even belong to the server owner.

are non-malicious but the main goal of the mechanism is just to empower legitimate clients with the ability to compete with large swarms of zombies to reduce the amount of malicious traffic overworking the cloud intermediary and protected server resources.

7.1.2 Deployability

To combat DDoS, a mechanism needs to possess more aggregated resources than attackers; aggregated resources can be harnessed from multiple distributed moderately provisioned locations (11; 13; 18; 24; 32; 38; 48; 59; 60; 62; 65; 73; 82; 83; 105; 123; 126) or few localized well-provisioned clusters (63). The majority of research adopts the former approach, clearly reflecting the opinion of the community that harnessing sufficient localized resource for specialized DDoS defense is neither cheap for end-users to unilaterally deploy nor such endeavor is profitable for ISPs to build a service. sPoW debunks this trend by using utility-based cloud computing to overcome the deep-rooted issue of cost and deployability in 3 ways: (1) the use of just a few cloud infrastructure is sufficient for DDoS defense thus easing unilateral deployment without relying on ISPs or other parties for cooperation, (2) protected servers pay for only cloud resource that they use, reducing the initial cost outlay substantially and (3) self-verifying PoW enables the protected server to set a peak resource utilization, which is similar to other existing mechanisms, but with the unique property that sPoW ensures the resource utilization is allocated as much as possible in favor of legitimate clients. The complex issues of preventing a protected server's hidden location from being leaked and accounting can be side-stepped because cloud computing platforms are built by trusted organizations who are highly motivated to prevent such leakage and are equipped with their own comprehensive accounting system.

7.2 Assumptions

Resilient Naming Service sPoW relies heavily on a naming service. It can utilize existing DNS by just making some changes to the authoritative server (see Section 7.9 for details). This implies that sPoW's resiliency is no worse

than the existing Internet. For improved resiliency, existing research such as CoDoNS (89) or ConfiDNS (85), can be adopted.

Rich Cloud Computing Resource Each cloud infrastructure is very-well provisioned because it is a shared environment for multiple customers. By using a single cloud or an aggregate of a few, we assume that it is possible to harness sufficient resource to thwart even the largest of DDoS attacks.

Hidden Server Location Enforcement A cloud is utilized as an intermediary to receive traffic on behalf of a protected server in order to grant the server the ability to accept traffic from clients that have expended sufficient resource to discover the obscured channels the server is listening on at the cloud. Since the server can rely on only one or a few of these clouds for protection and these clouds belong to large trusted entities, e.g., Amazon, Google, etc., the hidden server location can be well-preserved, preventing attackers from discovering its location and attacking it directly.

7.3 Overview

Elastic cloud computing (9; 46) promises an affordable and highly dependable computing infrastructure that is easy to setup, maintain, and scale. These value propositions have spurred many businesses to adopt cloud computing technology (8). However, this dream setup can become a nightmare during a DDoS when the cloud adopter must pay for resources used, whether by a server to handle a large amount of undesired traffic, or by a defense mechanism to filter them. Either way, attack packets stretch the cloud elasticity resulting in an economic DDoS (eDDoS).

For cloud computing to remain attractive, our key goal is to combat the *yet* unaddressed eDDoS threat (5; 52; 106) by proposing sPoW, a unilaterally deployable and affordable on-demand cloud-based eDDoS mitigation mechanism. sPoW is designed to mitigate eDDoS triggered by both filterable and non-filterable attack packets, which can occur at either the network or application layer. Today's cloud billing mechanisms are based on network-layer traffic handling; if a cloud can determine how to forward a packet to its destination server by observing the

network-layer headers, it will do so even though a packet is malicious, unsolicited, or malformed at the application layer, and billing will occur. Two strategies to decrease the impact of eDDoS are to (1) detect and filter DDoS packets before they trigger a cloud's billing mechanism by mere observation of the packets' *network-layer* headers and where it is impossible to do so (2) employ a system that enables packets' origins to compete with each other for the servers resources by expending their own resources to generate and embed a "signal" within their headers, a system generically known as Proof-of-Work (PoW). A signaling system increases the efficiency of a server's resources; legitimate clients, when denied service, will attempt to generate stronger signals in order to receive higher priority, which has the impact of reducing DDoS traffic getting through to the server. However, unlike existing signaling systems that require resources to verify those signals, which in turn makes them a DDoS or eDDoS target sPoW is the first system, to our knowledge, that can prevent eDDoS traffic from incurring any costly cloud resources during verification of a packet's signal strength.

Our work extends previous work on capabilities (13) and PoW. Capabilities grant *existing connections* the highest priority to server resources because they were once *initial connection* requests but they successfully competed for the right to the server resources through PoW. With PoW, a client expends its resources to solve a "crypto-puzzle" and submits proof of the solution (the signal), which also indicates the puzzle difficulty, together with its initial connection request thereby enabling the PoW system to verify the puzzle correctness and prioritize the request, based on the puzzle difficulty, in a queue that forwards requests to the server. PoW enables initial connections from legitimate clients and zombies to compete for the resources remaining from existing connection handling.

During DDoS, legitimate clients will expend more resources to solve more difficult crypto-puzzles to elevate their initial connection requests' priority at the expense of longer connection request generation time. An attacker, with huge but finite resources has 2 options: (a) expend more resources to generate even higher priority connection requests to drown those legitimate clients but sacrifice the *quantity* of requests it can generate, resulting in other possibly lower priority legitimate clients "sneaking" access to the server, or (b) maintain its

current connection request priority to continue to generate the maximum *quantity* of connection requests but permit those higher priority legitimate requests to gain access to the server. Either way, the introduction of capabilities and PoW, empower existing connections and legitimate clients' connection requests to compete effectively to reduce the portion of non-filterable eDDoS connection requests that reach the server while increasing their own. The initial connection is set up only once per client-server pair and subsequently, it can be used to send multiple application protocol requests and receive corresponding replies between the pair. Refer to the explanation after Step 10 in Section 7.6 for details.

Capabilities and PoW are known to work handily against DDoS but they have yet to be made deployable and affordable due to their lofty requirements such as making modifications to proprietary existing core systems spanning multiple parties in order to incorporate new features nor have they been constructed to handle eDDoS. Our contribution is using capabilities in an innovative way to fit the cloud platform and designing a conceptually new PoW—a self-verifying PoW, to build a first-of-its-kind, immediately and unilaterally deployable, cloud-based mitigation platform for eDDoS.

7.4 Design Goals

Unilateral Deployment It must be reasonably affordable for a single party interested in protecting its servers against eDDoS and DDoS to deploy sPoW unilaterally.

On-Demand eDDoS Mitigation The cloud adopter should not incur significant infrastructure setup or maintenance costs under normal circumstances while being able to scale easily to meet with traffic demands. During an eDDoS attack, the cloud adopter should only incur the cost from legitimate traffic as much as possible.

Network-level eDDoS Defense An on-demand eDDoS mitigation mechanism must be capable of distinguishing as much network-level eDDoS traffic as possible and drop them to avoid billing the adopter for it.

Non-filterable eDDoS Defense Non-filterable eDDoS traffic that is indistinguishable from legitimate client traffic cannot be filtered out accurately. Thus a scheme that enables a server to prioritize initial connection requests based on PoW and clients that are capable of demonstrating PoW are necessary. sPoW empowers legitimate clients to compete with and reduce non-filterable eDDoS initial connection requests that enter the cloud and trigger the billing mechanism.

Immediate Deployability To make sPoW attractive, we want it to be deployable on existing cloud infrastructure without requiring any changes.

7.5 Architecture

We describe the architecture used to fulfill the design goals in Section 7.4.

Localized Cluster(s) DDoS mitigation is most effective when deployed at multiple heterogeneous points as DDoS can be stemmed the soonest possible at those points leaving downstream resources unclogged. Such an approach, however, entails numerous deployment points as well as substantial cooperation among multiple parties with possibly conflicting incentives and priorities. sPoW adopts an alternative approach similar to Phalanx (38) and KUMO (Section 6), which trades-off the ability to eliminate attacks near sources for unilateral deployment by clustering bandwidth and resources at a single or a few localized sites to withstand the largest of DDoS onslaught expected. The constant drop of CPU and bandwidth prices has made such a unilateral approach appealing. However, the similarity ends there, unlike Phalanx, we do not require crypto-puzzle verifiers, to be installed anywhere. Further advantages of not requiring verifiers are elaborated in Section 7.5. In Figure 7.1, sPoW is deployed as a localized cluster in Cloud #1 to protect servers in cloud, e.g., Cloud #2, or other servers in general. It can also be deployed as a multi-site localized clusters on a few clouds concurrently.

Cloud-based DDoS Mitigation Mechanism A constant problem with DDoS mitigation proposals is the cost in acquiring a large deployment base and

the trustworthiness of the deployed base. A cloud-based DDoS mitigation mechanism removes the need for an initial expensive infrastructure outlay, increases the trustworthiness of the deployed base due to its controlled environment as well as ownership by trusted providers and is easier to grow since a general purpose computing cloud is flexible in meeting customer needs thus easier to share and has a wider customer base thereby achieving economies of scale, which drives down the cost, faster than a shared DDoS specific infrastructure.

Ephemeral Server Channels To prevent network-level flood from chalking up server utilization, sPoW mediates initial connections to the server through *ephemeral server channels*—a short-lived (single initial connection request serving) varying set of channels that connects to the server. A channel is just a named entity, e.g., a message queue in a cloud, where data can be stored and prioritized before being forwarded/retrieved to/by the server. A client needs to find a server channel through which it can communicate with the server. It does so by querying a sPoW name server that functions like a Domain Name Server (DNS), which returns the connectivity details of a server associated channel when a server hostname is provided. The constantly varying server channel set protects the server from persistent network-level eDDoS that usually involves sending attack packets to a server's single IP address. With sPoW, the attacker is forced to (1) mimic legitimate clients by querying a sPoW name server continuously to seek out valid server channels in order to send attack packets or (2) spread its attack over the entire channel identity namespace. The former results in an non-filterable eDDoS and is handled using *capabilities and self-verifying PoW scheme* described next. The latter means sPoW successfully transforms network-level eDDoS into a packet pattern matching filtering problem; a perimeter that can drop traffic whose destination is not within the server channel set identities is sufficient to eliminate most network-level eDDoS before the billing mechanism is triggered. However, some brute force traffic does get through and incur utilization cost. The trade-off of using a larger channel identity namespace, which increases packet header-to-data ratio

but reduces the network-level eDDoS that incurs billing is a subject left for future work. Server channels can be implemented at any OSI layer as long as there is sufficiently large namespace for server channel names at that layer to obscure server channels. For example, at the *application-layer*, we can utilize names of message queues of Amazon's SQS (6) or OnlineMQ (81) to represent server channels; only data sent to message queue names setup as server channels will be forwarded to the server and billed to the adopter while the rest is dropped without charges. For other alternative server channel implementations, refer to Section 7.9.1.

