University of California
Los Angeles

D-WARD: Source-End Defense Against Distributed Denial-of-Service Attacks

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Jelena Mirkovic

2003

The dissertation of Jelena Mirkovic is approved.

Fernando Paganini

Songwu Lu

Deborah Estrin

Peter Reiher, Committee Co-chair

Mario Gerla, Committee Co-chair

University of California, Los Angeles

2003

To my loving parents, numerous dear friends and my love. They all believed in me much more than I did, patiently endured my explanations of strange research problems and invariantly offered same great advice — to work less, rest more and chill.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I am deeply grateful to my advisor, Peter Reiher, who has been such an amazing mentor. Enticing but not pushing, flexible but not lenient, he generously provided his time, effort and knowledgable advice at all times. He possesses a rare talent to organize and steer students with advice rather than commands, which both gives one an unparalleled freedom to explore bold ideas and limits procrastination. He also genuinely cares for his students' success, expediting their degree progress, supporting fair amount of conference travel and guiding them through the job search process. This is all done through regular weekly meetings, a few counselling sessions, an occasional deadline, and always coupled with unfaltering confidence in each one of us. Shaped by this gentle and unfaltering guidance, we all finally grow up and learn to stand on our own feet, which is the greatest wisdom one can be taught.

My sincere thanks goes to all the members of the LASR group at UCLA. Without their constant support, critique and clever suggestions, the road to PhD would have been desolate, long and hard. They warmly accepted me and cheered and helped along the way, providing also the right amount of fun, jokes and chocolate cookies. They endured with great patience my long and confused dry-runs for numerous talks, participated in extended weekly meetings and genuinely considered each of the strange ideas I insisted to discuss.

Janice Martin-Wheeler, our LASR assistant and English guru, has proofread and corrected innumerable technical papers that I and others vigorously produced. She not only corrected writing errors but spent long hours adjusting the style so that everything flows smoothly, like a poem. I would not be at all surprised if she eventually got a PhD in Computer Science — after such dedicated effort to understand and fix our papers she must be thoroughly educated in the

field. Each deadline was met and each paper brought close to perfection, only thanks to her keen eye, infinite patience and calm perseverance.

My friends were of invaluable help — I would never have made it without them. First, they made my life joyous and meaningful, and filled my hearth with warmth, and my days with laughter. They gave me peace when I needed peace and adventure when I craved it, but most of all they offered their shoulder whenever I wanted to cry, complain and stress about life, Universe and everything else. They always had time to listen to my whining, forgave my fits of nervousness, let me sleep on their sofas, cooked delicious dishes and coffee, and helped me understand that worry is not directly proportional to effort.

My love, Nikola, was simply that missing piece of myself that once found, made me wonder how I could ever have lived without him. He loved me invariantly regardless of how awful I was, fulfilled my impossible requests, and stood by me in every situation. He approached with great enthusiasm every research problem I wanted to discuss, and thoroughly examined it with me. Sometimes this provided a new insight I was missing, other times it simply organized my thoughts letting me see more clearly. He was always ready to listen to my elaborate descriptions of every little worry I had in life and, unlike any other person I know, he treated my problems with simple blend of ignorance, humor and love. This never failed to make me understand, that I am simply too dramatic and laughable.

I am eternally indebted to my loving parents. They readily and selflessly let me pursue any dream I had, even when this meant moving across half of the Earth, and visiting them only twice a year. They also gave me everything I ever needed, and loved and supported me without ever holding back.

And finally, even though it may sound cheesy, I am honestly grateful to the United States of America. This country accepted me with only two-month-living

money in my pocket, sponsored my education, awarded my degree and offered me a great job, without asking for anything in return. I have never felt like a stranger, and I have found help and support for every need I had. This, I believe, is "an American dream."

Vita

1974	Born, Belgrade, Serbia
1989–1993	Graduated Economics High School, Belgrade, Serbia. Then made a career change and enrolled in School of Electrical Engineering.
1998	B.Sc. in Computer Science, School of Electrical Engineering, University of Belgrade, Belgrade, Serbia. Awarded Best Student Award in Computer Science Department.
1998-2000	Teaching Assistant for Introductory and Advanced C++ courses at UCLA Computer Science Department.
2000	M.S. in Computer Science, UCLA, Los Angeles, California.
2000-2001	Graduate Student Researcher in Internet Research Laboratory, Computer Science Department, UCLA, under direction of Professor Lixia Zhang
2001-2003	Graduate Student Researcher in Laboratory for Advanced Systems Research, Computer Science Department, UCLA, under direction of Professor Peter Reiher.
2003-	Assistant Professor at Computer Science Department, University of Delaware, Newark, Delaware.

Publications and Presentations

M. Burns, G. Prier, J. Mirkovic and P. Reiher, "Implementing Address Assurance in the Intel IXP Router," *Proceedings of NPC 2002*.

J. Li, J. Mirkovic, M. Wang, P. Reiher and L. Zhang, "SAVE: Source Address Validity Enforcement Protocol," *Proceedings of INFOCOM 2002*.

J. Li, J. Mirkovic, M. Wang, P. Reiher and L. Zhang, "SAVE: Source Address Validity Enforcement Protocol", presented at *INFOCOM 2002*, New York, June 2002.

J. Li, J. Mirkovic, M. Wang, P. Reiher and L. Zhang, "SAVE: Source Address Validity Enforcement Protocol," *UCLA Technical Report CSD-TR-010004*.

V. Milutinovic, D. Cvetkovic and J. Mirkovic, "Genetic Search Based on Multiple Mutations," *IEEE Computer*, pp. 118-119, November 2000.

J. Mirkovic, J. Martin and P. Reiher, "A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms," *UCLA CSD Technical Report CSD-TR-020018.*

J. Mirkovic, G. Prier and P. Reiher, "Source-End DDoS Defense," *Proceedings of Network Computing and Applications Symposium NCA 2003*.

J. Mirkovic, G. Prier and P. Reiher, "Source-End DDoS Defense," presented at

NCA 2003, Boston, April 2003.

J. Mirkovic, G. Prier and P. Reiher, "Attacking DDoS at the Source," *Proceedings of ICNP 2002*.

J. Mirkovic, G. Prier and P. Reiher, "Attacking DDoS at the Source," presented at *ICNP 2002*, Paris, France, November 2002.

J. Mirkovic, G. Prier and P. Reiher, "Source Router Approach to DDoS Defense," *UCLA CSD Technical Report CSD-TR-010042*.

J. Mirkovic, G. P. Venkataramani, S. Lu and L. Zhang, "A Self-Organizing Approach to Data Forwarding In Large-Scale Sensor Networks," *Proceedings of ICC 2001*.

J. Mirkovic, G. P. Venkataramani, S. Lu and L. Zhang, "A Self-Organizing Approach to Data Forwarding In Large-Scale Sensor Networks", presented at *ICC 2001*, Helsinki, Finland, June 2001.

J. Mirkovic, M. Robinson, and P. Reiher, "Forming Alliance for DDoS Defense," *Proceedings of New Security Paradigms Workshop NSPW 2003*.

J. Mirkovic, Z. Xu, J. Li, M. Schnaider, P. Reiher and L. Zhang, "iSAVE: Incrementally Deployable Source Address Validation," *UCLA CSD Technical Report CSD-TR-020030*.

Abstract of the Dissertation

D-WARD: Source-End Defense Against Distributed Denial-of-Service Attacks

by

Jelena Mirkovic

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2003

Professor Mario Gerla, Co-chair

Professor Peter Reiher, Co-chair

Distributed denial-of-service (DDoS) attacks are a grave and challenging problem. Perpetration requires little effort on the attacker's side, since a vast number of insecure machines provides fertile ground for attack zombies, and automated scripts for exploit and attack can easily be downloaded and deployed. On the other hand, prevention of the attack or the response and traceback of perpetrators is extremely difficult due to a large number of attacking machines, the use of source-address spoofing and the similarity between legitimate and attack traffic.

