ASSURING NETWORK SERVICE WITH BANDWIDTH AND INTEGRITY BASED FAIRNESS

BY

FARIBA MAHBOOBE KHAN

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Doctoral Committee:

Professor Carl A. Gunter, Chair
Assistant Professor Matthew Caesar
Professor Sanjeev Khanna, University of Pennsylvania
Professor Klara Nahrstedt

Abstract

During an Internet distributed denial-of-service (DDoS) attack, attackers pose as a superpower overloading bandwidth and services that otherwise would have been lightly used by genuine users. These legitimate users send few packets and occasionally back-off and fail while competing for resources. The Internet architecture provides only modest support for verifying the true origin of a packet or intention of a sender. This makes identification and filtering of attack traffic difficult. DDoS attacks could be limited greatly if there were a way to fairly distribute the resources among the parties despite limited origin integrity.

In our work, we propose two methods for achieving fairness despite no or partial implementation for integrity verification. *Adaptive Selective Verification (ASV)* provides legitimate clients service despite large but bounded attack rates without any integrity infrastructure. ASV can be implemented, without the cooperation of the core routers, by slight modification of the client and server applications. The other system is *Integrity Based Queuing (IBQ)*. In this work, we expect that integrity will not be perfect, but observe that even an imperfect implementation can improve the effectiveness of queuing when parities with better a integrity level are incentivized. ASV and IBQ together create a mechanism for incentives, infrastructure and independence for network service assurance.

ASV is shown to be efficient in terms of bandwidth consumption using network simulations. It differs from previously-investigated adaptive mechanisms for bandwidth based payment by requiring very limited state on server. Our study of IBQ includes proof of direct relationship of integrity to service, a network simulation for comparative study, simulation with real attack traffic and security analysis.

Our network assurance architecture provides a synergistic approach for defending against DDoS attacks. With moderate infrastructure support, IBQ can be an architecture to provide graded source validation on the Internet. Clients that do not have the support from the ISP, use their spare bandwidth with ASV for service.

To Ammu.

Acknowledgments

At the time I deposit this dissertation, I will have been in Illinois for seven years, three months, eleven days. In that time, life has offered me an army of friends, colleagues and mentors that have helped me through this great life challenge:

- My advisor who challenged me in research and taught me to speak up: Carl Gunter.
- The agencies that gave me financial support: Department of Computer Science, UIUC, Sarah and Shohaib Abbasi Fellowship program, National Center for Supercomputing Applications, Office of Naval Research, National Science Foundation, MacArthur Foundation.
- My faculty mentors and supporters: Klara Nahrstedt, Matthew Caesar, Sanjeev Khanna, Susan Hinrichs, Steve LaVelle, Mehdi Harandi, Roy Campbell, Sam King, Indy Gupta and José Meseguer.
- The department staff who were always there: Barb Cicone, Rhonda McLeroy, Mary Beth Kelly, Angie Bingaman, Holly Bagwell, Trisha Benson, Andrea Whitesell and Sonya Harris.
- My fellow travelers for all the gossip and coffee breaks: Sruthi Bandakavi, Jehanzeb Hameed Chowdhury, Omid Fatemieh, Nana Arizumi, Mehedi Bakht.
- My mentors: Tanya Crenshaw, Erin Chambers, Shamsi T. Iqbal, Maifi Hasan Khan, Ragib Hasan and Adam Lee.
- The BUETian friends who had my back all along: Sonia Jahid, Imranul Hoque, Ahsan Arefin, Ahmed Khurshid and Abul H. Samee.
- My village in a foreign land who were always there with food, fun and love that cannot be explained to someone who has not migrated: Liza Zahid, Sadia Zabeen, Nadia Nasreen, Syeda Akhter, Rumana Wahid, Rani Rahman,

Shahneela Chowdhury, Karishma Muntasir, Zeenat Rashid, Salim Rashid, Taher Saif, Sharif Ullah, Rezwana Silvi, Tawhid Ezaz, Abdullah Muzahid and Kallol Das.

- My *aa jao* gang, my life support in Champaign-Urbana. : Saujanya Gosangari, Abhishek Agrawal, Sameer Talreja, Sama Usmani, Pratik Chaturvedi.
- My oldest gang: Amit Rahid, Tanzeeba Kishwar, Shaila Sharmeen, Sifat Ferdousi, Tanzeem Iqbal, Adnan Eshaque, Abdullah Al Zayeed, Khadem Iftekhar, Ishita Nasreen Khan, Abeer Reza, Afreen Tabassum, Zeeshan Amin, Towsif Mannan, Shikder Riyad, Zurnama Tarannum, Atiya Sobhan and Adnan Kabir.
- My BUET friends who asked me every now and then when I would graduate: Samiun Nabi, Naimul Basher, Helali Mortuza and Saiful Chowdhury.
- Friends that I cannot put in a group: Moz Saleem, Jodie Boyer, Abdullah Moyeen, Atonu Rabbani and Nadia Shams.
- The Nurses who made it possible for me to concentrate on my work.
- My guilty pleasure: Facebook. Thanks for all the stalking opportunites.
- Those who were born: Naajid Sharif, Yafee Khan, Zoya Farhan and Sadit Issam.
- The kids that are growing up too fast into elegant and smart tweens: Tsunami Rahman, Tashriq Khandaker, Tabeeb Khandaker, Farzad Saif, Faaiza Saif, Nuha, Muntaha
- The kids back home: Nafisa, Zahra, Wahzee, Tahzee and Tasfia.
- My adopted son whom I am proud of: Sakeeb Ahsan.
- My friend and foe: Atanu Khan.
- My sister and best friend: Samira Khan. And her best friend: Omar Chowdhury.
- My weird family: my twin aunts Farnaz and Farah Nobi, aunt-sister Tamanna Bashar and uncle-rival Tasleem Bashar, young grand Taz Goznobi and crazy-nerd cousin Harunur Rashid.

- My mother-in-law who took care of the household last few months: Dr Maleka Banu.
- My east coast support hot-line who have magical ways to be here at moments of crisis: Wasima Parveen and Ahmed Mushtahid. All the best for their upcoming twins, Ronon and Shaoli.
- My *nanu* who was there for my mom and sister when I couldn't: Noasi Begum.
- My parents who have made great sacrifices to see me here: Dr. Habib Ibrahim Khan and Azizunnesa Mary.

Table of Contents

LIST OF ABBREVIATIONS

AS	Autonomous System
ASV	Adaptive Selective Verification
BGP	Border Gateway Protocol
DDoS	Distributed Denial-of-Service
DNS	Domain Name Service
HMAC	Hash-based Message Authentication Code
IBQ	Integrity-Based Queuing
IP	Internet Protocol
MAC	Message Authentication Code
ns2	Network Simulator v2
ns3	Network Simulator v3
ISP	Internet Service Provider
RFC	Request for Comments
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VoIP	Voice Over Internet Protocol

1 Introduction

The core of the Internet is structured with a few thousand links and routers that carry traffic long distances, *i.e.*, trans-ocean, trans-continent. These are highly provisioned links and the routers operate at link speed to route traffic. ISPs and servers connect to this core. Regular clients connect to the server through their ISP. A communication is established from the client to server through the ISPs and the core. The attackers use the same infrastructure as the clients use to launch a distributed denial-of-service attack towards a server. The attack spans the clients, the edge of the autonomous system (AS), the core and the server. Defense mechanisms can also be deployed at one or many of these locations. We refer to security investments as infrastructure services that can be installed by the client, its AS or the core. Our proposal is architecture for network availability assurance where such infrastructure services are matched by a network service that incentivizes such investment directly. The incentivizing network service could be located at the core or at the edge of the server AS. Additionally, infrastructure services are graded. Organizations invest in a of grade infrastructure based on their business priorities. But a higher grade of availability is provided for a higher grade of infrastructure. We have two proposals that fit in this graded incentive architecture, Adaptive Selective Verification (ASV) and Integrity-Based Queueing (IBQ). In ASV, clients are in bandwidth competition to get service. IBQ ASes provide their clients with graded source authentication and better authentication is prioritized at the service. Figure 1.1 shows this architecture.

1.1 Problem

Distributed Denial of service (DDoS) attacks are a growing concern as they continue to pose an elevated threat to the reliability of the Internet. Many attacks aim to deplete scarce resources (*e.g.*, CPU, memory, disk) by generating illegitimate requests from one or many, possibly compromised, attacker-controlled hosts [1, 2, 3, 4, 5, 6]. The army of attacker agents (*bots*) can comprise from

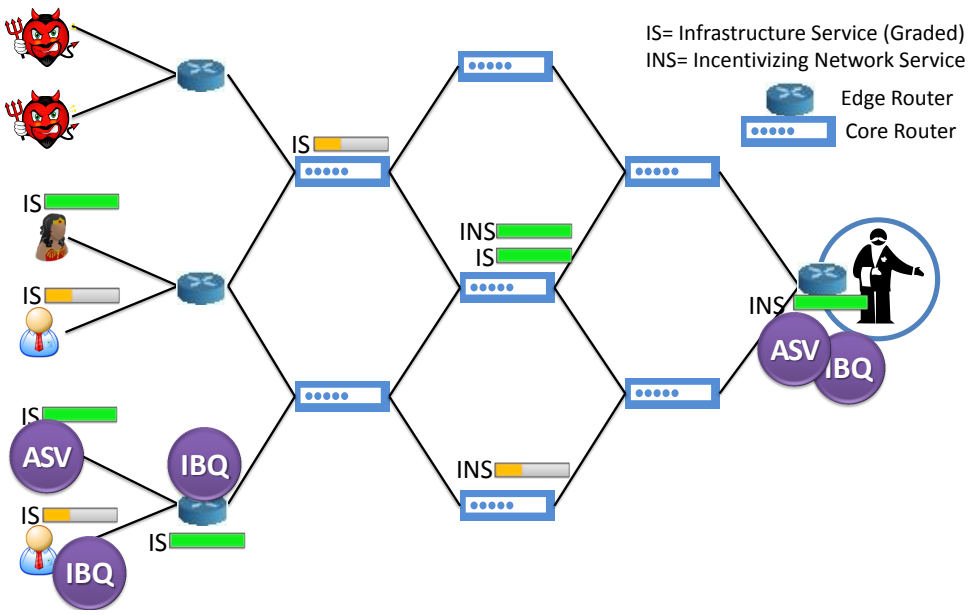

Figure 1.1: A network architecture with IBQ. The nodes that provide a IS or INS service have power bars. The level shows their participation level. They are most likely to use combination of different kinds of IS and INS services.

hundreds to millions of bots located Internet-wide. The time required to process these requests degrades the service to available clients to an unacceptable degree or forces costly over-provisioning by the service provider. Instances of potentially vulnerable services include IKE key exchanges for gateway security association setup [7], legacy and digitally-signed DNS services [8], large file retrievals from web servers, and computationally expensive query processing at database front-ends. These malicious agents send traffic in a high magnitude and frequency and the sheer volume of traffic can render chaos at the host. Hosts under attack exhaust their resources such as computing power, memory and bandwidth. Legitimate clients, on the other hand, each send a single packet requesting a service. These requests will fail while competing with attack traffic for scarce resources at the end host or congested edge routers. In recent years attacks have been launched equally against private hosts and nations [9]. Recent attacks on Estonia and Georgia [2, 10] lasted more than days. Everything tied to the Internet, which includes government and financial services, were at a halt. Attacks have disconnected us from social networks (Twitter) [1] and cost businesses billions of dollars [11]. Table 1.1 summarizes some recent attacks over the decade. The attacks target a diverse set of victims and motives.

Year	Victim	Bandwidth	Motive	Protocol
2003	BetCris	1G	Extortion	TCP SYN, DNS, email, Ping
2007	Estonia	90M	Cyber-war	Ping, TCP SYN
2009	Facebook, Twitter	200M	Political	app layer, http
2011	Sony		Camouflage breach, protest the arrest of a PS3 hacker	

Table 1.1: Some well-publicized DDoS attacks over the decade.

There is a big gap between commercially used defense mechanisms and the research in the area. Commercial tools for defense focus highly on over-provisioning resources and intelligence based intrusion detection systems. Over the past few years it has become easy for small servers to have great amount of memory, cpu and bandwidth available through data-centers. The intrusion detection systems keep a constant check on the traffic pattern for the organization and the data-center. They trigger alerts at any suspicious activity. Servers are also better capable at balancing load. For example, they do not allow a complete downtime of the web traffic if the attack is targeting the mail server. Network administrators are better prepared for typical DDoS attacks with SYN cache management and limits on ICMP messages. Commercially. the focus is at the server and protecting its network.

DDoS has a wealth of research. Broadly the research efforts can be classified as being on identification, filtration and allocation. Finding the source of an attack is difficult. Attackers employ various redirections and spoofed identities to hide their origin. An attack packet is structurally similar to a packet from a legitimate client; just that there is more of them. Research on identification uses cryptographic techniques to lightly authenticate the source of the packet. There are proposals for deployment by clients, the edge or core routers — many times by a combination of these. Researchers have studied deployment of static and dynamic filters at different locations of the Internet once attack flows have been identified. Once legitimate traffic is identified and attack traffic is filtered, researchers have looked into mechanisms to allocate resources among the clients. There also has been lot of focus into separate defenses for making a connection to the server and sustaining that connection.

1.2 Requirements

Our proposal looks into ways that ideas in research could be ported into deployment. It is hard to deploy defense that includes the core. But one deployed only on the attacked server is not sufficient. Our goal is to involve the client and its ISP and provide them assurance of availability.

We observe that, if legitimate clients can get their requests through, the magnitude of the attack traffic is less important. Such situation can be created if the network resources can be divided fairly among all the flows. Under such a regime, attackers are not able to create widespread congestion because they mainly attack themselves with abusive flows. Whatever the magnitude of the abusive traffic, they cannot disrupt service to a legitimate client as they are not able get more resources than their share.

We consider strategies for DDoS protection based on bandwidth payments and anti-spoofing integrity measures where the benefit of implementation primarily accrue to the party that invest in establishing and acting on security assurances. Moreover, the goal is to provide this benefit incrementally so that even if only some ASes provide only some assurance to some servers that implement our technique, then those that make this effort are rewarded for it by improved resilience to DDoS flooding attacks. Ideally the system

- Does not require adoption by core routers but could benefit from their participation;
- Offers direct benefit to parties who participate in it;
- Enables a small number of parties to participate in the system and gain a benefit, but provides greater benefit if many parties participate;
- Provides at least one practical implementation strategy but does not rule out others, including simultaneous use of distinct mechanisms.

In our work we propose two methods for achieving fairness despite no or partial implementation for integrity verification. *Adaptive Selective Verification (ASV)* provides legitimate clients service despite large but bounded attack rates without any integrity infrastructure. The other is *Integrity Based Queuing (IBQ)*. In this work we expect that integrity will not be perfect, but observe that even an imperfect implementation can improve the effectiveness of queuing when parities with

better a integrity level are incentivized. We introduce a system called Cheater that integrates both and builds an architecture that protects clients with or without cooperating ISPs.

1.3 Incentives for Source Integrity

Though the attackers have set up complex underground mechanisms to compromise machines and build botnets, the actual attacks are still simple. Typically an attacker targets a server and uses its large botnet to send a massive volume of packets with invalid source IP addresses [4]. This makes identification and filtering of attack traffic difficult. Spoofing protections have modest incentives for the party applying them—the main benefits are to the party under attack—and sometimes are too coarse-grained when classifying the origin as valid or invalid.

In the Internet it is to be expected that integrity will not be perfect and implementation will be partial. However, even an imperfect implementation can improve the effectiveness of queuing as defense against DDoS attacks when, rather than treating each flow equally, a party with a better integrity level is better treated as an incentive. Our approach is called *Integrity Based Queuing (IBQ)*. IBQ gateways classify packets according to the likelihood that they are from a spoofed origin and allocate bandwidth to high, medium and low integrity flows. Fair queuing for high integrity flows has high effectiveness as each flow gets its own bucket. Gateways impose a differential rate limit while fair-queueing medium integrity flows. The rate limit is imposed as a function of the integrity. The low integrity flows receive general queuing.

Approach. With IBQ, an ISP that properly authenticates the source IP address of a packet gets the best guarantee of service when the server is under an attack. An ISP that authenticates a packet only to its domain but does not bind the source IP to the packet gets a good guarantee of service but not the best. An ISP that lets the attack bot spoof the IP addresses on the packets gets service at approximately the rate as it would without IBQ. As ISPs invest more in infrastructure, they may choose to provide better integrity for clients. Better source validation enables them get better service in the face of an attack and works as an incentive for an ISP to spend on integrity-enhancing infrastructure. In Figure 1.2, we refer to this as *the cycle of integrity assurance*.

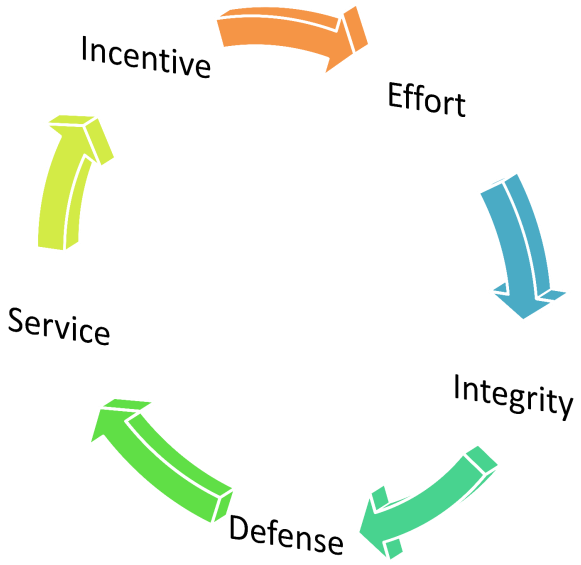

Figure 1.2: The Cycle of Network Availability Assurance.

This proposal has number of advantages. First, it gives a graded definition of integrity: good, bad and middle. Second, it provides a direct measure of incentive to the ISP: as packets are queued based on integrity, an ISP can see how the performance of applications have improved by increasing a grade of integrity. Third, the middle gradation provides a strategy for ISPs to improve their integrity and performance step-by-step.

1.4 Fight for the Right

Recent work has proven the effectiveness of *bandwidth* as a currency. In order to get service, the clients are encouraged to spend more bandwidth by either sending repeated requests from which the server selectively verifies (processes) some [12, 13] (Figure 1.4), or dummy bytes on a separate channel to enable a bandwidth auction [14]. Currency-based mechanisms impose a cost on the system, particularly on the clients, so it is desirable to have *adaptive* counter-measures that are deployed dynamically and proportionally to blunt attacks at minimal cost. [14] describes how to do this for auction-based bandwidth payments but the proposed solution potentially requires significant server state such as tens

of thousands of TCP sessions. The selective verification algorithm in [12, 13] requires almost no server state, but does not include any mechanism for adaptation.

In this dissertation we introduce *Adaptive Selective Verification* which is a distributed adaptive mechanism for thwarting attackers' efforts to deny service to legitimate clients based on selective verification. Our scheme uses bandwidth as currency but the level of protection employed by the clients dynamically adjusts to the current level of attack. At a high level, the clients exponentially ramp-up the number of requests they send in consecutive time-windows, up to a threshold. The server implements a reservoir-based random sampling to effectively sample from a sequence of incoming packets using bounded space. This enables adaptive bandwidth payments with server state whose size remains small and constant regardless of the actions of the attacker. The protocol itself is both natural and simple.

A contribution of this work is an evaluation of the adaptive selective verification protocol with the aim of understanding its performance in practice. Our simulations show that under a time-varying attack, the performance of ASV protocol adjusts quickly to the prevailing attack parameters. The performance is measured in terms of success probability of each client, and the total bandwidth consumed by the clients.

In theory, its performance is compared to an "omniscient" protocol in which all attack parameters are instantaneously made known to all clients as well as the server. Surprisingly, ASV closely approximates the performance of this omniscient protocol. The theoretical evaluation of ASV is out of the scope of this thesis but can be found in [15, 16].

Approach. ASV is a distributed adaptive mechanism for thwarting attackers' effort to deny service to legitimate clients based on selective verification [12] (Figure 1.4). Our scheme uses bandwidth as currency but the level of protection employed by the clients dynamically adjusts to the current level of attack. At a high level, the clients exponentially ramp-up the number of requests they send in consecutive time-windows, up to a threshold. The server implements a reservoir-based random sampling to effectively sample from a sequence of incoming packets using bounded space. This enables adaptive bandwidth payments with server state whose size remains small and constant regardless of the actions of the attacker [15, 16] (Figure 1.4).

The theory is validated with ns2 experiments. We also measure the performance of ASV with pulse attacks and over lossy networks and analyze effect on TCP crosstraffic. The details of these experiments and the proofs can be found in the relevant publications [15, 16].

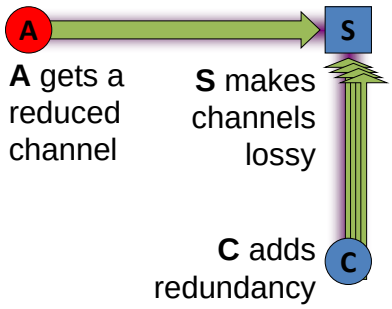

Figure 1.3: Attacker A uses the highest bandwidth available. To invert this effect the server S makes the channel lossy and legitimate clients send redundant requests.

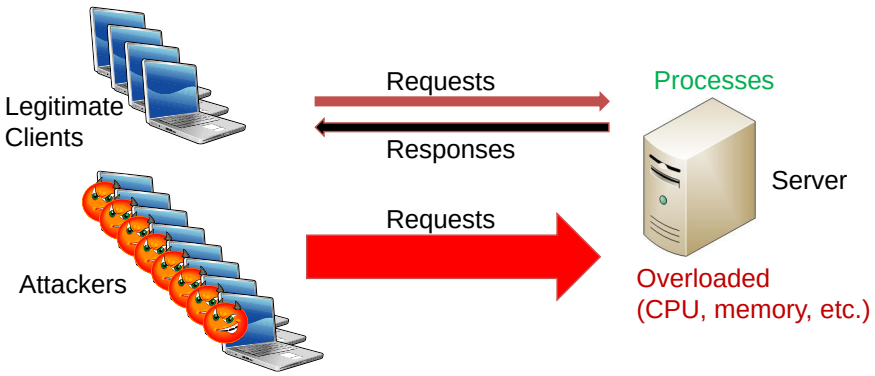

Figure 1.4: Attackers send request at a high rate. Legitimate clients send a request and wait for the response.

1.5 Thesis Statement

ASV and IBQ mitigate DDoS between two domains by directly incentivizing the amount of security effort expended by clients and their ISPs.

1.6 Contribution

To summarize the contributions made by this dissertation are:

- This is the first architecture supporting incentive structures for DDoS defense;
- We introduce two systems, ASV and IBQ, supporting this architecture. ASV exploits bandwidth; IBQ introduces "spoofing index" — gradation of valid and invalid source addresses;
- We conduct the first scalable simulation analysis of a queueing defense driven by a non-conjectural data-set;
- We built "fksim" — first scalable tool capable of supporting a DDoS queueing analysis driven by such dataset.

This graded incentive network availability assurance architecture is presented in Figure 1.1. Both, IBQ and ASV work within the bounds of this architecture. IBQ is evaluated against 2007 attack dataset. We conduct the first comprehensive analysis of this attack. Existing network experiment tools cannot scale for this dataset. *fksim* is tested to scale with ten thousand client nodes. Also, it integrates high-bandwidth trace with TCP flow simulation.

1.7 Organization

Distributed denial of service attacks and defenses are well-studied in the literature. In the next chapter (Chapter 2) we classify and overview some of the classical and state-of-the-art approaches. Chapter 3 present experiments exhibiting the legitimate clients success with Adaptive Selective Verification. We present the loss rate, delay and bandwidth overhead observed by ASV and compare it a naive and an aggressive approach. We also present results on how TCP cross-traffic is effected by ASV.

Integrity Based Queueing is presented in Chapter 4. We discuss the architectural components and related design choices there. In Chapter 5, we bring forth mathematical equations for incentives and gradations and qualitatively discuss the threats that may be present in the architecture. The DDoS TCP simulation tool fksim is presented in Chapter 6. We also discuss excellent performance of

VoIP traffic when using IBQ. These set of experiments were conducted in ns2. We evaluate IBQ with attack data from the wild collected by CAIDA [17] in §6.2 and §6.4. Our simulator *fksim* plays out the CAIDA attack and measure the degree to which IBQ affects clients at various integrity levels. Unfortunately simulators like ns2 cannot handle the load levels for such a study so we have implemented our own simulation framework including a fresh implementation of TCP. Our simulations show TCP flows from a high integrity domain go on with a slightly lower throughput, whereas for other mechanisms, similarly deployed only at the edges, connection rate is 30% or lower.

Cheater in Chapter 7 combines IBQ and ASV for an integrated defense architecture for the Internet. Experiments in ns2 show how they complement each other to fight DDoS.

2 Related Work

In this chapter we are going to discuss network DDoS protections and what each kind has to offer. At a higher level DDoS defenses can be categorized in quite a few different ways, based on where in the network they are implemented and based on the approach. Towards the source of the attack approaches try to focus on intrusion detection and filtering at the client. At the end host measures are taken to detect attack traffic and deploy measures to protect the server so that it can still serve legitimate requests. There are measures that are taken at the network to detect attack traffic and prevent them from reaching the destination host. Most modern approaches require deployment almost network-wide. We are going to discuss four broad approaches: filters, capability, bandwidth based and puzzles. For each approach we will discuss where in the network the implementation goes. We are also going to discuss integrity, fairness and incentives on the Internet. Interested readers are encouraged to go through [18, 19, 20, 21] for detailed accounts of DDoS attack and defense mechanisms.

2.1 DDoS Defenses

2.1.1 Cookies

A cookie verifies that a request from a client comes from a given address and delays the creation of state on the server until there is proof of some level of commitment from the client. Cookies assure, to a reasonable degree of certainty, that a client can receive a packet at its claimed source address. This is done by sending a hard-to-guess value, the cookie, to the client in response to its initial request. The client has to send that cookie back to the server on the next packet. This makes spoofing of source addresses difficult without requiring the server to save state. The idea is used in many contexts: TCP [22, 23], IKEv2 [7]. Lets have a more detail look at the TCP SYN cookie (Figure 2.1).

With TCP, first, the initiating host (client) sends a synchronization packet (the SYN flag set to 1) to initiate a connection. It sets the packet's sequence number to a random 32-bit value x. The other host receives the packet, records the sequence number x from the client, and replies with an acknowledgment and synchronization (SYN-ACK). The acknowledgment is a 32-bit field in the TCP segment header. It contains the next sequence number that this host is expecting to receive $(x + 1)$. The host also initiates a return session. This includes a TCP segment with its own initial sequence number of value y. The initiating host responds with the next sequence number $(x + 1)$ and a simple acknowledgment number value of $y + 1$, which is the sequence number value of the other host + 1.

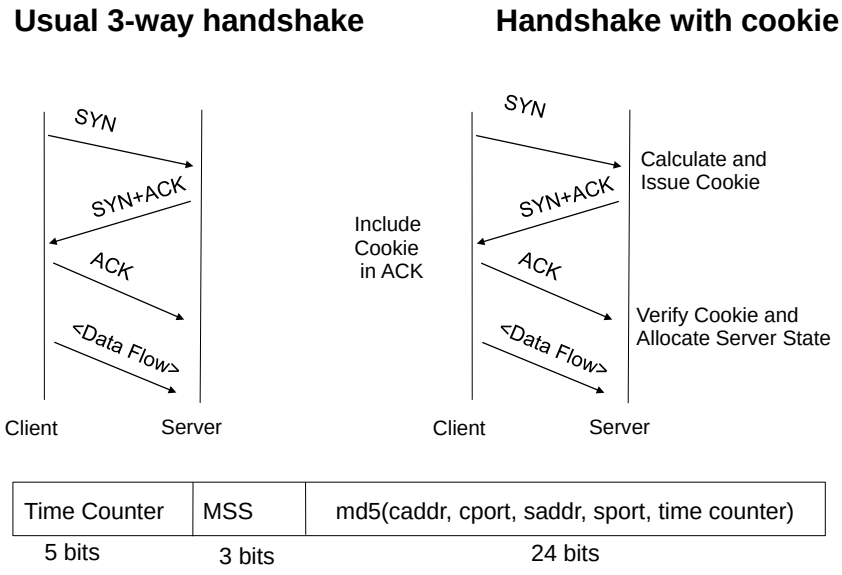

Figure 2.1: TCP with and without SYN cookie. The cookie replaces the 32 bit sequence number and tries to encode information in the original request.