Capabilities and Self-Verifying PoW Scheme To address non-filterable eDDoS, we need to prioritize traffic. We adopt a two prong approach. First, we utilize capabilities, to prioritize existing connection traffic over initial connection request traffic and secondly, we prioritize among the initial connection requests using self-verifying PoW. The two prioritization schemes empower existing connections and legitimate client initial connection requests to compete effectively and reduce non-filterable eDDoS connection requests that triggers undesirable billing mechanism in the cloud.

Existing connection traffic is afforded higher priority than initial connection requests because the former is more "trusted"; it originated as an initial connection request and became an existing connection after successfully expending its resources to compete for the right to access the server. Unlike the original capabilities, we do not use cryptographic tokens agreed between a client-server pair to mark existing connection traffic, instead, the traffic is carried over established one or more *communication channels*. whose identities are privy only between the client-server pair. Note that a *communication channel* is different from a *server channel* because the former is used for existing connections and each channel identity is privy to only a client-server pair while the latter is used for initial connection requests and its identity is openly available to any client that queries a sPoW name server (see Figure 7.1). The privy of a communication channel identity serves the same purpose as the secrecy of a capability token; it identifies traffic associated with that existing connection and minimizes possibility

that attackers can spoof traffic claiming to belong to that connection resulting in unwanted traffic (eDDoS) being transmitted to the server. By default all existing connections have the same priority but an application can assign priorities based on the activity carried out over those connections, e.g., casual browsing traffic should have lower priority than purchase transaction traffic, by communicating the priority assigned to each existing connections (using channel names) to sPoW. Note that at any time, an existing connection can be terminated by the server if it is deemed harmful or needs to make way for other higher priority connections.

Prioritizing among *initial connection requests* is trickier since they can come from anywhere, including malicious sources thus we need a PoW scheme where all clients' initial connection requests are prioritized based on the resources each expended in solving "crypto-puzzles" with the difficulty level reflecting the resources expended, which in turn signals each client's connection establishment urgency. Unlike conventional DDoS PoWs (38; 83), where the clients solve crypto-puzzles and submit solutions for each initial connection request followed by solution verification, prioritized queuing and forwarding to the server by some *PoW verifier*, sPoW does not need verifiers. It is self-verifying because the crypto-puzzle consists of both the encryption of a server channel connectivity details, e.g., IP address, port number, etc., and the partial encryption key (the encryption key that can be used to recover the channel details but with k bits concealed). By brute-forcing these k bits, which is also the puzzle difficulty indicator, a client discovers the server channel where it can submit an initial connection request, which is then queued for forwarding based on the puzzle difficulty associated with that server channel. The subtle difference of not requiring a verifier and delegating verification to client-side offers sPoW 2 advantages: (1) the cloud infrastructure is freed of verification computations and more importantly, (2) we can be reasonably sure that each initial connection request is submitted by a client who has expended enough resources to successfully discover that server channel and during DDoS, as explained in Section 7.3, with PoW, the origin of this traffic is less likely to be attacking agents due to fierce competition from legitimate clients thus reducing the

probability that traffic entering the cloud is malicious. Contrast this to conventional PoWs that have to implement verifiers in the cloud resulting in un-verified traffic entering the cloud in order to be verified, which incurs cloud billable resources thus subjecting those PoWs to eDDoS. For (1), we would like to point out that the main purpose is to eliminate the ability of attackers to ramp up the cost incurred by puzzle verification and to some extent save processing resources in the cloud. Although compared to non self-verifying PoWs that leave puzzle generation to the clients by just offering them seeds to guide unique and time-limited puzzle creation, sPoW needs to generate puzzles for clients, thus consuming slightly more resource but we argue that overall, self-verification consumes less resource since a generated puzzle is re-used until a client/zombie solves it, i.e., it cannot be excessively triggered by zombies, while non self-verification schemes can be over-worked by zombies relentlessly submitting incorrect puzzles.

Existing Technologies We have carefully selected the technologies used to achieve each of our design goals to ensure that they can be easily supported out of existing cloud platforms for immediate deployment.

7.6 A Walkthrough

To show how the different pieces of technology work together to deliver an eDDoS mitigation mechanism, we provide a walkthrough here. In order to use sPoW, we need client-side and server-side components. The server-side component shields sPoW details from a server in order for it to utilize sPoW unmodified. The client-side can be implemented as a zero-installation Java Applet or Java Web Start (JWS) (109) client and embedded in a static web page hosted on resilient content distribution networks (CDN) (4; 45; 120). It is dynamically and transparently downloaded by users visiting the web page of sPoW-protected servers. The server-side component can be installed on a server itself or on a disparate system. In our description below, each number in parentheses corresponds to the step number in Figure 7.1.

A client that wants to contact the server performs a DNS resolution (1) to obtain the location of the client-side component on the CDN and proceeds to download it together with the server-side component's public key, which can be retrieved using HTTPS (2). The client-side component then performs a sPoW name resolution specifying the server hostname and the puzzle difficulty, k, to obtain the crypto-puzzle for the destination server (3). The sPoW name server forwards the puzzle request to the server-side component handling puzzle generation for that domain name (4). The server-side component randomly creates an ephemeral server channel (5), encrypts the channel details and sends back both the encrypted details as well as the encryption key with k bits undisclosed as the crypto-puzzle (6). Note that under certain circumstances, the server-side component may return an existing puzzle, instead of creating a new one. See Table 7.2, which is discussed in Section 7.7.3 for details. The client-side component brute-forces and recovers the server channel details and submits an initial connection request (7), which includes a randomly generated shared key, encrypted using the server-side component's public key to protect its secrecy, through that server channel. If its initial connection request is not established within a timeout period, it can request for a more difficult crypto-puzzle and re-submit the connection request through the higher priority channel. When the server receives the initial connection request (8), it creates a communication channel (9), encrypts the channel details using the client generated shared key and sends the information back to the client-side component (10). Communication between the client-server component pair can proceed using the established communication channel. Destination server application protocol-specific commands sent by a client-side component terminate at the server-side component after traversing the communication channel. The server-side component echoes the command to the target application through connections that it initiates as an immediate client of the application. The application replies also terminate at the server-side component before they are echoed back to the client-side component. Thus the communication channel is a one-time setup for each client-application pair that can handle multiple commands/replies, e.g., if a client-side component is communicating with a HTTP server, throughout the entire HTTP session spanning

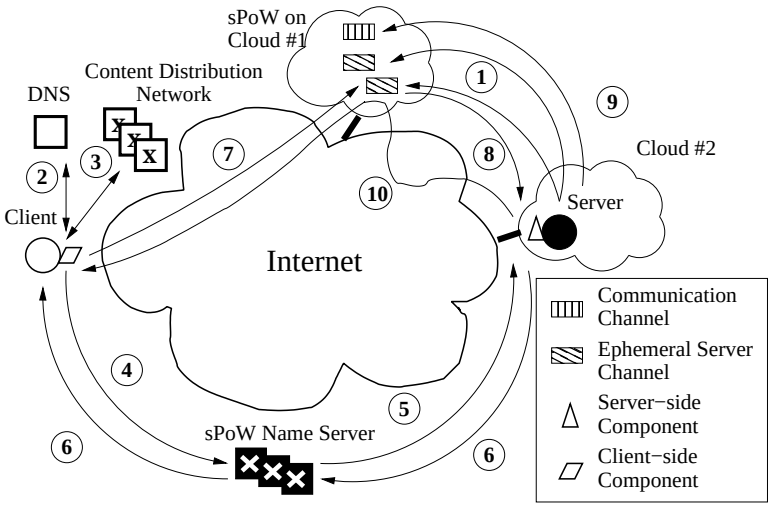


Figure 7.1: A walkthrough of how a client can communicate with a server protected by sPoW.

multiple HTTP commands, multiple connections are created between the server-side component and the HTTP server but only one communication channel is used to send/receive data between the client-side and server-side components.

7.7 sPoW Implementation

sPoW utilizes the KUMO framework (Section 6) to harness cloud platforms as intermediaries where communication and server channels are created. Architecturally, sPoW adopts from KUMO, but with a few important changes/additions—the puzzle generator and puzzle requester/solver replaces the KUMO DNS updater and KUMO DNS resolver respectively with the connection manager as an addition. Figure 7.2 shows how the sPoW components are added on top of the KUMO framework (compared to Figure 6.1). The client-side and component-side is implemented by KUMO KC-side and KUMO KS-side respectively. In this section, we provide a big picture of how the components tie in together before discussing about possible ways the system can be attacked and finally explaining how the components work together to deal with the attack.

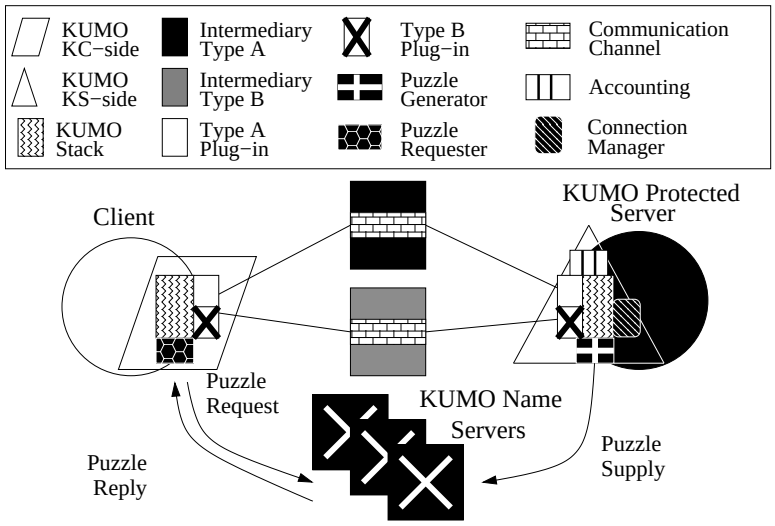


Figure 7.2: The sPoW architecture adopts after KUMO with the exception of replacing the KUMO DNS Updater with Puzzle Generator and KUMO DNS Requester with Puzzle Requester as well as adding a Connection Manager to the server-side component.

7.7.1 Big Picture

Connection Manager Manages the channels that a server is listening for initial connection requests. This is a critical function because a server has limited capacity and to defend against DDoS, the manager needs to ensure the channels, with the help of self-verifying PoW, are used to serve mostly legitimate client requests, instead of being hogged by zombies. It is also responsible for creating and maintaining multiple communication channels between each client-server pair after a client's initial connection request has been handled.

Puzzle Generator Generates a puzzle to protect the connection server channel by encrypting the channel information, e.g., the channel IP, port, name, etc. The encrypted data and the encryption key with k bits concealed together are referred to as a puzzle. A client that receives a puzzle has to expend resources to brute-force the k hidden bits to retrieve the channel information. This concealment determines the puzzle difficulty since each additional bit concealed increases the resource expended by a client to retrieve the request channel. In this study, the terms puzzle difficulty and

channel protection level are synonymous—the latter refers to the difficulty of the puzzle created to conceal the channel information.