Many defense systems have been designed in the research and commercial communities to counter DDoS attacks, yet the problem remains largely unsolved. This thesis explores the problem of DDoS defense from two directions: (1) it strives to understand the origin of the problem and all its variations, and provides a survey of existing solutions, and (2) it presents the design (and implementation) of a source-end DDoS defense system called D-WARD that prevents outgoing attacks from deploying networks. Source-end defense is not the complete solution to DDoS attacks, since networks that do not deploy the proposed defense can still

perform successful attacks. However, this thesis shows that a source-end defense (implemented in the D-WARD system) can detect and prevent a significant number of DDoS attacks, does not incur significant cost for its operation, and offers good service to legitimate traffic during the attack. By performing successful differentiation between legitimate and attack traffic close to the source, source-end defense is one of the crucial building blocks of the complete DDoS solution and essential for promoting Internet security. The thesis also includes a description of two joint projects where D-WARD has been integrated into a distributed defense system, and extensively tested. In all of the experiments, the operation of the system significantly improved with the addition of D-WARD.

CHAPTER 1

Introduction

Distributed denial-of-service attacks (DDoS) attacks consist of an overwhelming quantity of packets being sent from multiple attack sites to a victim site. These packets arrive in such a high quantity that some key resource at the victim (bandwidth, buffers, CPU time to compute responses) is quickly exhausted. The victim either crashes or spends so much time handling the attack traffic that it cannot attend to its real work. Thus legitimate clients are deprived of the victim's service for as long as the attack lasts.

Distributed denial-of-service attacks are widely regarded as a major threat to the Internet. They have adversely affected service to individual machines, major Internet commerce sites, and even core Internet infrastructure services. Occasionally, a very large-scale DDoS attack occurs (usually as the byproduct of a virus or worm spread), crippling Internet-wide communications for hours. While services are restored as soon as the attack subsides, the incidents still create a significant disturbance to the users and costs victim sites millions of dollars in lost revenue. Furthermore, the Internet is used daily for important communications such as stock trades, financial management and even some infrastructure services. Many of these transactions must be processed in a timely manner and can be seriously delayed by the onset of a DDoS attack. The seriousness of the threat is further increased by the ease om which these attacks are performed. Any unsophisticated user can easily locate and download DDoS tools and engage

them to perform successful, large-scale attacks. The attacker runs almost no risk of being caught. All of these characteristics have contributed to a widespread incidence of DDoS attacks ([MVS01] reports more than 12,000 attacks per week).

The first large-scale appearance of distributed denial-of-service (DDoS) attacks occurred in mid-1999. Today, four years later, researchers are still struggling to devise an effective solution to the DDoS problem. Although many commercial and research defenses have appeared, none of them provide complete protection from the threat. Rather, they detect a small range of attacks that either use malformed packets or create severe disturbances in the network; and they handle those attacks by non-selectively dropping a portion of the traffic destined for the victim. Clearly this strategy relieves the victim from the high-volume attack, but also inflicts damage to legitimate traffic that is erroneously dropped.

There are two DDoS attack features that hinder the design of more effective defenses:

1. **DDoS traffic is highly similar to legitimate traffic.** The attack usually consists of legitimate packets, generated in high quantity. They blend completely with the small amount of legitimate client traffic, so no differentiation can be made on a packet-by-packet basis. To perform traffic separation, the defense system must group all packets targeting the victim into higher-semantic structures (such as "all traffic exchanged between two IP addresses," "all HTTP traffic," "all traffic generated from a given source IP address," etc.), then keep many statistics on the dynamics of those structures to detect high-volume or anomalous communications. Packets that belong to suspect structures will then be policed, while packets belonging to structures that exhibit legitimate behavior will be forwarded.

2. **DDoS traffic is distributed.** Attack streams are generated from numerous attack machines spread all over the Internet and converge only in the proximity of the victim. The defense system must control a large portion of the total attack to alleviate the denial-of-service effect on the victim. This indicates that a system must either be a single-point system located near the victim or a distributed system whose defense nodes cover a significant portion of the Internet.

These two features create contradictory requirements for DDoS defense. In order to perform accurate traffic separation, the defense system requires a lot of resources for record-keeping. Therefore, it can only handle small to moderate traffic volumes. On the other hand, the need to control a large portion of the attack traffic requires placement at points that relay a high traffic volume. Those two requirements can hardly be satisfied at a single deployment point. A majority of DDoS defense systems sacrifice the first goal — traffic separation — to achieve the second goal — control of a large portion of the attack traffic. Those systems are located at or near the victim site, which enables them to detect and control the majority of DDoS attacks, but also places the defense system on the path of high-volume traffic, which impairs its selectiveness.

This thesis takes a different approach. It proposes a **source-end** defense system, called D-WARD, located at networks that are hosting some of the attack machines. D-WARD monitors and polices the outgoing traffic from those networks, thus controlling attacks. Placing the defense at the source-end exposes the defense system to low-to-moderate traffic volumes, thus enabling sophisticated profiling and traffic separation. The system thus provides a highly selective response to DDoS attacks, inflicting almost no damage to legitimate traffic.

Source-end defense is not a complete answer to DDoS attacks. Clearly, it

can control only the traffic from the networks that deploy the proposed defense system. If the critical mass[1] of the attack machines is located in unprotected networks, the attacker will still successfully deny service to legitimate clients of the victim. On the other hand, source-end defense provides two very important features:

- It places the response close to the sources, thus relieving shared Internet resources from the attack as soon as possible.
- It provides a selective response, minimizing collateral damage to legitimate traffic.

Both of these features make source-end defense highly attractive for integration with other DDoS defense systems that are placed closer to the victim. As those systems police all incoming traffic to the victim, they can improve their response selectiveness by detecting and forwarding packets already policed by a source-end defense. This approach creates a dedicated channel to the victim for those legitimate clients that deploy a source-end defense. They feel no denial-of-service effect and communicate unhindered with the victim.

1.1 Key Contributions

This thesis makes several key contributions to the DDoS defense field:

1. It demonstrates the feasibility of autonomous source-end DDoS detection. D-WARD detects a broad range of attacks using simple traffic models while keeping a reasonable amount of traffic statistics.

[1] The smallest number of attack machines that can overwhelm the targeted resource

2. It proposes a new approach to traffic profiling that successfully separates legitimate from attack traffic. This separation enables D-WARD to apply a selective attack response, thus inflicting almost no collateral damage to legitimate traffic.

3. It proposes a dynamic rate-limit assignment that quickly adjusts the system's response to observed traffic changes. This enables the system to promptly constrain the attack traffic, and to remove rate limits shortly after the attack has subsided.

4. It provides detailed analysis and classification of current DDoS attack and defense mechanisms, thus improving an understanding of the problem and current solutions.

1.2 Roadmap of the Dissertation

This thesis is organized in the following manner. Chapter 2 provides a detailed analysis of the DDoS threat. Chapter 3 provides an overview of DDoS defense challenges and current solutions. Chapter 4 organizes the information presented in Chapters 2 and 3 and structures it into DDoS attack and DDoS defense taxonomies. This facilitates a global view of the problem and solution space. Chapter 5 discusses benefits and challenges of source-end defense. Chapter 6 presents the philosophy and design of the D-WARD system. It discusses our motivation for the current design and also presents in great detail key components of the system. Chapter 7 provides a highly detailed overview of D-WARD implementation in a Linux router. Chapter 8 discusses the problem of evaluating DDoS defense systems and presents the experiment setup that will be used for D-WARD evaluation. Chapter 9 presents extensive performance results that demonstrate

D-WARD effectiveness in numerous attack scenarios. Chapter 10 describes cooperation efforts with three other DDoS defense projects: COSSACK, DefCOM and DDoS-DATA. Chapter 11 provides an overview of the related work. Chapter 12 discusses new directions for DDoS research that have been opened with D-WARD. We conclude the thesis in Chapter 13.

CHAPTER 2

Distributed Denial-of-Service Attacks

Denial-of-service (DoS) and distributed-denial-of-service (DDoS) attacks pose a grave danger to Internet operation. They are, in essence, resource overloading attacks. The goal of the attacker is to tie up a chosen key resource at the victim, usually by sending a high volume of seemingly legitimate traffic requesting some service from the victim. The overconsumption of the resource leads to degradation or denial of the victim's service to its legitimate clients.

In the absence of effective defense mechanisms, the denial-of-service effect lasts for the entire duration of the attack (i.e., as long as key resources are being tied with malicious traffic), and vanishes quickly once the attack is aborted. Since machine resources are usually shared among many applications, the DoS effect inflicts significant damage — not only on client transactions with the victim, but on the victim's total operation. The victim experiences a significant slowdown in all applications sharing the targeted resource, and frequently also connectivity disruption.