In TCP with SYN cookie, instead of any sequence number y the server sends the cookie. The first 5 bits are t mod 32, where t is a 32-bit time counter that increases every 64 seconds. Next 3 bits are an encoding of a Maximum Segment Size (MSS) selected by the server in response to the client's MSS. The final 24 bits are a server-selected secret function of the client IP address (caddr) and port number (cport), the server IP address (saddr) and port number (sport), and time counter.

TCP SYN cookies are backwards compatible, transparent to clients, conform to RFC requirements for TCP and have no performance impact until SYNs become a problem. They have good performance under attack. But there are also drawbacks: TCP hangs if the ACK is lost and it cannot use large window sizes. The server only has 3 MSS bits to encode all options and rebuild the initial SYN from that. Also the cookie only offers a 24-bit protection. But the idea of cookies have been really useful for recent DDoS resistant protocol developments.

SYN cookies reverse the situation where the client rather than the server saves state. The server is under no obligation before the client completes a round-trip. Cookies are the simplest example where the load is shifted on the clients until they provide some proof of trustworthiness. In a later subsection (2.1.5) we look at some of others. There are also work that specialize in cookies. In CAT [24], all flows must perform a three-way handshake with an in-network element to obtain permission to send data. The three-way handshake dissuades source spoofing and establishes a unique handle for the flow, which can then be used for revocation by the receiver. CAT offers the protection qualities of network capabilities, and yet does not require major architectural changes. Interestingly, dFence [25], a proposal that selectively provides paying customers with protection, uses a middle box with CAT for source authentication.

2.1.2 Filters

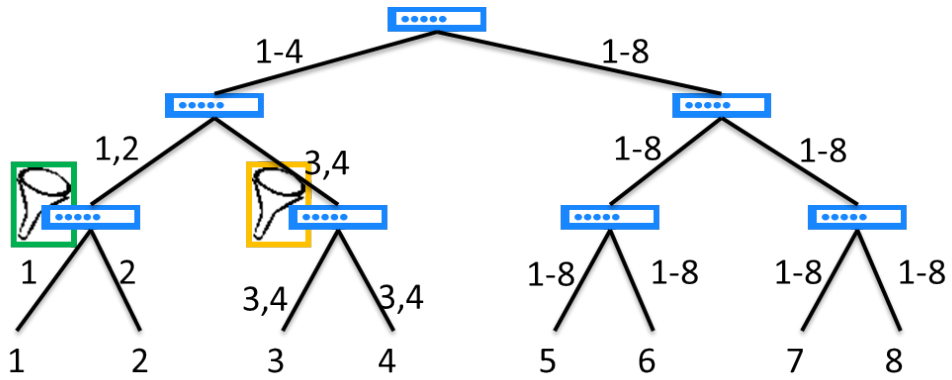

Figure 2.2: Ingress Filtering. The green domain ensures that each node sends packets with its IP address. The yellow one only ensures that packets leaving the domain carry an address from the domain. The unfiltered domains let nodes use any IP address.

Filtering means to detect attack packets and drop them, possibly close to the source. With ingress filtering [26] an ISP or AS can filter all packets sent with an spoofed IP address. That is, if a packet sent from its subnet has an source IP address that is not from that subnet it will be dropped. Though very effective, ISPs have very less incentive to deploy such a scheme. Research by the Spoofer project [27] shows that, as of July 2009, 16% netblocks are spoofable and about 25% AS's do not execute any kind of filtering. Wang *et al.* [28] propose a simple and robust mechanism for detecting SYN flooding attacks. Instead of monitoring the ongoing traffic at the front end (like firewall or proxy) or a victim server itself, they detect the SYN flooding attacks at leaf routers that connect end hosts to the Internet.

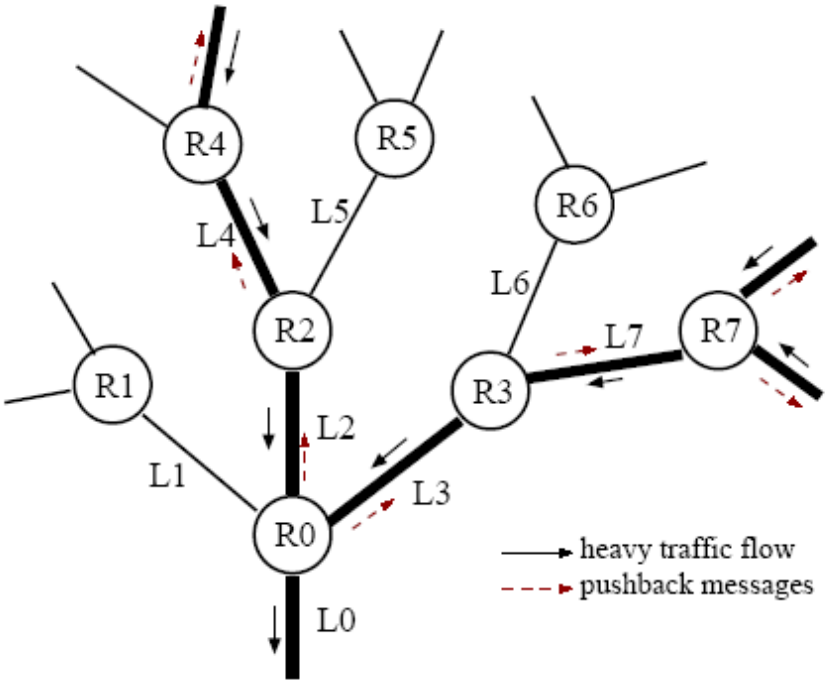

Figure 2.3: Workings of the Pushback filter.

In Pushback [29, 30] (Figure 2.3), severe congestion from the server is back-traced to the sources and filters are installed at routers close to congestion source to limit the attack flow. IP Traceback [31], first described a general purpose trace-back mechanism based on probabilistic packet marking in the network. It allows a victim to identify the network path(s) traversed by attack traffic. This requires a balance of good number of markings on packets, good number of packets with markings and fast algorithms to reverse the markings to path information. Other

schemes followed with the goal to optimize the amount of marks on each packet and processing and storage on routers [32, 33, 34]. The problem is that, attackers are using a million number of bots located in different parts of the Internet. It is lot of effort for a server under attack to detect all those flows and send a filter request and eventually stop the flow. Also as the attackers might keep rotating the attack bots, the attack may have moved when eventually a filter is installed.

Recent work on filtering [35, 36, 37, 38, 39, 40] is a combination of these two basic schemes but use source address validation schemes to minimize spoofing. Routers are designed to mark packets with a signature so that the end host can verify the source of a packet through those signatures. Attack packets then can be differentiated and filtered by use of those markings at upstream routers and the end host. These approaches require infrastructure support from the routers and hosts. The domains are still required upgrade the router functionality.

Route-based filtering [41] takes a different approach. It proposes taking advantage of the power-law property of the Internet and detecting routing anomalies caused by a spoofed packet. The anomalies facilitate filtering or demoting such packets. Alternatively, in Hop Count Filtering [42], using a mapping between IP addresses and their hop-counts to an Internet server, the server can distinguish spoofed IP packets from legitimate ones. But as the number of AS's has grown to almost 40,000 keeping a graph of connectivity and deducing such information has become expensive and tricky. Efforts to create an reasonable map [43, 44] of the Internet with core and PoP information using multiple sources of data, such as BGP announcements and DNS, are active, so is the study to understand its complexity [45, 46]. We discuss further about Internet complexity in Section 2.2). In practice, [41] might be implemented with BGP route announcements and for the least be used for ranking the validity of a packets origin.

In Mayday [47], the overlay nodes perform client authentication and protocol verification, and then relay the requests to a protected server. The server is protected from outside attack by packet filtering rules that can be efficiently deployed in backbone routers. The Detour [48] study noted that re-routing packets between hosts could often provide better loss, latency, and throughput than the direct Internet path. The RON project experimentally confirmed the Detour observations, and showed that an overlay network that performs its own network measurements can provide improved reliability [49]. In fact, Content Delivery Networks such as Akamai [50] that use overlay networks to provide faster service to clients by caching or eliminating redundant data transmission are the prime way for smaller

end host to protect themselves from a DDoS attack. Essentially, they are increasing the amount of resource allocated to their service.

Another thing to note is that these approaches have a free rider problem. Once majority of ASes start participating the rest can benefit. But nobody wants to put lot of work to start off as there is no benefit when only few parties are doing it.

2.1.3 Capabilities

Proof-based defense mechanisms instead of focusing on detecting attack traffic, ask legitimate clients to provide a proof. This proof can be as simple as a cookie transmission stage or a puzzle. Also it could be one of more complex, like a capability. In a network-wide capability defense [51, 52, 53, 38, 40, 54, 55], legitimate clients are given an unforgeable capability to send traffic. Routers can verify this capability and forward each packet. But a client may require round-trips to acquire this capability and special routers are needed . Also, this is orthogonal to filtering and integrity verification. We have to be able to identify a client to give them a token and we need to filter out packets based on those tokens.

In TVA [55, 54] a server assigns its clients capability tokens. A token defines how much traffic that client can send within a time window. Routers on the path to the destination keep track of this allocation and forward packets that have an unexpired token. The token are assigned fairly to all clients. To stop an attacker from spoofing multiple source IP addresses and acquire multiple tokens special routers are installed. To validate the source address, when a packet is travelling through the chain of routers each router stamps it a verifiable hash. These hashes provide the path signature to upstream routers and the destination to identify a flow.

SNAPP [38] uses the path verifiers as the capability token. No separate allocation is made by the servers. This has the benefit of relieving the routers of maintaining bandwidth allocation information for all the senders. Reputation-based ticket-granting [53] proposes how long-term behavior of a client can influence whether future tickets are granted.

Instead of using a fixed allocation, NetFence [56] uses feedback on the pathpins to dynamically change congestion policing. The difference with [30] is that congestion markers are put on request packets by the routers in network so that

they can be applied on the data packets. This way NetFence avoids the problem of tracing back to the congestion as Pushback did.

These defenses require a complex set-up including integrity infrastructure to identify flows so that capabilities can be assigned to them.

2.1.4 Bandwidth

In DDoS defense mechanism based on bandwidth as a currency [12, 15, 16, 13, 14, 57], instead of sending one packet the client expands its traffic so that it is a substantial fraction of the request-pull to the server. This gives the client equal chance of success at the server as the attacker. But this puts much load on the routers and the server (Figure 2.4).

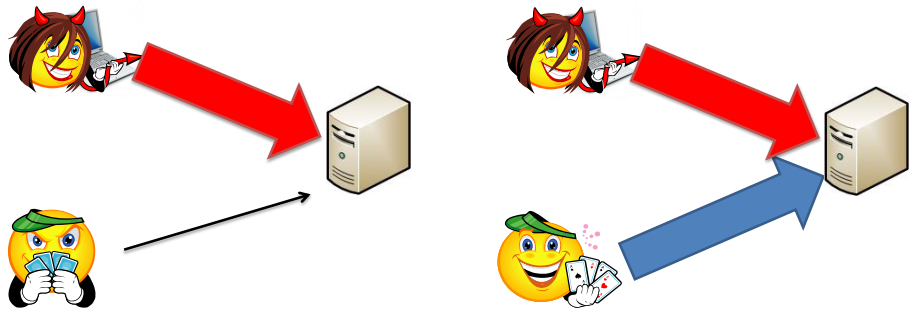

Figure 2.4: The basic tit-for-tat approach of bandwidth based payments.

Perhaps, Burch and Cheswick [58] implement the first bandwidth-based scheme while tracing back. They propose a controlled flooding of links to determine how this flooding affects the attack stream. Flooding a link will cause all packets, including packets from the attacker, to be dropped with the same probability. One can conclude from this that if a given link were flooded, and packets from the attacker slowed, then this link must be part of the attack path. Then recursively upstream routers are coerced into performing this test until the attack path is discovered.

Yau *et al.* [59] propose router throttles to pushback attack traffic. It limits the rate at which an upstream router can forward packets to the server, so that the server exposes no more than its designed capacity to the global network. They

introduce the notion of level-k max-min fairness, where the upstream routers are maximum k hops away. This, too, reflects a bandwidth based approach rather than a filter-based one that it was proposed as. Also, it is an excellent proposal on how to push a filter rule to the edge of network few hops away. The assumption is that the edge has more bandwidth capacity to handle it.

In ASV [16], the level of aggressiveness of the protection employed by the clients dynamically adjusts to the current level of attack. At a high level, the clients exponentially ramp-up the number of requests they send in consecutive time-windows, up to a threshold. This enables them to come par with attackers who are sending at a high rate. There are two benefits of doing this adaptively rather than sending a burst of packets. First, this separates legitimate clients from attackers. Second this enables clients to adapt to any attack rate without response from network or the end server. The server implements a reservoir-based random sampling to effectively sample from a sequence of incoming packets.

Speakup [14] throws bandwidth auctions that allow clients s to build credit by sending bytes to an accounting system and the server takes requests from clients that have built the most credit. With bandwidth based schemes no support is required from the routers. Only the server maintains some state for the reservoir sampling. But the aggressive behavior of a legitimate client may cause problem for other users using the network. To a third party the legitimate clients and the attacker both will look mischievous. So it is important that bandwidth based schemes be used in a controlled setting, such as at the request channel for capability based approaches.

2.1.5 Puzzles

Puzzle-based protections [60, 61, 62, 63, 64, 65] use puzzles to validate a legitimate client. Puzzles can both authenticate a real client and provide a proof of computational effort put by them. A client puzzle protocol is an one-way algorithm, where serving the puzzle and validating the client response is easy thus it is infeasible to make abuse of server resources.

CPP [60] is one of the earliest client-puzzle protocols for DDoS where clients need to use computational power to compute the puzzle and thus prove to the server that are incurring some cost for the service requested. Wang *et al.* [62] propose use of puzzles as payment for bandwidth from routers. One weakness of

this approach is that spoofing attackers might be able to choke legitimate clients just by sending many bogus packets.

Kill-Bots [63], a kernel extension to protect Web servers against DDoS attacks that masquerade as flash crowds. Kill-Bots provides authentication using graphical tests but after the IP address of the client has been validated by a SYN cookie and bloom-filtered against a black-list. Third, Kill-Bots combines authentication with admission control. As a result, it improves performance, regardless of whether the server overload is caused by DDoS or a true Flash Crowd.

Portcullis [64] use puzzles with dynamically adjusted hardness to protect the request channel of a capability protection system.

Gligor [66] has strongly advised against proof-of-work protections. As the attack is hosted from commandeered client machines, he suggests, they would have similar capability to solve such puzzles. He argues for simpler rate-control agreements and proves that they provide a stronger guarantee based on waiting time limits.

2.2 Reflections on The Internet

1. Internet communication must continue despite loss of networks or gateways.
2. The Internet architecture must accommodate a variety of networks.
3. The Internet architecture must permit distributed management of its resources.
4. The Internet must support multiple types of communications service.
5. The Internet architecture must be cost effective.
6. The Internet architecture must permit host attachment with a low level of effort.
7. The resources used in the internet architecture must be accountable.

   D. Clark, 1988 [67]

15 years into the Internet, the year this paper [67] was written, congestion control [68] was introduced in TCP [69]. Thoughts on security were yet to come. The end-to-end system design goals [70] highlighting the importance to keep the core light and fast were presented just a few years back. And as Clark notes, accountability of the resources on the Internet were least prioritized.

2.2.1 Fairness and Queueing

The internal nodes or routers of the Internet have a complicated role in DDoS defense. They operate with the goal to forward traffic fairly at line rate with minimum processing delay and hence avoiding any memory and computation intensive operation. Attack traffic, if not exactly at their door, does not impede on this regular operations. On the other hand, defense mechanisms may add to the delay. We are going to discuss a little history of Internet bandwidth and how packet scheduling and queuing has changed with it.

The Internet was designed with a fairness criteria, that is every party has equal right to service and none should starve. That is if a router needs to schedule, sort and send its packets it has to have a fair policy. Fairness usually is measured based on allocation to flows. A flow is identified by its source and destination addresses and ports. Fair-queuing is one of the earlier methods for such scheduling. In fair-queuing each flow receives its own bucket or mini-queue. If a system can serve at rate r and there are f flows, to preserve fairness, the system would serve an unit from each flow every f/r time unit. There are three problems with implementing this in a network of packets. First, how to implement the system so that it simulates sending a packet from each flow every f/r time. Second, the implementation has to take care of the fact that all packets might not be equal. And finally, with the growing size of the Internet the total number of flows have high space requirements.

Fair-queuing has been an active avenue for congestion control at the core of the Internet [71, 72, 73, 74, 75, 76, 77] whereas use of TCP at the edge. Earlier work discuss what is fair and defines fair queueing [71, 72, 74, 75]. According to max-min fairness criterion every user has equal right to the resource, less demanding users get the resources they want and large users split the rest. To implement such a system Demers *et al.* [71] proposed a bit round robin system. That is, if the line capacity is r bps, a bit is sent every $1/r$, the flows can take turns in sending the bit. To implement this, they proposed pre-calculating a packet's final bit sending time upon its arrival. Deficit round robin (DRR) [78] eliminates this calculation by providing a quantum of service to each mini-queue. If a large packet cannot be sent in the current round the remainder of the quantum is added to the quantum next round. DRR is the implemented version of fair-queuing on routers. In stochastic fair-queuing (SFQ) [79] tries to minimize per flow space requirement. Rather than having a mini-queue or bucket for every possible queue,

the flows that hash to same key share a bucket. Each bucket is than served in deficit round-robin order. Fairness is not affected if hash index is sufficiently larger than the number of active flows.

Fair-queuing was very useful when queuing delay was negligible compared to propagation delay. This line of work was subdued for high state requirements in optical fiber speed [76]. Also Random Early Detection (RED) gateways [80] got adopted providing very good congestion avoidance with the over-provisioned networks rarely requiring additional scheduling. But with increasing volume of DDoS attacks there has been a renewed interest in fair-queuing algorithms [30, 59]. Approaches such as Pushback [30], TVA [55] and NetFence [56] use fair-queuing with specially tagged packet flows as part of the protocol. Preferential queuing is used by routers and ISP's to provide QoS required by some service level agreement. But use of it, specifically based on integrity as a protection against DDoS is yet to be proposed.

Also with renewed interest in fairness the cost of queueing comes into play again. The Coremelt attack [81] describes how among N attackers there are $O(N_2)$ connections, which can cause significant congestion and filter rule overhead in the network core.

2.2.2 Integrity

There are quite a few implementations and proposals provide integrity verification for defense mechanisms. Definitions of a valid client given by them are sometimes either too narrow or too broad. For example, IPsec [82] provides public-key authentication in IP but the cost of signature verification on core routers and the general complexity of tunnel configuration is hindering to its deployment.

In TVA [54, 55], Pi [39] and SIFF [40], each router uses a self-verifiable MAC to verify that the initial request packet had passed through it. This sets up and authenticates the path for the flow. The cost here is incurred by the core routers. MACs are lightweight to be verified at line rate at commercial routers. Passport [37] and StopIt [52] use MACs with pre-shared keys. Diffie-Hellman public keys [83] are shared between all routers. This way they can be verified by others. Here the cost is more on the on the party that puts the MACs. As this is a shared key the initiator has to figure out for whom to put the MAC for. Also an infrastructure for key distribution sometimes becomes complex to integrate with

an Internet-wide protocol. These approaches,too, have a free rider problem [84]. Each party needs either some incentive to participate or an enforced payment for rides. For example, if a small fraction of the ISPs invest in an advance protection the general security of the Internet rises very little, but sufficient benefit is not propagated to the investor for the high cost incurred.

2.2.3 Complexity

Internet has become more and more complex over the years. It has connected billions and changed the world for better but also there are avenues for evil. As Floyd-Kohler [85] noted, networking researchers work from mental models of the Internet. The scenarios used in simulations and experiments reveal aspects of these mental models. All of these modeling assumptions affect simulation and experimental results, and therefore our evaluations of research. But none of them are confirmed by measurement studies, and some are actively wrong. At the same time, simulating how the global Internet behaves is an immensely challenging undertaking because of the network's great heterogeneity and rapid change. The heterogeneity exists at the individual links that carry the network's traffic, protocols that operate over the links, mix of different applications used at a site, levels of congestion seen on different links [86]. But, nonetheless we try to understand it.

Efforts to create an reasonable map [43, 44] of the Internet with core and PoP information using multiple sources of data *i.e.*, BGP announcements and DNS, are active, so is the study to understand its complexity [45, 46, 87]. Many theories and experiments have emerged over the years to map and measure the features of the Internet.

Yook *et al.* [88] suggest that the physical layout of nodes form a fractal set, determined by population density patterns around the globe. The placement of links is driven by competition between preferential attachment and linear distance dependence, a marked departure from the then used exponential laws.

Coates *et al.* [89] observe that the problem of extracting the hidden information from active or passive traffic measurements falls in the realm of statistical inverse problems, an area which has long been of interest to signal and image processing researchers. Paxson [90] observed variance in the end-to-end packet dynamics conducted by tracing 20000 TCP bulk transfers between 35 Internet sites.

Given this complexity tools to simulate the Internet are too simplified or too complex to scale at Internet-wide experiments. Fujimoto *et al.* [91] try to characterize quantitatively the capability of parallel simulation tools to simulate large-scale networks. These experiments include runs utilizing as many as 1536 processors yielding performance as high as 106 Million pps. One can understand the hardness of building such a parallelized testing framework to be used by researchers world-wide. Planet-lab [92, 93, 94] is one such effort. It is a collection of machines distributed over the globe, most of the machines are hosted by research institutions, that is an overlay network testbed, a deployment platform and a microcosm of the next Internet. Emulab [95] is a network testbed, giving researchers a wide range of environments in which to develop, debug, and evaluate their systems. There are installations of the Emulab software at more than two dozen sites around the world, ranging from testbeds with a handful of nodes up to testbeds with hundreds of nodes. The DETER testbed is a public facility for medium-scale repeatable experiments in computer security. Built using Utah's Emulab software, the DETER testbed has been configured and extended to provide effective containment of a variety of computer security experiments, including defense against attacks such as DDoS, worms, viruses, and other malware, as well as attacks on the routing infrastructure. But in any case it is rarely possible to simulate or emulate a DDoS attack with more than few thousand nodes [55, 96].

2.2.4 Validity Gradations and Incentives

Though computers are a binary world we see much more fine-grained detail in computer security now. Access control, for example, has fine-grained detail permissions. Email and spam ranking have a continuous approach. Email goes through series of tests before being discarded as a spam. Sender Policy Framework [97] lets domains define rules for origin of emails from that domain. Domains have control over how fine-grained rules they want to define. At verification, when in doubt email falls into a soft-fail category.

In the recent years p2p systems such as file sharing and ad-hoc networks have widely benefited from incentive-based protocols [98]. Internet security, even with all the complex relationships of all the non-cooperating parties, would advance with a good incentive mechanism.

In recent years BGP and DNS vulnerabilities have caused much havoc. Similar to DDoS defenses, the security benefits provided by the S*BGP protocols do not kick in until a large number of ASes have deployed them. Recently, Gill *et al.* [99], proposed a strategy that governments and industry groups can use to harness ISPs' local business objectives and drive global S*BGP deployment.

Accountability is key to effective DDoS defense [100, 101]. Real-time applications benefit greatly with tiered incentives for integrity protection [102]. An ISP can see direct profit when clients receive great quality streaming media and live sport even during a DDoS attack.

3 Adaptive Selective Verification

In ASV, We consider DoS attacks within the province of a shared channel model in which attack rates may be large but are bounded and client request rates vary within fixed bounds. In this setting it is shown that the clients can respond effectively to an attack by using bandwidth as a payment scheme and time-out windows to adaptively boost request rates. The server will be able to process client requests with high probability while pruning out most of the attack by selective random sampling. ASV is efficient in terms of bandwidth consumption using both a theoretical model and network simulations. It differs from previously-investigated adaptive mechanisms for bandwidth-based payment by requiring very limited state on the server. [1]

3.1 The Setting

Consider the following one-round client-server protocol. The first step of the protocol is an REQ packet from a client $\mathcal{C}$ to the server $\mathcal{S}$. In response, the server sends back an ACK to the client. Each client employs a *time-out window* of duration T determined by the worst case expected round-trip delay between the clients and the server: if after transmission of an REQ a client does not receive an ACK within T seconds he assumes that the attempt has failed. The parameter T is known to the clients as well as the server.

It will be convenient to partition time into a sequence of windows $W_1, W_2, \ldots$, each of duration T. We suppose that the server $\mathcal{S}$ can process requests at a rate of S REQ packets per second so that the number of requests that it can process in any given window is $S\mathsf{T}$. In any given window W, new clients arrive at a rate of $\mathsf{R}(W) = \rho(W)S$ clients per second. The *client request factor* $\rho(W) = \mathsf{R}(W)/S$ determines the fraction of the server's computational bandwidth that is required to process new clients in the window W. We suppose that the client request factors

[1] This chapter includes material from previous publications by Khanna, Venkatesh, Fatemieh, Khan and Gunter [15, 16]

are uniformly bounded away from both zero and one, $0 < \rho_{\min} \leq \rho(W) \leq \rho_{\max} \leq 1$, for some fixed $\rho_{\min}$, $\rho_{\max}$ in the unit interval. We are particularly interested in the situation where $\rho_{\max} \ll 1$ as it may reasonably be expected that REQ requests are typically small packets with most of the server capacity dedicated to servicing the bulk of the communication to follow.

We will assume that a diffuse, distributed, denial of service attack $\mathcal{A}$ on the server takes the form of a potentially time-varying flood of spurious REQ packets aimed at overwhelming the server's capacity to process new REQs. We suppose that, in any given window W, the attack $\mathcal{A}$ sends spurious REQs at a rate of $A(W) = \alpha(W)S$ packets per second. The *attack factor* $\alpha(W) = A(W)/S$ determines the excess computational bandwidth that will be required of the server to process the illegitimate requests in window W. In keeping with the guiding philosophy of the shared channel model that was articulated by the authors to model DoS attacks [12], we assume that the attack factors are uniformly bounded, $0 \leq \alpha(W) \leq \alpha_{\max}$, for some fixed $\alpha_{\max}$, though the upper bound on the attack factors may be very large. Clearly, when $\alpha(W) > 1$ the attack *prima facie* overwhelms the server's capacity to process all requests and, abeyant a protocol to handle overflows, there is the potential for the successful execution of a DoS attack. Our interest is in the particular case where $\alpha_{\max} \gg 1$ and the attack can occur on a scale much larger than the available server computational bandwidth.

In order to focus on the DDoS attack at the receiver, in the next two sections we idealize the situation and assume that REQ and ACK packets are transmitted instantaneously, the round-trip delay occasioned solely by processing time at the server, and that no REQ or ACK packets are lost in transmission. Packet drops at the server are then occasioned only because the arriving request stream from clients and attackers combined exceeds the server's computational bandwidth. Thus, if $\rho_{\max} + \alpha_{\max} > 1$ then it cannot be guaranteed that an individual client's REQ will be processed by the server. If $\alpha_{\max} \gg 1$ it is in principle then possible to almost completely throttle the clients of service and effect a successful DoS attack.

In the sequel, logarithms are to base e. Long proofs are eliminated due to space constraints and will appear in the full version of the paper.

3.2 The Omniscient Protocol

Consider any time-out window W. Suppose that $0 < \rho = \rho(W) < 1$ and $0 < \alpha = \alpha(W)$ denote the client request factor and the attack factor, respectively, over the window W. If clients and server clairvoyantly know ρ and α then it is easy for them to thwart the DDoS attack using a modicum of repetition combined with selective randomized sampling at the server. This simple, if unrealistic, situation is worth analyzing as it provides benchmarks for more realistic situations.

OMNISCIENT CLIENT PROTOCOL: Given α and ρ, each new client in a given window W transmits $\lceil \alpha/\rho \rceil$ copies of the REQ packet in that window. Clients who do not receive an ACK from the server within T seconds leave never to return.

OMNISCIENT SERVER PROTOCOL: Given α and ρ, the server accepts an arriving REQ packet in the window W, independently of other arriving packets, with probability

$$p = \frac{1}{\alpha + \rho\lceil \frac{\alpha}{\rho} \rceil}$$

and discards it with probability $q = 1 - p$. The server sends out an ACK for each accepted REQ.