Puzzle Distributor The puzzle distributor stores the generated puzzles indexing them using the hostname those puzzles are created for. Puzzle distribution is de-coupled from generation because the generator, which coordinates its generation activity with the stateful connection manager algorithm that keeps track of all server connections, is difficult to replicate, and may become a single-point-of-attack. A puzzle distribution system, whose purpose is restricted to storing hostname-to-puzzle mapping is easier to replicate, and thus is used to hand out puzzles while concealing the puzzle generator from attacks. The KUMO name servers can play the role of puzzle distributor without modification.

Puzzle Requester/Solver A puzzle requester running on the client-side obtains a puzzle by specifying the hostname and puzzle difficulty level in its puzzle request sent to the puzzle distribution system. The requester then expends the client resources to solve the puzzle and submit its connection request through the channel whose information is retrieved from the puzzle. The requester is responsible for retrying connection request submission until its request is processed by the server. Each retry involves using an algorithm to select a puzzle difficulty that improves its chances of connection establishment.

7.7.2 Attacks

The cloud intermediaries, connection manager and puzzle generator are key components in sPoW that protects a server from DDoS thus they themselves will now have to bear the brunt of attacks. We assume that cloud intermediaries are resource-rich thus are not susceptible to attacks. In this section, we will elaborate on attacks on the other two components.

PoW Violation with Proof Sharing If clients can share a puzzle proof then the main concept of prioritizing client requests based on resources expended will be violated; clients can obtain high priority service by resubmission of

someone else's proof without expending any resource. PoW schemes need to protect against proof sharing.

Puzzle Generation Resource Exhaustion Zombies can request for a lot of puzzles without any intention of solving them. This leads to 2 types of resource exhaustion—processing power (CPU) and/or network connectivity (memory/storage/configuration limit). Unlike conventional PoWs (38; 83), where clients themselves generate the puzzles based on seeds/random numbers provided by the server-side, in sPoW, the server-side itself is responsible for puzzle generation, thus is susceptible to puzzle generation resource exhaustion attack. Therefore, the "low resource consumption" task of puzzle request performed by many zombies can easily overwhelm the server-side CPU resource required for "high resource consumption" puzzle generation. Moreover, since each puzzle leads to a channel, through each puzzle acquisition, zombies can hog those puzzles' channels by not solving those puzzles nor submitting any request through those channels, resulting in the server-side waiting in vain and tying up network connectivity resources, i.e., the memory/storage or configuration limit, required to maintain each channel.

Puzzle Level Inflation If the connection manager simply replaces lower protection level channels with higher ones when its network connectivity resource runs out in order to favor high priority clients, zombies can inflate puzzle difficulty by merely requesting for the most difficult puzzles. This results in clients having to solve un-necessarily high puzzles to submit connection requests.

In this study, we also use the term *puzzle accumulation attacks* to refer to both resource exhaustion and puzzle level inflation attack.

7.7.3 Connection Management and Puzzle Generation Algorithm

In this section, we describe the details of connection management and puzzle generation algorithm, and how they are designed to thwart the described attacks.

For our self-verifying PoW, a server needs to maintain connectivity to its server channels or keep track of them to periodically poll them for connection request arrivals. This consumes the server's finite resource, so server channels are limited at a any point in time. Under normal circumstances, each initial connection request can be given a unique puzzle that leads to a unique server channel. However, during non-filterable eDDoS, puzzle requests will exceed the server channels. It becomes necessary to hand the same puzzles of previously created server channels, which have yet to receive any initial connection requests, as replies to multiple puzzle requests. Server channel sharing results in only the quickest puzzle solver being successful in initial connection request submission. Frustrated clients can choose to solve increasingly difficult puzzles until their connection requests are handled. As previously noted, given that an attacker's resource is finite, at a certain difficulty level, the attacker will no longer be able sustain the DDoS on all server channels enabling some legitimate clients to prevail. Note that, at all times, the puzzle to channel mapping always remain one-to-one, thus we use the term puzzle/channel interchangeably, e.g., creation of a puzzle implies the creation of a channel, etc.

Furthermore, the algorithm needs to take into consideration attacks described in the previous section 7.7.2. Puzzle sharing can be ameliorated if channels are *ephemeral*; each channel will close upon reception of a initial connection request and its resources will be freed up for new channel creation. Puzzle accumulation attacks can be mitigated by sharing request channels while puzzle inflation attack alleviation requires the algorithm to keep track of puzzle resolution rate for each puzzle difficulty level in two consecutive periods and only permitting new puzzles to be generated at that level if the puzzle resolution rate for that level in the previous period is similar or higher than the puzzle request rate of that level in the current period, i.e., almost all previously requested puzzles need to be solved before sPoW permits generation of new ones. In short, the connection management algorithm's role here is to determine (a) when a new channel should be created, (b) when and which channels should be shared, e.g., oldest and least shared channel, (c) which channel should be dropped when a request for a higher difficulty puzzle arrives, e.g., oldest and lowest difficulty channel, and (d) when a channel should be closed.

Event	Significance	Puzzle Management
A) Channel request received at a channel with protection level k	Puzzle at difficulty k solved	Close the channel after single use to avoid puzzle sharing attacks
B) Puzzle request for $k<k_{lowest}$	Puzzle request from client not very determined to reach protected server	If capacity exceeded, drop request, otherwise create new channel
C) Puzzle request $k\geq k_{lowest}$, $s_{k,t-1}=r_{k,t}$ and $O_k=0$	No channel at k yet	If capacity exceeded, drop oldest and k_{lowest} channel to acquire new channel for puzzle generation at k, otherwise create new channel
D) Puzzle request for $k\geq k_{lowest}$, $s_{k,t-1}\geq r_{k,t}$ and $O_k\leq s_{k,t-1}$	Puzzles at k are solved as quickly as they are handed out	If capacity exceeded, drop oldest and k_{lowest} channel to acquire new channel for puzzle generation at k, else create new channel
E) Puzzle request for $k\geq k_{lowest}$ but $s_{k,t-1}<r_{k,t}$ or ($s_{k,t-1}=r_{k,t}$ and $O_k>s_{k,t-1}$)	Puzzles for k are requested faster than they are resolved indicating possible puzzle accumulation attack	Oldest existing channel at k used for puzzle generation at k

Table 7.2: Connection manager events, their significance and puzzle manager reactions

Before proceeding, we define a few terms that facilitate the explanation of our algorithm. k is the difficulty level of a puzzle, which is also the protection level of the channel associated with the puzzle. The condition $0<k_{min}<k<k_{max}$ holds with k_{min} selected such that the effort expended to recover the channel information is non-negligible while k_{max} is dependent on the cipher key size, e.g., for Data Encryption Standard (79), $k_{max}=64$. We also define C as the server-side capacity catered for request channel handling (requests per second). The algorithm keeps track of the number of puzzle requests in the current period, $r_{k,t}$ where period$=t$ and number of puzzle resolutions in the previous period, $s_{k,t-1}$. The optimal length of each period, t, is left for future work. At any time, each of the channels may have different protection levels with k_{lowest} being the lowest protection level. Also O_k indicates the number of channels with protection level k.

Table 7.2 shows a summary of events at the connection manager, their signif-

icance and the manager's reaction. Upon puzzle request, the server-side acquires a server channel, creates a puzzle of the specified difficulty from it and hands it back to the puzzle requester. The server-side listens to that channel until a connection request is received, upon which the channel is closed (Event A). The puzzle handed out to clients are unique if the capacity, C, is not exceeded. Issues arise under adverse conditions of puzzle accumulation attack or non-filterable DDoS. In both cases, even after C is exhausted, new puzzle requests keep arriving. To handle DDoS, we abide by the principle to service connection requests based on k thus when a $k \geq k_{lowest}$ puzzle request arrives, we close the oldest and k_{lowest} channel and acquire a new channel to generate the new puzzle at k. Otherwise, if $k < k_{lowest}$, we just reject the puzzle request (Event B). To mitigate puzzle accumulation attacks, the algorithm ensures that the oldest and k_{lowest} channel is dropped in favor of the creation of a new higher or similar level one *only* if the $s_{k,t-1} \geq r_{k,t}$. In other words, a new channel replaces an older and lower priority one only if it can been proven that the combination of clients and attackers have the capacity to solve such a higher level puzzle if it is issued; the proof is demonstrated by the puzzle resolution rate of in the previous period. This rule is split into Events C and D with the former handling the special case of allowing at least one k channel to be created even if none has been solved previously, without which, no client can request for a higher level puzzle. To mitigate possible puzzle generation resource exhaustion attacks, existing yet-to-be-utilized channels should be re-used, i.e., their corresponding puzzles should be handed out as response to puzzle requests, when possible signs of such attack as elaborated in Event E occurs, namely, the puzzle request rate in the current period exceeds the puzzle resolution rate in the previous period.

7.8 Evaluation

As explained in Section 7.1.1, with sPoW, a legitimate client is guaranteed to establish connection within bounded time. The poignant question is "how much is the connection establishment time prolonged when DDoS occurs?" We attempt to answer the question by first performing a theoretical analysis, followed by

experiment-cum-simulations to validate our analysis. Before doing so we introduce the assumptions our analysis and experiments are based upon.

7.8.1 Model Assumptions

We assume that the cloud intermediary has multiple huge network pipes from various tier-1 ISPs resulting in rich path diversity. In a DDoS attack, the cloud network connectivity cannot be degraded because of its multiplicity as well as well-provisioned capacity, and neither can the paths from legitimate clients to the cloud be congested due to the rich path diversity and distribution of clients and zombies. This is a reasonable assumption because even in the current Internet, during DDoS, congestion often occurs not on the client-side but on the server-side, which lacks path diversity. With sPoW, however, the paths from the cloud to the server are unaffected by DDoS because the cloud intermediary is responsible for dropping unsolicited traffic. We also assume that the cloud intermediary has sufficient CPU resource to process all connectivity requests in its network queue to filter unsolicited ones fast enough so that the queuing delays do not dominate the connection establishment time. Since the cloud intermediary act as a traffic control point for a protected server—the server can control the number of requests forwarded to it or the server itself can control the rate of request arrivals by pulling them from the cloud; the limited server resources never becomes the bottleneck.

7.8.2 Theoretical Analysis

7.8.2.1 Worst Case Connection Establishment Time

Since in our assumptions, there is no network queuing delays or network bottlenecks that dominate connection establishment time, the bulk of a client's connection establishment time is incurred during puzzle resolution and awaiting server handling. In this section, we analyze the puzzle resolution time theoretically and as we will soon explain how that can represent a client's connection establishment time. Previous work has either not analyzed this metric (38) or the simulations

performed assume that all the requests that is successfully forwarded to the protected server can be handled, thus offering little insight into how puzzle resolution time is affected if the protected server capacity has limited capacity (83).