Both DoS and DDoS attacks are seemingly simple in design and operate without requiring any special skill or resource for their perpetration. The attack tools can be obtained easily online and the attack goal (resource exhaustion) is attained whenever a sufficiently large amount of malicious traffic is generated. The targeted resource dictates the type and contents of attack packets, e.g. exhaustion of CPU resources requires computation-intensive packets such as CGI or authen-

tication requests, while network resources can be exhausted by any high-volume traffic.

The main difference between DoS and DDoS attacks is in scale — DoS attacks use one attack machine (to generate malicious traffic) while DDoS attacks use large numbers of attack machines. The scale difference also invokes differences in operation modes. The large number of attack machines allows DDoS perpetrators a certain recklessness — they frequently trade sophistication for brute force, using simple attack strategies and packet contents to overload victim resources. However, the simplicity in both attack types arises from convenience, not necessity. The lack of effective defense mechanisms, even for simple attacks, offers no motivation for perpetrators to design more sophisticated ones. Once defenses successfully counter one attack class (e.g., like ingress filtering [FS00] has countered random IP source spoofing), attackers quickly deploy slight modifications in their attacks to bypass defensive actions.

There are many attack variations and many dimensions in which attacks can still evolve while preserving the ability to inflict damage on the victim. This feature makes it very challenging to design successful defenses. Due to attack variety, defense systems must maintain a volume of statistical data in order to detect attacks and sieve legitimate from attack traffic. This incurs high operation costs. On the other hand, attackers can easily bypass or trick defenses with slight modifications to their attacks. Any such modifications require added complexity in defense mechanisms (in order to handle the new attack class), thus skyrocketing the cost.

This chapter explains details of the denial-of-service phenomena.

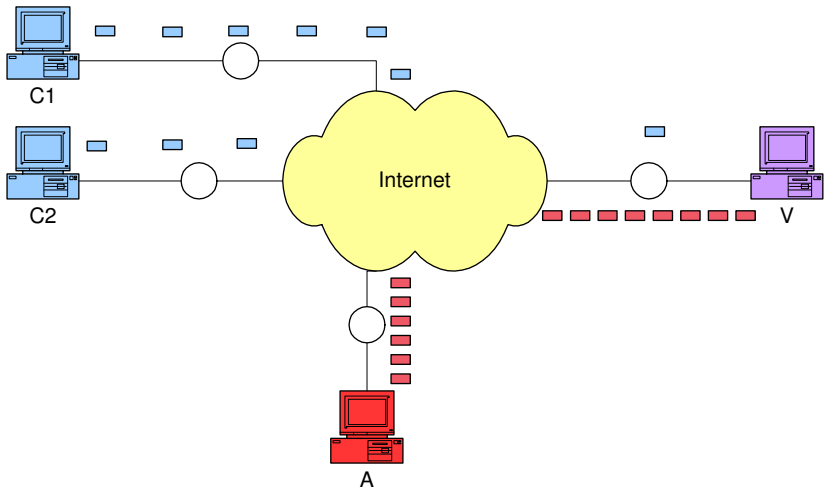

Figure 2.1: Denial-of-service attack scenario

2.1 Denial-of-Service Attacks

A denial-of-service (DoS) attack occurs when the victim receives a malicious stream of packets that exhausts some key resource; this results in denial-of-service to the victim's legitimate clients. Figure 2.1 depicts a typical denial-of-service attack scenario in which an attacking machine A sends a stream of malicious packets to victim V, denying its service to legitimate clients C1 and C2. Attackers rarely use their own machines to perform attacks, so machine A is, in fact, an *agent machine*, an unwitting participant subverted by the attacker.

The attack may exhaust a key resource by misusing some vulnerability in the software running at the victim (*vulnerability attacks*) or by simply sending a higher volume of traffic than the victim is provisioned to handle (*flooding attacks*). Vulnerability attacks usually contain packets of a special type or content to perform the exploit. As vulnerabilities can frequently be exploited by a few packets, vulnerability attacks are of a low-volume. Both of these features (special type packets and low volume) simplify handling of vulnerability attacks — the

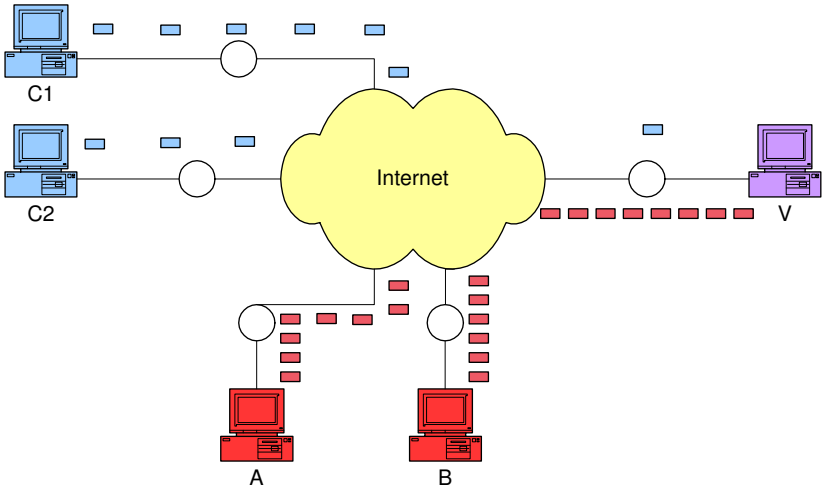

Figure 2.2: Distributed denial-of-service attack scenario

victim can either patch its vulnerability or detect the special-type packets and handle them separately. Flooding attacks overwhelm the victim's resource by sheer volume. This strategy is more difficult to counter, as malicious packets can be of any type or content and the high volume hinders detailed traffic analysis. As DoS attacks involve only one attacking machine, a common approach to defending against flooding attacks is to equip the victim with abundant resources. The attacker then needs to find and subvert a better-provisioned machine to perform a successful attack. The difficulty of the attacker's task to find the adequate agent machine increases with the amount resources allocated to the victim.

2.2 Distributed Denial-of-Service Attacks

Distributed denial-of-service (DDoS) attacks are simply denial-of-service attacks performed from multiple subverted machines (agents). In the strawman and most frequently used scenario, all machines are engaged simultaneously and start generating as many packets as they can toward the victim. A large number of

participating agents enable the attacker to overload resources of very highly provisioned victims, with modest capabilities of agent machines. Figure 2.2 depicts a simple distributed denial-of-service attack scenario in which attacking machines A and B send streams of malicious packets to victim V, denying its service to legitimate clients C1 and C2.

There are several features of DDoS attacks that severely challenge the design of successful defenses:

- **Use of IP source spoofing.** Attackers frequently use *source address spoofing* during the attack — they fake information in the IP source address field in attack packet headers. One benefit attackers receive from IP spoofing is that it is extremely difficult to trace the agent machines. This, in turn, brings several dire consequences. Since agent machines run a very low risk of being traced, information stored on them (i.e., access logs) cannot help to locate the attacker himself. This greatly encourages DDoS incidents. Furthermore, hiding the address of agent machines enables the attacker to reuse them for future attacks. Last, as attack packets carry a wide variety of addresses, they appear as if they come from many disparate sources; this defeats fair-sharing techniques that are a straightforward solution to resource overloading problems. The other advantage that IP spoofing offers to the attackers is the ability to perform *reflector attacks*[Pax01]. The attacker requests (in the victim's name) a public service that generates large replies to specific small-size requests (amplification effect). The attacker generates as many requests for service as his resources permit, faking the victim's source address, and sends them to public servers. These servers direct a manyfold volume of replies to the victim (thus reflecting and multiplying the attack force) and overload its

resources. A common case of reflector attack is described in [CERe]. The attacker sends a large number of UDP-based DNS requests to a nameserver using a spoofed source IP address of a victim. Any nameserver response is sent back to the spoofed IP address as the destination. Because nameserver responses can be significantly larger than DNS requests, there is potential for bandwidth amplification. Even if the traceback problem[1] were solved, it would not help to address reflector attacks. The public servers are unwitting participants whose legitimate service is misused in the attack. They possess no information about the attacker. Also, their service cannot be disabled (i.e., to stop the attack) as this would inflict damage on numerous other clients. Depending on these servers' resources and the request volume, they could prevent reflector attacks by limiting the number of replies they are willing to generate to a particular IP address. This approach would require servers to cache requesting addresses, thus potentially consuming significant memory resources.