The total number of REQs transmitted by clients in window W is $\lceil \frac{\alpha}{\rho} \rceil \rho ST$. It follows that, *for any given window W, the cumulative mean transmission bandwidth consumed by client REQs in the omniscient client-server protocol is approximately αS packets per second*. As the number of attack packets received in this window is αST, the total number of REQs received by the server during window W is given by

$$N = N(W) = \lceil \tfrac{\alpha}{\rho} \rceil \rho ST + \alpha ST = \big(\alpha + \rho\lceil \tfrac{\alpha}{\rho} \rceil\big) ST.$$

Accordingly, the expected number of packets processed by the server in window W is given by $pN = ST$ so that the server processes REQs at its rate of S packets per second.

Theorem 1 (Omniscient Connection Confidence)**.** *Suppose $0 < \delta < 1$ is a given confidence parameter. If*

$$\rho_{\max} \leq \frac{1}{-2\log\delta} \tag{3.1}$$

then the probability that a given client has an REQ accepted is at least $1-\delta$ under the omniscient client-server protocol.

Proof. A given client $\mathcal{C}$ transmits $\lceil \alpha/\rho \rceil$ REQs in a window W. The probability that each of these REQs is discarded by the server is given by

$$Q := q^{\lceil \alpha/\rho \rceil} \leq e^{-1/2\rho} \leq e^{-1/2\rho_{\max}} \leq \delta \tag{3.2}$$

in view of the elementary inequality $1 - x \leq e^{-x}$. ∎

Thus, for all sufficiently small client request factors $\rho_{\max}$, the omniscient client-server DDoS protocol accepts REQs from all but a small fraction of at most δ of all clients at a cost in transmission bandwidth of (about) αS client packets per second.

3.3 The Adaptive Protocol

The assumption in the omniscient client-server protocol that clients are continuously aware of the client request factor and the attack factor current in each window is clearly unrealistic, especially given the distributed and—until connection is established—as yet unknown location and legitimacy of the clients and, more critically, the ability of the attack to vary rates continuously and unpredictably. Designing a protocol for the worst-case attack is, of course, possible, but unnecessarily congests the network during periods when the attack is quiescent or at low levels. Our goal, hence, is to design a client-server DDoS protocol which adapts to the behavior of the attack $\mathcal{A}$ *without clients having access to explicit current information about the nature and intensity of the attack.*

In view of our experience with the omniscient protocol, on the client side we are led to seek a replicating protocol where the replication rate used by the clients should ideally be proportional to the current attack factor (and inversely proportional to the current request factor though this is likely to be under better regulation). While the client does not have direct access to this information, he can infer the state of the attack indirectly based on whether he receives an ACK or not in response to REQ(s) sent in the previous window. The failure to receive an ACK in response to transmitted REQ(s) can be construed provisionally as evidence of an attack in progress and the client can then ramp up his replication rate in an effort to counter current attack conditions. Experience with doubling algorithms (or, on the flip side, exponential back-off in TCP protocols) suggests that it would be

profitable to have the replication rate grow exponentially with repeated connection failures (up to a worst-case maximum).

On the server side, a more detailed picture about current conditions can be directly obtained from the ensemble of packets arriving in each time-out window. The server can now very simply maintain the advertised service rate by reservoir sampling to generate a random sample of the sequentially arriving packets. The randomized sampling of incoming packets helps obviate timing attacks or the exercise of other overt control by the adversary over the decision making process at the server, while the adaptive changes in sampling rates that reservoir sampling accords allows the server to respond to changes in attack factors across windows while staying within the budgeted service bandwidth. These considerations lead to the following adaptive client-server protocol.

Adaptive Client Protocol Given $\rho_{\max}$, $\alpha_{\max}$, and T, after each unsuccessful attempt the client adaptively increases the number of REQs sent in the succeeding time-out window up to a maximum number specified by the given parameters.

C1. [*Initialize replication count.*] Set $j \leftarrow 0$ and $J \leftarrow \left\lceil \log\left(\frac{\alpha_{\max}}{\rho_{\max}}\right) \Big/ \log(2) \right\rceil$.

C2. [*Double replication.*] Send 2^j REQ packets to the server.

C3. [*Time-out.*] If no ACK packet is received within T seconds, set $j \leftarrow j + 1$; if an ACK packet is received, exit the initiation protocol and proceed to the next phase of communication.

C4. [*Iterate till exit condition.*] If $j \leq J$, go back to step C2; else exit without communicating with the server.

Adaptive Server Protocol The server performs reservoir sampling on incoming REQ packets during each time-out window. Given S and T, the server processes a random subset of the arriving REQs at a rate not exceeding S packets per second.

S1. [*Initialize window count.*] Set $k \leftarrow 1$.

S2. [*Form reservoir.*] Store the first $\lfloor ST \rfloor$ REQ packets arriving in window W_k in a reservoir. If time-out expires without filling the reservoir, go to step S4. Else, set REQ packet count to $j \leftarrow \lfloor ST \rfloor + 1$.

S3. [*Randomly sample incoming packets.*] If there is an incoming REQ numbered j, accept it for placement into the reservoir with probability $\lfloor ST \rfloor / j$ and discard it with probability $1 - \lfloor ST \rfloor / j$. If the REQ is accepted for placement in the reservoir, discard an REQ from the reservoir uniformly at random and replace it with the accepted packet. Set $j \leftarrow j + 1$ and iterate until the time-out window expires.

S4. [*Time-out*] Accept the packets in the reservoir and send out an ACK for each accepted REQ.

S5. [*Iterate.*] Empty the reservoir, set $k \leftarrow k + 1$, and go back to step S2.

We have streamlined the protocols to focus on the critical ideas. In particular, we adopt the convenient fiction that step S4 in the server protocol occurs *instantaneously*. Thus, there is no gap in time between the expiration of a time-out window expires and the identification of the random subset of packets that is accepted by the server over that window. The reservoir sampling procedure is due to Fan, Muller, and Rezucha [103] and guarantees that if N REQ packets arrive in a given time-out window and $N > ST$, the server accepts a subset of size ST REQ packets uniformly at random from the N packets, and discards the remaining packets.

We call the quantity

$$J = \left\lceil \log\left(\frac{\alpha_{max}}{\rho_{max}}\right) \Big/ \log(2) \right\rceil \tag{3.3}$$

the *retrial span* of a client. In the event that the attack is launched at maximum severity, a client can replicate packets over a period of J windows until he achieves a maximum replication rate of α_{max}/ρ_{max} matched to the peak attack.

Blocking Probabilities Our results say that each client succeeds in establishing a connection with essentially the same confidence guarantee as in the omniscient case at the expense of some added delay.

Theorem 2 (ASV Connection Confidence)**.** *Suppose* $0 < \delta < 1$ *is a given confidence parameter. If*

$$\rho_{max} \leq \frac{1}{-5 \log \delta} \tag{3.4}$$

then the probability that a given client has a REQ accepted within JT *seconds is at least* $1 - \delta$ *under the adaptive client-server protocol.*

Note that the bound on ρ_{max} as given in inequality (3.4) differs from the one in inequality (3.1) by only a small constant factor.

Theorem 3 (ASV Bandwidth)**.** *The expected bandwidth consumption of the adaptive client-server protocol is only* $\mathcal{O}\left(\log(\alpha_{max})/\log(\frac{1}{\rho_{max}})\right)$ *times larger than the bandwidth consumed by the omniscient selective verification protocol.*

It is worthwhile to contrast this bound with a non-adaptive approach that stays in the high protection mode at all times. The bandwidth consumed by such an approach can be $\mathcal{O}(\alpha_{max})$ times larger than the bandwidth consumed by the omniscient selective verification protocol. Thus the adaptive scheme can improve the bandwidth consumption *exponentially*.

3.4 Extensions

In this section we focus on extending the basic adaptive protocol from two aspects. First, we identify and address two concerns with respect to the practical deployment of the protocol. Specifically these relate to the possibility of an unreliable server and network. Second, we elaborate on the ways by which the clients and the server could use the protocol in a more flexible and cost-effective fashion. Specifically we consider regulation of bandwidth devoted to ASV by the server and client, respectively.

Suppose that a server being protected by ASV goes down. Under the current ASV protocol, this would cause a flood of requests from the clients, which would only aggravate the situation. It would be desirable to avoid this unwanted side-effect of the protocol. A potential solution is for the server to provide a special type of ACK, *DACKs (Drop ACK)* at step S3 in Section 3.3 for every request it receives but is not able to process. DACKs serve as an encouragement mechanism which communicates a "please retry more aggressively" message to the clients. A client in round i (which sends 2^i requests), waits for the server's ACK or DACK(s) before moving to round $i+1$. He quits upon receipt of an ACK; else, if he receives any DACK in T time units, he moves on to the next round; failing these two possibilities, he quits. In addition, in order to obstruct the opportunity of creating smurf-type attacks [104], the clients can put a nonce (as a weak authenticator) in each of their REQs, and expect the server to return it in DACKs.

We have so far assumed that the network is reliable. But, what if the network is lossy (*e.g.* due to congestion) and REQs and/or server responses are sometimes dropped? This could result in undesirable scenarios such as a client quitting under perception of a down server. A potential solution is to modify the client's protocol as follows. If no DACK is received for K consecutive packets sent by the client, he quits. This check is only performed at the beginning of each round. Therefore, if the path from a client to the server experiences a drop rate of d, this modification reduces the probability of a client incorrectly quitting to the order of d^K. In addition, this can serve as a crude congestion control mechanism, particularly if K is set to low values. We do not consider this addition a full-fledged congestion control mechanism. In particular, we intentionally do not want the clients to be overly reactive to low-rate packet drops, since this would easily make them back off and provide an opportunity for the attackers to build an advantage. However, if due to a network link attack (which is out of the scope of this work), or any other reason there exists *heavy* congestion in the network, it would be desirable for ASV clients to back off and not further flood the network. We believe by properly setting the value of K, the above mentioned effect could be achieved. We experimentally investigate the effect of network congestion on ASV in Section 3.5.

We can also envision situations in which a server that uses ASV for protection would be interested in devoting less bandwidth to the ASV process. For instance, consider a number of co-located services that share bandwidth. Under certain circumstances, it may not be economical for one service to consume too much bandwidth just to prevent denial-of-service attacks. Another example could be a server that would prefer to reduce the ASV bandwidth consumption in favor of having more bandwidth available for a period of heavy bulk transfers. Concerns of this type may be addressed by having the server inform the clients *not* to send too much traffic. However, this comes at the cost of a potential degradation in service guarantees to the legitimate clients. Translated to our protocol, this means that the server provides clients with a new retrial span J when it is desired that the clients reduce bandwidth consumption.

Consider a function $F\colon r \mapsto (BW, SuccProb)$ where r is a candidate retrial span, BW is an upper-bound on bandwidth consumption, and $SuccProb$ is the expected probability that each client succeeds in getting a request served. Now, consider a utility function $U\big((BW, SuccProb), P\big)$ which takes these bandwidth consumption and success probabilities together with a *system priorities* input P, and outputs the expected utility for the system as a value between 0 and 1. P is a generic

input for system parameters that encode the current priorities and constraints of the system such as the ones discussed above: The goal is to pick a retrial span that maximizes the utility of the system. The new J calculated below is communicated to the clients to be used as their retrial span: $J = \arg\max_r \{U(F(r), P)\}$.

Suppose the server has successfully communicated the value of the retrial span J to the clients. In simple words, J is the maximum bandwidth (price) that the server is willing to accept from each legitimate client. However, what if the client is not willing to spend this much bandwidth to get service? We use a client cost-benefit analysis inspired by [61] to formulate this problem. Suppose each client has a *valuation function* V indicating the value of receiving the service in *some* units (for simplicity *dollars*). Also suppose that the client knows a non-decreasing success function that maps the retrial span r of the client to the probability of succeeding in getting service when the server is heavily loaded. Such a function could be obtained from the server through DACKs. Based on this the client can compute its retrial span as in [61]. On the other hand, there could be a client that is willing to send the required number of repeated requests, but its low bandwidth connection does not allow it to. Suppose that at a round i its connection allows it to send (in a single window) only $\frac{2^i}{f}$ packets for some integer $f \geq 1$. It is easy to verify that if the client continues to send $\frac{2^i}{f}$ packets, in $\mathcal{O}(f)$ windows it will succeed with the same probability as if it sent 2^i packets in a single window (provided that neither the attack rate nor the client traffic changes).

3.5 Experimental Evaluation

To measure the effectiveness of the proposed adaptive mechanism, we perform several network simulations. The simulations provide an opportunity to test the full protocol in settings that reflect the real world situations more closely. Specifically, the simulations aim to evaluate the effectiveness of the adaptive scheme against its non-adaptive counterparts and verify the accuracy of our analytical results. In addition, we study ASV's behavior in the presence of network congestion, as well as its effect on TCP-based cross traffic.

3.5.1 Simulation Setup

The simulations are performed using the *ns2* network simulator for the topology shown in Figure 3.1. The topology shown is dynamic in the sense that, in each

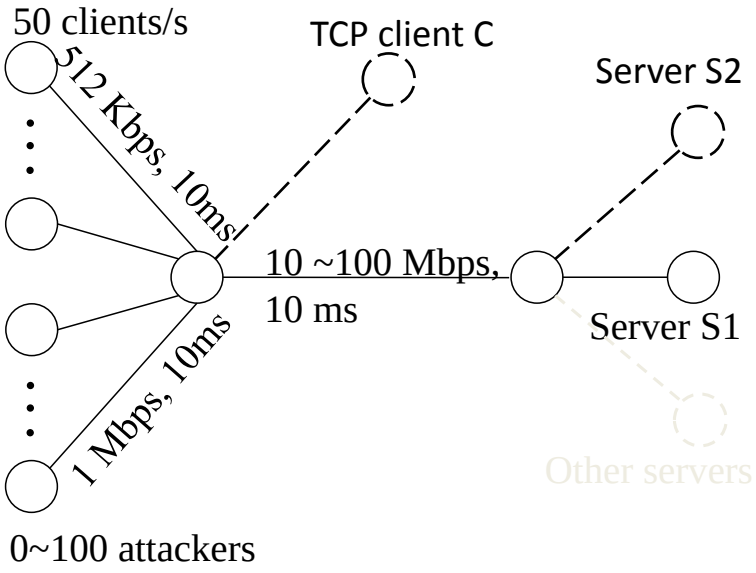

Figure 3.1: Simulation topology.

simulation scenario, the number of clients increases with time. Every second, 50 new clients join the topology and start sending REQs toward the server. Each client that joins the topology needs to get one REQ served. The number of attackers can range between 1 and 100 to represent different attack rates. Each attacker constantly issues 400 REQs/s. So, depending on the number of attackers for each scenario, the attack rate would range between 400 and $40,000$ REQs/s. S, the number of requests that the server can process in a second, is set to 600. Translated into our notation $\alpha_{\max} = 66$, and $\rho = \rho_{\max} = 0.08$. RTT is 60ms and T is set to 0.4s. REQs are 200 bytes, and server DACKs and ACKS are 50 and 200 bytes, respectively. All the communications are over UDP unless otherwise noted. In most of the experiments, the capacity of the bottleneck link is over-provisioned to 100Mbps to avoid any network congestion. However, in particular experiments, we reduce this capacity and create network congestions as needed. The arrival times of the clients and attackers, as well as the inter-packet intervals for attackers are randomized in order to avoid any undesirable deterministic pattern. Given the parameters above, based on the theoretical guidelines in Section 3.3, the retrial span J should be set to 10. We performed separate experiments (not reported in

this paper) using $J = 10$ which easily verify the theorems in that section. In view of practical cost-benefit considerations outlined in Section 3.4, we present here experiments for a retrial span of $J = 7$.

3.5.2 Comparing Adaptive and Non-Adaptive

In order to evaluate ASV against its non-adaptive counterparts, we implement three different client behaviors and compare them against each other in various attack conditions (solid line topology in Figure 3.1). The three client behaviors are:

- Naive: Send one REQ every T seconds. Quit if an ACK is received or JT seconds pass.
- Aggressive (Non-Adaptive): Send 2^J REQs. Quit if an ACK is received or JT seconds pass.
- ASV: Implement ASV for one REQ (which means for a maximum of JT seconds).

Each experiment is performed with one type of client and a fixed average attack rate for 30 seconds which proves to be sufficiently long for the system to stabilize. The results are obtained by changing attack rates across different simulation runs.

Figure 3.2 shows the ratio of clients that succeed in getting one REQ served against different attack levels. Figure 3.3 illustrates the expected time to service for clients that succeed in getting service. And finally, the aggregate bandwidth consumption from legitimate client REQs is depicted in Figure 3.4. We only report on the *client* bandwidth overhead since in each scenario, the number of REQs issued by attackers (and therefore the respective bandwidth consumption) is fixed and the same across all three cases. Client bandwidth consumption numbers are used to compare the bandwidth overhead (cost) introduced by each of the three client behaviors.

It can be immediately concluded from Figures 3.2 and 3.4 that ASV outperforms the Aggressive scheme in terms of success ratio and bandwidth consumption. This proves the effectiveness of the adaptation strategy in raising the costs (bandwidth) in accordance with the attack rate. As expected, the average time to service is lower for Aggressive, however, given the ASV's benefits in terms of success probability and bandwidth consumption, service latencies of at most 2.3s for

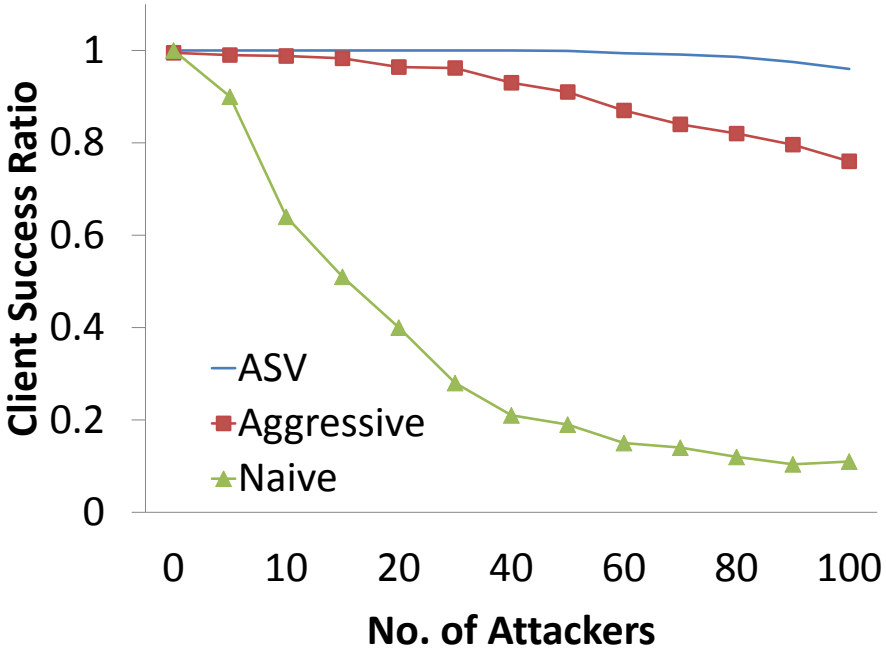

Figure 3.2: The ratio of the successful clients to all clients (1500 in 30s) vs. attack rate.

the fiercest attacks should be considered an acceptable trade-off (Figure 3.3). The Naive clients, as expected, suffer serious failure rates, which underscores the effectiveness of ASV. End-to-end delay for the few successful naive clients is lower than ASV for high rates of attack. This could be due to the uniformness of ns2 scheduler for constant rate traffic. The results also quantify the overhead of ASV, which is a factor of 16 in terms of bandwidth, and 1.5 in terms of service latency in the worst attack scenarios.

3.5.3 Pulse and Variable Rate Attacks

In each of the previous experiments the attack rate was fixed during the simulation. Here, we explore the effect of varying attack rates on clients that implement ASV. In the first set of experiments we subject the system to *pulse* attacks. In these experiments we observe the system's behavior under a 5 second no-attack period, followed by 10 seconds of heavy (but fixed rate) attack, and another 10 seconds with no attack. We performed this experiment for 25, 50, and 100 attackers. The

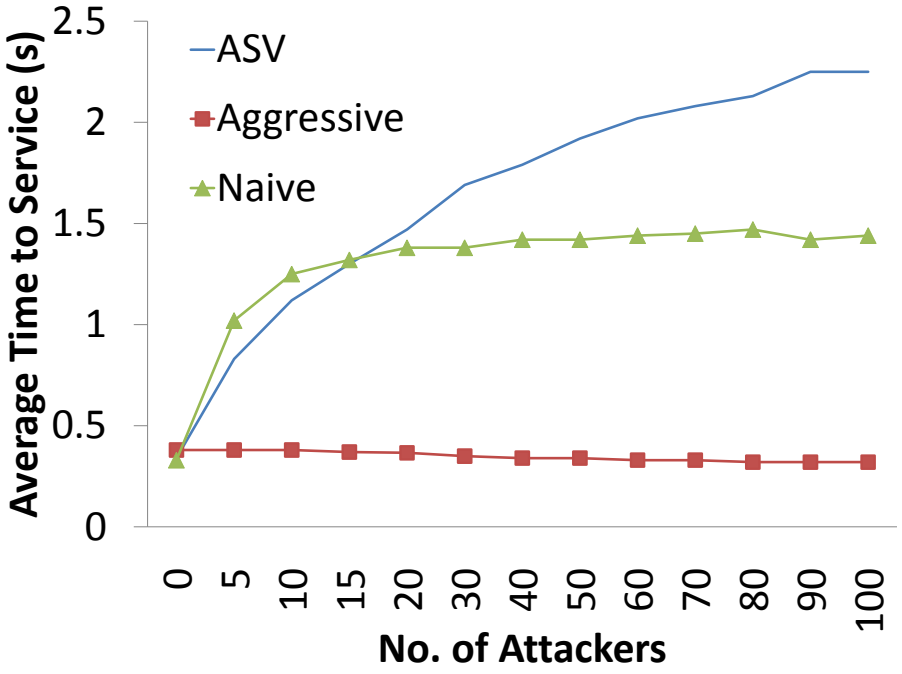

Figure 3.3: Average time to service (for clients that succeed in getting service) vs. attack rate.

detailed results are omitted due to space constraints. However, the most important outcome is that in all three scenarios, in less than two seconds the system fully adapts itself to attack conditions, *i.e.*, success ratio, time to service, and bandwidth consumption numbers converge to the corresponding values in Figures 3.2, 3.3, and 3.4. In addition, after the attack stops, the system recovers to its pre-attack conditions in less than two seconds.

To better understand the effect of highly variable-rate attacks we simulate 45 seconds of variable rate attacks, preceded and followed by 5 second periods with no attack. The number of attackers changes and is $\lceil \exp(r) \rceil$ where r is a floating point number chosen at random from $[0, \ln(100))$ each second. The results are depicted in Figure 3.5.

These experiments show how quickly the system adapts and then recovers to the pre-attack pattern in the presence of pulse attacks. This significantly reduces the attackers' ability to disrupt the operation (and bandwidth consumption) of multiple ASV protected servers' at the same time by attacking them in rotation. The

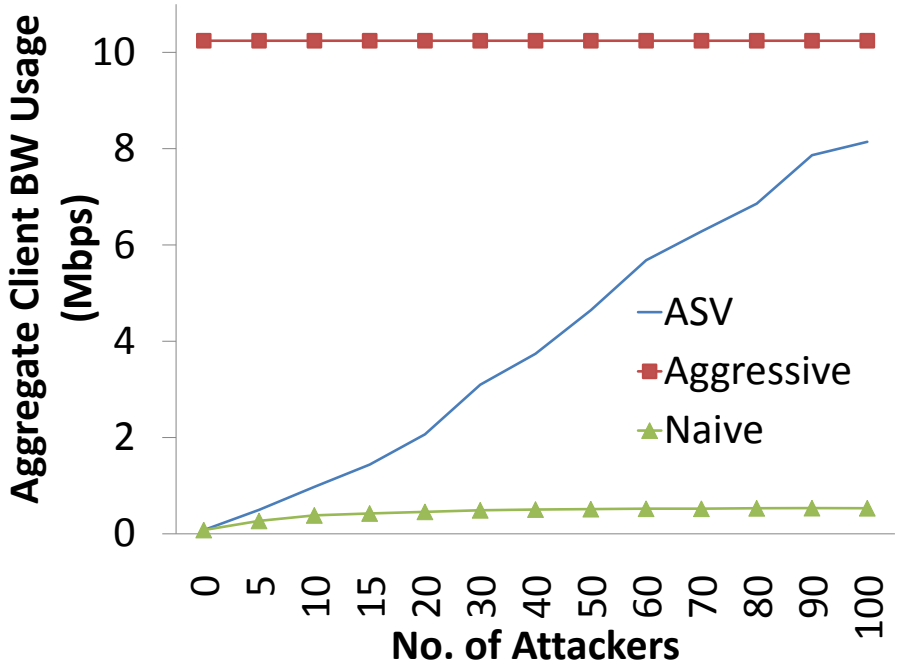

Figure 3.4: The aggregate bandwidth consumption for all the clients vs. attack rate.

variable rate attack experiments (Figure 3.5) show how ASV preserves success ratio, time to service, and bandwidth consumption within reasonable bounds.

3.5.4 Lossy Network

So far, we assumed links are over-provisioned, and thus there is no packet loss in the network. In order to assess the effect of a lossy network, we make the bottleneck link drop packets at different rates. The experiments are performed for $K = 3$ and $K = 7$ with 50 attackers present. In brief, in both cases, for drop rates of up to 30%, there is almost no quitting and client bandwidth consumption stays approximately fixed. However, for network drop rates of 40% to 80%, the quit ratio ranges from 0.08 to 0.71 for $K = 3$, and from 0.01 to 0.32 for $K = 7$. The corresponding client bandwidth consumption ranges from 4.26Mbps to 1.04Mbps, and 4.62Mbps to 4.08Mbps respectively.

Even though enforcing a cap on the maximum number of outstanding REQs (with no DACK) is not meant to be a full-fledged congestion control mechanism,

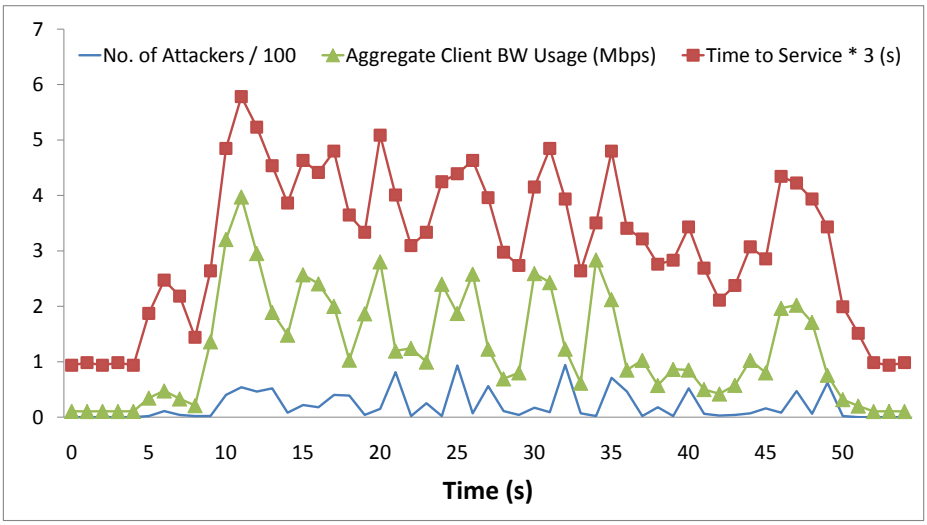

Figure 3.5: The effect of 45 seconds of variable rate attacks on success ratio and aggregate client bandwidth consumption. Success ratio for clients is always 1. Note that clients *joining* the system between times i and $i + 1$ are represented in front of Time =i.

it would still be desirable for the ASV clients to react to very serious network congestions by backing off (please refer to Section 3.4 for details). Additional simulations that we do not report here for lack of space provide evidence that if (for any reason) clients face heavy congestion in the network, they eventually react and stop aggravating the situation.