One of the best strategy an attacker can employ to prolong a legitimate client's connection establishment time is to utilize its resource to solve the most difficult puzzles to clog all channels (83). We assume that the attacker is lucky enough to obtain all C_{lower} distinct puzzle channels, i.e., lower of the two capacity that determines maximum puzzle generation rate of a server-side—connection request handling (since puzzles must be associated with a request channel) C_{net} (requests per second) or puzzle generation, C_{cpu} (puzzles per second). Thus, we are calculating the upper-bound of puzzle resolution time. In such a scenario, the attacker can concentrate N_{bot}/C_{lower} on a single puzzle where N_{bot} is the botnet size or the number of zombies. Assuming that puzzle-solving effort is linear, the puzzle-solving time is inversely proportional to the botnet size and a legitimate client has similar capacity to a zombie, then for a single legitimate client, the time taken to solve a puzzle of the same difficulty is N_{bot}/C_{lower}. Solving a puzzle with additional one-bit difficulty requires twice the effort, $2{*}N_{bot}/C_{lower}$. If a client solves a puzzle and its difficultly level, which determines its initial connection request order in the service queue, is high enough that the request can be serviced prior to its short timeout period, the connection establishment time will be approximately similar to its puzzle-solving time. The puzzle-solving time is dominant since the request's queuing time in the cloud intermediary is minimal and there is no bottleneck along the client-server path (see Section 7.8.1). Assuming that connection request handling is similar to an Apache web server request, with a carrier class 8-CPU Linux cluster that can handle around 8000 requests per second (51) running the server-side component, the time taken by a legitimate client to successfully solve a puzzle when there is 1 million bots attacking is a mere 250 seconds. Note that C_{net} is lower than C_{cpu} because even on a Lenovo ThinkPad X60 laptop with a specification of one Intel Core 2 T5500 1.66GHz and 1 GB RAM, we can generate approximately 3850 puzzles from 133-byte strings (the estimated size of a connection request consisting of server channel data such as application-layer channel name, intermediary IP and port number) in a second.

7.8.3 Experiment

7.8.3.1 Tick-based Emulation

Performing DDoS experiments is difficult since carrying it out at real-scale requires extreme amount of resources. A theoretical analysis is best supplemented by simulations or experiments on a testbed, e.g., Emulab (122) or DETERlab (35). However, both are less than ideal. The former lacks realism while the latter suffers from complexity in emulating a real-world environment at a smaller scale in terms of legitimate/attack traffic source and Internet topology (72). For sPoW, we apply a novel approach, which involves actual deployment of sPoW code on the Planet Lab infrastructure (84). This overcomes realism concern about the topology. Since our model assumes that there is minimal network traffic congestion and queuing delay even during DDoS due to the high capacity nature of cloud intermediaries and rich path diversity offered by the clouds' multiple tier-1 ISP connectivity, the influence of background traffic is negligible thus ameliorating the necessity to simulate it. Our challenge is then to simulate how legitimate clients and attackers compete for a server's limited server channels by solving different puzzle level that varies in resolution time. We introduce "tick-based activity" to emulate the interaction between clients, zombies and a protected server. Also as the experiments are long-running, in the scale of days, the instability of some PL nodes reduces the nodes available for experiments. To make up for this, we introduce the concept of virtual hosts; a physical node can be emulating multiple virtual hosts that represent multiple clients or zombies. In tick-based activity, all virtual hosts' activities in a physical node are governed by a shared tick controller. At each tick, each virtual host is permitted to execute an activity which is one of the following: get puzzle for connection request (*get_puzzle*), brute force a single puzzle combination in an attempt to decrypt the puzzle to obtain the concealed server channel information (*brute_force*), submit connection request through uncovered server channel (*submit_request*), wait for connection establishment request from server (*wait_reply*) and receive information to establish communication channels (*conn_establish*). The activity flowchart is shown in Figure 7.3. To emulate competition, the clients and zombies can be configured with different parameters as shown in Table 7.3.

Parameter	Description	Significance
group	Group name	Virtual hosts defined under this group will be assigned names using the group name suffixed by a unique number, which is incremented for each virtual host in the group starting from zero.
qty	Quantity	Number of virtual hosts defined in the group. See *group*.
power	Power of a virtual host	Determines how often (in ticks) a virtual host will execute its action relative to each other. A higher power virtual host will execute more often. We use the concept of a virtual host and its *power* to represent zombies collaborating to compete against legitimate clients. Example, in the case where the number of zombies is N_{zombie} and legitimate clients is N_{legit}, a virtual host of $power = N_{zombie}/N_{legit}$ will represent zombies banding together to compete against a single legitimate client.
algo	Puzzle difficulty selection algorithm	Determines how puzzle difficulty is selected with each attempt to establish connection. A zombie may just request for the same puzzle level continuously while a legitimate client will request for increasing puzzle difficulty with each failed attempt.
stamina	Connection establishment quantity	How many established connections a client desires. A zombie desires "infinite" connections while legitimate clients usually want 1.
delay	Delay (in ticks) prior to activity commencement	Used to control when legitimate clients commence connection request. This can be used to delay client requests so that zombies can gain initial momentum to speed up the effect of full-fledge DDoS within a short time.

Table 7.3: sPoW experiment emulation configuration parameters

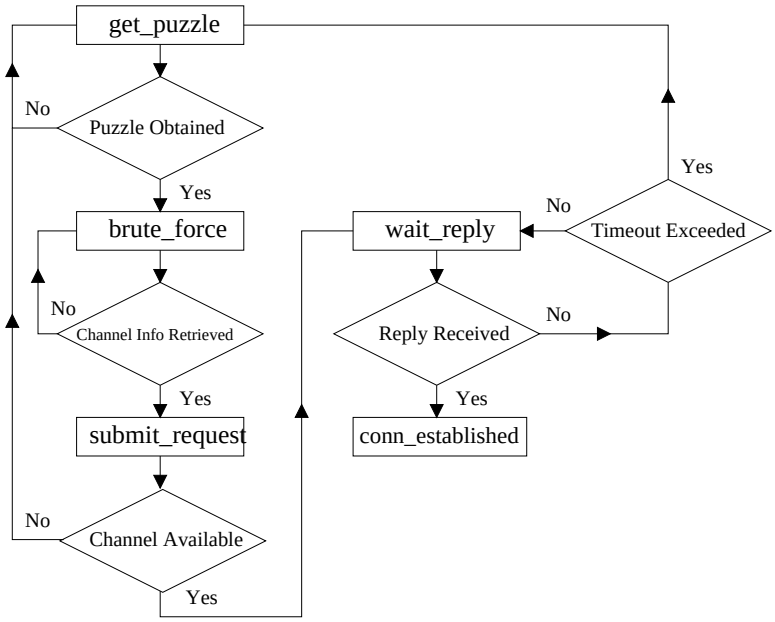


Figure 7.3: Tick-based activity flowchart

We provide an example of a example configuration (Listing 7.1) here in YAML format (124). because it is the best way to explain the parameters.

We define two groups, one with *group* name "att" and the other "legit". Each group will have 5 virtual hosts (*qty*=5). The "att" group will start immediately (*delay*=0) and request/solve level "4" puzzles (*algo*=4). It will continue to request/solve puzzle until the experiment is terminated (*stamina*=infinite). The "legit" group will start after 30 ticks (*delay*=30) and request/solve puzzle that varies according to the formula x-((rand(x)-rand(x))/4), where x is the attempt number. The legit virtual hosts will stop after succeeding in establishing a single connection request (*stamina*=1). Note that the *power* of the attack group vs. the legitimate client group is 20 vs. 1. This represents 20 zombies collaborating to compete with a single legitimate client. Thus, in total this configuration is emulating 5 legitimate clients competing with 5x20=100 zombies. This competition is reflected in the emulation through the relationship between each tick and virtual host activity. All activities except *brute_force* occupies on a single tick, since, regardless of collaboration, all of those activities cannot be accelerated. *brute_force* activity however can be parallelized and therefore, with more power

the virtual host is permitted to brute force more combinations. In other words, the configuration will result in legitimate clients only carrying out *brute_force* activity once every 20 ticks while attackers perform a brute-force combination at every tick.

Listing 7.1: Sample example configuration file to emulate 5 clients vs. 5x20

```
1  attacker_group:
2    group: att
3    qty: 5
4    algo: 4
5    delay: 0
6    stamina: infinite
7    power: 20
8  legit_group:
9    group: legit
10   qty: 5
11   algo: x-((rand(x)-rand(x))/4)
12   delay: 30
13   stamina: 1
14   power: 1
```

On top of those parameters, the connection manager implementation has two tunable parameters: (1) period for determining puzzle request and resolution rate, t and (2) server capacity, C. Both of which have been explained in Section 7.7.3.

7.8.3.2 Caveats

The existence of virtual hosts and the requirement for virtual hosts to perform multi-stage processing, *get_puzzle*, *brute_force*, etc., make the concept of virtual hosts in our experiment different from existing ones where the virtual hosts are used purely to send attack traffic without any need to process server replies. This makes measuring connection establishment time—the time a *get_puzzle* is issued to the time *conn_establish* is achieved, difficult since the connection establishment time of each virtual host is actually influenced by the processing of other virtual hosts that are running simultaneously on the same physical system. In fact, thus far, no experiments use DETER emulation test bed for measuring connection establishment time (72) when virtual hosts are involved. We intend to overcome

this by measuring connection establishment time in terms of ticks and convert the ticks into a measurement of "tardiness". The measurement in tardiness is necessary because we want to compare the connection time in different experiments using different configuration, e.g., *power*. An experiment with zombie *power*=5 means that a legitimate client only performs an action every 5 ticks while the zombies perform one every tick. However, in another experiment where zombie *power*=3, a legitimate client performs an action every 3 ticks. Assuming that a legitimate client completes connection establishment by performing each activity sequence in the connection establishment cycle once (total of 4 activities), then the legitimate client in the former experiment will clock 4x5=20 ticks while the legitimate client in the latter experiment clocks 4x3=12 ticks. However, both actually should complete roughly within the same amount of time, barring major differences in CPU load and traffic conditions, if the experiment was carried out in real time since in both cases, the legitimate clients completed their actions without any retry. Thus we need a metric tardiness that is obtained by dividing the connection time in ticks with ticks required for a legitimate client to complete connection in the absence of DDoS, i.e., zombie *power*x4.