- **Large number of agent machines.** Even if traceback could be successfully performed in the face of IP spoofing, it is difficult to say what actions could be taken against hundreds or thousands of agent machines. Such a large number prevents any but crude automated responses aimed at stopping attack flows close to the sources.

- **Similarity of attack to legitimate traffic.** Any type of traffic can be used to perform a successful denial-of-service attack. Some traffic types require a higher attack volume for success than others, and attack packets of different types and contents target different resources. However, if the

[1]The goal of traceback approaches is to locate those machines that generate packet streams overwhelming the victim. In the case of reflector attacks, traceback can only locate intermediate parties that unwittingly participated in the attack.

goal is simply to cripple the victim's operation, it can be met by sending sufficiently large volumes of any traffic and clogging the victim's network.

Attackers tend to generate legitimate-like packets to perform the attack, obscuring the malicious flow within legitimate traffic. Since malicious packets do not stand out from legitimate ones, it is impossible to sieve legitimate from attack traffic based purely on examination of individual packets. A defense system must keep a volume of statistical data in order to extract transaction semantics from packet flows and thus differentiate some legitimate traffic (e.g. belonging to lengthy well-behaved transactions) from the attack traffic.

2.3 Origin of Denial-of-Service Phenomenon

Denial-of-service is not simply another weak spot in the Internet, a slip that can be mended with slight protocol changes or by deployment of sophisticated defenses at potential target sites. The origin of denial-of-service lies in the very core of the Internet architecture. Design decisions reached several decades ago, that brought us connectivity and information wealth beyond our wildest dreams, carry within their key concepts the root of the DDoS threat.

The Internet was designed with functionality, not security, in mind, and it has been very successful in reaching its goal. It offers participants fast, simple and cheap communication mechanisms at the network level that provide "best effort" service to a variety of protocols. The only claim made is that the Internet will make a best attempt to move packets from a sender to a destination. Packet loss, reorder or corruption, sharing of Internet resources, different service levels for different traffic types and similar performance issues are handled by higher-level

transport protocols deployed at the end hosts — the sender and the receiver. These two principles, *best-effort service* and the *end-to-end paradigm* are the cornerstones upon which the Internet was built. Simple basic service provided by the IP protocol and the "best effort" principle enabled the building of numerous transport protocols on top of the IP to provide various performance guarantees: TCP for reliable delivery, RTP, RTCP and RTSP for streaming media, ICMP for control, etc. The end-to-end paradigm enabled end users to manage their communication any way they desired, adding complexities such as encryption and authentication, while the intermediate network remained simple and efficient.

Problems arise when one of the parties in the end-to-end model becomes malicious and acts to damage the other party. In that scenario, end-to-end protocols are violated and provide no more guarantees. At the same time, the end-to-end paradigm prevents the intermediate network from stepping in and policing the violator's traffic. Instead, it continues passively forwarding packets to their destination, where they overwhelm the victim's resources.

This problem first became evident in October 1986 when the Internet suffered a series of congestion collapses [Nag84]. Although the problem was quickly addressed by the design and deployment of several TCP congestion control protocols [Flo00], end-to-end flow management was unable to ensure a fair allocation of resources in the presence of aggressive flows (i.e., those that would not deploy congestion control). This problem was recognized and finally handled by enlisting the help of intermediate routers to monitor and police bandwidth allocation among flows to ensure fairness. There are two major mechanisms deployed in today's routers for congestion avoidance purposes — *active queue management* and *fair scheduling* algorithms [BCC98]. A similar approach that engages intermediate routers in flow management may be needed to completely solve the

DDoS problem. We explore this idea further in Section 3.4.

The following list summarizes several features of Internet design that open security issues and create opportunities for denial-of-service attacks:

1. **Internet security is highly interdependent.** DDoS attacks are commonly launched from systems that are subverted through security-related compromises. Regardless of how well secured the victim system may be, its susceptibility to DDoS attacks depends on the state of security in the rest of the global Internet[CER01].

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

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

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

5. **Intelligence and resources are not collocated.** An end-to-end communication paradigm led to storing most of the intelligence needed for service

guarantees with end hosts, limiting the amount of processing in the intermediate network so that packets could be forwarded quickly and at minimal cost. At the same time, a desire for large throughput led to the design of high bandwidth pathways in the intermediate network, while the end networks invested in only as much bandwidth as they thought they might need. Thus, malicious clients can misuse the abundant resources of the unwitting intermediate network for delivery of numerous messages to a less provisioned victim.

6. **Accountability is not enforced.** The source address field in an IP packet is assumed to carry the IP address of the machine that originates the packet. This assumption is not generally validated or enforced at any point on route from the source to the destination. This creates the opportunity for source address spoofing — the forging of source address fields in packets. Source address spoofing gives attackers a powerful mechanism to escape accountability for their actions, and sometimes even the means to perpetrate attacks (reflector attacks, such as the Smurf [CERj] attack).

2.4 Attacker Goals

The goal of a DDoS attack is to inflict damage on the victim. Frequently the ulterior motives are personal reasons (a significant number of DDoS attacks are perpetrated against home computers, presumably for purposes of revenge), or prestige (successful attacks on popular Web servers gain the respect of the hacker community). However, it is not unlikely that some DDoS attacks are performed for material gain (damaging competitor's resources, such as the recent case of Linux fans attacking SCO [Sha03] because of its lawsuit against IBM) or for

political reasons (a country at war could perpetrate attacks against its enemy's critical resources, potentially enlisting a significant portion of the entire country's computing power for this action). In some cases, the true victim of the attack might not be the actual target of the attack packets, but others who rely on the target's correct operation. For example, in September 2002 there was an onset of attacks that overloaded the Internet infrastructure rather than targeting specific victims [Nar02].

It also frequently happens that a DDoS attack is perpetrated accidentally, as a byproduct of another malicious activity, such as worm spread [Moo, Sym]. Inefficient worm-spreading strategies create massive traffic that congests the Internet and creates a denial-of-service effect to numerous clients.

While ordinary home users are less likely to become victims of DDoS attacks than large corporate networks, no one is free from the DDoS threat. The next attack may target AOL servers, denying service to many home users, or the next worm may congest the Internet so severely that no one can receive service. DDoS is an Internet-wide problem and all parties should cooperate to find a suitable solution.

2.5 Modus Operandi

A distributed denial-of-service is carried out in several phases. The attacker first **recruits** multiple agent (*slave*) machines. This process is usually performed automatically: the attacker downloads a scanning tool and deploys it from other compromised machines under its command (*masters*). The tool scans remote machines, probing for security holes that will enable subversion. Vulnerable machines are then **exploited** — broken into using the discovered vulnerability. They

are subsequently **infected** with the attack code. The exploit/infect phase is also automated, and the infected machines can be used for further recruitment of new agents.

Attackers attempt to cover the fact that agent machines have been compromised. They erase all logs showing malicious activity to destroy evidence that could incriminate them. They also hide attack scripts under system directories and give them obscure, non-suspicious names so they will not attract a user's attention and be erased. Sometimes they patch the vulnerability used for the exploit, to prevent other hackers from taking over the machine. Current exploit/infection scripts contain automated tools for covering tracks, so even inexperienced attackers do not leave much evidence of the subversion.

During a DDoS attack, agent machines are **engaged** to send the attack packets to the victim. The attacker orchestrates the onset of the attack, and scenario details such as the desired type and duration and the target address from the master to the agent machines. Agent machines usually fire out the packets at a maximum possible rate to increase the attack's chances of success. However, there have been attacks where agents were generating packets at a small rate (to prevent agent discovery) or where agent machines were periodically pausing the attack to avoid detection (*pulsing attacks*). Attackers usually hide the identity of subverted machines during the attack through spoofing of the source address field in attack packets. Note, however, that spoofing is not always required for a successful DDoS attack. With the exception of reflector attacks that use spoofing as an attack tool, all other attack types use spoofing only to hinder detection and discovery of agent machines.