3.5.5 Effect on TCP Cross Traffic

To measure the effect of ASV on cross traffic we set up the following simulation scenario. We create a client C that is communicating with a data-backup server S2, co-located with the ASV-protected server S behind the bottleneck link. The capacity of the bottleneck link is set to 10Mbps (see the shaded lines in Fig-

ure 3.1). Client C is backing up data on S2, and thus uploads data on S2 at the rate of 512Kbps over TCP. In parallel, we simulate DDoS attacks on S with a clientele of 50 clients per second (Naive, and ASV with K = 3). As before, Naive behavior represents a no-defense base for comparison reasons. The attack rates and the queuing disciplines used in the bottleneck link vary in different scenarios. The queuing disciplines in the bottleneck link are DropTail and Stochastic Fair Queuing (SFQ) with 80 buckets. The amount of data that C can upload to S2 in 30s in each scenario is plotted in Figure 3.6.

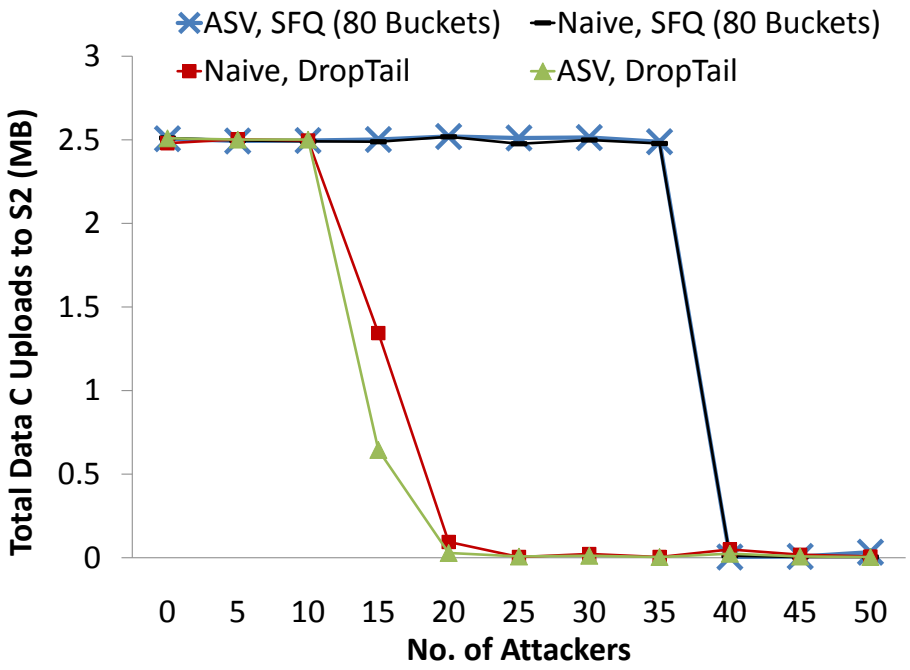

Figure 3.6: The amount of data that client C can upload to sever S2 in 30s. The lines that are close to the horizontal axis represent values in the 7.5KB to 15KB range.

The figure shows that when TCP cross traffic shares a bottleneck link with non-congestion controlled traffic from attackers, it could be seriously throttled. It confirms that unless the network links around a UDP-based service are highly over-provisioned and protected against network link attacks, TCP cross traffic would be seriously harmed in the face of fierce attacks. In addition we observe that Stochastic Fair Queuing (SFQ) would provide better guarantees compared to DropTail *only* until client C's traffic is hashed into the same bucket as attackers

packets. This results in C's traffic being dropped, which in turn causes it to back-off. However, the main result of this experiment is that the attack traffic is the major cause of the TCP client's suffering, and thus compared to Naive (which represents no-defense attack-only scenarios) ASV does not cause any significant *extra* harm to TCP cross-traffic.

3.6 Conclusions

In conclusion, ASV advances the state-of-the art in bandwidth based DDoS defense mechanisms by introducing a distributed adaptive solution based on selective verification. In ASV, the clients exponentially ramp-up the number of requests they send in consecutive time-windows, up to a threshold. The server implements a reservoir based random sampling to effectively sample from a sequence of incoming packets using bounded space. ns2 network simulations of the protocol verify and quantify the effectiveness of ASV against its non-adaptive counterparts and illustrate that under highly variable-rate attacks, the performance of ASV adjusts quickly to prevailing attack parameters. In addition, it is shown that the effect of ASV on the Internet cross traffic is minimal, and comparable to that of its naive non-adaptive counterpart, which represents no-defense attack-only scenarios.

4 IBQ Design

A key challenge to effective countermeasures for Distributed Denial of Service (DDoS) attacks on the Internet is that some of the best strategies require hosts and routers to implement altruistic measures that protect the network as a whole, but provide little immediate benefit to the party implementing the measure. We consider strategies for DDoS protection based on anti-spoofing integrity measures and queueing where the benefit of implementation primarily accrue to the party that invest in establishing and acting on integrity assurances. Moreover, the goal is to provide this benefit incrementally so that even if only some ASes provide only some integrity assurance to some servers that implement our technique, then those that make this effort are rewarded for it by improved resilience to DDoS flooding attacks that rely on spoofing.

4.1 Introduction

Countermeasures like ingress filtering, in which Autonomous Systems (ASes) prohibit spoofed packets from passing into the Internet, can help, but these benefits are limited by the fact that only about three quarters of the ASes perform ingress filtering and most ingress-filtering ASes do not provide full integrity assurances. This situation is at least partially driven by the economics of source integrity protections: in many cases these protections do not directly benefit the party that implements them except to the degree they protect the Internet as a whole.

The idea with IBQ is to enable routers or servers to implement classification of packets based on source origin integrity and then assign different priority and queuing disciplines based on this classification. To illustrate and analyze the IBQ we introduce a design that uses authentication for ASes and fair queuing to implement differential treatment of packets according to whether they have high, medium, or low integrity assurances. We show that resilience to DDoS based on spoofing can be incrementally improved by individual ASs in such a way that

the primary beneficiary of an improvement is seen by the ASs that implement the improved integrity assurances and the routers and servers that recognize this improvement.

We motivate IBQ in Section 4.2, provide the concept in Section 4.3, give and overview of the design in Section 4.4 and provide the detail of the components and protocol in Section 4.5, 4.6 and 4.7.

4.2 What is Not Working Now?

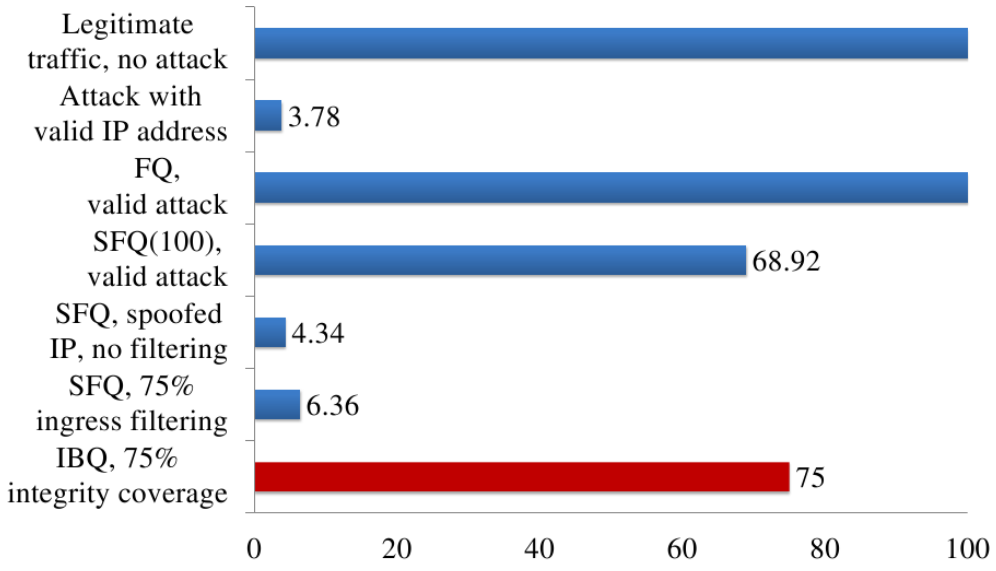

Figure 4.1: Effect of fair-queue(FQ), stochastic fair-queuing (SFQ), spoofing and ingress filtering in DDoS defense.

In this section, we look at the problem of why DDoS defense is hard and even harder to have deployment. To set the goals of IBQ clear we conduct a simple experiment in ns2 to show what happens with fundamental identification and filtration approaches with bots.

We set up a topology with 1024 clients. The clients and attacked server have a link distance of 10 hops and the bottleneck bandwidth is 10 Mbps. Legitimate clients are sending at the rate of only 10 kbps. Clearly, the link is well-provisioned and if there is no attack traffic it is not congested — there is no packet loss (Figure 4.1, first blue bar from the top). We then setup 103 of the clients are attackers each sending UDP traffic at the rate of 10 Mbps. The rate is in accordance with

what typically is seen on the Internet [105]. But as the attack traffic kicks in, about 94% of the legitimate packets are dropped (second bar).

Now, if we introduce *fair-queuing(FQ)* of the packets per source, at the server, the attack packets and legitimate traffic go into separate buckets. Packets from an given attacker are thrown in the same small bucket and are dropped. In the meantime, the more modest legitimate flows are not impaired. When we also do this experiment with *stochastic fair-queuing (SFQ)* where the number of buckets are limited, legitimate clients still do moderately well. SFQ is employed with 100 bins, roughly equal to the number of attackers; when protections are put at the server and the core network, legitimate clients lose only one-third of their packets.

Unfortunately, in real life attackers often use *spoofed* source IP addresses which means they muddle into legitimate flows. As a result, the good effect of fair queuing completely disappears no matter how deeply it is installed in the network. With ingress filtering [26] an ISP or AS can filter packets sent with a spoofed source address. That is, if a packet sent from its subnet has an source IP address that is not from that subnet it will be dropped. Though effective, not all ISPs deploy such a scheme. Research by the Spoofer project [27] shows that, as of July 2009, 16% net blocks are spoofable and about 25% ASes do not execute any kind of filtering. Note that an attacker in those ASes can spoof any IP address on the Internet. Now let's assume Alice is a filtering ISP and has a client with IP address $\mathfrak{i}$. When the victim Bob receives a packet with IP address $\mathfrak{i}$ it has no way to verify if it is from Alice or not. There are two problems of such filtering. First spoofing cannot be stopped until the last IP address is filtered. And second, the good ISP does not observe any benefit of filtering as her clients cannot connect to Bob. For example, in the ns2 experiment we also randomly picked 25% of the clients to originate from domains that allow spoofing. Ingress filtering is applied to the 75% of the clients. Still, about 94% of valid traffic is lost. One would naturally have expected that the result for partial spoofing to be partial success. But the results are worse. The attackers in the unfiltered domain have the capacity to overwhelm valid packets from everywhere. The results do not reflect the proportion of the filtered regions to the unfiltered.

We introduce a strategy for being partially successful in the face of partial spoofing. Identifying a spoofed packet or a legitimate packet on the Internet is a challenging task. Authentication, as we know, is not cheap. Neither it is something that the victim can easily control. The nodes helping out with authentication

have put up with extra work. Additionally, the client needs to identify itself on the first packet. A multi-message protocol may suffer heavy losses during an attack and never be able to complete the task. In this work any effort put by an ISP for integrity protection and verification is awarded by treating its packets preferably.

4.3 IBQ Concept

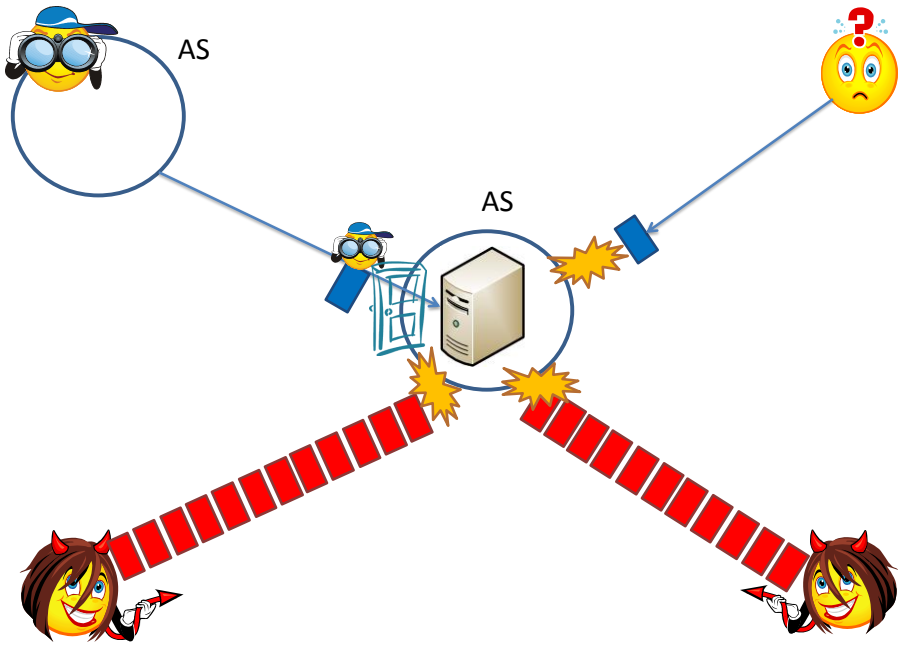

Figure 4.2: The special access entry at an IBQ server opens only for packets from domain that marks them after filtering. Packets from unmarked domains get best effort service.

Our goal is to have availability between domains even when one is suffering a heavy DDoS attack. When two ASes install IBQ, the client AS puts an authenticator on packets from its domain. During a DDoS attack packets carrying the authenticator get preferential service. Attack bandwidth does not affect this incentive-based service for early adopters. Legitimate clients that do not carry such authenticator do not get such service. To achieve this, the participating ASes set up some infrastructures.

When the ISP installs "source integrity token" service it makes an announcement to its customers to validate her spoofing index to the server's "spoofing index service provider". With just few clients responding this is set. Now when a packet

leaves for the server, the source ISP puts a token on it. This token is generated as a MAC with a key shared between the ASes. The shared key is generated from their public keys. The server prefers packets with valid tokens and queues them according to the "spoofing index" which is defined in Section 4.5.

4.4 Architecture

In this section we motivate key components of IBQ. The goal of IBQ is to facilitate communication between two hosts despite attack on the server. The clients, server, client's ISP and en-route AS gateways all can choose to have a role in this facilitation in accordance to our incentivizing network services architecture (Figure 1.1). We define two types of roles: source integrity validating "integrity services" and "integrity-based service providers". Participation in an integrity service or integrity-based service provider is optional and involvement can be at one of multiple graded levels. A party can decide if it wants to install an integrity service and how fine-grained a service the integrity service provides. An integrity service has higher chances of early adoption if the implementer is well incentivized by an integrity-based service provider.

For example, a content-provider *e.g.*, ESPN.com and an ISP, *e.g.* Comcast could diminish the effect of DDoS between them if the ISP installs a high-grade integrity service and the content-provider prioritizes traffic from the ISP with its integrity-based queueing service. The ISP filters spoofed packets to achieve an "source IP integrity level" and puts an "integrity token" on the packets for the server as an assurance that packets originated from her domain. The server has an IBQ enabled gateway and filters packets based on the presence of integrity tokens and the integrity level of the originating ISP. A global "spoofing index table" measures and maintains the integrity level or the spoofing index of all ASes through crowd sourcing the Internet hosts. In Section 6.1 we look at live sports and video services and measures how exactly quality of service improves.

This extends readily to others customers the content-provider might have and who are interested in some change in their infrastructure for a better quality of service. Packets from all such integrity-service domains are treated with high priority during a denial-of-service attack on the server. Moreover, packets originated from domains with a higher integrity level are prioritized over the ones from a

lower one. Thus IBQ incentivizes not only ingress filtering but also promotes better quality of filtering.

Figure 1.1 shows the network design with IBQ. The power meters signify the gradation of service provided. There could be many infrastructure services and network services incentivizing them other than IBQ and ASV. Many existing mechanisms may have these features. But all integrity providing technologies are not integrity services. It must have two features; it should provide a rank of integrity and that should be verifiable by others. Ingress filtering by itself is not an integrity service. Though it gives a rank, it provides no proof. An incentivizing network service, on the other hand, provides a preferential service based on single or multiple security infrastructure ranks and directly improves the service for the implementer. The Internet society might benefit some but the installers benefit a lot. Variables other than integrity could be the unit for ranking and incentives too.

In the Section 4.5 4.6 and 4.7 we discuss the integrity service infrastructures, *i.e.*, spoofing index and source integrity token, and queueing service incentivizing those.

4.5 Spoofing Index Table

Though a good 75% of the autonomous systems on the Internet deploy ingress filtering [27], they differ in depth of filtering (Figure 4.3). We propose 'spoofing index table' for ingress filtering data on domains. A host from, University of Illinois at Urbana-Champaign, can spoof 511 neighboring addresses within its \23 prefix. That is it has a 9-bit freedom to spoof an IP address. We define 9 as *spoofing index* or integrity level for University of Illinois. The lower this number is the better integrity that AS provides. A *spoofing index table (SIT)* is a table providing spoofing index information for all net-blocks. We propose a *spoofing index server (SIS)* that crowd sources clients for spoofing index.

Clients opt to run an integrity level checking program. This sends spoofed packets to the SIS. The program starts by sending a packet with the valid IP address. Next it sends packets with a single bit of the IP address spoofed and then with the two lowest bits. This goes on for the 32-bit space of the IP address. The spoofing index is determined based on the packets received by the SIS. The SIS waits for a few minutes for the spoofed packet. If up to the nth bit of the IP address could be spoofed and received by the SIS then n is the integrity level of the domain and the

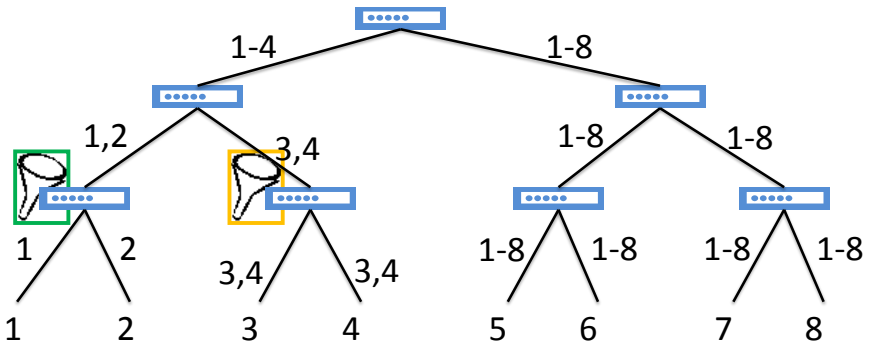

Figure 4.3: Quality of ingress filtering. The green domain filters strictly and checks that each node send with its own IP address. The yellow domain performs a less strict ingress filtering and only verifies that packets leaving the domain carry IP addresses owned by that domain.

SIS stores that information in the spoofing index table indexed by the $32 - n$ bit IP prefix. MIT ANA Spoofer project [27] has been collecting spoofing data from clients similarly since 2003. Their data is anonymized and publicly available.

SIS also functions as provider of spoofing index information. IBQ gateways query the SIS with the integrity level of a domain and use it to decide what grade of queueing service to provide it. SIS entries are cached locally for a week on the server, too. Spoofing index information is mostly stable over years [27].

Crowd sourcing the clients takes the ISP out-of-the loop of spoofing index measurement. The ISP might have a motive to report falsely and get a high integrity index. One way the ISP could achieve false reporting is through the use of specially colluded clients. Malicious clients could pollute the report, for example, by not sending any of the spoofed packets. But to get a conservative assessment, the worst integrity level among those reported is chosen. The ISP could specifically ingress filtering packets to the SIS servers but no to all end hosts. This involves, if not more, equal level of effort as ingress filtering. And studies [106] suggest

that lack of ingress filtering in many cases just represents lack of knowledge or hardware. reported chosen.

An SIS could be the target of a DDoS attack itself. But as a SIT could be cached for a long time the attack will not be successful. We propose to integrate SISes with Domain Name Service (DNS) in future. An SIS has DNS like characteristics and its just natural for hosts to collect spoofing index information along with the DNS entry. Even if multiple hosts start creating their own SIS they can be collected together at DNS easily once there is wide adoption. This way hosts do not need to wait for integration with DNS to start using IBQ.

4.6 Source Integrity Tokens.

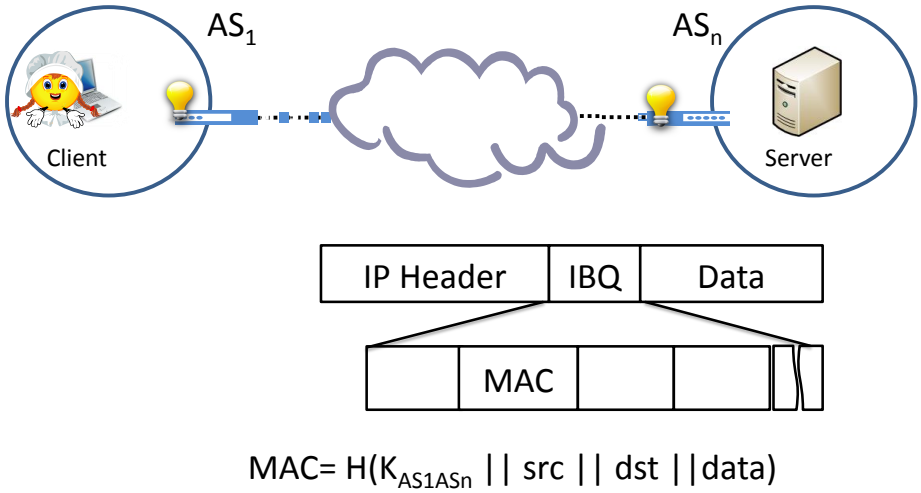

Figure 4.4: Integrity tokens carried by the packet from the originating AS. It has a MAC for the destination AS.

On the Internet, the initial request packets from a client do not bear any proof of origin. IBQ domains attach integrity tokens to their packets. These tokens authenticate the vouching domain to the end host (and en-route domains). An integrity token, represented as a keyed MAC, is inserted for each AS hop that is an integrity based service provider. The key used in the MAC is a shared DH key

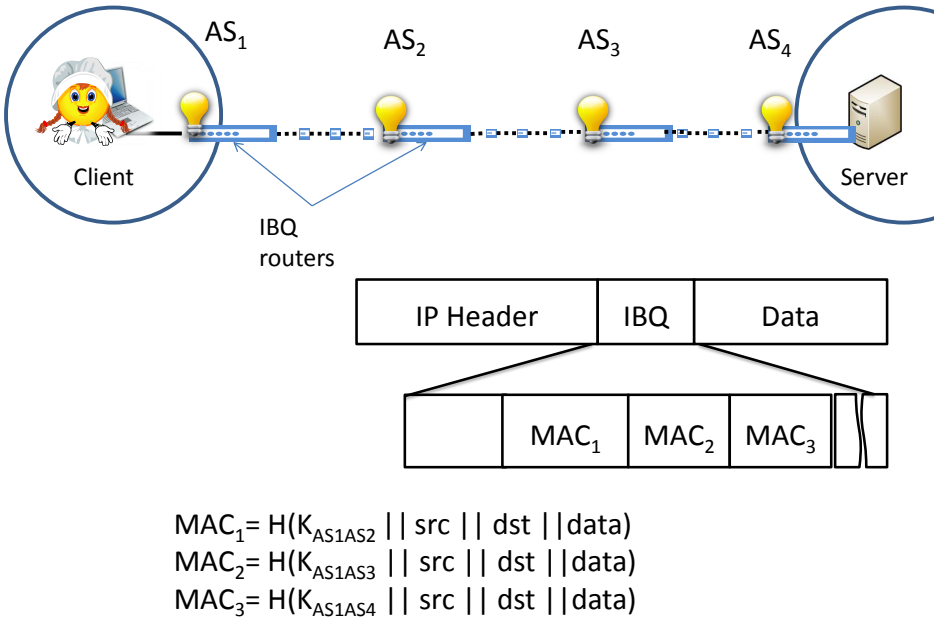

Figure 4.5: Integrity tokens carried by the packet from the originating AS. It has a MAC for every en-route AS and one for the destination AS.

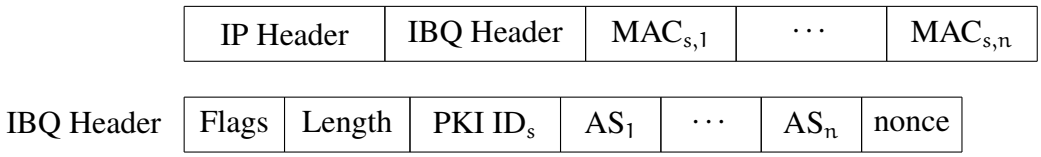

Figure 4.6: IBQ Header

derived from the public keys of the domains. Figure 4.6 shows how the integrity tokens are created. The header contains a PKI ID to identify the party that put the token. Each token is 64-bits. The MAC is computed on the source and destination IP addresses, the IP ID field and the length and the first 8-bits of payload. The token includes the source address so that it cannot be spoofed. Also it has part of the packet data so that attackers cannot steal them from valid clients. This technique is well examined. For the purposes of DDoS defense it was first used at Passport [37]. We suggest a 64-bit tag as per the RFC [107]. This gives us a 2^{64} nonce or sequence number space.

Even if only the client and the server implement integrity tokens IBQ would has great benefits (Figure 4.4). DDoS is mitigated within those two domains. As more and more autonomous systems deploy it, they form a click that has service

despite attack floods. Also the spoofed attacker flow is filtered much closer to the origin.

The Diffie-Helman public keys for the integrity tokens are distributed by the SIS, too. Each AS generates their own public-private key pair and shares the public key value with the SIS. The AS also re-keys at regular interval to provide stronger security. ASes use a known CA for certificates. BGP announcements are used to bind an IP address to a prefix and ASN. There are also commercial and non-profit whois services for this.

Figure 4.5 shows how the integrity tokens are created are passed on. If only the client and the server implement integrity tokens IBQ would have little benefit. But as more and more autonomous systems deploy it, the spoofed attacker flow is filtered much closer to the origin.

MACs are used to provide path verification in multiple DDoS defense approaches. Passport [37] and StopIt [52] use a similar mechanism as ours. One could imagine a MAC being added only for the next hop. The next hop after processing the request could add a new MAC verifying itself to be on the route. This adds the burden of signature to the intermediate nodes but has the benefit of path verification for less stable routes. There could be further study in terms of relative costs of both.

Signatures provide much stronger security and we would need only one per packet. Recent work [108] demonstrated line rate signature verification at a 10Gbps link. But historically network providers have shown less interest in signatures. IBQ implementation will get much easier if signatures are adopted. But for this paper we work with MACs and show how even with the added overhead of key distribution they can be effective (Section 6.5).

Another challenge in this design is verifying the ownership of an IP prefix and certificate distribution. DNS is an option. But DNS itself is vulnerable. DNSSEC has just recently rolled out signed root zone certificates. Many protocols have relied on unsecure DNS for distribution of core information. One great example is sender policy framework for SPAM [97]. SPF uses DNS to specify the IP address range that can send emails for that domain. An email received is then ranked based on this policy. SSL certificates, on the other hand, can be obtained commercially and the vendor decides how to validate ownership for that domain name and the mapping to an IP address. One key point to notice here is that an attack on the signing infrastructure deteriorates the performance of the client if also there is an DDoS attack going on at the server. With IBQ packets are demoted

to a lower priority when something cannot be verified. With wide deployment of IBQ, DNS may take on the role of a distributed SIS. IBQ can use in-band distributions mechanisms such as IPA [96], but there is no added benefit. Also distribution through SIS is more direct and motivates early adoption.