7.8.3.3 Experiment Scenarios

With all the assumptions, modeling and connection establishment time measurement metrics carefully thought out, we ran experiments to seek insight about the following:

Legitimate client bounded connection time To determine if a legit client can establish a connection within bounded time, we configure attackers to compete with increasing level of k in each experiment, starting from 2 to 4. In those experiments, we set zombie *power*= 3, legitimate client *power*= 1, *qty*= 5, C= 5, t= 120 and deploy the clients/zombies on 4 physical nodes with another node to act as a server. This experiment reflects 4x5=20 legitimate clients competing against 4x5x3=60 zombies, i.e., 20 zombies in groups of 3. As shown in Figure 7.4, as zombies increase k from 2 to 3, legitimate clients have to expend more resources to solve more difficult puzzles before successfully gaining higher priority in connection

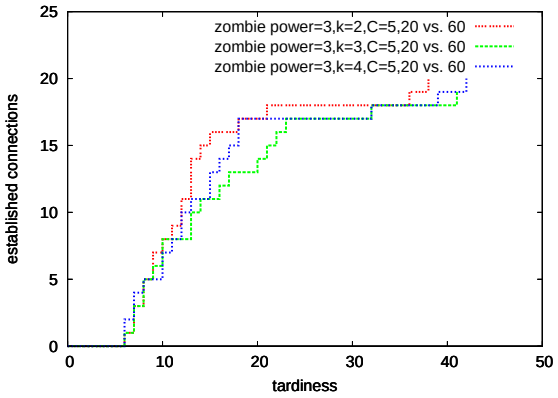


Figure 7.4: Connection establishment time normalized to "tardiness" metric for legitimate clients when zombies attack by solving puzzles with different puzzle level, k.

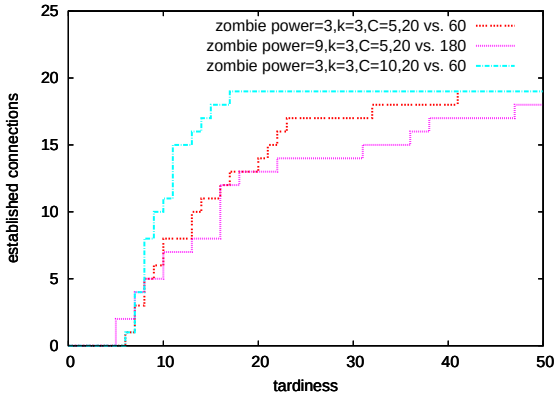


Figure 7.5: Connection establishment time normalized to "tardiness" metric for legitimate clients when zombies attack by solving puzzles with k= 3 but with tunable parameters, server capacity, C and zombie *power* varied.

establishment thus resulting in longer connection establishment time, i.e., the CDF for k= 3 is further to the right. However, when the zombies increase k from 3 to 4, the connection establishment time for legitimate clients improves, i.e., k= 4 is further to the left. This indicates that as zombies attempt to solve more difficult puzzles, their resources become over-stretched leaving them unable to solve sufficient puzzles to dominate all available channels to the server, resulting in legitimate clients solving easier puzzles successfully establishing their connection in shorter time.

Effect of tunable parameters Besides k, we explore how connection establishment time is affected by the parameters *power* and C. As shown in Figure 7.5, when power of attackers are increased from 3 to 9, i.e., the effective zombie quantity becomes 5x4x9=180 competing with 20 legitimate clients, the CDF shifts to the right, indicating that connection establishment times degrade, which is expected since there are more zombies competing for server channels. The effect of increasing C from 5 to 10 improves establishment time since with more channels available the chances that legitimate clients can establish connections increase.

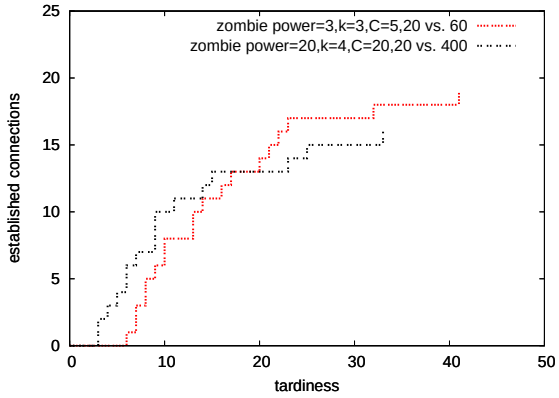


Figure 7.6: Connection establishment time as the experiment is scaled up to more realistic numbers of legitimate client/zombie ratio.

Approaching realistic experimentation We scale up the power of zombies to 20, thus emulating 20 legitimate clients competing with 5x4x20=400 zombies and we assume that the server is catered for peak loads, i.e., we set C equal to the number of legitimate clients. We chose the power of 20 because this was the also the ratio between legitimate clients/zombies used in the Portcullis simulation (83). Moreover, as noted by Rajab et al. (88), the number of live zombies is usually only in the magnitude of a few thousands, thus if legitimate clients numbering in at least hundreds, we believe the power of 20 can be a reflection of the ratio between legitimate clients/zombies in real scenarios. Figure 7.6, shows that in the 20 legitimate clients vs. 400 zombies (20 vs. 400) experiment, connection establishment for a majority of the connections improve; approximately 13 out of 20 connections, are faster than in the 20 vs. 60 because the in the 20 vs. 400, server is catered for peak legitimate load, i.e., C=number of legitimate clients. Some clients suffer from longer connection times as they attempted to solve puzzles that are much harder because they fail in their initial attempts while others failed to complete because of the nodes they were running on were disrupted since the experiments often last beyond two days. Overall, in the realistic experiment, 75% of legitimate clients can establish connections within a tardiness of 35. In order words, assuming, that with no DDoS, the connection establishment time is 0.5 seconds, under DDoS where zom-

bies outnumber legitimate clients by 20 to 1, and with the server catered to handle all legitimate clients, legitimate clients are expected to establish connection within 35x0.5=17.5 seconds.

7.9 Discussion

7.9.1 Alternative Manifestations

To implement server channels, we can utilize any technology, which enables creation of different "channels" with distinct names that can transport or store data. The channel namespace must also be large enough to force an attacker to spread its brute-force network-layer DDoS attack over the large namespace resulting in only a small fraction of DDoS getting through to the protected server.

Therefore, although we utilize a message queuing cloud as an example of *application-layer* channel provider. We can easily utilize Amazon's Simple Storage Service (S3) (7) as well but the server-side component would need to continually poll S3 for initial connection request arrivals since S3 does not push data to the server (Step 8 in Figure 7.1).

Another more general and flexible but complex alternative is to use generic clouds, e.g., Amazon's Elastic Cloud (EC2) (9). Inside the cloud we roll out our own *application-layer* channel providers, such as IRC servers (80) or Berkeley's I3 (104). The IRC channel and I3 trigger names have large namespaces that are resistant to brute-force. Another alternative is to utilize the *transport layer* as channel providers by combining an IP address and a TCP/UDP port number to form a namespace for implementing server channels. These channel providers need to work with generic cloud firewalls to constantly update the rules to permit only traffic destined to the selected server channels in order to mitigate against network-level eDDoS; only traffic headed towards the ephemeral server channels is permitted and billed while the rest are dropped at the firewalls, incurring no charges.

7.9.2 Utilization of Existing Name Service

sPoW can rely on existing DNS servers as a puzzle delivery mechanism. Instead of returning the IP address of the requested hostname, it returns the puzzle as the DNS address (A) resource record (76). Puzzle requester in the client-side component will be able to request and solve the puzzle.

7.9.3 sPoW Client Code

We have talked about how Java Applets or Java Web Start can be used to transparently coax a users web browser to load sPoW-based clients to achieve resilient connectivity to web applications. For non-browser based applications, we have two options: (1) recompile the client with sPoW libraries or (2) upgrade the operating system (OS) with sPoW-enhanced network stack. The former is ideal for developers to recompile their applications without no changes and making the sPoW-enhanced binaries available for use. The latter requires meticulous OS recompilation but is ideal since all applications can acquire sPoW capabilities without changes.

7.10 Limitations

Cloud Provider Billing System The example manifestation of sPoW provided, relies on Amazon SQS, which bills the cloud adopter for only traffic stored/retrieved in/from her message queues and her queue management (creation/deletion) requests. SQS offers defense against network-level with its vast infrastructure; it drops DDoS traffic at no cost to the adopter because of SQS's inability to determine the destination of the traffic since it is targeted at queue names that belongs to no adopter. sPoW exploits this billing mechanism idiosyncrasy to avoid adopters from having to foot the bill for handling unsolicited traffic, i.e., eDDoS, by changing the queue names frequently. The cost for changing queue names are currently very low but may subsequently change in the future (possibly due to our exploitation) and a sharp increase may impact the unilateral deployability of sPoW.

Lack of Realistic Experiment Testbed Even though in Section 7.8, we showed that the time legitimate clients take to establish connections under DDoS is indeed bounded, and how the bounded time varies under different DDoS scenarios, the experiments are not carried out in realistic environments. The devastating and large-scale nature of DDoS attack prevent experiments from being carried out realistically or at the same scale. Most DDoS research have resorted to pure simulation (83), or cleverly introducing environmental parameters to simulate DDoS in experiments, e.g., Phalanx (38) simulates DDoS by making 50% of its intermediaries drop 75% of received traffic. In our evaluation, we simulate DDoS, by performing experiments using tick-based activity.

Discrepancy in Computational Power sPoW empowers legitimate clients to expend own resources in order to get higher priority service from a server. However, low computational power legitimate clients, e.g., mobile phones or personal desktop assistants (PDAs), will end up with the short-end in such a scheme. This has been pointed out by Parno et al. (83). In order to mitigate this, providers of mobile Internet services may offer their clients access to a proxy that computes a rate-limited number of puzzles on behalf of each client. Re-designing puzzles to use non-parallelizable decryption as well as exploring memory-based function puzzles (1; 39; 40) since memory discrepancy among devices are much smaller 5-10 times versus 38 times in computational power discrepancy, are other alternatives.

7.11 Future Work

The real acid test is to deploy sPoW to defend a server application in a real environment. For a start, our intention is to create a zero-install sPoW-based basic web browser initially, i.e., without all the complicated support for Javascripts etc., using JWS and deploy a server-side component on a powerful server. Through these two components, together with Amazon's S3 and SQS, we can offer a pay-as-you-use DDoS resilient service to web servers that has static web pages without any modification. From there, we will increase the functionality of the JWS sPoW

web browser to support more features in order to offer DDoS resiliency to a wider base of web pages and applications.

In the connection management algorithm, we would also like to investigate further how the value of t, the period used for puzzle request and resolution rate calculation, and a larger number of zombies can affect a client's connection establishment time.

Currently a client attempts to retry failed connection establishment by slowly increasing puzzle difficultly chosen, perturbed by some randomness. It would be interesting to see if other puzzle difficulty selection algorithms can result in a faster connection establishment time, and whether there is a different optimal algorithm for different traffic conditions. If so, we want to find out what those conditions are, and how a client can probe/monitor those conditions in order to adopt different puzzle difficulty selection algorithms.

7.12 Conclusion

We created sPoW, which is a unilaterally deployable "pay-as-you-use" cloud-based eDDoS mitigation mechanism that offers filterable and non-filterable eDDoS protection to servers deployed in clouds or DDoS protection to servers in general. sPoW itself, in turn, needs to be protected against eDDoS. By mediating connectivity to servers and varying the channel identities used to reach the servers frequently, sPoW transforms network-level eDDoS into traffic that can be filtered. To handle non-filterable eDDoS, sPoW adapts conventional capabilities through an innovative way to work with our self-verifying PoW, enabling the ratio of non-filterable eDDoS in legitimate-looking traffic to be reduced thereby minimizing the cost incurred by non-filterable eDDoS but at the expense of protracted legitimate client connection establishment time.