Figure 2.3 illustrates the recruitment, exploitation, infection and engagement phases, depicting also the master/slave architecture of compromised machines.

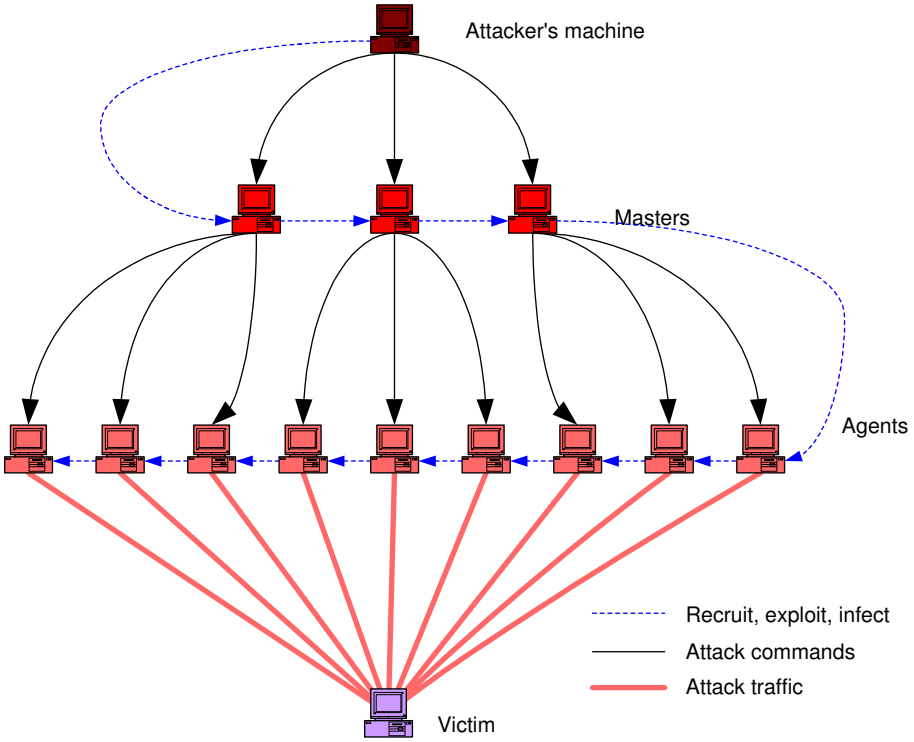

Figure 2.3: Distributed denial-of-service attack: modus operandi

2.6 Commonly Observed Attacks

While there are many ways to create the denial-of-service effect, there are a handful of attacks that have been commonly observed in the majority of DDoS incidents.

UDP flooding attack. During this attack the victim is flooded by numerous UDP packets that overwhelm its network bandwidth. To fully exploit bandwidth resources, packets usually have a large size. This attack is very simple to perpetrate, as the attacker need not discover (and take advantage of) any vulnerability at the victim. Simply by deploying a large number of agents, he can ensure the attack's success. On the other hand, many victim sites do not regularly receive incoming UDP traffic and can discard attack packets by deploying simple filtering rules. If filters are deployed at a high bandwidth point (e.g., an upstream router), this attack can be handled successfully.

TCP SYN flooding attack (open port). An attacker takes advantage of a vulnerability in the TCP protocol design to perpetrate a TCP SYN flooding attack. A TCP session starts with negotiation of session parameters between a requesting party — a client and a server. The client sends a TCP SYN packet to the server, requesting some service. In the SYN packet header, the client provides his *initial sequence number*, a unique per-connection number that will be used to keep count of data sent to the server (so the server can recognize and handle missing, reordered or repeated data). Upon SYN packet receipt, the server allocates a connection buffer record, storing information about the client. He then replies with a SYN-ACK, informing the client that its service request will be granted, acknowledging the client's sequence number and sending information about the server's initial sequence number. The client, upon receipt of the SYN-ACK packet, allocates a connection buffer record. The client then replies with an

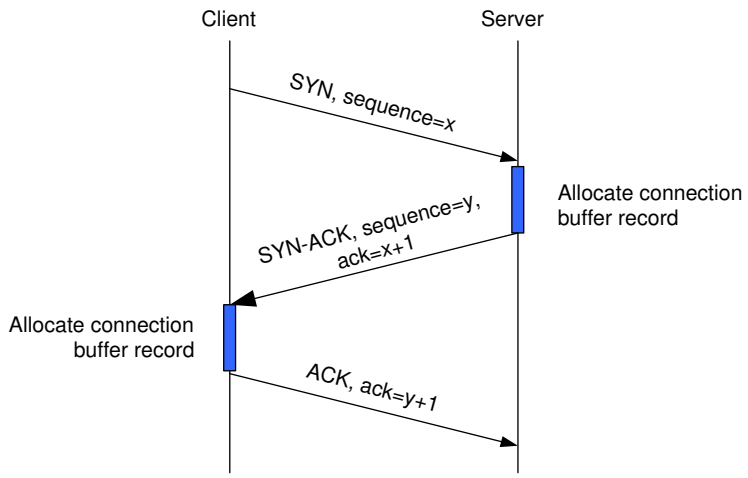

Figure 2.4: Opening of TCP connection: three-way handshake

ACK to the server which completes the opening of the connection. This message exchange is called a *three-way handshake* and is depicted in Figure 2.4.

The potential for abuse lies in the early allocation of the server's resources. When the server allocates his connection buffer space and replies with a SYN-ACK, the connection is said to be *half-open*. The server's allocated resources will be tied up until the client sends an ACK packet, closes the connection (by sending an RST packet) or until a timeout expires and the server closes the connection, releasing the buffer space. During a TCP SYN flooding attack, the attacker generates a multitude of half-open connections by using IP source spoofing. These requests quickly exhaust the server's connection buffer space, and the server can accept no more incoming connection requests. Established TCP connections usually experience no degradation in service. In rare cases, the server machine crashes, exhausts its memory or is otherwise rendered inoperative. In order to keep buffer space occupied for the desired time, the attacker needs to generate a steady stream of SYN packets toward the victim (to reserve again those resources that have been freed by timeouts).

The TCP SYN flooding attack is described in detail in [CERk, SKK97]. This is an especially vicious attack, as servers expect to see large numbers of legitimate SYN packets and cannot easily tell apart the legitimate from the attack traffic. No simple filtering rule can handle the TCP SYN flooding attack because legitimate traffic will suffer collateral damage. Several solutions for TCP SYN flooding have been proposed and their detailed description is given in [SKK97]. We also revisit the discussion of potential solutions in Section 6.4.3.

In order to perform a successful TCP SYN flooding attack, the attacker needs to locate an open TCP port at the victim. Then he generates a relatively small volume packet stream — as few as ten packets per minute[SKK97] can effectively tie up victim's resources. Another version of the TCP SYN flooding attack — *random port TCP SYN flooding* — is much less common. In it, the attacker generates a large volume of TCP SYN packets targeting random ports at the victim, with the goal of overwhelming the victim's network resources. Because TCP SYN packets are small, this is a very inefficient way of exhausting bandwidth, and is thus an unlikely attack.

ICMP flooding attack. During an ICMP flooding attack, the attacker generates a flood of *ICMP_ECHO* packets directed at the victim. The victim replies to each ICMP request, consuming its CPU resources (for reply generation) and network resources. This attack is as simple to perpetrate as a UDP flooding attack. As machines usually receive a very low volume of incoming ICMP packets, they can substantially defend against ICMP flooding attacks by deploying a simple rate-limiting rule at a high-bandwidth point (e.g., an upstream router), at the cost of dropping a few legitimate ICMP requests in the process.

Smurf attack. The Smurf attack[CERj] is a reflector attack. The attacker directs a stream of *ICMP_ECHO* requests to broadcast addresses in intermediary

networks, spoofing the victim's IP address in their source address fields. A multitude of machines then reply to the victim, overwhelming its network. This attack is easily countered either at the source network (generating forged *ICMP_ECHO* requests) by deploying ingress filtering [FS00] or at the intermediary network by ignoring/filtering out *ICMP_ECHO* requests targeting broadcast addresses.