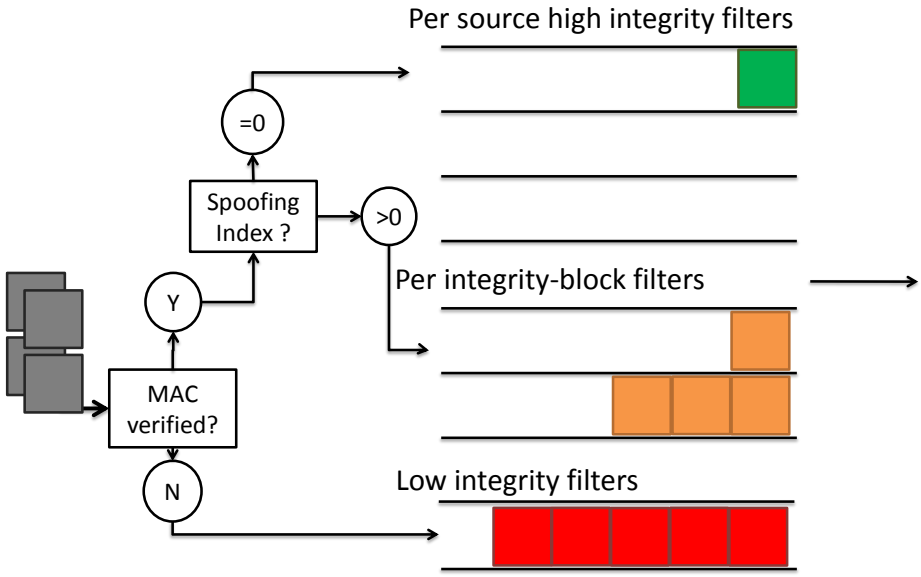

Figure 4.7: Integrity Based Queuing.

4.7 Availability Based on Spoofing Index

The basis of integrity-based queuing is standard fair-queuing where each flow is put into a separate bucket and all the buckets are served in a round robin fashion (assuming equal sized packets). In IBQ, a flow is defined as a group of packets from the same identifiable origin. All packets being identifiable to be from a source are sent into the same bucket. This means any group of packets that can be identified to be from same prefix-block compete for bandwidth. When packets arrive from non-IBQ domains and their origin cannot be verified, they receive general queuing. These are the *low integrity* packets.

Packets that come with an integrity token for the IBQ router are verified against the spoofing index table for their integrity level. If the home AS has an spoofing

index of 0, the flow is classified as having *high integrity* and a filter is created for it. All the high integrity packets are queued in individual per-source buckets. If the spoofing index is higher than zero the packet is in the *middle integrity* category. It is queued with all other packets sharing a IP prefix of ($\backslash 32 -$ *spoofing index*) with it. For example, University of Illinois packets from AS38 will be filtered by $\backslash 23$ prefix address. All the packets having the same $\backslash 23$ prefix will be queued together. We call these filters *per-integrity block filters*. Integrity-block filters are part of the fair queue system. But they achieve weighted queuing as the number of sources addresses that are hashed into a filter differs. For example, for an AS that has a spoofing index of 4 only 16 IP addresses hash into a filter whereas for a spoofing index of 8 there are 256 possible sources. But both of these filters are served equally. This means fewer packets from a high spoofing index AS are forwarded.

Exact implementation of this protocol can be done in quite a few ways. Queues for each integrity netblock might result in an explosion of number of queues. This can be handled by concatenating lower integrity queues to make room for higher integrity ones. A practical challenge for this scheme is adjustment of queue numbers dynamically at runtime. Another option is too set up filters to rate limit flows to follow the preference levels. This really gives the server freedom to decide a particular implementation based on the resources it has. Many newer routers might have better queuing capacity, while filters might be useful for others. In the literature sometimes filters, queues and priority schedulers are used interchangeably. But we felt it is necessary to delineate the differences.

5 IBQ Analysis

In this chapter we analyze the incentive and security provided by IBQ. In Section 5.1, we build an analytic model of the incentive scheme that mathematically relates the spoofing index of a client to its defense against a DDoS attack. In Section 5.2, we take a look at the IBQ protocol itself and and look for security strengths and weaknesses. [1]

5.1 Mathematical Analysis

In this section, we mathematically show that lower spoofing index decreases the loss rate observed by a legitimate client. First we analyze a simple scenario with one client and one attacker. Next, we analyze the probability of the client sharing the same integrity information (Integrity IP prefix) as a bot in the presence of multiple bots. We provide an analysis of how a better integrity level enables a legitimate client to have a better service.

5.1.1 Single Attacking Node

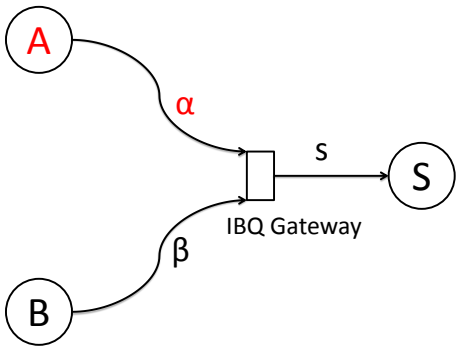

Figure 5.1: A simple three-node topology.

[1] This chapter includes material from a previous publication by Khan and Gunter [102]

Consider the simple topology in Figure 5.1. An attacker A and a legitimate client B are communicating with a server S. The server can process s packets in a particular time window. We provide an analysis of how a better integrity level enables B to have a better service.

The server is over-provisioned for a legitimate client. The client B sends packets at the rate of β per time window where $\beta \ll s$. That means all packets from the legitimate client are processed by the server when there is no attack. The attacker, on the other hand, seeks to overwhelm the capacity of the server to process packets by sending many spurious packets. A sends packets at the rate of α, where $\alpha \gg s \gg \beta$.

In such a scenario the server can process only portion of the incoming packets. With general best-effort service the probability of a client packet being processed is $\beta/(\alpha+\beta) \approx 0$. The attacker traffic takes over all the capacity of the server. When IBQ is deployed by the server this scenario changes. Flows are queued based on their integrity. If all the flows have high integrity, the capacity of the server is equally shared between them. So A and B both get a fair share of $s/2$. Client B sends at a much lower rate β than its fair share, so all of its packets are processed and it is not adversely affected by the DDoS attack.

Now let us consider the scenario where B is in a domain with a spoofing index of i. Also consider that the attacker agent is likely to spoof any address on the Internet that it can. The probability that the attacker is spoofing the address of B is one in a few billions (assuming a 32-bit address space). But the probability that the attacker is in the same domain as B and will carry the same degree of source authentication is, $p = 1/2^{c-i}$, where i is the spoofing index and $c = 32$, for IPv4 addresses. In that case the success probability for client is, $\beta/(\alpha+\beta)$, tends to zero. This same as the case where there was no defense.

The expected number of packets processed for B is,

$$E(B) = sp\frac{\beta}{\alpha+\beta} + \beta(1-p) = \beta - \beta p + \frac{sp\beta}{\alpha+\beta}$$

So the loss rate l for B is,

$$l = 1 - \frac{E(B)}{\beta} = p\left(1 - \frac{s}{\alpha+\beta}\right) = 1/2^{c-i}\left(1 - \frac{s}{\alpha+\beta}\right) \tag{5.1}$$

If the spoofing index i is close to zero the loss rate is negligible. If there is no source authentication ($i = c$) the loss rate is same as the regular best-effort

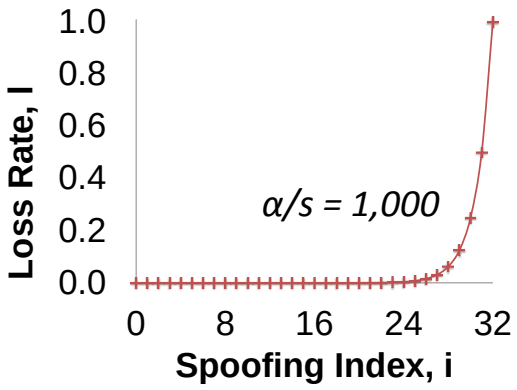

Figure 5.2: Relationship of spoofing index i and the loss rate experienced by a client for a simple three-node topology.

service$(1 - \frac{s}{\alpha+\beta})$. But with each grade of integrity (smaller values of i) the chance gets better. This exponential trend is shown in Figure 5.2. This exponential result extends for any topology as DDoS attack rate is always much higher than what the server can process and p dominates.

5.1.2 Attack with Multiple Bots

Now let us consider that the attacker A is using n bots for his DDoS attack against the server. We show the relationship of number of bots, spoofing index and probability of sharing source integrity with attack traffic.

With general queuing the probability of a client request being processed is $\beta/(n\alpha + \beta) \approx 0$. The attack traffic takes over all the capacity of the server. If all the flows have high integrity, the server's capacity is equally shared between them. All the clients, B and the bots, get a fair share of $s/(1+n)$. If B has a request rate lower than the fair share, that is $\beta > s/(1+n)$, all of its requests are processed. The client can be overwhelmed by the DDoS attack if $n \gg s$, *i.e.*, there is a massive number of bots.

Now let us consider the scenario where B is in a domain with a spoofing index of i. Also consider that the attacker agents are likely to spoof any address on the Internet that they can. The probability that the attacker is spoofing the address of B is one in a few billions, $1/2^{32}$ (32-bit address space). But the probability that a bot is in the same domain as B and will carry the same authenticating source IP prefix is, $1/2^{c-i}$, where $c = 32$. We are going to use $m = 2^{c-i}$.

So far this is similar to the situation where there is only one attacker. The client observes heavy loss when it shares identity with attack traffic. We analyze the probability of that. We can see that this problem is similar to the birthday problem. What is the probability with Internet full of n attackers one or more has same authenticating IP-prefix as A? The probability that a particular bot does share integrity prefix with A is,

$$\frac{m-1}{m} = 1 - \frac{1}{m}$$

The probability of having one or more bots with same integrity IP prefix can be presented with q.

$$q = 1 - \left(1 - \frac{1}{m}\right)^n$$

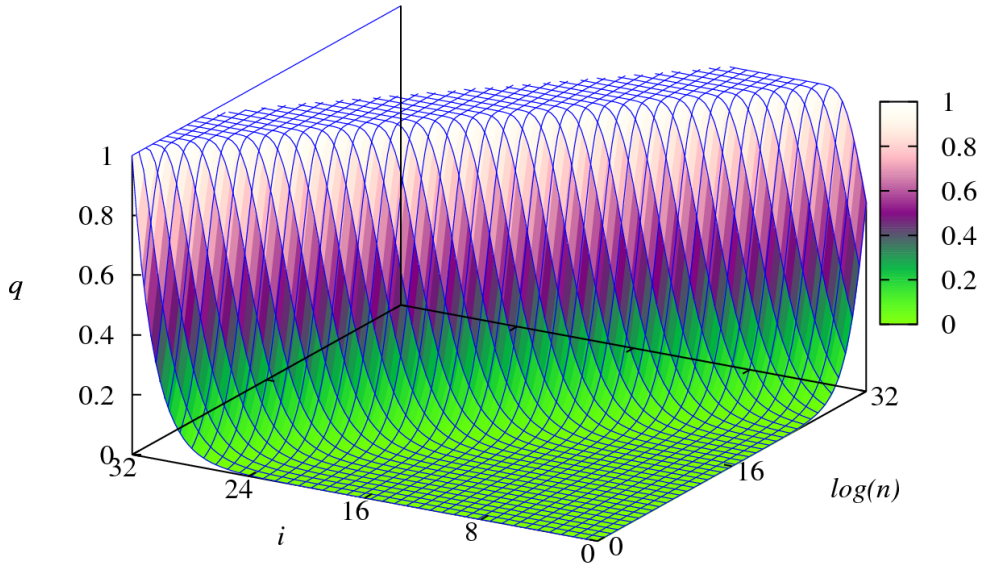

Figure 5.3: Relationship of the spoofing index i of an AS and the probability q that bots are spoofing the IP prefix of a client; n is total number of bots in the attack.

This is plotted in Figure 5.3. If there is no source authentication ($i = c, m = 1$), the client receives general queueing and shares the queue with bots ($q = 1$). In that case there is less chance of getting a legitimate packet through. As i grows smaller, m grows exponentially and $\left(1 - \frac{1}{m}\right) \to 1$. The attacker needs many more bots to exponentiate this fraction to close to zero and make an attack effective (Figure 5.4).

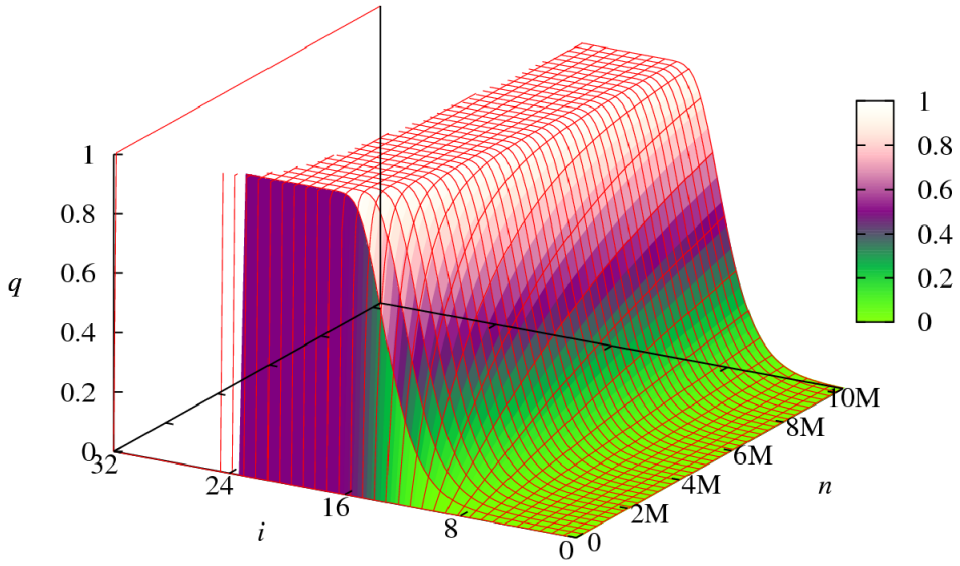

Figure 5.4: Relationship of the spoofing index i of an AS and the probability q that bots are spoofing the IP prefix of a client. Millions of bots are not sufficient to launch a successful attack if the spoofing index is good (low).

5.2 Threat Analysis

In this section we provide an analysis of how our design withstands threats such as malicious clients and gateways collusion and different attack patterns.

Malicious clients

Malicious clients do not get much leverage in IBQ. Malicious clients cannot spoof a valid IBQ header. The authenticator is created by the source AS and the key is known only to the source and the destination AS.

Packets with illegitimate IBQ header cannot overload the destination. MACs can be verified at network line rate. Additionally, if the source AS is an IBQ AS it detects and replaces fake headers. It can limit the bandwidth usage of or blacklist the client. This, in fact, is an excellent way for network administrators to detect bot host machines and inform the owners. If the source AS is not an integrity service provider the IBQ gateway will detect these malformed packets and put them into low priority queueing. Once IBQ has good coverage we recommend to drop such packets as SPF is planning to do for email.

We have discussed Section 4.5 potential malicious behavior for spoofing index crowd-sourcing. To get a conservative assessment, the worst integrity level among those reported are chosen.

Malicious Source Gateway

IBQ is well-protected against malicious AS. Such an AS cannot affect availability for clients of other AS. Also it can enable its own clients to get higher integrity results then the integrity it is providing them. It cannot collude with a client to provide wrong information for the spoofing index table. The spoofing index is verified from the globally available table. The the table is created with information from many clients.

Malicious Routers

A malicious downstream router can override the IP and IBQ header of a packet. It can also intentionally drop packets. This is unusual for few reasons. First packets traverse through domains that they have service level agreement with. So it is economically not profitable for a core router to drop packets. It is hard for a malicious core router to sustain malicious behavior without detection. The Internet-wide faults caused by them initiates fast discovery of the root cause.

Colluding Routers

IBQ is not robust against certain threats. When there are colluding malicious routers the situation is prett bad. However, it has happened on the Internet. Malicious routers may collude and share their private keys for the source authentication. This way they can generate IBQ headers with each others identity which is synonymous to spoofing. This is unlikely as it would violate the security infrastructure investments of the ISP. Traffic with spoofed authenticators would share the authentication prefix with its clients and be queued in same bucket. Also blacklisting such ISP upon detection would discourage them from such an malicious action.

6 IBQ Evaluations

In this Chapter, we evaluate performance of IBQ with simulated VoIP traffic and real attack traces, and analyze overhead of implementing IBQ. Our results show that IBQ incentivizes adoption by ISPs that appreciate reliable performance of real-time traffic. Our work contains the first published use as a test scenario of the 2007 CAIDA DDoS attack dataset. Our tool *fksim* integrates this trace with TCP flows and shows how two domains can mitigate the effect of DDoS between them with IBQ. [1]

6.1 Incentives for Real-time Traffic

We are particularly interested in how real-time applications such as VoIP are affected by IBQ. Traditional circuit-based telephony systems are being replaced by VoIP services on clients (*e.g.*, Skype, Vonage) and ISPs (*e.g.*, Comcast, Turner). Many ISPs are also teaming up with broadcasters for live-streaming of events (*e.g.*, ESPN 360). It is critical for a client that these services perform well under all circumstances. A provider that keeps up the quality even in extreme situations will keep the market share.

6.1.1 Topology Setup

The Internet has grown into billions of nodes and thousands of autonomous systems. Three quarters of the ASes implement some level of ingress filtering and rest none. We use the ingress filtering statistics available from the Spoofer [27] project to model our simulations. As ns2 only scales up to few thousand nodes, we scale down the topology accordingly. We simulate 2048 nodes and 256 autonomous systems. We assume only 3% of the nodes from any AS are active clients. The nodes are connected to a high-speed core network with 64 Mbps links. The con-

[1] This chapter includes material from a previous publication by Khan and Gunter [102]

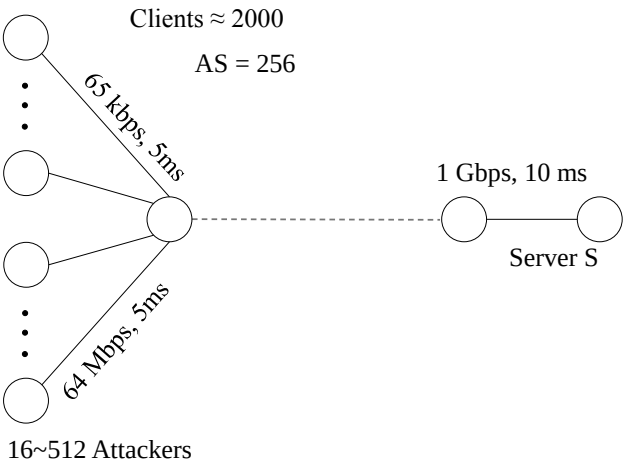

Figure 6.1: ns2 Topology.

gested link at the server has a capacity of 1 Gbps. The link capacities have been scaled down for the scaled down topology.

6.1.2 Attack Parameters

Attackers use the complete link capacity available to them. They also use their spoofing ability. Attackers are placed uniformly randomly within the clients. For our topology we scale down the address space to 2^{16} bits. Each AS has a prefix of $\backslash 8$.

6.1.3 VoIP Parameters

We set the parameters of the VoIP communication for our clients according to the Cisco guidelines [109]. Each client sends packets at a rate of 65 kbps to maintain a good quality communication. Each packet is around 190 Bytes. A reasonable quality call should observe less than 1% loss of packets, less than 150ms of delay and a packet delay dispersion of less than 30ms based on International Telecommunication Union-Telecommunication (ITU-T) standards. At this rate the client links and the bottleneck link to the server both are underutilized. Total client bandwidth is about 3Mbps. Without an attack the packets observe an average delay of 50ms and no variation. Loss rate is 0%.

6.1.4 The Effect of the Attacks

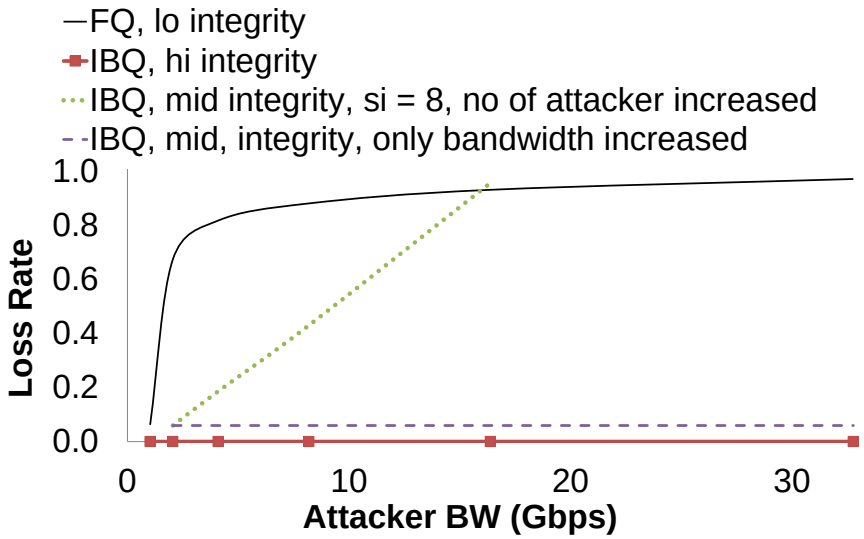

Figure 6.2: Queuing and source authentication have immense effect on performance.

To compare our approach to base cases, we observe what happens to the VoIP service with changing attack rates. We simulate attacks ranging from 1 to 32 Gbps in bandwidth. The attacks are on a lightly-used link. Though the clients have very good VoIP capability when otherwise that capability vanishes very fast with an attack. Any mechanism that tries to sort out the good and bad packets fail due to the lack of source authentication. We observe that, IBQ performs very well even when using mid-range source validation and only fail when attackers fall in the gray area on the integrity block and choke up the queues. Figure 6.2 shows the result of this analysis.

6.1.5 Comparison to other fairness schemes

Next we compare IBQ to existing methods that deploy source authentication and fairness as a measure of defense. The best comparison should be when both protocols are fully deployed. We experiment with deploying a per-AS fairness in ns2 topology. We use the same parameters as before. The results are shown in Figure 6.3. Even at its best, a per-AS scheme has a performance similar to IBQ with mid-integrity flows. Attackers distribute their agents globally and a single bot in such a domain could choke all the legitimate clients in that domain.

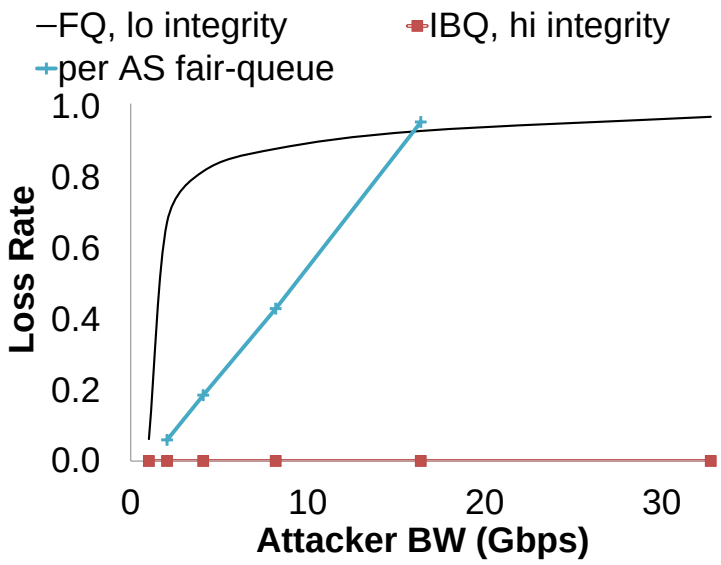

Figure 6.3: IBQ compared to fully deployed per-AS queuing such as TVA.

Partial deployment with per-AS or path queues are tricky. If we imagine the link between the server and the clients to be replaced by a complex topology, one would wonder how would the authentication mechanism effect the communication. TVA request channel prefers the newest token on packet rather than the old one while queuing. That might defend against some edge cases of attack, but hampers performance in a regular spoofed packet attack. In the next section we evaluate and compare results for partial deployments.

6.1.6 Performance Incentive

In this set of experiments, we fix our attack rate to 8 Gbps and relate application performance to the spoofing index. We have shown in Chapter 5 how performance improves exponentially with integrity level. The results in Figure 6.4 validate that analysis.

One important question that rises from these results is this: should all packets be sent through a preferential service or should that service be used only for the request packets? Every protocol starts with few initial packets that request entry into the protocol. Any authentication mechanism used by the application layer starts thereafter. But the tricky part is how to identify and sort those initial packets. In our experiments we observe that the VoIP packets suffer from a loss rate higher

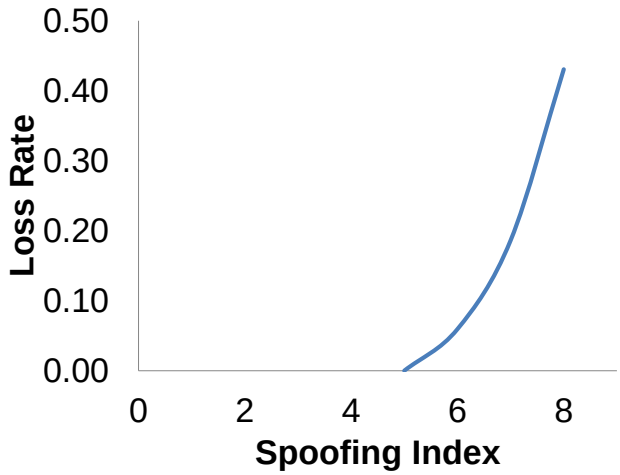

Figure 6.4: Loss rate as integrity level increases.

than the application can tolerate. So VoIP would benefit from using IBQ to make the call and then establish a secure communication channel. But VoIP, like other network protocols, comes in many flavors, some open and some proprietary. It would help if request packets were easily recognizable by core-routers.

6.2 DDoS Attack Dataset Analysis

CAIDA has recently made anonymized data on a DDoS attack on the day of August 4, 2007 available by subscription. We show that a client in an ISP using IBQ and a server can indeed communicate when this attack happens. First we analyze the characteristics of this data. Next we use this to set an attack profile in ns2 and verify IBQ performance for clients using both UDP and TCP communication.

The CAIDA 2007 DDoS attack dataset has very interesting traits. We provide the first comprehensive analysis of this data. The CAIDA trace is for a bandwidth based attack that lasts about an hour. As CAIDA describes it, the attack starts around 21:13 in some unspecified host. Within a few minutes the network load increases from about 200 kbps to about 80 Mbps. Though there is no mention of the actual bandwidth of the bottleneck link it would be fair to assume that it is 80 Mbps. The trace itself contains only attack traffic. The victim's response rate is about 700 kbps.

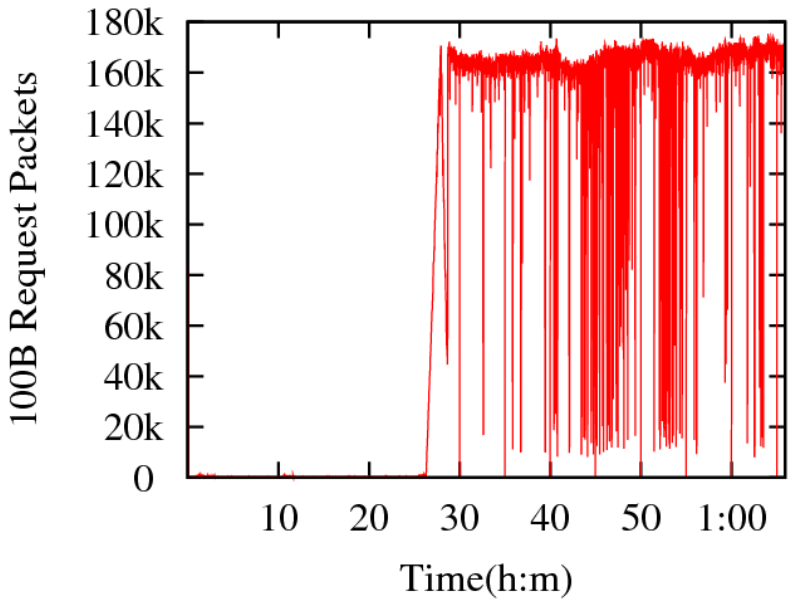

Figure 6.5: CAIDA Data Set for a DDoS Attack.

Our analysis of this trace shows interesting results. The attacker uses around 9311 unique source IP addresses. These bots sent about 400 million packets over the hour of the attack. The packets were mostly ICMP and TCP SYN packets of about 100 bytes. The destination address on all the packets were the same.

We also looked at how many packets were sent using each IP address and at what rate packets were sent every second (Figure 6.5, 6.6, 6.7, and 6.8). The attacker seems to be testing the network capacity for about first 20 minutes and then there is a rapid hike in the rate. The choice of addresses seems to be pretty well distributed over the IP range except two gaps. The attacker might be avoiding the pool of reserved and local addresses. But we cannot be sure as the trace has been anonymized.