Chapter 8

Conclusion

We started out by defining our goal of deployable resiliency; a DDoS mitigation mechanism should be (1) unilaterally deployable, i.e., it is low in resource consumption while requiring minimal modification to existing core infrastructure and (2) resilient against the newly-emerged eDDoS as well as all types of DDoS described by Mirkovic's taxonomy (74), with exception of semantic and router infrastructure targeted attacks. Our proposal also includes the additional features of DDoS attack deterrent and client empowerment DDoS congestion bypass mechanisms. We then identify economic phenomena, *negative externality*, *incentive misalignment* and *ossified infrastructure* as the main impediments to DDoS research deployability. To overcome them, we design the Burrows framework; its self-scaling model empowers end-users, who have the most incentive to deploy DDoS mitigation mechanisms, to collaborate on the construction of a traffic control reception overlay comprising of many systems known as intermediaries, without assistance from disinterested parties. This realigns economic incentive. By forcing all server-destined traffic through this overlay of intermediaries, a server's DDoS defense is solely dependent on the intermediaries thus minimizing negative externality, i.e., the need to rely on disinterested parties for defense. Moreover, the intermediaries can be deployed on Internet edge nodes instead of difficult-to-modify ossified core Internet infrastructure. It is crucial for the traffic control mechanism to possess resources sufficient to drop all filterable DDoS traffic while continuing to accept all non-filterable DDoS and legitimate traffic that it needs to prioritize. This implies that the overlay must be capable of amassing

large number of intermediaries, which is questionable.

Thus besides self-scaling, we explore the possibility of using any existing Internet system as an intermediary in KUMO, which is designed based on the Burrows framework. KUMO tunnels client traffic to a protected server through any existing Internet system by utilizing that system's protocol. The elegance of KUMO is that no modifications are required on the client, protected server and even the Internet systems that play the role of intermediaries. Furthermore, a digital signature-less accounting mechanism can be implemented to compensate the Internet systems for the use of their resources. Overall, we believe KUMO enhances resource accumulation for DDoS defense since any existing Internet system can be recruited as an intermediary without modifications and its owner will have the financial incentive to contribute the system's often idle over-provisioned resources for intermediary purposes.

In both Burrows and KUMO, the haphazard recruitment of intermediaries implies that there may be malicious systems among them leading to confidentiality, integrity and availability (CIA) issues. Restricting intermediaries used in traffic control to those belonging to trusted entities or those from entities a protected server has existing relationships with can alleviate this security exposure.

We look at the possibility of reducing this trust issue by exploring 2 other ways to amass intermediaries for traffic control: (1) utilizing the large number of an ISP's access routers (Overfort) and (2) utilizing a few resource-rich cloud computing platforms (sPoW). Both eliminates trust issues since the intermediaries are trusted. Overfort is a first-of-its-kind unilaterally deployable automatic traceback and black-list mechanism. It enables a server to traceback to clusters that harbor zombies and subsequently black-list those clusters by refusing their queries about the server's whereabouts. This feature enables Overfort to quickly find and suppress DDoS even though the aggregate intermediary resource is less than those of attacking zombies. sPoW is a unilaterally deployable mechanism that is different from Burrows, KUMO and Overfort because it has a mechanism to deal with all three filterable DDoS, non-filterable DDoS and eDDoS. The usage of resource-rich cloud platforms facilitates sufficient resource harness for defense against filterable DDoS. For non-filterable DDoS defense, sPoW enables clients to compete for a server resources with attackers by expending more of

their own resources to discover obscure channel names that lead to the server with each channel's priority proportional to its name obscurity. sPoW's unique self-verifying PoW enables a deployer to modify the cloud's billing mechanism without the cloud providers assistance and without violating contractual terms, such that the cost of eDDoS is not solely borne by her; eDDoS traffic will not know the obscure channel names created within the cloud thus it will be dropped by the cloud provider's firewall at no cost to the sPoW deployer.

Finally, AI-RON-E is a slight deviation from the traditional DDoS defense because instead of defending against it, it empowers legitimate clients to find alternate paths to bypass congested caused by DDoS.

Appendix A

DDoS Mitigation Mechanism Metrics

We classify DDoS mitigation research using 3 metrics: (1) deployability, (2) incentive and (3) effectiveness. A high rating for all 3 indicates a very effective and high deployable resilient mechanism. At the very least, 2 of the 3 metrics, deployability and incentive ratings of a research must be high, otherwise the research will become a "white elephant" that never make it beyond the doors of academia. This classification is not perfect and is still evolving. We hope to obtain feedback in particular from authors of those research we classified to understand the accuracy of these ratings as well as evaluate the necessity to include other metrics that enables more objective and accurate ratings.

A.1 Deployability

We define deployability as the ease at with which a DDoS mitigation mechanism can be deployed. To measure deployability objectively, we introduce the following metrics: (1) deployment location and (2) deployment size. Based on these metrics, we assign deployability ratings from 0 to 10 with 10 being easiest to deploy. The possible values for each metric and ratings are shown in Table A.1.

Server-side	Edge	Client-side	Network	Rating
Few	None	None	None	10.0
None	Few	None	None	9.5
Few	Few	None	None	9.0
None	None	Few	None	8.5
Few	None	Few	None	8.0
None	Few	Few	None	7.5
None	None	None	Few	7.0
Few	None	None	Few	6.5
None	Few	None	Few	6.0
None	None	Few	Few	5.5
None	Many	None	None	5.0
Few	Many	None	None	4.5
None	None	Many	None	4.0
Few	None	Many	None	3.5
None	*[a]	Many	None	3.0
None	None	None	Many	2.5
Few	None	None	Many	2.0
None	*	None	Many	1.5
None	None	*	Many	1.0
None	Many	Many	Many	0.0

[a] As long as there is a column with "many" then the other value of columns marked by "*" is unimportant, i.e., they can be "few" or "many", the deployability rating is unaffected.

Table A.1: Deployability Rating

The ease of deployability at deployment locations in diminishing order is: server-side, Internet edge ("edge" for short), client-side and network. The deployability ratings assigned to those locations are influenced by (1) the modification flexibility of the systems at those location to incorporate DDoS defense and (2) the likelihood of technical skills available to perform those modifications. At the server-side, both rank high. At the edge both also rank quite high but edge nodes often needs to be specially installed for certain purposes thus mandating slightly higher skill set, which may not be easily available, at the client, skill level drops drastically, while at the network, modification is extremely difficult due to the complex and intricately interwoven systems. Also mechanisms that require components to be deployed at more than one classified location is rated lower, e.g. mechanisms that require network and edge components are rated lower

than those that require only network components. We use the terms *many*, *few*, *none* to indicate deployment size, a DDoS defense's deployability degrading as the number of deployment points increase.

A.2 Incentive

We define incentive as the willingness of an owner of a system, e.g., router, server, computer, etc., to deploy DDoS defense mechanisms. To measure incentive objectively, we introduce the following metrics: (1) level of DDoS affliction at the deployment location and (2) deployment size. The affinity between the afflicted location and the deployment location contributes to incentive rating; if a mitigation mechanism is designed to be deployed at a location that is highly afflicted by DDoS, the incentive to deploy, thus the rating will be high and vice versa. The possible values for each metric are shown in Table A.2.

Afflicted	Fairly Afflicted	Not Afflicted	Rating
Most	None	None	10.0
Most	Few	None	9.5
Most	Most	None	9.0
Few	Most	None	8.5
None	Most	None	8.0
Most	None	Few	7.5
Most	Few	Few	7.0
Most	Most	Few	6.5
Few	Most	Few	6.0
None	Most	Few	5.5
Most	Few	Most	5.0
Most	Few	Most	4.5
Few	Most	Most	4.0
Few	Few	Most	3.5
Few	None	Most	3.0
None	Most	Most	2.5
None	Few	Most	2.0
None	None	Most	1.5

Table A.2: Incentive Rating

The level of DDoS affliction is ranked in diminishing order as *afflicted*, *fairly afflicted*, and *not afflicted*. *Afflicted* indicates that the systems in that location bears the full brunt of DDoS, e.g., the server under attack, while *fairly afflicted* refers to the systems that suffers collateral damage, i.e., they are not under direct attack but nonetheless are affected, e.g., servers that share the uplink with the server under attack or networks and routers along the attack path. We want to highlight a particular case that is easily misunderstood in this area; usually clients do not bear the brunt of attacks, i.e., they are classified as *not afflicted*, unless they want to connect to the server under attack, which means clients should be classified under *fairly afflicted* in that circumstance.

We use the terms *most*, *few*, *none* to indicate deployment size. Although the deployment size terms here look similar to those used as metric for deployability in the previous section, they differ because unlike in deployability where the terms describe the absolute quantity of the nodes where installation is required, here the terms are used in a relative sense, with respect to the total number of nodes required in the entire deployment. For example, for the combination of "afflicted - most, fairly afflicted - most, non-afflicted - none", it means that most of the deployment nodes are spread equally in both afflicted and non-afflicted areas, and this is equivalent to the combination of "afflicted - few, fairly afflicted - few, non-afflicted - none". Due to this, not all combinations are shown in the table, as this can be easily worked out by the reader. Although it can be argued that, the number of absolute nodes also affect the incentive, we believe that this has been catered for when we used absolute node quantity as a metric for rating deployability.

A.3 Effectiveness

We measure the effectiveness DDoS mitigation mechanism by rating it based on the type of defense it offers and the type of DDoS it protects against. The effectiveness ratings are shown in Table A.3.

Most descriptions can be easily associated with the DDoS attacks that we describe in this main text with the exception of the following two, which we offer

DDoS Defense	Rating
eDDoS (including non-filterable, spoofed net-level DDoS)	10.0
Non-filterable DDoS (without requiring user authentication)	9.0
User authenticated initial and established connection protection	8.0
Established connection DDoS protection only (no initial connection protection)	7.0
Pushback spoofed/net-level DDoS to client-side	6.0
Pushback spoofed/net-level DDoS to network-side	5.0
Avoid hotspots	4.0
Filter DDoS (spoofed/net-level, non-filterable, eDDoS) near server	3.0
Filter spoofed/net-level DDoS near server	2.0
Filter spoofed/net-level non-distributed DoS near server	1.0

Table A.3: Effectiveness Rating

further explanation.

Established connection DDoS protection only (no initial connection protection) DDoS defense that has some means to tag established connection traffic with unforgeable tokens so that they can be given priority, (13). However, to obtain these tokens, i.e., in the initial connection request stage, legitimate clients and zombies have to compete without any way to differentiate them thus putting legitimate clients at a disadvantage.

User authenticated initial and established connection protection Similar to the above except that initial connections from legitimate clients and zombies can be differentiated by offering clients the opportunity to use authentication or solve reverse Turing tests to gain priority high priority during initial connection request than automated zombies that are incapable of doing so.