Domain Name Service (DNS) reflector attack. This attack sends a stream of DNS requests to multiple nameservers, spoofing the victim's address in their source address fields [CERe]. Because nameserver responses can be significantly larger than DNS requests, there is potential for bandwidth amplification. Attackers usually request the same valid DNS record from multiple nameservers. If the target nameserver allows the query and is configured to be recursive or to provide referrals, the response could contain significantly more data than the original DNS request, resulting in a higher degree of bandwidth amplification. A target nameserver configured without restrictions on DNS query sources may not log malicious queries at all. An available defense at the source side (the network generating spoofed DNS requests) is to deploy ingress filtering. As intermediary servers receive legitimate-like requests, they cannot detect and prevent the attack (unless they exchange statistics on requesting addresses, or limit the number of responses to a given address).

2.7 Commonly Used Attack Tools

While there are numerous scripts that are used for scanning, compromise and infection of vulnerable machines, there are only a handful of DDoS attack tools that have been used to carry out the engagement phase. A detailed overview of these tools, along with a timeline of their appearance, is given in [CER01]. DDoS attack tools mostly differ in the communication mechanism deployed between

masters and slaves, and in the customizations they provide for attack traffic generation. The following paragraphs provide a brief overview of these popular tools. The reader should bear in mind that features discussed in this overview are those that have been observed in instances of attack code detected on some infected machines. Many variations may (and will) exist that have not yet been discovered and analyzed.

Trinoo[Dita] deploys a master/slave architecture, where an attacker sends commands to the master via TCP and masters and slaves communicate via UDP. Both master and slaves are password protected to prevent them from being taken over by another attacker. Trinoo generates UDP packets of a given size to random ports on one or multiple target addresses, during a specified attack interval.

Tribe Flood Network (TFN) [Ditc] also deploys a master/slave architecture. Agents can wage a UDP flood, TCP SYN flood, ICMP_ECHO flood and Smurf attacks at specified or random victim ports. The attacker communicates with masters using any of a number of connection methods (e.g., remote shell bound to a TCP port, UDP based client/server remote shells, ICMP-based client/server shells such as LOKI[rou97], SSH terminal sessions, or normal "telnet" TCP terminal sessions.) Remote control of TFN agents is accomplished via *ICMP_ECHOREPLY* packets. All commands sent from master to slaves through ICMP packets are coded, not cleartext, which hinders detection.

Stacheldraht[Ditb] (German for "barbed wire") combines features of Trinoo and TFN tools and adds encrypted communication between the attacker and the masters. Stacheldraht uses TCP for encrypted communication between the attacker and the masters, and TCP or ICMP for communication between master and agents. Another added feature is the ability to perform automatic updates of agent code. Available attacks are UDP flood, TCP SYN flood, ICMP_ECHO

flood and Smurf attacks.

Shaft[SD00] is a DDoS tool similar to Trinoo, TFN and Stacheldraht. Added features are the ability to switch master servers and master ports on the fly (thus hindering detection by intrusion detection systems), a "ticket" mechanism to link transactions, and a particular interest in packet statistics. Shaft uses UDP for communication between masters and agents. Remote control is achieved via a simple telnet connection from the attacker to the master. Shaft uses "tickets" for keeping track of its individual agents. Each command sent to the agent contains a password and a ticket. Both passwords and ticket numbers have to match for the agent to execute the request. A simple letter-shifting (Caesar cipher) is used to obscure passwords in sent commands. Agents can generate a UDP flood, TCP SYN flood, ICMP flood, or all three attack types. The flooding occurs in bursts of 100 packets per host (this number is hard-coded), with the source port and source address randomized. Masters can issue a special command to agents to obtain statistics on malicious traffic generated by each agent. It is suspected that this is used to calculate the yield of a DDoS network.

Tribe Flood Network 2000 (TFN2K)[CERd] is an improved version of the TFN attack tool. It includes several features designed specifically to make TFN2K traffic difficult to recognize and filter, to remotely execute commands, to obfuscate the true source of the traffic, to transport TFN2K traffic over multiple transport protocols including UDP, TCP, and ICMP, and features to confuse attempts to locate other nodes in a TFN2K network by sending "decoy" packets. TFN2K obfuscates the true traffic source by spoofing source addresses. Attackers can choose between random spoofing and spoofing within a specified range of addresses (to defeat ingress filtering [FS00]). In addition to flooding, TFN2K can also perform some vulnerability attacks by sending malformed or invalid

packets, as described in [CERl, CERg].

mstream[DWD] generates a flood of TCP packets with the ACK bit set. Masters can be controlled remotely by one or more attackers using a password-protected interactive login. The communications between attacker and masters, and a master and agents, are configurable at compile time and have varied significantly from incident to incident. Source addresses in attack packets are spoofed at random. The TCP ACK attack exhausts network resources and will likely cause a TCP RST to be sent to the spoofed source address (potentially also creating outgoing bandwidth consumption at the victim).

Trinity is the first DDoS tool that is controlled via IRC or ICQ. Upon compromise and infection by Trinity, each machine joins a specified IRC channel and waits for commands. Use of legitimate (IRC or ICQ) service for communication between attacker and agents eliminates the need for a master machine and elevates the level of the threat, as explained in Section 4.1. Trinity is capable of launching several types of flooding attacks on a victim site, including UDP, IP fragment, TCP SYN, TCP RST, TCP ACK, and other floods.

2.8 Summary

Distributed denial-of-service are simple attacks. They rarely use any sophisticated mechanism or complicated and covert actions (like viruses, worms or intrusion tools do). Instead they attack with brute force, gathering resources of numerous agents to overwhelm the victim. The difficulty in handling DDoS attacks lies exactly in their simplicity. Because they misuse legitimate protocols to perform denial-of-service, it is extremely difficult to separate attack traffic from legitimate traffic; this hinders both detection and response. IP spoofing

additionally complicates the problem.

This section has provided an overview of attack methods, most frequently seen incidents and popular attack tools. However, DDoS attacks are adversarial and constantly evolving. Once a particular kind of attack is successfully countered, a slight variation is designed that bypasses the defense and still performs an effective attack. DDoS attacks can afford to vary many of their features, such as IP header values, deployed protocols, agent sending rate, etc. As long as the victim receives a flood of packets that overwhelms its resources, the attack succeeds. This high variability needs to be sufficiently understood to design effective defenses. Chapter 4 provides a taxonomy of DDoS attacks (designed as part of this thesis work) with exactly this goal in mind.

CHAPTER 3

DDoS Defenses

The seriousness of the DDoS problem and the increased frequency, sophistication and strength of attacks have led to the advent of numerous defense mechanisms. Yet, although it has been several years since the first distributed attacks were perpetrated,[1] and many solutions have been developed since then, the problem is hardly dented, let alone solved. Why is this so?

3.1 Defense Challenges

The challenges to designing DDoS defense systems fall roughly into two categories: technical challenges and social challenges. Technical challenges encompass problems associated with the current Internet protocols and characteristics of the DDoS threat. Social challenges, on the other hand, largely pertain to the manner in which a successful technical solution will be introduced to Internet users, and accepted and widely deployed by these users.

The main problem that permeates both technical and social issues is the problem of large scale. DDoS is a distributed threat that requires a distributed solution. Attacking machines may be spread all over the Internet. Clearly, attack streams can only be controlled if there is a point of defense between the agents

[1] In [CER01] authors report that the first wide-spread incidences of DDoS attacks occurred in July 1999.

and the victims. One approach is to place one defense system close to the victim so that it monitors and controls all of the incoming traffic. This approach has many deficiencies (see Section 3.4), the main one being that the system must be able to efficiently handle and process huge traffic volumes. The other approach is to divide this workload by deploying distributed defenses. Defense systems must then be deployed in a widespread manner to ensure effective action for any combination of agent and victim machines. As widespread deployment cannot be guaranteed, the technical challenge lies in designing effective defenses that can provide reasonable performance even if they are sparsely deployed.

The social challenge lies in designing an economic model of a defense system in a manner that facilitates large-scale deployment in the Internet.