The trace has another fascinating trait. The attack while sorted by address or time looks uniform. But if we look at how many packets were sent by each attacker we can see that it has a staircase pattern (Figure 6.8). Some IP addresses occur only once while the largest two occurrences are 340,067 and 225,399. The average number of packets sent by an IP address is 38,626.97 while the standard deviation is 38,430.12. If we sort them by number of packets sent and take the difference between each pair the average is 24.21 with a standard deviation of 150.08.

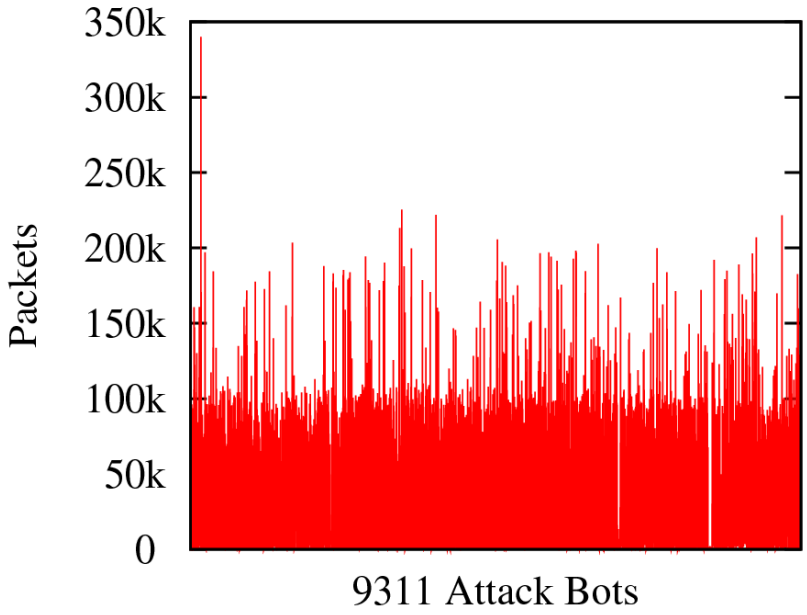

Figure 6.6: CAIDA Data Set for a DDoS Attack.

Let us look at the sending pattern from few of these addresses. Packets from each address also arrive in a staircase pattern. Rather than using new bots existing bots keep increasing the number of packets sent. Figure 6.9 shows the sending pattern for the addresses that were used the most. Also, given that the attack took 15 minutes to start it is surprising how abruptly it ends.

6.3 fksim: Trace and TCP Tool

Our C++ tool fksim replays the CAIDA trace along with simulated TCP traffic in less than a second. fksim is optimized to maintain minimal state information for the nodes and the flows. This enables it to run efficiently. Simulators such as ns2 and ns3 do not scale for 10,000 nodes—ns2 scales up to 2,000 and ns3 up to 1,000. They are general purpose network simulators that maintain the overhead of packet headers and complex routing information. Each node added to the topology introduces new possibilities that are stored throughout the simulation. For DDoS experiments the ability to simulate many clients is important. The added overhead with each new entity in existing systems make that impossible. Also replaying traces with simulations is not trivial.

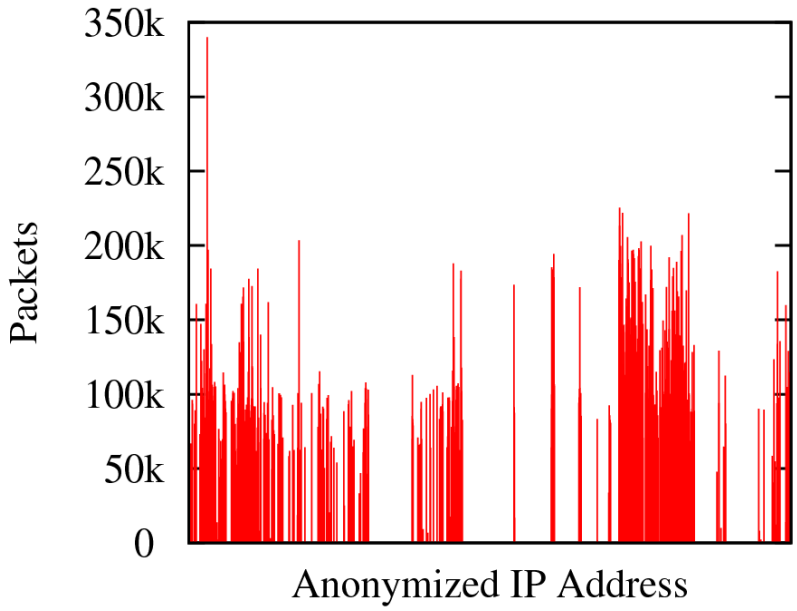

Figure 6.7: CAIDA Data Set for a DDoS Attack.

fksim does not maintain any header, topology or per node state. Instead, loss and delay are calculated frequently and results are stored. The reduced overhead also makes it possible to simulate TCP clients and assign some memory to TCP state information. We implement TCP according to the related RFCs. Our implementation yields same results for bulk transfer as an ns2 agent. This supports our claim of correctness.

6.4 Simulation of Legitimate Traffic

We validate that indeed if a server is using a incentivizing integrity and a client domain is using an integrity authentication, DDoS attack is mitigated between them. We use our custom C++ simulator to use this trace as the attack traffic and introduce lab-created legitimate traffic. We have multiple objectives for these simulations. First we want to evaluate how the legitimate client performs while such an attack is going on. Second, we compare performance of IBQ with TVA. This experiment is conducted for simulated TCP.

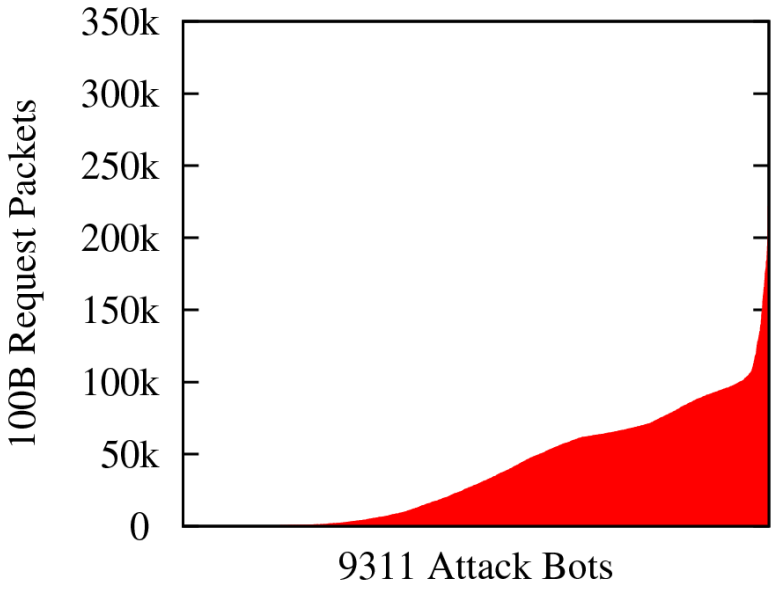

Figure 6.8: CAIDA Data Set for a DDoS Attack.

6.4.1 TCP Experiment Setup

We evaluate throughput and connectivity of long running TCP connections between the client and the server. The objective is to show that with only the client domain installing source integrity infrastructure and the server providing service based on quality of integrity attack is mitigated between them. Though more services at other places in the network have a more global effect on DDoS, local services benefit the parties installing them the most. It is interesting to see how TCP, itself a fascinating protocol, work outs its traits during an attack and with other queuing protocols such as IBQ and TVA. TVA service differentiates between the first packets and others whereas IBQ does not.

Next, we discuss how each protocol is set up for the experiment.

IBQ. IBQ clients carry a MAC from the domain that is verifiable by the server. We take the spoofing index of the domain to be zero. There is only a single domain using IBQ. So, the server, accordingly, allocates only 1% of its resources for IBQ integrity-token packets and rest for legacy traffic.

TVA Setup. TVA [55] uses per-source fair-queuing for request packets and per-destination queues once the channel is set up. For different protocols a channel setup may take varying number of messages. TVA uses the first packet to iden-

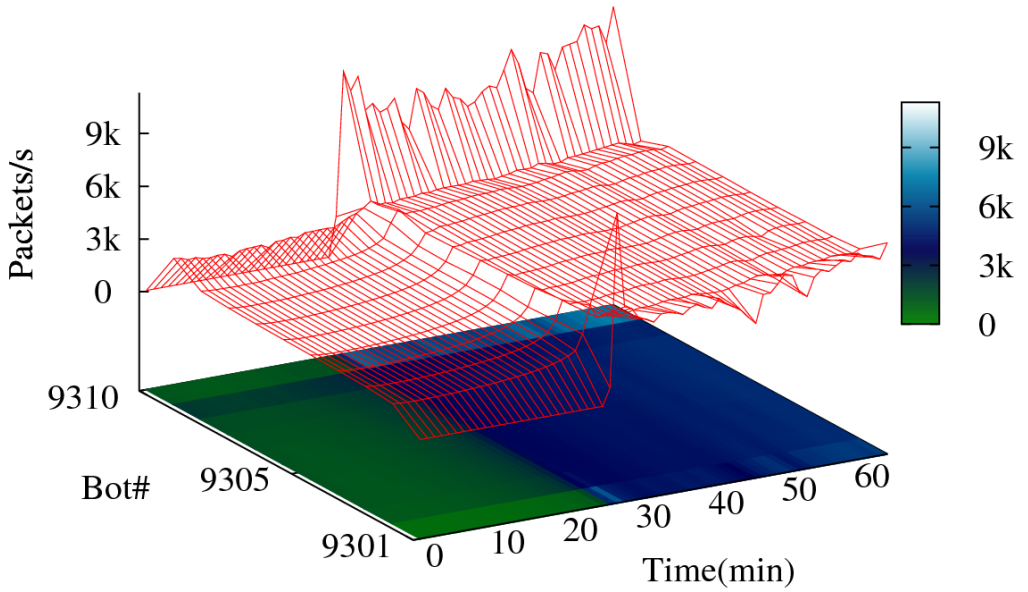

Figure 6.9: Top 10 highest sending bots

tify the path based on the self-verifiable MACs put by en-path routers. We test partially deployment of TVA only at the originating ISP and the server. So for this setup legitimate packets get a tag from the originating domain. It is as simple as spoofing an IP address to add a fake tag to its packet as no other party can verify the tag. This is challenging for incremental implementation of TVA on the network. Early adopters see no benefit. In fact there request packets are put in a constricted channel with the attack packets.

TCP. We configure TCP with regular bandwidth setting. The receiver window size is 64KB with a segment size of 536 bytes. The slow start threshold is also 64KB. The connection timeout is 3s and the persistent timeout is 6s. We use a long lasting TCP connection to transfer a stream of bytes to the victim. Web traffic follows a pattern more like UDP where after the connection is typically one or two more transmissions fetch the webpage. The disconnection that the user experiences frequently is reflected with UDP loss rates.

Server. The victim server reflects the victim in the CAIDA attack. Its victim capacity is set up to 1Mbps. The attack traffic throughput is 80Mbps. This capacity

is much greater than the 20kbps traffic it was observing before that attack as well as sets the attack to be severe.

The parameters used for network protocols here are comparable to the CAIDA attack. As the Internet bandwidth increases we will see much higher bandwidth attacks on higher bandwidth links as well as higher bandwidth network protocols. The characteristics of the results hold true there too.

6.4.2 TCP Experiment and Results

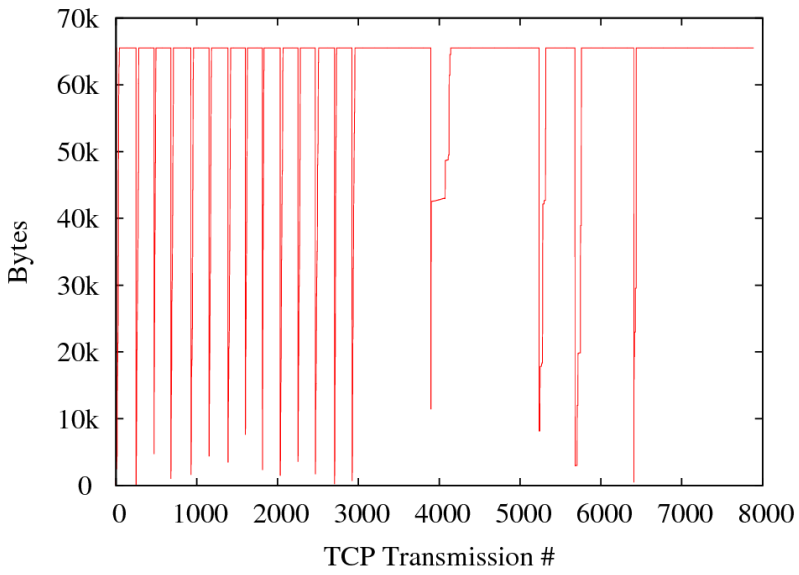

Figure 6.10: Transmission of a TCP agent without any other traffic.

We analyze the sensitivity of TCP to the CAIDA attack traffic in terms of longevity and throughput and compare the result with IBQ and TVA employed. We start a TCP agent regularly every second. We show the results 5 minutes apart for clarity of the figures. The attack varies over time. TCP connection setup works differently than data transfer in a connection that is already set up. Starting new agents regularly tests both of these behaviors. Some results are presented in Figure 6.10, 6.12, 6.11, and 6.13.

Without any attack traffic the client yields a good-put of about 970 kbps, transferring about half a gigabyte of data. The agent keeps sending the maximum window size of 64 kilobytes every roundtrip (500 ms). It back-offs a bit when a few packets are dropped as her sending rate is higher than the line rate. Without any defense in place TCP performs well at the beginning of the attack. The agent

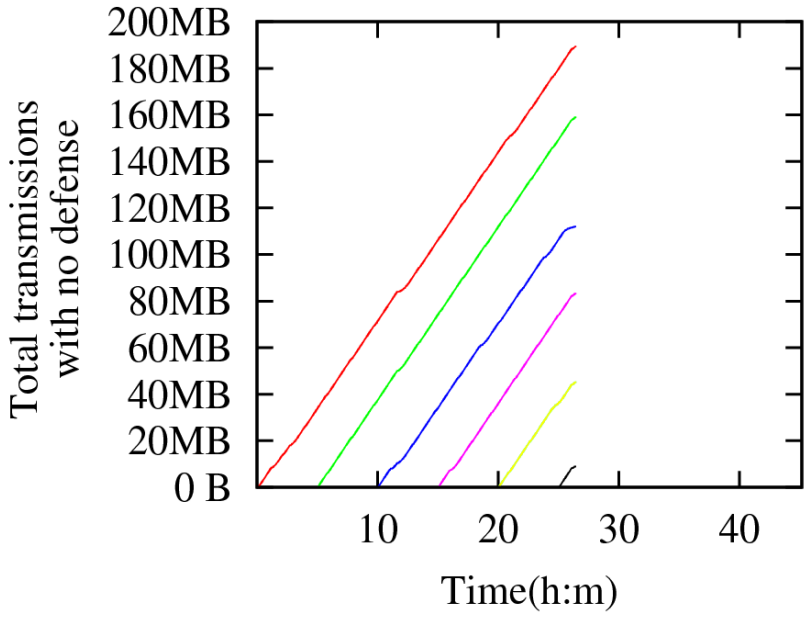

Figure 6.11: Transmissions of TCP agents under attack without any defense.

starting at the beginning of attack yields 180 MB but disconnects once the attack hits link capacity. But after some time the TCP agents fail to even establish a connection. The agents already connected rarely can make a successful transmission. TCP's exponential back-off during congestion effects its throughput. After several timeouts it makes very few transmission attempts during rest of the attack period.

With IBQ even with constricted capacity the client observes good consistent throughput of about 546–661 kbps throughout the attack. TVA suffers extensively as the attacker is using spoofed tags and all the request packets are queued together. Deployment of more TVA routers at the core network or even more at the edges alleviates the effect of spoofing. As, we have argued, the limitation of this approach is that such deploying parties are not directly incentivized because only traffic between two other parties is aided.

But it cannot be defined as a directly incentivizing network service.

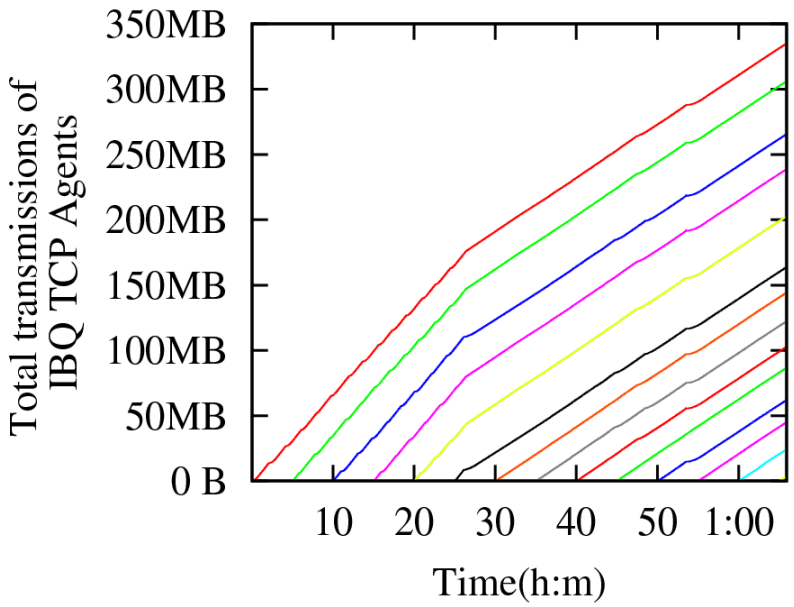

Figure 6.12: Transmissions of TCP agents under attack with IBQ.

6.5 Overhead

6.5.1 Spoofing Index Table

The spoofing index table has similar overhead of a BGP table. Each entry is indexed by an IP prefix and contains the spoofing index and the PKI ID for that prefix. Spoofing indexes verified by clients in that prefix-block are available in the table. The size of the spoofing index table depends on the granularity of spoofing and filtering boundaries. We make a design decision to use BGP IP prefix for indexing the spoofing index table. This limits the growth of the table but challenges the correctness.

For example, University of Illinois at Urbana-Champaign announces a 16-bit IP prefix in BGP but has a spoofing index of 9. This means that it has 128 netblocks, each of size 512, within this prefix. A client would validate spoofing index of one of these netblocks. To reflect this we add a fourth column in the table indicating how many netblocks have been validated. The spoofer project [27] found 35% of the spoofing boundaries to be exactly the BGP prefix. It makes sense for domains to filter at traffic at the border. Nevertheless, this check keeps the table and the domains honest.

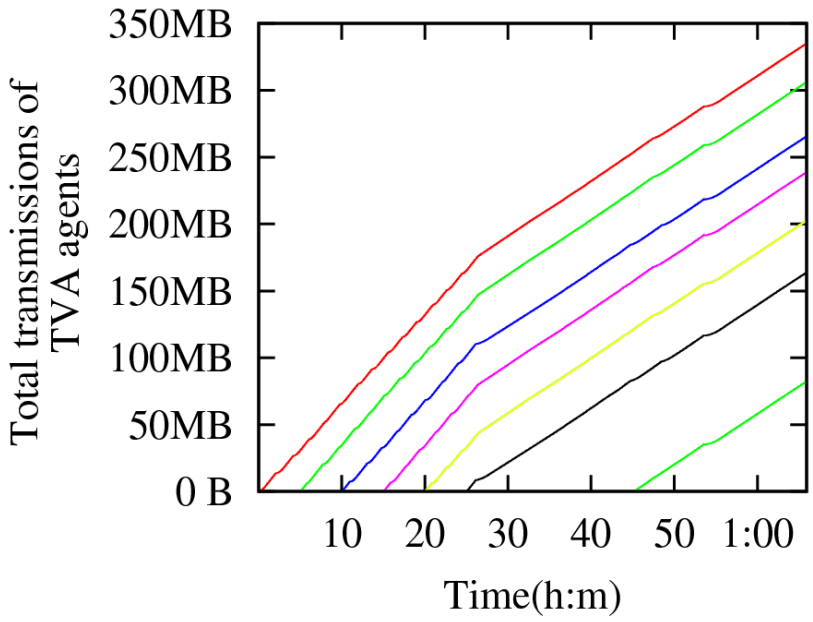

Figure 6.13: Transmissions of TCP agents under attack with TVA.

Any contradictions are reflected with a separate entry in the table. For example, may be the Department of MoreSecure filters more rigorously and spoofing index is 0. Though they share IP prefix with a bigger organization their longer prefix is an entry in SIT, too.

The size if SIT is comparable to the global BGP table which currently has 40,000 entries. Each entry being 12 - 24 bytes (IP-v4 and IP-v6) the table is under 1MB.

6.5.2 Spoofing Index Server

The spoofing index server has three roles. It coordinates with clients to determine the spoofing index of an IP prefix. It responds to queries for the spoofing index and public key certificates for domains. It works as a CA to validate prefix-ownership requests, rekey and revoke keys.

The load on SIS for crowd-sourcing is lower than that of a web-hosting site. The majority of the work is done by the client. The SIS gives a timestamped cookie to the client to verify a spoofed packet received is indeed originated by its request. We use connection timeout of 3s similar to TCP for this. The client sends out 10 spoofed packets for testing each spoofing index and 20 for reserved, bogon and internal addresses. 300 packets will time out in approximately 2.5 minutes.

The SIS starts storing state once it receives the first spoofed packet with a valid cookie. An attack on this is possible only using a valid IP address. A server capable of storing millions of spoofing index can process 24 million requests per hour. It denies requests from a domain from which it already has fresh spoofing index information.

The spoofing index table is cached locally at the edge routers. Queries responses are fast as the table is indexed by IP prefix. Software routers and switches such as CLICK [110] and OpenFlow [111] have optimized implementations for IP prefix classifiers.

SIS also distributes keys for the domains. The overhead of rekeying every week by all ASes has been measured to be less than 17ms per minute [52]. Let us analyze how much traffic is tagged with this week long key. Some of the Internet2 core links have up to 5 Gbps data usage [112]. With a conservative packet size of 500 bytes this link will process 604.8G packets in a week for all destinations. A 64-bit nonce space in the hash of the MAC can handle much more than this. Nevertheless, our proposal allows the origin ISP to set the nonce size based on its network load.

6.5.3 The IBQ Tokens

The IBQ header adds up to 8 bytes for each token. For 4 ASes the header length could be at least 24 bytes for 32 bit Tokens and 44 bytes for 64 bit Tokens. We recommend a maximum of 4 tokens. It takes about half a microsecond to put or verify a 32-bit token on a commodity GHz device [37]. A throughput of two million packets per second verification is excellent for a server with a GHz of processing capacity.

6.5.4 Queuing and Filtering

Ideally, an integrity-based queueing service sorts packets per authenticated IP prefix. Even if routers can install thousands of filters that is not enough for a network with many high and medium integrity flows. The 32-bit IP space can have a maximum of 4 billion filters. If we look at the adoption of ingress filtering [27] and base our calculation on the probability distribution of spoofing indices, IBQ will have 125 million integrity blocks. A host might get only a fraction of these as

clients and attacker flows, but those filters need to get installed at runtime. But dynamic memory management is neither recommended for hardware nor for software routers. For example, in CLICK router the spoofing index table is installed as a classifier. Flows from each class go into separate BandwidthMeter elements and separate queues for round-robin and priority scheduling.

IBQ scales much better with monitors that look for high-bandwidth flows and punish them. RED-PD [113] and DiffServe are two such implementations. OpenFlow [111] switches have support for DiffServ. We design IBQ with three high capacity queues for – high ($0 \leq si \leq 8$), medium ($9 \leq si \leq 16$), moderate($17 \leq si \leq 24$) and low integrity traffic. Bandwidth limiters such as RED-PD are implemented for the three higher class flows. These detect and limit the sending rate of flows sending at a higher rate than the set threshold. Traffic from the higher integrity flows are prioritized according to the spoofing index distribution. The high integrity queue also uses a shadow backup queue. This way the overhead of IBQ is the related to the number of higher integrity attack flows. For example for the CAIDA attack dataset in the worst most unlikely case, 9311 filters would have been installed. High bandwidth legitimate traffic follows TCP congestion rules and would not trigger filters.

7 Cheater

Cheater combines multiple DDoS defense systems to complement each other. It looks into the solutions that make big differences and builds a defense in depth. There are four layers of protection in Cheater. First, the server implements IBQ and incentivizes based on integrity. Second, the server uses cookies to enable legitimate users not from an IBQ domain to send packets. Third, we enable legitimate clients to *scavenge* for cookies. A client can feed on a spoofed agent request that scatters back to her and use the capability on it for itself. Finally, the legitimate client does not back off as it would traditionally and uses ASV to send more requests. So putting these all together, the client that is not from an integrity domain has two other approaches. She will peek in any packet that reaches her and check if it contains a cookie from a server she is targeting. She will also periodically make the same request again and again and in larger number each time.

7.1 Architecture

7.1.1 Scavenging

When a legitimate client makes a request she listens for the reply on any port. That way she can pick up any reply from the server on any port. This way the client can 'scavenge' off the spoofed packets. The server sends a scavenger cookie with each response. This cryptographic cookie contains the source and destination IP address and a timestamp.

Attacker agents, to avoid detection, often spoof their source IP addresses. Replies to these requests go back the address that was spoofed. Legitimate clients make some benefit of this situation at zero cost by scavenging off the spoofed requests. The attackers cannot scavenge a cookie unless they are willing to use their own IP address.

7.1.2 IBQ and the Cookie Channel

Cheater uses the allocation for low-integrity flows as the request channel. Packets that are selected get a cookie that they can use for later packets. As mentioned before, this cookie is self-signed by the server. Clients that do not have an integrity provider ISP can use the cookie channel to request for a cookie and include the cookie in future packets. The returned cookie proves to the server that the client has finished a round-trip and somewhat ownership of that IP address. Packets carrying a cookie are prioritized over other packets in this channel. It is as if they move into the $31\frac{1}{2}$ spoofing index queueing class.

7.1.3 ASV and the Cookie Channel

We propose that the legitimate clients ramp up their requests in the face of an attack. This is similar to the proposals in Chapter 3. Our proposal is similar to ASV with the server implementing the reservoir. But this is deployed only for the low-integrity flows. The client keeps retrying up to certain rounds or if a maximum number of packets in a sequence go unacknowledged. The client ramps up its requests if it does not get an ACK from the server. To avoid situations where the server is down or there are other network problems, the server sends back a DAC to the legitimate clients before dropping a packet. For the packets that are accepted a self-signed cookie mentioned before is sent back.

7.1.4 Defense-in-depth

Cheater is a defense-in-depth approach combining these three mechanisms for the low-integrity channel (Figure 7.1). Cheater dedicates a small fraction of the link for cookie requests. Using this channel clients can request for a token of service. Once a user has this token she can use the data channel. The specific token can be chosen by the server. For example an e-commerce site might have a complex token like a capability. A news site might opt for a simple puzzle token. For our experiments we use a simple hash cookie with the source and destination address and a time stamp. To enable scavenging, client host listens on all ports (promiscuous port listening). Also the client is set up for ASV. Clients wait for an acknowledgement from the server. If none is received within a time-period the clients send two requests. The sending rate doubles in each round. The server is

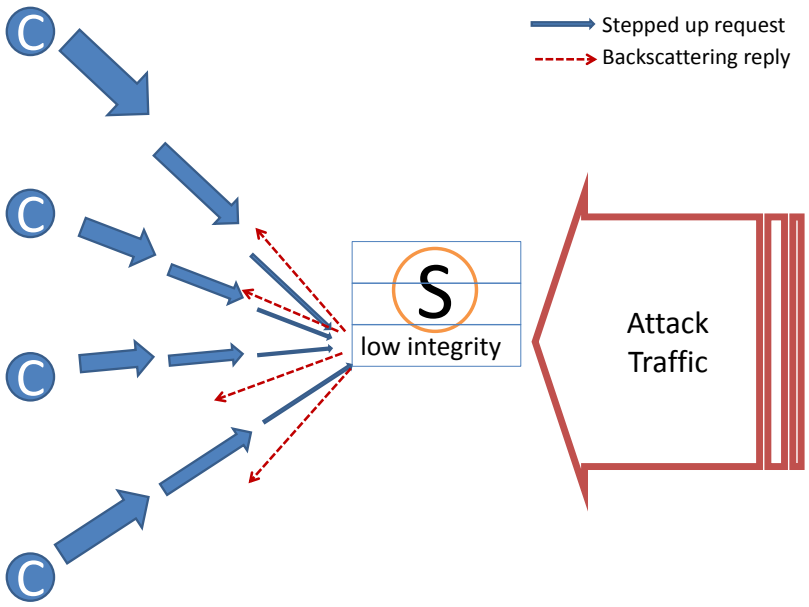

Figure 7.1: Cheater with scavenging, IBQ and ASV on low-integrity traffic.

set up for ACK and DACK responses. The server also initiates IBQ to sort out the requests it receives. Once a request is received and selected the server sends back a cookie. If this was a spoofed source request it will backscatter to another host. That host can scavenge the cookie.