References

[1] M. Abadi, M. Burrows, M. Manasse, and T. Wobber. Moderately Hard, Memory-bound Functions. In *Network and Distributed System Security Symposium (NDSS)*, 2003. 142

[2] J. Abley and K. Lindqvist. RFC 4786: Operation of Anycast Services (http://www.ietf.org/rfc/rfc4786.txt), 2006. 12

[3] Akamai Technologies. State of the Internet Quarterly Reports (http://www.akamai.com/stateoftheinternet/), 2008. 2

[4] Akamai Technologies. Akamai Technologies (http://www.akamai.com/), 2009. 1, 103, 122

[5] Amazon Cloud Computing Forum. Unaddressed DDoS Concerns in Amazon's Cloud (http://developer.amazonwebservices.com/connect/search.jspa?q=DDoS), 2009. 115

[6] Amazon Web Services. Amazon Simple Queue Services (SQS) (http://aws.amazon.com/sqs/), 2009. 120

[7] Amazon Web Services. Amazon Simple Storage Services (S3) (http://aws.amazon.com/s3/), 2009. 91, 140

[8] Amazon Web Services. Amazon's Web Services Case Studies (http://aws.amazon.com/solutions/case-studies/), 2009. 115

[9] Amazon Web Services. Elastic Compute Cloud (EC2) (http://aws.amazon.com/ec2/), 2009. 115, 140

[10] American Registry for Internet Numbers (ARIN). ARIN IPv4 Depletion Notice (https://www.arin.net/knowledge/about_resources/ceo_letter.pdf), 2009. 70

[11] David Andersen. Mayday: Distributed Filtering for Internet Services. In *USENIX Symposia on Internet Technologies and Systems (USITS)*, 2003. 20, 114

[12] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient Overlay Networks. In *ACM Symposium on Operating Systems Principle (SOSP)*, 2001. 26, 74, 75

[13] Thomas Anderson, Timothy Roscoe, and David Wetherall. Preventing Internet Denial-of-Service with Capabilities. In *ACM Hot Topics in Networks (Hotnets)*, 2002. 25, 114, 116, 151

[14] Farooq M. Anjum. TCP Algorithms and Multiple Paths: Considerations for the Future of the Internet. 2004. 85

[15] Arbor Networks. Annual Infrastructure Security Report (http://www.arbornetworks.com/report), 2005. 2

[16] Arbor Networks. Arbor PeakFlow TMS (http://www.arbornetworks.com/peakflowsp), 2009. 23

[17] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Resource Records for DNS Security Extensions (http://www.ietf.org/rfc/rfc4034.txt), 2005. 90

[18] Katerina Argyraki and David R. Cheriton. Active Internet Traffic Filtering: Real-time Response to Denial-of-Service Attacks. In *USENIX Annual Technical Conference (ATC)*, 2005. 24, 114

[19] Tuomas Aura, Pekka Nikander, and Jussipekka Leiwo. DOS-resistant Authentication with Client Puzzles, 2000. 42

[20] Steve Bellovin, Marcus Leech, and Tom Taylor. ICMP Traceback Messages. *Internet Drafts*, 2003. 20, 55

[21] Phillippe Biondi and Arnaud Ebalard. IPv6 Routing Header Security. In *Canada Security West Conference (CANSECWEST)*, 2007. 89

[22] Bram Cohen. BitTorrent (http://www.bittorrent.com), 2009. 77

[23] Kevin Butler, Toni Farley, Patrick McDaniel, and Jennifer Rexford. A Survey of BGP Security Issues and Solutions. In *Proceedings of IEEE*, 2009. 91

[24] Martin Casado, Aditya Akella, Pei Cao, Niels Provos, and Scott Shenker. Cookies Along Trust-Boundaries (CAT): Accurate and Deployable Flood Protection. In *USENIX Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2006. 9, 17, 21, 24, 114

[25] ccNSO. DNSSec Survey Report 2009. *ICANN*, 2009. 91

[26] M. Cha, S. Moon, C. D. Park, and A. Shaikh. Placing Relay Nodes for Intra-Domain Path Diversity. In *IEEE INFOCOM*, 2006. 76

[27] Xin Chen, Haning Wang, Shansi Ren, and Xiaodong Zhang. DNScup: Strong Cache Consistency Protocol for DNS. In *IEEE International Conference for Distributed Computing Systems(ICDCS)*, 2006. 63

[28] Cisco Networks. Cisco Anomaly Guard (http://www.cisco.com/en/US/products/ps6235/index.html), 2009. 23

[29] Cisco Networks. Understanding Unicast Reverse Path Forwarding (http://www.cisco.com/web/about/security/intelligence/unicast-rpf.html), 2009. 87

[30] Computer Emergency Response Team (CERT). Build Security In (https://buildsecurityin.us-cert.gov/daisy/bsi/home.html), 2009. 15

[31] Computer Security Institute. CSI Computer Crime and Security Report (http://www.gocsi.com/), 2008. 2

[32] Debra L. Cook, William G. Morein, Angelos D. Keromytis, Vishal Misra, and Daniel Rubenstein. WebSOS: Protecting Web Servers From DDoS Attacks. In *IEEE International Conference on Network (ICON)*, 2003. 20, 114

[33] David Dagon, Cliff Zou, and Wenke Lee. Modeling Botnet Propagation Using Time Zones. In *Network and Distributed System Security Symposium (NDSS)*, 2006. 20

[34] Drew Dean and Adam Stubblefield. Using Client Puzzles to Protect TLS. In *USENIX Security Symposium*, 2001. 42

[35] DETERlab. DETERlab Testbed (http://www.isi.edu/deter/), 2000. 133

[36] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol v1.2 (http://www.ietf.org/rfc/rfc5246.txt), 2008. 20

[37] Roger Dingledine and Nick. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium*, 2004. 91

[38] Colin Dixon, Thomas Anderson, and Arvind Krishnamurthy. Phalanx: Withstanding Multimillion-Node Botnets. In *USENIX Network Systems Design and Implementation (NSDI)*, 2008. 13, 17, 18, 21, 26, 114, 118, 121, 127, 131, 142

[39] C. Dwork, A. Goldberg, and M. Naor. On Memory-bound Functions for Fighting Spam. In *CRYPTO*, 2003. 142

[40] C. Dwork and M. Naor. Pricing via Processing or Combatting Junk Mail. In *CRYPTO*, 1993. 142

[41] T. Fei, S. Tao, L. Gao, and R. Guerin. How to Select a Good Alternate Path in Large Peer-to-Peer Systems. In *IEEE INFOCOM*, 2006. 75

[42] D. Ferguson and P. Senie. RFC 2827: Network Ingress Filtering (http://www.ietf.org/rfc/rfc2827.txt), 2000. 18, 87

[43] P. Ferguson and D. Senie. RFC 2267: Network Ingress Filtering: Defeating Denial of Service Attacks which employs IP source address spoofing (http://www.ietf.org/rfc/rfc2267.txt), 1998. 19

[44] Jason Franklin, Vern Paxson, Adrian Perrig, and Stefan Savage. An Inquiry into the Nature and Causes of the Wealth of Internet Miscreants. In *ACM Computer and Communications Security (CCS)*, 2007. 2

[45] Michael Freedman, Eric Freudenthal, and David Mazieres. Democratizing Content Publication with Coral. In *USENIX Network System Design and Implementation (NSDI)*, 2004. 1, 103, 122

[46] Google. Google App Engine (http://code.google.com/appengine/), 2009. 115

[47] Barry Raveendran Greene, Chris Morrow, and Brian Gemberling. ISP Security - Real World Techniques II. *North America Network Operators Group (NANOG)*, 2001. 22, 23

[48] Adam Greenhalgh, Mark Handley, and Felipe Huici. Using Routing and Tunneling to Combat DoS Attacks. In *USENIX Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2005. 17, 21, 114

[49] Krishna P. Gummadi, Harsha V. Madhyastha, Steven D. Gribble, Henry M. Levy, and David Wetherall. Improving the Reliability of Internet Paths with One-hop Source Routing. In *USENIX Operating Systems Design and Implementation (OSDI)*, 2004. 26, 75, 80

[50] ha.ckers.org. Slow Loris HTTP DoS (http://ha.ckers.org/slowloris/), 2009. 15

[51] Ibrahim Haddad. Open-Source Web Servers: Performance on a Carrier-Class Linux Platform (http://www.linuxjournal.com/article/4752), 2001. 132

[52] Chris Hoff. Cloud Computing Security From DDoS (http://www.rationalsurvivability.com/blog/?p=66), 2008. 115

[53] H. Y. Hsieh and R. Sivakumar. A Transport Layer Approach for Achieving Aggregate Bandwidths On Multi-Homed Mobile Hosts. In *ACM Mobicom*, 2002. 85

[54] Geoff Huston. Interconnection, Peering, and Settlements. In *Internet Society INET*, 1999. 83

[55] Intelliguard. Intelliguard dps (http://www.intelliguardit.net/Docs/Whitepapers/IntelliGuardIT_Inline_Deployment_Whitepaper.pdf), 2009. 23

[56] Internet World Stats. Usage and Population Statistics (http://www.internetworldstats.com/stats.htm), 2009. 3, 61

[57] Cheng Jin, Haining Wang, and Kang G. Shin. Hop-Count Filtering: An Effective Defense Against Spoofed DDoS Traffic. In *ACM Computer and Communications Security (CCS)*, 2003. 19

[58] S. Kent, C. Lynn, and K. Seo. Secure Border Gateway Protocol (SBGP). In *IEEE Journal on Selected Areas of Communication*, 2000. 18, 90

[59] A. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In *ACM Special Interest Group in Communications (SIGCOMM)*, 2002. 20, 114

[60] Soon Hin Khor, Nicolas Christin, Tina Wong, and Akihiro Nakao. Power to the People: Securing the Internet One Edge at a Time. In *ACM Large Scale Attack and Defense (LSAD)*, 2007. 21, 58, 61, 114

[61] Soon Hin Khor and Akihiro Nakao. AI-RON-E: The Prophecy of One-hop Source Routers. In *IEEE Globecom Next Generation Networks Symposium*, 2008. 92, 110

[62] Soon Hin Khor and Akihiro Nakao. Overfort:Combating DDoS with Peer-to-Peer DDoS Puzzle. In *IEEE Secure Systems and Nework Workshop*, 2008. 17, 114

[63] Soon Hin Khor and Akihiro Nakao. sPoW: On-Demand Cloud-based eDDoS Mitigation Mechanism. In *Hot Topics in Dependency WorkShop (HOTDEP)*, 2009. 16, 17, 114

[64] Aleksandar Kuzmanovic and Edward W. Knightly. Low-Rate TCP-Targeted Denial of Service Attacks. In *ACM Special Interest Group in Communications (SIGCOMM)*, 2003. 15

[65] Xin Liu, Xiaowei Yang, and Yanbin Lu. To Filter or to Authorize: Network-layer DoS Defense Against Multimillion-Node Botnets. In *ACM Special Interest Group for Communications (SIGCOMM)*, 2008. 24, 114