3.1.1 Technical Challenges

The distributed nature of DDoS attacks and use of legitimate traffic models and IP spoofing represent the main technical challenges to designing effective DDoS defense systems. In addition to that, the advance of DDoS defense research is hindered by the lack of attack information and absence of standardized evaluation and testing approaches. The following list summarizes and discusses technical challenges for DDoS defense:

1. **Need for a distributed response at many points on the Internet.** Section 9.1 elaborates on the fact that there are many possible DDoS attacks, very few of which can be handled only by the victim. Thus it is necessary to have a distributed, possibly coordinated, response system. It is also crucial that the response be deployed at many points on the Internet to cover diverse choices of agents and victims. Since the Internet is administered in a distributed manner, wide deployment of any defense system (or

even various systems that could cooperate) cannot be enforced or guaranteed. This discourages many researchers from even designing distributed solutions.

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

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

4. **Difficulty of large-scale testing.** DDoS defenses need to be tested in a realistic environment. This is currently impossible due to the lack of large-scale testbeds, safe ways to perform live distributed experiments across

the Internet, or detailed and realistic simulation tools that can support several thousands of nodes. Claims about defense system performance are thus made based on small-scale experiments and simulations, and are not credible.

3.1.2 Social Challenges

Many DDoS defense systems require certain deployment patterns to be effective. Those patterns fall into several categories:

1. Complete deployment
2. Contiguous deployment
3. Large-scale, widespread deployment
4. Complete deployment at specified points in the Internet
5. Modification of widely deployed Internet protocols, such as TCP, IP or HTTP
6. All (legitimate) clients of the protected target deploy defenses

None of the above requirements are practical for general purposes (although they may work well to protect an important server or application that communicates with a selected set of clients). The Internet is extremely large and is managed in a distributed manner. No solution, no matter how effective, can be deployed simultaneously in hundreds of millions of disparate places. On the other hand, there have been quite a few cases of an Internet product (a protocol, an application or a system) that has become so popular after release that it was

very widely deployed within a short time. Examples include Kazaa, SSH (Secure Shell) protocol , Internet Explorer, Windows OS, etc. The following factors determine a product's chances for wide deployment:

1. **Good performance.** A product must meet the needs of customers.

2. **Good economic model.** A customer must gain direct economic benefit, or at least reduce the risk of economic loss, by deploying the product. Alternately, the customer must be able to charge others for improved services resulting from deployment.

3. **Incremental performance.** As the degree of deployment increases, customers might experience increased benefits. However a product must offer considerable benefit to its customers even under sparse partial deployment.

3.2 Defense Goals

The primary goal of DDoS defense is to provide good service to a victim's legitimate clients during the attack, thus cancelling the denial-of-service effect. Ideally, clients should perceive little or no service degradation while the attack is ongoing. The secondary goal is to alleviate the effect of the attack on the victim so that its resources can be dedicated to legitimate clients or preserved. Last, attack attribution (locating with high accuracy agent machines and perpetrators of the attack) will serve as a strong deterrent to DDoS incidents, as attackers could face the risk of discovery and punishment.

3.3 Defense Approaches

DDoS defense approaches can roughly be divided into three categories: *preventive*, *survival* and *responsive* approaches.[2]

Preventive approaches introduce changes into Internet protocols, applications and hosts, in order to patch existing vulnerabilities and reduce the incidence of intrusions and exploits. Their goal is to prevent vulnerability attacks, and to impede the attacker's attempts to gain a large agent army. While preventive approaches are necessary for improving Internet security, they need to be deployed widely to constrain the DDoS threat. As long as large numbers of machines are insecure, attackers can still wage large-scale attacks. There is no reason to believe that preventive approaches will successfully undermine the power of the DDoS threat in the foreseeable future.

Survival approaches enlarge a victim's resources, enabling it to serve both legitimate and malicious requests during the attack, thus cancelling the denial-of-service effect. The enlargement is achieved either statically — by purchasing more resources, or dynamically — by acquiring resources at the sign of possible attack from a set of distributed public servers and replicating the target service. Enlargement approaches can significantly enhance a target's resistance to DoS. Replication approaches offer successful DDoS protection (and also load balancing) to static Internet content. The disadvantage is that not all public services (those that may be subject to the DDoS attacks) are replicable. For instance, dynamic Web pages, databases, remote login services, etc., can be replicated only with a great effort invested into synchronization and emulation. The effectiveness of survival approaches is limited to cases in which enlarged resources are greater

[2]In Section 4.2, preventive and survival approaches are both classified into the *preventive* category, while responsive approaches are classified into the *reactive* category.

than the attack volume. As an attacker can easily gather hundreds of thousands of agent machines, survival approaches are not likely to offer a complete solution to DDoS problem.

Responsive approaches detect the occurrence of the attack and respond to it ("fight back") either by controlling attack streams, or by attempting to locate agent machines and invoking human action. In order to be successful, response approaches must meet following requirements:

1. **Accurate detection.** The system must be able to detect all attacks that inflict damage at the victim.

2. **Effective response.** The system must stop the attack flows, regardless of their volume or distribution. Alternately, in the case of response by agent identification, the system must be able to accurately identify the majority of attack machines regardless of their distribution. This identification must be prompt so that the action can be taken while the attack is on-going. Ideally, identification responses should identify not only the agent machines, but also the master and the attacker machines.

3. **Selective response.** The system must differentiate between legitimate and attack packets, and ensure good service to legitimate traffic during the attack. Collateral damage due to the response must be lower than the damage suffered by legitimate clients in the absence of response. This requirement does not pertain to agent identification approaches.

We call these requirements *responsive defense requirements*. The remainder of this chapter will discuss only responsive approaches that react to incidents by controlling attack streams.

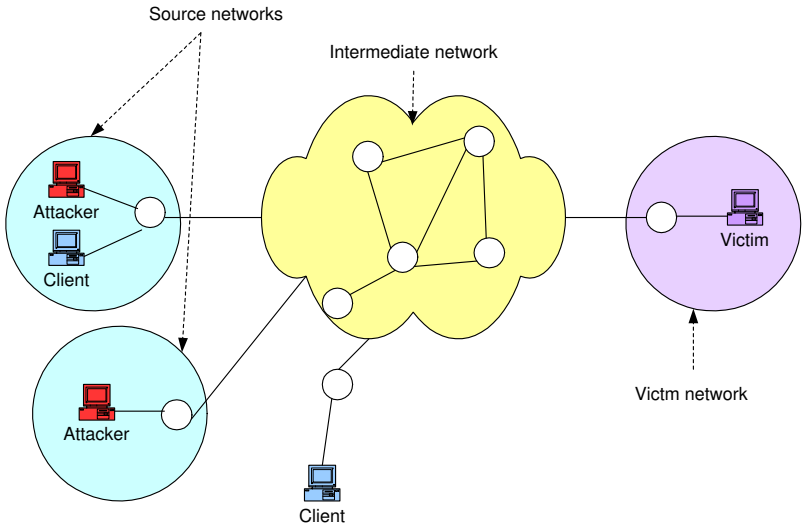

Figure 3.1: Points of defense

3.4 Points of Defense

A DDoS defense system can either be deployed as an autonomous (single-point) system or as a distributed system. Autonomous systems consist of a single defense node that observes the attack and applies the response. Distributed systems consist of multiple defense nodes (frequently with same functionality) that are deployed at various locations and organized into a network. Nodes communicate through the network and coordinate their actions to achieve a better overall defense.

3.4.1 Autonomous Defense

DDoS attack streams originate from distributed attack machines, are forwarded by core routers and converge at the victim network or some nearby core router. We observe this process as an interaction of three types of networks: *source networks* that unwittingly host attack machines, several *intermediate networks*

that forward attack traffic to the victim, and the *victim network* that hosts the target. Figure 3.1 depicts this interaction. Each of the involved networks (source, intermediate, or victim) can host DDoS defense systems. We now observe how responsive defense requirements are met by an autonomous system deployed at *only one* of these points.

3.4.1.1 Victim-End Defense

Historically, the majority of DDoS defense systems have been designed for victim-end deployment. This is understandable since the victim suffers the largest damage from a DDoS attack and is therefore motivated to invest in a defense system. A victim-end DDoS defense system facilitates easy detection because it can closely observe the victim, model its behavior and notice any anomalies. However, the range of response is limited. The defense system is on the path of the full-force attack, and may be overwhelmed by a large traffic volume. The point of failure is then simply moved from the target to the DDoS defense system. Alternately, the attacker may send enough traffic to overwhelm the victim's network connection in front of the defense system. The point of DoS is then beyond the defense system's scope. The other consequence of a large traffic volume is the limited amount of processing and storage that defense system can commit to. The differentiation of legitimate streams from attack streams is complex at this point, since they have been heavily aggregated by the time they reach the victim network. To perform sophisticated traffic profiling a system needs a large amount of storage and computational power to store and examine statistics on each stream. In the presence of IP spoofing, this is infeasible as each packet will appear to come from a different source. Poor traffic separation, in turn, leads to large collateral damage during a response.