7.2 Evaluations

In this section we simulate Cheater in ns2 to evaluate how well it combines ASV and IBQ to defend DDoS attacks on the Internet. We also compare Cheater to TVA and ASV to highlight the performance differences. Cheater is implemented as discussed in Section 7.1. The clients are set up to send a request at regular intervals and step-up accordingly if no reply is received. IBQ is implemented on the server to filter the requests for attackers. We set up the queue parameters based on the simulation topology. Also the clients listen in promiscuous port mode to scavenge on spoofing attacker's responses. The attackers are set to send requests at a very high rate to disrupt the service. First, we describe our experimental setup and methodology. Then we discuss the results.

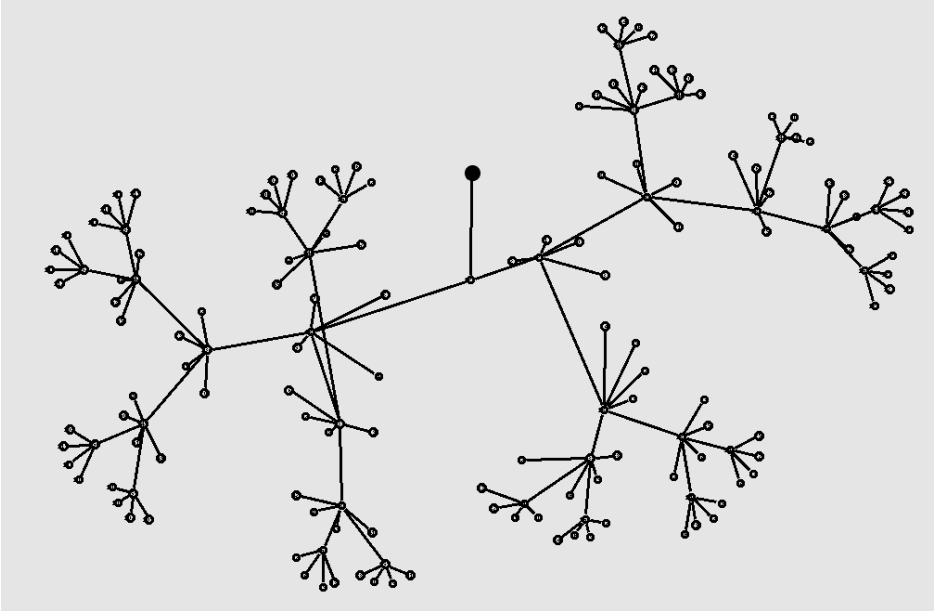

Figure 7.2: Sample experiment topology. Subnets = 32, subnet size is 256 (4 nodes shown for the clarity of the figure).

7.2.1 Topology Setup.

The Internet DDoS attacks are too large to be simulated in existing tools. If we consider the AS-level network topology there are more than 25K ASes. Unfortunately, existing simulation tools cannot scale to such a large number. In particular, ns2 can handle topologies with 1000—2000 nodes. To address this issue, we scale down the Internet topology into a size that can be handled by ns2 and at the same time that will capture the relational properties of clients, routers and servers. We analyzed the available Internet topologies such as Abilene [114], commercial backbone maps and AS maps. We also consulted connectivity and subnets of University of Illinois and University of Pennsylvania. Based on those we propose a simulation topology of 1058 nodes: 1024 clients, 33 routers, and a destination server. We propose 32 subnets, where we may very the subnet address size from 32 to 512, that is 2^5 to 2^9. We refer to this index as the spoofing index. By varying the spoofing index we can vary the degree an attacker can use spoofed IP address. The spoofing index is scaled to our topology. Figure 7.2 shows a trimmed topology with 4 nodes per subnet.

We assign 5% of the link capacity to the low integrity cookie request channel. All the links except the bottleneck are 200Mbps. All the links have 10ms of delay. The bottleneck link from the edge router to the server is 2Gbps. Our

study of the network shows that servers are closer to the backbone and sit on high capacity links. On the Internet this link would be 40Gbps to 100Gbps. We scale the bandwidth down for our much smaller number of end hosts. The Internet links are highly over-provisioned with less then 10% utilization. So allocating the request channel 5% is also over-allocation. We want to test Cheater in extreme situations so we have chosen parameters values as extreme as ns2 would allow.

7.2.2 Attack and Defense Parameters

Attack

We have 10% of our hosts set as attacker agents. Total attack bandwidth ranges from 100 to 700Mbps. These agents want to bombard the resources with packets to queue and process. We simulate their sending rate at the higher end of the request channel capacity. As discussed earlier, to avoid detection the attackers spoof their source IP address within a spoofing index.

Cheater

For Cheater and ASV the edge core routers have a legacy setup. Any commercial router can forward ASV and cheater packets. The sink link to the server is set up for IBQ for Cheater. There are 1024 low-integrity clients for IBQ.

ASV

Clients, on the other hand just try to send a 50B request packet. If they do not get an acknowledgement by a certain time, they resend their request, but twice this time. This is done for 12 rounds before timing out. ASV2 is more aggressive than ASV1. ASV1 and Cheater wait the same amount of time for a response from server before sending out the next round of packets. ASV2 time outs faster. For example, in Figure 7.4 ASV1 waits 200ms before sending out packets for the next round, whereas ASV2 waits 80ms.

TVA

For TVA 75% routers are setup to run TVA and H-PFQ [115, 116]. H-PFQ, not being a stochastic algorithm, can require as many queues as path identifiers. In our system this will be $32 * 2^8 \approx 8000$ queues on each router that has TVA installed. Unfortunately, according to [55] this a number is higher than what would be available to a similar size network. So we perform the TVA experiments in two settings. One version uses as many queues as needed. This gives it a advantageous setup compared to Cheater and ASV. In the other version uses 1024 queues. For the later queue-system maximum queue space required per router is approximately 200KB.

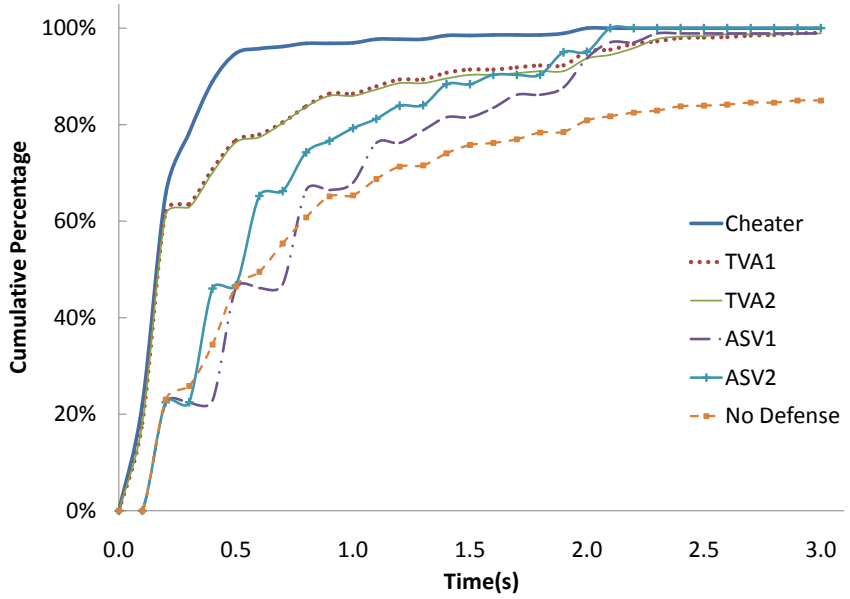

Figure 7.3: Delay observed by legitimate clients with Cheater, two setups of TVA and two setups of ASV when the attacks are sending a packet every 400µs. Attack bandwidth 102Mbps

7.2.3 Performance compared to TVA and ASV

The first experiment compares Cheater to two existing methods, TVA and ASV. Both variations of TVA and ASV are tested. We run this experiment for multiple attack bandwidths starting from 102 Mbps to 714 Mbps. With the 102 Mbps rate the attackers are bombarding the bottleneck link but they are not yet congesting the edge links or intra-AS links. Each attacker is sending at 1 Mbps and the request

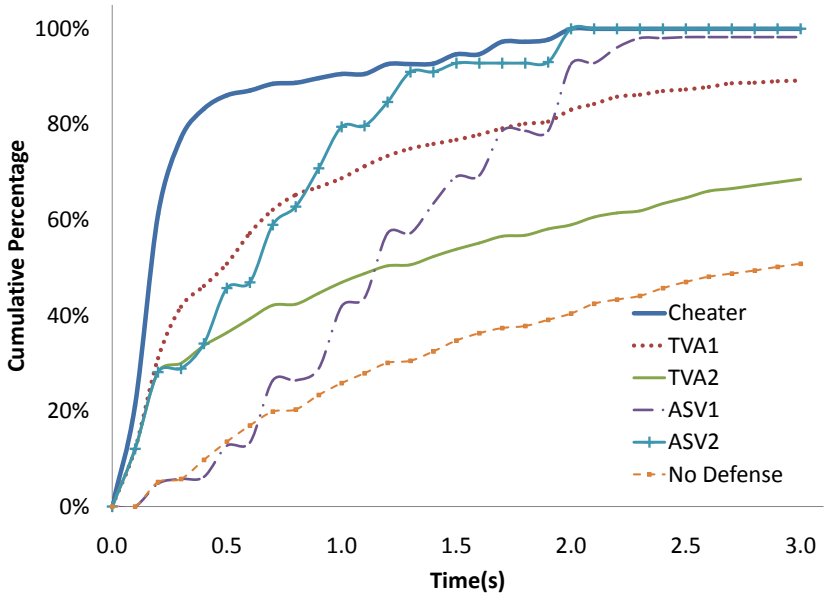

Figure 7.4: Delay observed by legitimate clients with Cheater, two setups of TVA and two setups of ASV when the attacks are sending a packet every 133µs. Attack bandwidth 306Mbps

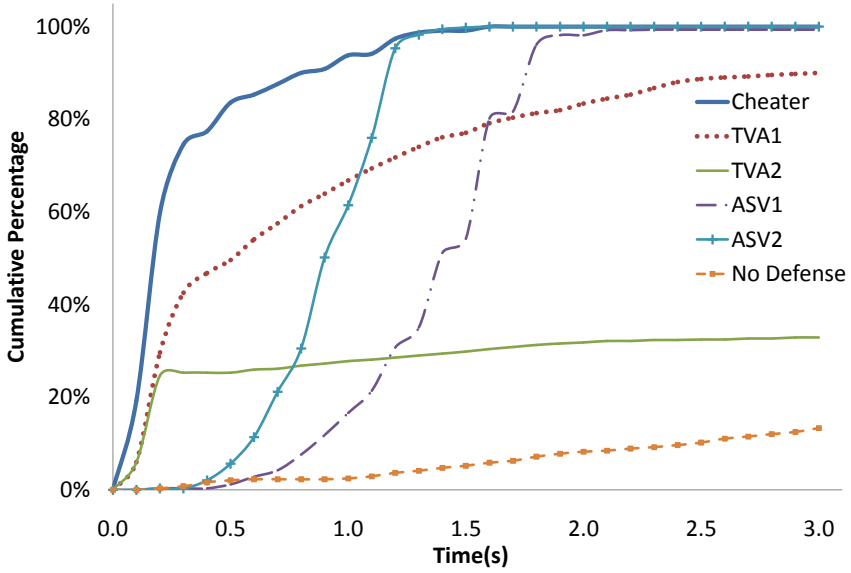

Figure 7.5: Delay observed by legitimate clients with Cheater, two setups of TVA and two setups of ASV when the attacks are sending a packet every 80µs. Attack bandwidth 510Mbps.

channel bandwidth for these links are 10 Mbps. With 306 Mbps attack bandwidth

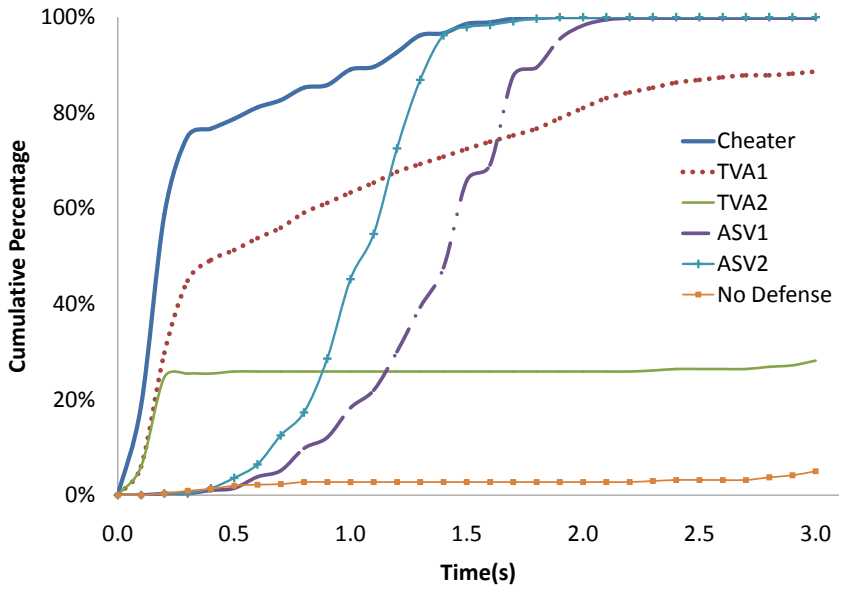

Figure 7.6: Delay observed by legitimate clients with Cheater, two setups of TVA and two setups of ASV when the attacks are sending a packet every 57µs. Attack bandwidth 714Mbps.

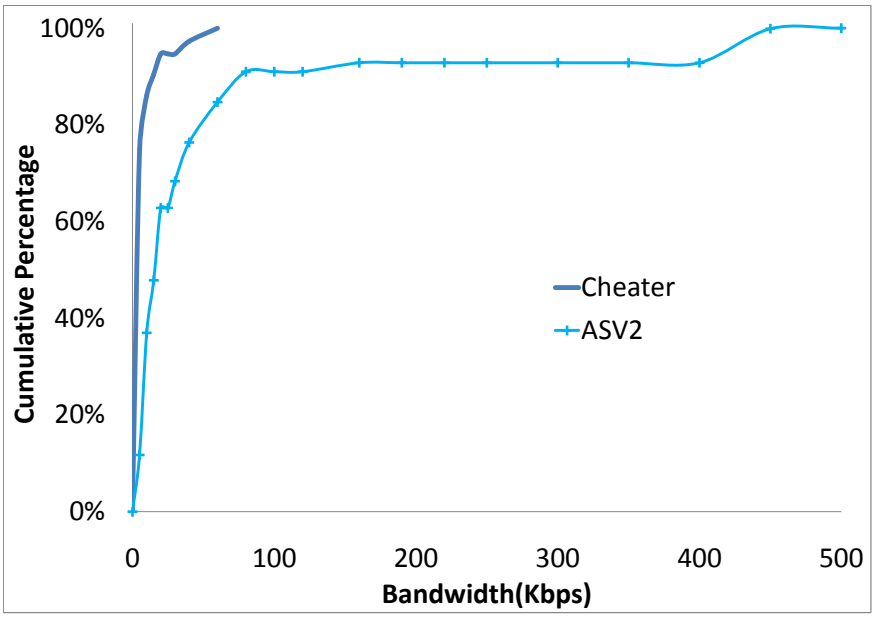

Figure 7.7: Bandwidth used by Legitimate clients in Cheater and ASV. Attack bandwidth 306Mbps.

the attackers start to congest some of the intra-AS links. 510 Mbps and 714 Mbps attack bandwidth congests all the links.

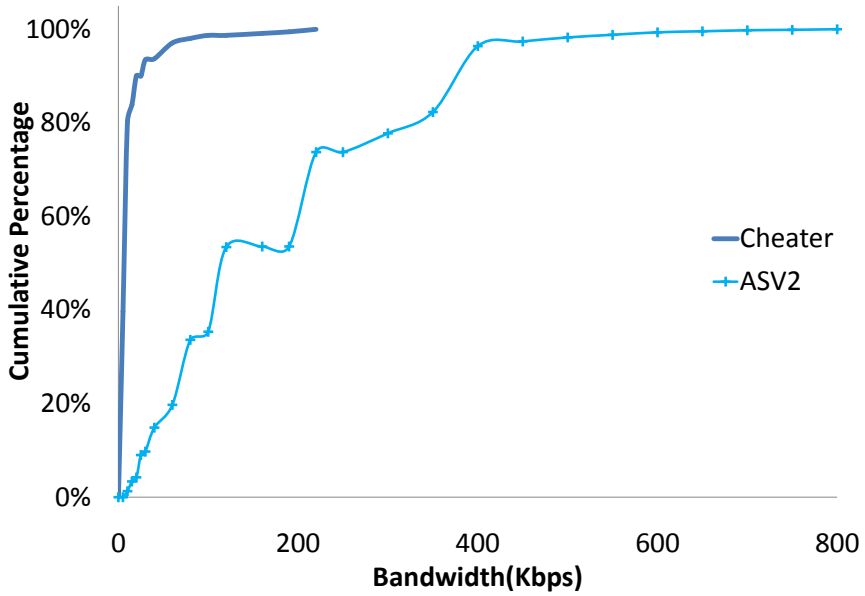

Figure 7.8: Bandwidth used by Legitimate clients in Cheater and ASV. Attack bandwidth 510Mbps.

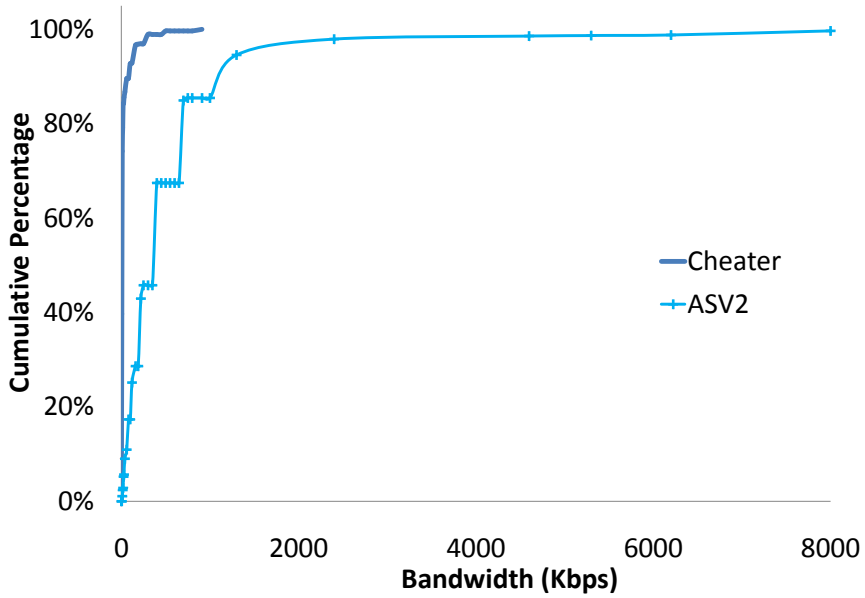

Figure 7.9: Bandwidth used by Legitimate clients in Cheater and ASV. Attack bandwidth 714Mbps.

Figures 7.3, 7.4, 7.5 and 7.6 show the results. With Cheater all clients get a response from the destination server within 2s for all the attack scenarios. TVA and ASV have good performance at lower attack rates, but stall as attack starts to

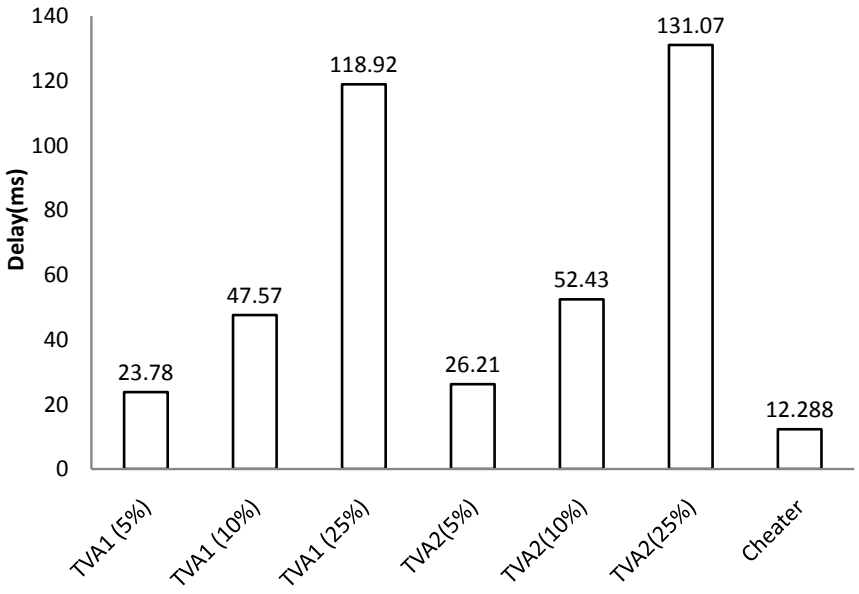

Figure 7.10: Delay caused by queuing in TVA1 and Cheater. TVA results are for average 5%, 1% and 0.5% full queues.

congest the links. With both variations of TVA when the attack rate is more than 1Mbps they fail to provide service to a big percentage of legitimate users. For example, in Figure 7.4 TVA1 can get only 90% of legitimate clients under 10s and TVA2 only 65%. The spoofing that goes on at the subnets that do not install TVA on routers stalls legitimate clients indefinitely in TVA1. In TVA2, routers are not able to differentiate between a legitimate client and an attacker within a subnet. This causes the attacker and legitimate client packets ending up in the same queue. As a result a large portion of clients are deprived of the service. The percentage of choked clients go up as the attack factor goes up. TVA performs worse than Cheater with higher memory requirements for queues. ASV performs better in terms of providing service to legitimate clients. In all attack scenarios ASV1 serves all the legitimate clients in less than 3s and ASV2 in less than 2s. But performance comes at cost. ASV2, the aggressive version of ASV, performs similar to Cheater but with much higher network bandwidth usage.

Figures 7.3, 7.4, 7.5, and 7.6 show the difference of bandwidth usage between Cheater and ASV at different attack rates. At attack bandwidth of 302Mbps, for 90% of the legitimate clients, Cheater bandwidth is less than 15kbps. For ASV2, the more aggressive ASV, it goes upto 472kbps and for the rest 10% of the clients it is more than 80kbps. At attack bandwidth of 510Mbps, for 90% legiti-

mate clients in Cheater bandwidth is less than 20kbps. Though the maximum is 200kbps only few legitimate clients reach that. For ASV2 it goes upto 790kbps and for 65% clients it is more than 100kbps. At attack bandwidth of 714Mbps, for 90% legitimate clients in Cheater bandwidth is less than 80kbps. Though the maximum is 905kbps. For ASV2 it goes upto 14Mbps and for 32.5% clients it is more than 400kbps. Cheater uses ASV as one of its protection steps. As it is strengthened by other steps, legitimate clients get service faster. As a result a legitimate client gets service without stepping up its sending rate to extreme levels.

Figure 7.10 shows the difference of queueing overhead between Cheater and TVA. Cheater only experiences 12.3ms of queuing delay if the queue on the bottleneck link is full 100% all the time. TVA1 has queues on 75% of the routers and the path length for end-to-end traffic is on average is 5. Total delay observed for queueing with the queues being on average 5% full is 23.78ms. The delay is 47.5ms if the queues are 10% full and 118.9ms for 25% full queues. TVA2 has queues only at a subnet level, so the path length is 4. But as this allows more spoofing by attackers, the queues are more full. In our experiment topology queueing delay is 26.2ms for 5% full queues, 52.4ms for 10% full queues and 131.1ms for 25% full queues. These are high delay on a 150ms long route. That is why we observed among the clients who received service, many of them did not get a response within 3s. Cheater uses IBQ only at the end host incurring minimal queuing delay.

7.2.4 Cheater Defense-in-depth.

This set of experiments analyzes the effect each protection has in overall outcome of Cheater. We separately run Cheater and its steps Scavenger, ASV and IBQ to two different attack setting (Figure 7.11, 7.12). At 102Mbps attack rate all three steps perform reasonably well. But none of them independently can provide service to all the legitimate clients. Both Scavenger and ASV stall at 99% clients. IBQ can provide service to 90% legitimate clients after 7s. At 514Mbps attack rate performance of individual steps suffer. Scavenger can serve 93% legitimate clients before timing out at 10s. In the same duration ASV serves 99% clients. IBQ serves only 11% legitimate clients. At higher attack rates congestion takes place all over the network. The defense steps individually cannot protect clients

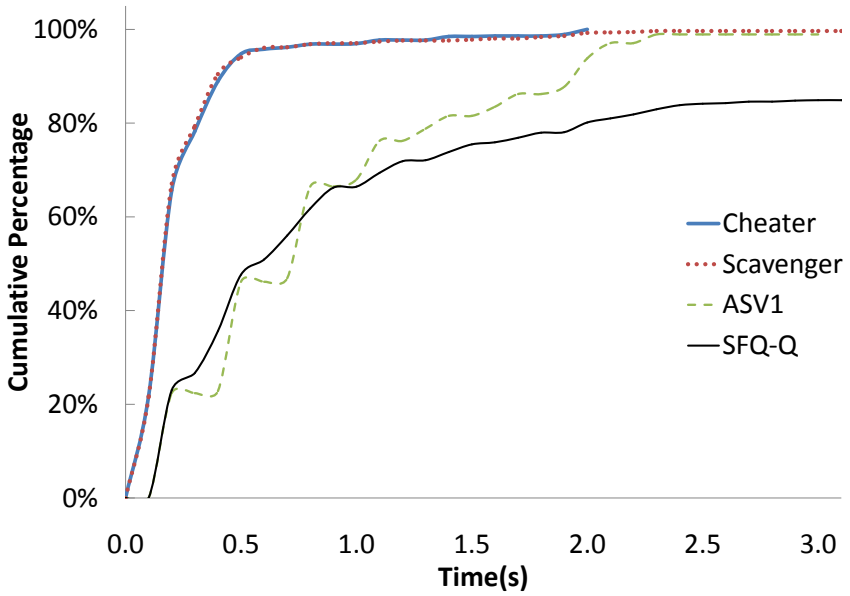

Figure 7.11: Comparing delay observed by legitimate clients with Cheater and the Cheater steps working independently. Attack bandwidth 102Mbps.

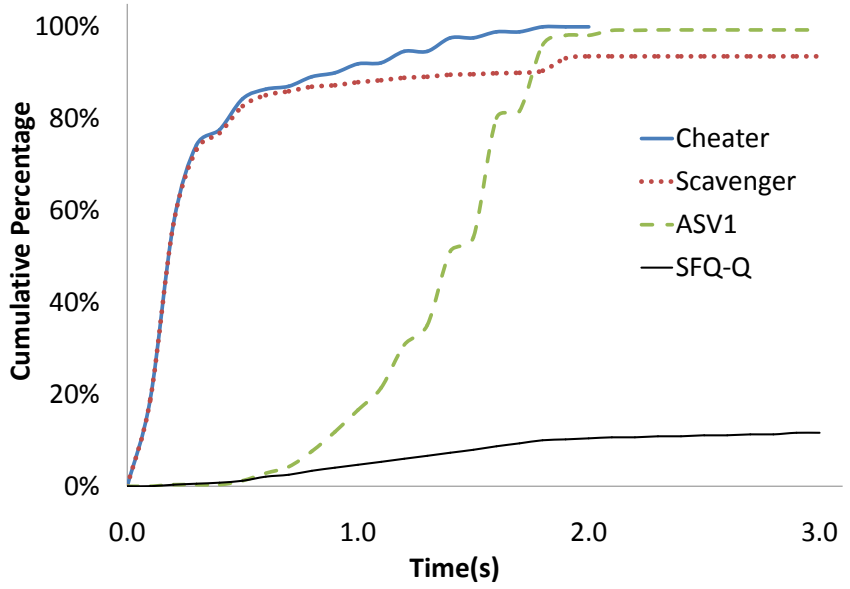

Figure 7.12: Comparing delay observed by legitimate clients with Cheater and the Cheater steps working independently. Attack bandwidth 510Mbps.

from flooding at all directions. ASV enables legitimate clients to overcome intermediate congestion. IBQ provides defense close to the server. Scavenger, being an opportunistic algorithm does not have guaranteed success. With cheater they strengthen each others shortcomings and provide more efficient and less expensive defense.