[66] Looking Glass. Looking glass (http://www.bgp4.net/wiki/doku.php?id=tools:ipv4_looking_glasses), 2009. 78, 81

[67] Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iPlane: An Information Plane for Distributed Services. In *USENIX Operating Systems Design and Implementation (OSDI)*, 2006. 74, 109

[68] Ratul Mahajan, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker. Controlling High Bandwidth Aggregates in the Network. *ACM Computer Communication Review (CCR)*, 2002. 24, 59

[69] A. Mahimkar, J. Dange, V. Shmatikov, H. Vin, and Y. Zhang. dFence: Transparent Network-based Denial of Service Mitigation. In *USENIX Network Systems Design and Implementation (NSDI)*, 2007. 22

[70] Athina Markopoulou, Gianluca Iannaccone, Supratik Bhattacharyya, Chen nee Chuah, and Christophe Diot. Characterization of Failures in an IP Backbone. In *IEEE INFOCOM*, 2004. 88, 89

[71] Massachusetts Institute of Technology. Spoofer Project: Current State of IP Spoofing (http://spoofer.csail.mit.edu/summary.php), 2009. 87

[72] Jelena Mirkovic, Sonia Fahmy, Peter Reiher, and Roshan Thomas. How to Test DDoS Defenses. In *Cybersecurity Applications & Technology Conference For Homeland Security (CATCH)*, 2009. 133, 136

[73] Jelena Mirkovic, Gregory Prier, and Peter Reiher. Attacking DDoS at the Source. In *IEEE International Conference on Network Protocols (ICNP)*, 2002. 18, 114

[74] Jelena Mirkovic and Peter Reiher. A Taxonomy of DDoS Attack and DDoS Defense Mechanism. In *ACM Computer Communications Review (CCR)*, 2004. 15, 28, 144

[75] Jelena Mirkovic, Max Robinson, and Peter Reiher. Alliance formation for DDoS defense. In *Applied Computer Security Associates (ACSA) New Security Paradigms Workshop (NSPW)*, 2003. 7

[76] P. Mockapetris. Domain Names: Implementation and Specification (http://www.ietf.org/rfc/rfc1035.txt), 1987. 141

[77] David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, and Nicholas Weaver. Inside the Slammer Worm. *IEEE Security and Privacy Magazine*, 2003. 94

[78] David Moore, Geoffrey M. Voelker, and Stefan Savage. Inferring Internet Denial-of-Service Activity. In *USENIX Security Symposium*, 2001. 2

[79] National Institute of Science and Technology (NIST). Data Encryption Standard (http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf), 1993. 129

[80] J. Oikarinen and D. Reed. RFC 1459: Internet Relay Chat (http://www.ietf.org/rfc/rfc1459.txt), 1993. 2, 91, 140

[81] OnlineMQ. OnlineMQ (http://www.onlinemq.com), 2009. 91, 120

[82] Kihong Park and Heejo Lee. On the Effectiveness of Route-based Packet Filtering for distributed DoS Attack Prevention in Power-Law Internets. In *ACM Special Interest Group for Communications (SIGCOMM)*, 2001. 19, 114

[83] Bryan Parno, Dan Wendlandt, Elaine Shi, Adrian Perrig, Bruce Maggs, and Yih-Chun Hu. Portcullis: Protecting Connection Setup from Denial-of-Capability Attacks. In *ACM Special Interest Group for Communications (SIGCOMM)*, 2007. 26, 114, 121, 127, 132, 139, 142

[84] Planet Lab. Planet Lab Consortium (http://www.planet-lab.org/consortium), 2002. 46, 71, 75, 77, 133

[85] Lindsey Poole and Vivek S. Pai. ConfiDNS: Leveraging Scale and History to Detect Compromise. In *USENIX Annual Technical Conference (ATC)*, 2008. 40, 54, 92, 115

[86] Prolexic. Network Protection Services (http://www.prolexic.com), 2009. 22

[87] S. Qazi and T. Moors. Scalable Resilient Overlay Networks Using Destination-Guided Detouring. In *IEEE International Conference on Communications (ICC)*, 2007. 75

[88] Moheed A. Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. My Botnet is Bigger than Yours. In *USENIX Hot Topics in Botnets (HOTBOT)*, 2007. 20, 61, 139

[89] Venugopalan Ramasubramanian and Emin Gun Sirer. The Design and Implementation of a Next Generation Name Service for the Internet. In *ACM Special Interest Group for Communications (SIGCOMM)*, 2004. 11, 40, 54, 92, 115

[90] Y. Rekhter and T. Li. RFC 1771: Border Gateway Protocol (BGP) (http://www.ietf.org/rfc/rfc1771.txt), 1995. 23, 83

[91] S. Rhea, B. Godfrey, B.Karp, J. Kubiatowicz, S. Ratnasamy, and S. Shenker. OpenDHT: A Public DHT Service and its Uses. In *ACM Special Interest Group for Communications (SIGCOMM)*, 2005. 47, 104

[92] Chris Rose and Jean Gordon. Internet Security and the Tragedy of the Commons. In *Journal of Business and Economics Research*, 2003. 39

[93] E. Rosen. RFC2547bis-03: BGP/MPLS IP VPNs, 2004. 44, 55

[94] RouteViews. University of Oregon Route Views Project (http://www.routeviews.org/), 2005. 70

[95] Craig Rowland. Covert Channels in the TCP/IP Suite. In *First Monday, Peer Reviewed Journal on the Internet*, 1996. 10

[96] sacrine. Sil v1.0 - A tiny banner grabber (http://www.netric.org), 2009. 107

[97] Jerome Saltzer and Michael Schroeder. The protection of information in computer systems, 1975. 19

[98] Stefan Savage, Tom Anderson, Amit Aggarwal, David Becker, Neal Cardwell, Andy Collins, Eric Hoffman, John Snell, Amin Vahdatand Geoff Voelker, and John Zahorjan. Detour: a Case for Informed Internet Routing and Transport. In *IEEE Micro*, 1999. 26, 74

[99] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical Network Support for IP Traceback. In *ACM Special Interest Group for Communications (SIGCOMM)*, 2000. 20, 55

[100] C. Shapiro and H. Varian. Networks and Positive Feedbacks, 1998. 43

[101] Elaine Shi, Ion Stoica, David Andersen, and Adrian Perrig. OverDoSe: A Generic DDos Protection Service Using an Overlay Network. 2006. 26

[102] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Beverly Schwartz, Stephen T. Kent, and W. Timothy Strayer. Single-packet IP traceback. In *IEEE/ACM Transactions on Networking (TON)*, 2002. 20, 55

[103] Neil Spring, Ratul Mahajan, David Wetherall, and Thomas Anderson. Measuring ISP Topologies with Rocketfuel. *IEEE/ACM Transactions in Networking*, 2004. 88

[104] Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana. Internet Indirection Infrastructure. In *ACM Special Interest Group for Communications (SIGCOMM)*, 2002. 21, 91, 140

[105] Robert Stone. CenterTrack: an IP overlay network for tracking DoS floods. In *USENIX Security Symposium*, 2000. 21, 114

[106] Krishnan Subramaniam. Cloud Computing Risks eDoS (http://www.cloudave.com/link/cloud-computing-risks-edos), 2008. 115

[107] Haibin Sun, John C. S. Lui, and David K. Y. Yau. Distributed Mechanism in Detecting and Defending Against the Low-rate TCP Attack. *Computer Networks Journal*, 2006. 59

[108] Sun Microsystems. Java Applets (http://java.sun.com/applets), 2009. 93

[109] Sun Microsystems. Java Web Start (http://java.sun.com/javase/technologies/desktop/javawebstart/index.jsp), 2009. 93, 122

[110] Symantec. Symantec report on the underground economy (http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_underground_economy_report_11-2008-14525717.en-us.pdf.), 2008. 2

[111] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Authentication Protocol (http://www.ietf.org/rfc/rfc4252.txt), 2006. 87

[112] Team CYMRU. CYMRU IP to ASN Service (http://www.team-cymru.org/Services/ip-to-asn.html), 2009. 70

[113] Renata Teixeira, Keith Marzullo, Stefan Savage, and Geoffrey M. Voelker. In search of path diversity in ISP networks. In *ACM Internet Measurement Conference (IMC)*, 2003. 26

[114] Rob Thomas and Jerry Martin. Underground Economy: Priceless. In *USENIX ;login:*, 2006. 2

[115] Traceroute. Traceroute (http://www.openbsd.org/cgi-bin/man.cgi?query=traceroute), 1987. 74

[116] J. Turner. Virtualizing the Net - a strategy for enabling network innovation [Keynote 2]. In *IEEE Symposium on High Performance Interconnects*, 2004. 4, 42, 69

[117] L. von Ahm, M. Blum, N. Hooper, and J. Langford. CAPTCHAS: Using Hard AI Problems for Security. In *EuroCrypt*, 2004. 20, 42

[118] Michael Walfish, Mythili Vutukuru, Hari Balakrishnan, David Karger, and Scott Shenker. Ddos defense by offense. In *ACM Special Interest Group for Communications (SIGCOMM)*, 2006. 26, 42

[119] Ju Wang, Xin Liu, and Andrew A. Chien. Empirical Study of Tolerating Denial-of-Service Attacks with a Proxy Network. In *USENIX Security Symposium*, 2005. 91

[120] Limin Wang, Kyoung Soo Park, Ruoming Pang, Vivek Pai, and Larry Peterson. Reliability and Security in the CoDeeN Content Distribution Network. In *USENIX Annual Technical Conference (ATC)*, 2004. 1, 103, 122

[121] XiaoFeng Wang and Michael K. Reiter. Defending Against Denial-of-Service Attacks with Puzzle Auctions. In *IEEE Symposium on Security and Privacy*, 2003. 42

[122] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. In *USENIX Operating Systems Design and Implementation (OSDI)*, 2002. 133

[123] Abraham Yaar, Adrian Perrig, and Dawn Song. SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks. In *IEEE Symposium on Security and Privacy*, 2004. 25, 114

[124] YAML. Yet Another Mark-up Language (http://www.yaml.org/spec/1.2/spec.html), 2009. 135

[125] Xiaowei Yang and David Wetherall. Source Selectable Path Diversity via Routing Deflections. In *ACM Special Interest Group for Communications (SIGCOMM)*, 2006. 75

[126] Xiaowei Yang, David Wetherall, and Thomas Anderson. A DoS-limiting Network Architecture. In *ACM Special Interest Group for Communications (SIGCOMM)*, 2005. 25, 114

[127] Ming Zhang, Chi Zhang, Vivek Pai, Larry Peterson, and Randy Wang. PlanetSeer: Internet Path Failure Monitoring and Characterization in Wide-area Services. In *USENIX Operating System Design and Implementation(OSDI)*, 2004. 74

[128] Zheng Zhang, Ying Zhang, Y. Charlie Hu, and Z. Morley Mao. Practical Defenses Against BGP Prefix Hijacking. In *ACM CoNEXT*, 2007. 1