8 Future Work and Conclusion

Summary. In conclusion, we provide an architecture for incentivizing investment in security infrastructure (Figure 1.1). We design and analyze two systems based on such incentives. In ASV, clients use bandwidth as a payment for service. Experiments quantify the effectiveness of ASV against its non-adaptive counterparts and illustrate that under highly variable-rate attacks, the performance of ASV adjusts quickly to prevailing attack parameters. ASV is a great tool available for clients in an Internet monopoly situation.

IBQ is a more structured way for a competitive ISP to provide graded source authentication and better service during a DDoS attack on a server. IBQ successfully mitigates DDoS attacks between two participating domains. This work shows its success for real-time traffic and TCP traffic. IBQ clients perform really well for both TCP connectivity and throughput with the 2007 DDoS attack trace. Our tests show that IBQ connectivity protection exceeds TVA and IBQ throughput matches it when both are deployed only at the edge of the network. Such results make IBQ suitable for early adoption. Optimizations can reduce overhead of IBQ; for instance, our proposal for implementation with rate-limit filters greatly reduces the queuing overhead.

IBQ is orthogonal to other techniques for differential queuing. For example, real time traffic can be separated from other traffic and IBQ can be applied to either type of traffic. Indeed, experiments in Section 6.1 show how VoIP can perform without jitter or loss under DDoS attack using IBQ.

Cheater shows how the seemingly very different techniques such as ASV and IBQ can complement each other and work for the defense of denial-of-service attack. We look forward to other security infrastructure to define incentive structures for protection. It would be interesting to explore incentive-based options for the Internet core. One driving force for them is the market and as larger content-providers start providing service based on integrity they may be able to influence the core providers.

Discussion. In future, it would be interesting to work out more details for practical impediments to Internet-wide use of IBQ. We proposed using a vouching and receiving table to enable smaller ISPs to participate. A receiving table stores the ASPATH that a packet would take to reach that AS. This is the inverse of ASPATH used by BGP forwarding table where it contains a path from its domain to all other domains. A receiving table enables to gateway to deduce spoofed packet based on the wrong route it has taken. But vouching directly benefits a domain only if there is a customer-provider relationship. Also asymmetry in Internet routes have an effect on forward tables. It is important to know the extent of that effect. Even after adding these capabilities there could be clients in parts of the network that cannot get an integrity service or do not have adequate bandwidth.

In practical terms IBQ has some kinship to SPF, which we mentioned earlier. The idea behind SPF is to provide a light-weight integrity service that can be used by recipients of email as they wish: to inform a spam protection weighting, to place a message in a low priority queue, or to drop the message. Thus SPF as an integrity service can combined with other integrity services and processed by diverse recipients in diverse ways. One speculates along these lines about how well one can unify some of these integrity sources or exploit new avenues for integrity assessment. This could provide help with other problems we encountered, like how to address variations in the spoofing index for large autonomous systems.

Future Directions. Distributed denial-of-service is a rich research area. It is important to give a deeper look at this research and classify defenses according to incentives provided for early adoption; put them in the incentive-based network service architecture. There two particular areas that need more focus: incentive model for the core network and efficient queueing. Proposals that provide solutions at the edge of the network have a hard time handling a massive attack on the core of the Internet. On the other hand, it is more challenging to find an incentive for deployment of solutions at the core routers. They operate at a high frequency and efficiency of the solution is important for them.

There has not been much overlap between the dense areas of denial-of-service and quality-of-service research. Many times it is not clear how the theories of QoS would scale with high rates of DDoS. We looked at the the limits of dynamism in queueing for routers in Section 6.5. This is true for many DDoS defense protocols. A theoretical and empirical study of limits of the underlying queueing mechanism in terms of scalability and state maintenance overhead in relationship to high bandwidth traffic is needed.

If the Internet stumbles, it will not be because we lack for technology, vision, or motivation. It will be because we cannot set a direction and march collectively into the future [117].

—*A Brief History of the Internet*, 2009.

References

[1] E. Mills, "Twitter, Facebook attack targeted one user," *cnet news*, August 6 2009. [Online]. Available: http://news.cnet.com/8301-27080_3-10305200-245.html

[2] J. Richards, "Georgia accuses Russia of waging 'cyber-war'," *The Times Online*, August 11 2008. [Online]. Available: http://technology.timesonline.co.uk/tol/news/tech_and_web/article4508546.ece

[3] "Three botnets responsible for half of all computer infections," *Infosecurity Magazine*, Feb 11 2010. [Online]. Available: http://www.infosecurity-us.com/view/7242/three-botnets-responsible-for-half-of-all-computer-infections/

[4] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage, "Inferring internet denial-of-service activity," *ACM Trans. Comput. Syst.*, vol. 24, no. 2, pp. 115–139, 2006.

[5] R. McMillan, "Two root servers targeted by botnet," *PC Advisor (pcadvisor.co.uk)*, Feb 7 2007.

[6] K. Poulsen, "FBI busts alleged DDoS mafia," *Security Focus (securityfocus.com)*, 08/26/2004.

[7] C. Kaufman, "Internet Key Exchange (IKEv2) Protocol," RFC 4306 (Proposed Standard), Internet Engineering Task Force, Dec. 2005, obsoleted by RFC 5996, updated by RFC 5282. [Online]. Available: http://www.ietf.org/rfc/rfc4306.txt

[8] D. Eastlake 3rd, "Domain Name System Security Extensions," RFC 2535 (Proposed Standard), Internet Engineering Task Force, Mar. 1999, obsoleted by RFCs 4033, 4034, 4035, updated by RFCs 2931, 3007, 3008, 3090, 3226, 3445, 3597, 3655, 3658, 3755, 3757, 3845. [Online]. Available: http://www.ietf.org/rfc/rfc2535.txt

[9] D. E. Sanger, J. Markoff, and T. Shanker, "U.S. Steps Up Effort on Digital Defenses," *New York Times Online*, April 28 2009.

[10] J. Nazario, "Estonian ddos attacks - a summary to date." 2005. [Online]. Available: http://asert.arbornetworks.com/2007/05/estonian-ddos-attacks-a-summary-to-date

[11] L. Belsie, "Iranian hacker attack: What will it cost twitter?" *Christian Science Monitor*, Dec 18 2009. [Online]. Available: http://www.csmonitor.com/Money/2009/1218/Iranian-hacker-attack-What-will-it-cost-Twitter

[12] C. A. Gunter, S. Khanna, K. Tan, and S. S. Venkatesh, "DoS protection for reliably authenticated broadcast," in *NDSS'04: Network and Distributed System Security Symposium*. The Internet Society, 2004.

[13] M. Sherr, M. B. Greenwald, C. A. Gunter, S. Khanna, and S. S. Venkatesh, "Mitigating DoS attacks through selective bin verification," in *IEEE ICNP Workshop on Secure Network Protocols (NPSec)*, 2005, pp. 7–12.

[14] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker, "DDoS defense by offense," in *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM Press, 2006, pp. 303–314.

[15] S. Khanna, S. S. Venkatesh, O. Fatemieh, F. Khan, and C. A. Gunter, "Adaptive selective verification," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, april 2008, pp. 529 –537.

[16] S. Khanna, S. S. Venkatesh, O. Fatemieh, F. Khan, and C. A. Gunter, "Adaptive selective verification: An efficient adaptive countermeasure to thwart dos attacks," *Networking, IEEE/ACM Transactions on*, vol. PP, no. 99, p. 1, 2011.

[17] P. Hick, E. Aben, kc claffy, and J. Polterock, "The CAIDA DDoS Attack 2007 Dataset," 2007. [Online]. Available: http://www.caida.org/data/passive/ddos-20070804_dataset.xml

[18] J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher, *Internet Denial of Service: Attack and Defense Mechanisms (Radia Perlman Computer Networking and Security)*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004.

[19] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *SIGCOMM Comput. Commun. Rev.*, vol. 34, pp. 39–53, April 2004. [Online]. Available: http://doi.acm.org/10.1145/997150.997156

[20] C. Douligeris and A. Mitrokotsa, "Ddos attacks and defense mechanisms: classification and state-of-the-art," *Computer Networks*, vol. 44, no. 5, pp. 643 – 666, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128603004250

[21] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the dos and ddos problems," *ACM Comput. Surv.*, vol. 39, April 2007. [Online]. Available: http://doi.acm.org/10.1145/1216370.1216373

[22] D. J. Bernstein, "SYN cookies," 1996. [Online]. Available: http://cr.yp.to/syncookies.html

[23] W. Eddy, "TCP SYN Flooding Attacks and Common Mitigations," RFC 4987 (Informational), Internet Engineering Task Force, Aug. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4987.txt

[24] M. Casado, A. Akella, P. Cao, N. Provos, and S. Shenker, "Cookies along trust-boundaries (cat): accurate and deployable flood protection," in *Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet - Volume 2*. Berkeley, CA, USA: USENIX Association, 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251296.1251299 pp. 3–3.

[25] A. Mahimkar, J. Dange, V. Shmatikov, H. Vin, and Y. Zhang, "dfence: transparent network-based denial of service mitigation," in *Proceedings of the 4th USENIX conference on Networked systems design & implementation*, ser. NSDI'07. Berkeley, CA, USA: USENIX Association, 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1973430.1973454 pp. 24–24.

[26] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing," RFC 2827 (Best Current Practice), Internet Engineering Task Force, May 2000, updated by RFC 3704. [Online]. Available: http://www.ietf.org/rfc/rfc2827.txt

[27] R. Beverly and S. Bauer, "The spoofer project: inferring the extent of source address filtering on the internet," in *SRUTI'05: Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop*. Berkeley, CA, USA: USENIX Association, 2005, pp. 8–8.

[28] H. Wang, D. Zhang, and K. G. Shin, "Detecting syn flooding attacks," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, june 2002, pp. 1530 – 1539.

[29] J. Ioannidis and S. M. Bellovin, "Implementing pushback: Router-based defense against DDoS attacks," in *Proc. Internet Society Symposium on Network and Distributed System Security*, 2002. [Online]. Available: https://www.cs.columbia.edu/~smb/papers/pushback-impl.pdf

[30] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling high bandwidth aggregates in the network," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 3, pp. 62–73, 2002.

[31] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Network support for ip traceback," *IEEE/ACM Trans. Netw.*, vol. 9, pp. 226–237, June 2001. [Online]. Available: http://dx.doi.org/10.1109/90.929847

[32] D. X. Song and A. Perrig, "Advanced and authenticated marking schemes for ip traceback," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, 2001, pp. 878 –886 vol.2.

[33] A. Snoeren, C. Partridge, L. Sanchez, C. Jones, F. Tchakountio, B. Schwartz, S. Kent, and W. Strayer, "Single-packet ip traceback," *Networking, IEEE/ACM Transactions on*, vol. 10, no. 6, pp. 721 – 734, dec 2002.

[34] J. Li, M. Sung, J. J. Xu, and L. E. Li, "Large-scale ip traceback in high-speed internet: Practical techniques and theoretical foundation," *Security and Privacy, IEEE Symposium on*, vol. 0, p. 115, 2004.

[35] K. Argyraki and D. R. Cheriton, "Active internet traffic filtering: real-time response to denial-of-service attacks," in *Proceedings of the annual conference on USENIX Annual Technical Conference*, ser. ATEC '05. Berkeley, CA, USA: USENIX Association, 2005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1247360.1247370 pp. 10–10.

[36] X. Liu, X. Yang, D. Wetherall, and T. Anderson, "Efficient and secure source authentication with packet passports," in *SRUTI'06: Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet*. Berkeley, CA, USA: USENIX Association, 2006, pp. 2–2.

[37] X. Liu, A. Li, X. Yang, and D. Wetherall, "Passport: secure and adoptable source authentication," in *NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2008, pp. 365–378.

[38] B. Parno, A. Perrig, and D. Andersen, "Snapp: stateless network-authenticated path pinning," in *ASIACCS '08: Proceedings of the 2008 ACM symposium on Information, computer and communications security*. New York, NY: ACM, 2008, pp. 168–178.

[39] A. Yaar, A. Perrig, and D. Song, "Pi: A path identification mechanism to defend against DDoS attacks," in *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2003, p. 93.

[40] A. Yaar, A. Perrig, and D. X. Song, "SIFF: A stateless internet flow filter to mitigate DDoS flooding attacks." in *IEEE Symposium on Security and Privacy*, 2004, pp. 130–.

[41] K. Park and H. Lee, "On the effectiveness of route-based packet filtering for distributed dos attack prevention in power-law internets," in *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2001, pp. 15–26.

[42] C. Jin, H. Wang, and K. G. Shin, "Hop-count filtering: an effective defense against spoofed ddos traffic," in *Proceedings of the 10th ACM conference on Computer and communications security*, ser. CCS '03. New York, NY, USA: ACM, 2003. [Online]. Available: http://doi.acm.org/10.1145/948109.948116 pp. 30–41.

[43] T. M. . D. M. k.c. claffy, "Internet tomography," *Web Matters*, Jan. 1999.

[44] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," *SIGCOMM Comput. Commun. Rev.*, vol. 32, pp. 133–145, August 2002. [Online]. Available: http://doi.acm.org/10.1145/964725.633039

[45] R. Pastor-Satorras and A. Vespignani, *Evolution and Structure of the Internet: A Statistical Physics Approach*. New York, NY, USA: Cambridge University Press, 2004.

[46] P. Mahadevan, D. Krioukov, M. Fomenkov, X. Dimitropoulos, k. c. claffy, and A. Vahdat, "The internet as-level topology: three data sources and one definitive metric," *SIGCOMM Comput. Commun. Rev.*, vol. 36, pp. 17–26, January 2006. [Online]. Available: http://doi.acm.org/10.1145/1111322.1111328

[47] D. G. Andersen, "Mayday: distributed filtering for internet services," in *Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems - Volume 4*, ser. USITS'03. Berkeley, CA, USA: USENIX Association, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251460.1251463 pp. 3–3.

[48] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson, "The end-to-end effects of internet path selection," *SIGCOMM Comput. Commun. Rev.*, vol. 29, pp. 289–299, August 1999. [Online]. Available: http://doi.acm.org/10.1145/316194.316233

[49] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," *SIGOPS Oper. Syst. Rev.*, vol. 35, pp. 131–145, October 2001. [Online]. Available: http://doi.acm.org/10.1145/502059.502048

[50] E. Nygren, R. K. Sitaraman, and J. Sun, "The akamai network: a platform for high-performance internet applications," *SIGOPS Oper. Syst. Rev.*, vol. 44, pp. 2–19, August 2010. [Online]. Available: http://doi.acm.org/10.1145/1842733.1842736

[51] T. Anderson, T. Roscoe, and D. Wetherall, "Preventing internet denial-of-service with capabilities," *SIGCOMM Comput. Commun. Rev.*, vol. 34, pp. 39–44, January 2004. [Online]. Available: http://doi.acm.org/10.1145/972374.972382

[52] X. Liu, X. Yang, and Y. Lu, "To filter or to authorize: network-layer dos defense against multimillion-node botnets," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 195–206, 2008.

[53] M. Natu and J. Mirkovic, "Fine-grained capabilities for flooding ddos defense using client reputations," in *LSAD '07: Proceedings of the 2007 workshop on Large scale attack defense*. New York, NY, USA: ACM, 2007, pp. 105–112.

[54] X. Yang, D. Wetherall, and T. Anderson, "A DoS-limiting network architecture," in *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM Press, 2005, pp. 241–252.

[55] X. Yang, D. Wetherall, and T. Anderson, "TVA: a dos-limiting network architecture," *IEEE/ACM Trans. Netw.*, vol. 16, no. 6, pp. 1267–1280, 2008.

[56] X. Liu, X. Yang, and Y. Xia, "Netfence: preventing internet denial of service from inside out," vol. 40. New York, NY, USA: ACM, August 2010. [Online]. Available: http://doi.acm.org/10.1145/1851275.1851214 pp. 255–266.

[57] M. AlTurki, J. Meseguer, and C. A. Gunter, "Probabilistic modeling and analysis of dos protection for the asv protocol," *Electron. Notes Theor. Comput. Sci.*, vol. 234, pp. 3–18, March 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1519544.1519753

[58] H. Burch and B. Cheswick, "Tracing anonymous packets to their approximate source," in *Proceedings of the 14th USENIX conference on System administration*. Berkeley, CA, USA: USENIX Association, 2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=1045502.1045544 pp. 319–328.

[59] D. K. Y. Yau, J. C. S. Lui, F. Liang, and Y. Yam, "Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles," *IEEE/ACM Trans. Netw.*, vol. 13, pp. 29–42, February 2005. [Online]. Available: http://dx.doi.org/10.1109/TNET.2004.842221

[60] A. Juels and J. G. Brainard, "Client puzzles: A cryptographic countermeasure against connection depletion attacks," in *Proceedings of the Network and Distributed System Security Symposium, NDSS 1999, San Diego, California, USA*. The Internet Society, 1999.

[61] X. Wang and M. K. Reiter, "Defending against denial-of-service attacks with puzzle auctions," in *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2003, p. 78.

[62] X. Wang and M. K. Reiter, "Mitigating bandwidth-exhaustion attacks using congestion puzzles," in *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*. New York, NY, USA: ACM Press, 2004, pp. 257–267.

[63] S. Kandula, D. Katabi, M. Jacob, and A. Berger, "Botz-4-sale: surviving organized ddos attacks that mimic flash crowds," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, ser. NSDI'05. Berkeley, CA, USA: USENIX Association, 2005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251203.1251224 pp. 287–300.

[64] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y.-C. Hu, "Portcullis: protecting connection setup from denial-of-capability attacks," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 289–300, 2007.

[65] E. M. Chan, C. A. Gunter, S. Jahid, E. Peryshkin, and D. Rebolledo, "Using rhythmic nonces for puzzle-based dos resistance," in *Proceedings of the 2nd ACM workshop on Computer security architectures*, ser. CSAW '08. New York, NY, USA: ACM, 2008. [Online]. Available: http://doi.acm.org/10.1145/1456508.1456518 pp. 51–58.

[66] V. Gligor, "Guaranteeing access in spite of distributed service-flooding attacks," in *Security Protocols*, ser. Lecture Notes in Computer Science, B. Christianson, B. Crispo, J. Malcolm, and M. Roe, Eds. Springer Berlin / Heidelberg, 2005, vol. 3364, pp. 80–96, 10.1007/11542322_12. [Online]. Available: http://dx.doi.org/10.1007/11542322_12

[67] D. Clark, "The design philosophy of the darpa internet protocols," *SIGCOMM Comput. Commun. Rev.*, vol. 18, pp. 106–114, August 1988. [Online]. Available: http://doi.acm.org/10.1145/52325.52336

[68] V. Jacobson, "Congestion avoidance and control," *SIGCOMM Comput. Commun. Rev.*, vol. 18, pp. 314–329, August 1988. [Online]. Available: http://doi.acm.org/10.1145/52325.52356

[69] V. Cerf and R. Kahn, "A protocol for packet network intercommunication," *Communications, IEEE Transactions on*, vol. 22, no. 5, pp. 637 – 648, may 1974.

[70] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Trans. Comput. Syst.*, vol. 2, pp. 277–288, November 1984. [Online]. Available: http://doi.acm.org/10.1145/357401.357402

[71] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *SIGCOMM '89: Symposium proceedings on Communications architectures & protocols*. ACM Press, 1989, pp. 1–12.

[72] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 19, no. 4, pp. 1–12, 1989.

[73] L. Zhang, "Virtual clock: a new traffic control algorithm for packet switching networks," *SIGCOMM Comput. Commun. Rev.*, vol. 20, no. 4, pp. 19–29, 1990.

[74] J. Nagle, "Congestion control in IP/TCP internetworks," RFC 896 (), Internet Engineering Task Force, Jan. 1984. [Online]. Available: http://www.ietf.org/rfc/rfc896.txt

[75] J. Nagle, "On packet switches with infinite storage," *IEEE Trans. On Communications*, vol. 35, no. 4, pp. 435–438, April 1987.

[76] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the internet," *IEEE/ACM Trans. Netw.*, vol. 7, no. 4, pp. 458–472, 1999.

[77] I. Stoica, S. Shenker, and H. Zhang, "Core-stateless fair queueing: a scalable architecture to approximate fair bandwidth allocations in high-speed networks," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 33–46, 2003.

[78] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," in *SIGCOMM '95: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*. New York, NY, USA: ACM, 1995, pp. 231–242.

[79] P. McKenney, "Stochastic fairness queuing," in *IEEE Conference on Computer Communications (INFOCOM '90)*, June 1990.

[80] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, 1993.

[81] A. Studer and A. Perrig, "The coremelt attack," in *Proceedings of the 14th European conference on Research in computer security*, ser. ESORICS'09. Berlin, Heidelberg: Springer-Verlag, 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1813084.1813088 pp. 37–52.

[82] S. Kent and K. Seo, "Security architecture for the internet protocol," RFC 4301 (Proposed Standard), Internet Engineering Task Force, Dec. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4301.txt

[83] W. Diffie and M. Hellman, "New directions in cryptography," *Information Theory, IEEE Transactions on*, vol. 22, no. 6, pp. 644 – 654, nov 1976.

[84] M. Lelarge and J. Bolot, "A local mean field analysis of security investments in networks," in *NetEcon '08: Proceedings of the 3rd international workshop on Economics of networked systems*. New York, NY, USA: ACM, 2008, pp. 25–30.

[85] S. Floyd and E. Kohler, "Internet research needs better models," *SIGCOMM Comput. Commun. Rev.*, vol. 33, pp. 29–34, January 2003. [Online]. Available: http://doi.acm.org/10.1145/774763.774767

[86] S. Floyd and V. Paxson, "Difficulties in simulating the internet," *IEEE/ACM Trans. Netw.*, vol. 9, pp. 392–403, August 2001. [Online]. Available: http://dx.doi.org/10.1109/90.944338

[87] P. Faratin, D. Clark, P. Gilmore, S. Bauer, A. Berger, and W. Lehr, "Complexity of internet interconnections: Technology, incentives and implications for policy," *35th Annual Telecommunications Policy Research Conference*, pp. 1–31, 2007.

[88] S.-H. Yook, H. Jeong, and A.-L. Barabási, "Modeling the Internet's large-scale topology," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 21, pp. 13 382–13 386, Oct. 2002. [Online]. Available: http://dx.doi.org/10.1073/pnas.172501399

[89] A. Coates, A. Hero III, R. Nowak, and B. Yu, "Internet tomography," *Signal Processing Magazine, IEEE*, vol. 19, no. 3, pp. 47 –65, may 2002.

[90] V. Paxson, "End-to-end internet packet dynamics," *Networking, IEEE/ACM Transactions on*, vol. 7, no. 3, pp. 277 –292, June 1999.

[91] R. Fujimoto, K. Perumalla, A. Park, H. Wu, M. Ammar, and G. Riley, "Large-scale network simulation: how big? how fast?" in *Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003. 11th IEEE/ACM International Symposium on*, Oct. 2003, pp. 116 – 123.

[92] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A Blueprint for Introducing Disruptive Technology into the Internet," in *Proceedings of HotNets–I*, Princeton, New Jersey, Oct. 2002.

[93] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak, "Operating system support for planetary-scale network services," in *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*, vol. 1. Berkeley, CA, USA: USENIX Association, 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251175.1251194

[94] L. Peterson, A. Bavier, M. E. Fiuczynski, and S. Muir, "Experiences building planetlab," in *Proceedings of the 7th symposium on Operating systems design and implementation*, ser. OSDI '06. Berkeley, CA, USA: USENIX Association, 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1298455.1298489 pp. 351–366.

[95] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 255–270, December 2002. [Online]. Available: http://doi.acm.org/10.1145/844128.844152

[96] A. Li, X. Liu, and X. Yang, "Bootstrapping accountability in the internet we have," in *Proceedings of the 8th USENIX conference on Networked systems design and implementation*, ser. NSDI'11. Berkeley, CA, USA: USENIX Association, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=1972457.1972474 pp. 12–12.

[97] M. Wong and W. Schlitt, "Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1," RFC 4408 (Experimental), Internet Engineering Task Force, Apr. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4408.txt

[98] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Comput. Surv.*, vol. 36, pp. 335–371, December 2004. [Online]. Available: http://doi.acm.org/10.1145/1041680.1041681

[99] P. Gill, M. Schapira, and S. Goldberg, "Let the market drive deployment: a strategy for transitioning to bgp security," in *Proceedings of the ACM SIGCOMM 2011 conference on SIGCOMM*, ser. SIGCOMM '11. New York, NY, USA: ACM, 2011. [Online]. Available: http://doi.acm.org/10.1145/2018436.2018439 pp. 14–25.

[100] A. Bender, N. Spring, D. Levin, and B. Bhattacharjee, "Accountability as a service," in *SRUTI'07: Proceedings of the 3rd USENIX workshop on Steps to reducing unwanted traffic on the internet*. Berkeley, CA, USA: USENIX Association, 2007, pp. 1–6.

[101] D. R. Simon, S. Agarwal, and D. A. Maltz, "As-based accountability as a cost-effective ddos defense," in *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*. Berkeley, CA, USA: USENIX Association, 2007, pp. 9–9.

[102] F. Khan and C. A. Gunter, "Tiered incentives for integrity based queuing," in *Proceedings of the 2010 Workshop on Economics of Networks, Systems, and Computation*, ser. NetEcon '10, Usenix OSDI. Vancouver, BC, Canada: ACM, October 2010. [Online]. Available: http://doi.acm.org/10.1145/1879082.1879093 pp. 8:1–8:7.

[103] I. R. C. T. Fan, M. E. Muller, "Development of sampling plans by using sequential (item by item) selection techniques and digital computers," *J. Amer. Statist. Assoc.*, vol. 57, pp. 387–402, 1962.

[104] "CERT CC. smurf attack. http://www.cert.org/advisories/ca-1998-01.html."

[105] C. Labovitz, "The internet goes to war," December 14 2010. [Online]. Available: http://asert.arbornetworks.com/2010/12/the-internet-goes-to-war

[106] D. Karenberg, "Ripe ip anti-spoofing task force," May 2006.

[107] E. T. Krovetz, "Umac: Message authentication code using universal hashing," RFC 4418 (Informational), Internet Engineering Task Force, Mar. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4408.txt

[108] S. Chhabra, Y. Solihin, R. Lal, and M. Hoekstra, "An analysis of secure processor architectures," *Transactions on computational science VII*, pp. 101–121, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1880392.1880400

[109] "Voice over ip - per call bandwidth consumption, document id: 7934," Feb 02 2006. [Online]. Available: http://www.cisco.com/en/US/tech/tk652/tk698/technologies_tech_note09186a0080094ae2.shtml

[110] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Trans. Comput. Syst.*, vol. 18, pp. 263–297, August 2000. [Online]. Available: http://doi.acm.org/10.1145/354871.354874

[111] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, March 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[112] "I2 Usage Data. http://dc-snmp.wcc.grnoc.iu.edu/i2net/."

[113] S. Floyd, D. Wetherall, and R. Mahajan, "Controlling high-bandwidth flows at the congested router," in *ICNP '01: Proceedings of the Ninth International Conference on Network Protocols*. Washington, DC, USA: IEEE Computer Society, 2001, p. 192.

[114] "Abilene. http://abilene.internet2.edu."

[115] J. C. R. Bennett and H. Zhang, "Hierarchical packet fair queuing algorithms," in *SIGCOMM '96: Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 1996, pp. 143–156.

[116] I. Stoica, H. Zhang, and T. S. E. Ng, "A hierarchical fair service curve algorithm for link-sharing, real-time and priority services," *SIGCOMM Comput. Commun. Rev.*, vol. 27, pp. 249–262, October 1997. [Online]. Available: http://doi.acm.org/10.1145/263109.263175

[117] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. Wolff, "A brief history of the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 39, pp. 22–31, October 2009. [Online]. Available: http://doi.acm.org/10.1145/1629607.1629613