3.4.1.2 Intermediate-Network Defense

The danger of a DDoS attack on network resources that is still present in victim-end defense was addressed by moving the defense further upstream, into the intermediate network. An intermediate-network defense system, usually installed at a core router, detects the attack through anomalies observed at this router. As core routers handle large-volume, highly aggregated traffic, they are likely to overlook all but large-scale attacks. Victim resources are frequently severely depleted by attacks that look like small glitches in the busy buffer of a core router. Detected attacks can be quickly suppressed, thanks to abundant network resources. However, response is likely to inflict collateral damage as core routers can only accommodate simple rate-limiting requests and cannot dedicate memory or processor cycles to traffic profiling.

3.4.1.3 Source-End Defense

As DDoS defense is pushed further from the victim to the source, detection capability diminishes. A source-end defense system can no longer easily observe the effect of incoming traffic on the victim. Further, as it may monitor only a small portion of the attack, the defense system has difficulties in detecting anomalies. On the other hand, response effectiveness increases with proximity to the sources. A small attack volume enables an effective response as it is unlikely to overwhelm the defense system. The small volume and degree of aggregation also facilitates complex profiling that, in turn, minimizes collateral damage.

3.4.2 Distributed Defense

Distributed systems for DDoS defense combine actions of victim-end, source-end and sometimes of intermediate-network defense systems. Victim-end defenses detect the attack and deliver the alert to other participants, who then cooperate to suppress attack streams. The goal is to install responses as close to the sources as possible, thus minimizing collateral damage.[3]

Distributed defenses are likely to be the proper solution for handling the DDoS threat. However, they are infrastructural solutions—they span multiple networks and administrative domains and represent major undertakings of many Internet participants. Such systems are difficult to deploy and maintain. Further, the required cooperation of defenses is hard to achieve due to distributed Internet management and strictly autonomous operation of administrative domains. Securing and authenticating the communication channels also incurs a high cost if the number of participants is large.

3.4.3 Where Does D-WARD Fit?

D-WARD is a source-end DDoS defense system. It is designed to operate both as an autonomous system and as a part of larger, distributed defense. As an autonomous system, it detects a wide range of attacks and controls them within seconds. The control is selective, thus enabling legitimate clients to receive good service from the victim during the attack. As part of a distributed system, D-WARD can receive an attack alert and rate-limiting directives, and apply a dynamic, effective and selective response. Its main feature, dynamic selective

[3] If a response inflicts collateral damage, i.e., it is not sufficiently selective, the effect will be imposed only on legitimate traffic coming from the source networks that also host attack machines. Thus, at least "purely legitimate" networks can deliver their traffic to the victim unimpeded.

response, greatly enhances the performance of any distributed defense system. While D-WARD is not a complete solution to DDoS attacks, it promotes Internet security, and is a valuable building stone for the complete solution.

3.5 Commonly Deployed Defenses

The commonly deployed approach to DDoS defense is to secure the potential victim so as to reduce the probability of denial-of-service effects for all but the very high-volume or stealthy DDoS attacks. This is done in the following manner:

- Keep all protocols and applications at the potential target up to date. Apply security updates and patches.
- Close all unused ports.
- Rate-limit, or filter, at the upstream router all incoming UDP and ICMP traffic that is not in reply to the requests issued from the target network.
- Purchase a large amount of network resources.
- If the target network provides a service (e.g., Web service), create a server pool and place the servers behind a load balancer.
- In the case of DDoS attack, devise the attack signature and ask the administrator of the upstream network to deploy the appropriate filters.

In addition to this, some networks use commercial solutions such as Mananet [Cs3] sold by CS3, Inc., PowerSecure [Maz] sold by Mazu Networks, PeakFlow [Arb] sold by Arbor Networks, Vantage System [Ast] sold by Asta Networks, etc., to protect themselves from DDoS attacks. Those solutions are deployed across the target network. They can usually identify (using proprietary techniques)

the ingress points that deliver a large portion of the attack traffic. Filtering or rate-limiting rules are then placed only at the identified ingress points.

3.6 Summary

The DDoS defense community faces technical and social challenges that hinder the design of effective and widely deployed defenses. Technical challenges lie in the design of defenses that detect a wide range of attacks, inflict small collateral damage to legitimate traffic and are effective in sparse deployment. The social challenge lies in the design of a good economic model of DDoS defense so that it can be widely deployed.

Defense approaches aim to prevent denial-of-service attacks (preventive approaches), to enable the victim to survive the attack without denying service to legitimate clients (survival approaches) or to detect and respond to the attack by selectively dropping attack traffic (responsive approaches). Preventive and survival approaches improve the security and raise the bar for DDoS attack success, but they cannot completely handle DDoS attacks. Responsive approaches show promise for complementing the action of preventive and survival approaches and completely addressing the DDoS problem.

To provide an effective defense, responsive approaches must accurately detect a wide range of attacks, effectively respond to detected attacks by stopping a large portion of the attack traffic, and apply selective response — thus inflicting low collateral damage to legitimate traffic. Attack detection is easiest at the victim network: a high-volume of incoming traffic or disturbed operation can be readily used as a sign of DDoS attack. Effective response, however, depends on the attack volume and victim network resources. No victim-end defense is possible

against sufficiently high-volume attacks — they overwhelm network resources even before they reach the defense system, leaving legitimate clients without service. Additionally, their high level of traffic aggregation hinders differentiation between legitimate and attack flows, leading to a non-selective response. Thus, while protecting the victim, the response penalizes some legitimate traffic, still leading to denial-of-service.

The selectiveness and effectiveness of response improve as the defense system is moved from the victim closer to the sources of the attack, but the detection accuracy deteriorates. Response is most effective at the source-end network, as attack streams can be stopped before they enter the Internet. Also, sophisticated profiling can be done to facilitate selectiveness of the response, since the attack traffic at the source is not highly aggregated. However, attack machines can be distributed among many source networks, thus each source network only observes a small amount of attack traffic that may appear legitimate, hindering detection.

Because none of the individual defense points can meet all three requirements of an effective defense, a distributed defense system presents itself as a likely solution.

CHAPTER 4

A Taxonomy of DDoS Attack and DDoS Defense Mechanisms

As previous chapters have discussed, the distributed denial-of-service attacks pose an immense threat to the Internet, and many defense mechanisms have been proposed to combat the problem. Attackers constantly modify their tools to bypass these security systems, and researchers in turn modify their approaches to handle new attacks. The DDoS field is quickly becoming more and more complex, and has reached the point where it is difficult to see the forest for the trees. On one hand, this hinders an understanding of the distributed denial-of-service phenomenon. Multitudes of known attacks create the impression that the problem space is vast, and hard to explore and address. On the other hand, existing defense systems deploy various strategies to counter the problem, and it is difficult to assess their effectiveness and cost, and to compare them to each other.

This chapter proposes a taxonomy of DDoS attacks and a taxonomy of DDoS defense systems. Together, they structure the DDoS field and facilitate a global view of the problem and solution space. By setting apart and emphasizing crucial features of attack and defense mechanisms, while abstracting detailed differences, these taxonomies can be used by researchers to answer many important questions:

- What are the different ways of perpetrating a DDoS attack? Why is DDoS

a difficult problem to handle?

- What attacks have been handled effectively by existing defense systems? What attacks still remain unaddressed and why?
- Given two defense mechanisms, A and B, how would they perform if attack C occurred? What is their deployment cost? What are their vulnerabilities? Can they complement each other and how? Are there some deployment points that are better suited for A than B and vice versa?
- What are the unsolved problems and how can one contribute to the field?

The proposed taxonomies are complete in the following sense: the attack taxonomy covers known attacks and also those which have not yet appeared but are potential threats that would affect current defense mechanisms; the defense system taxonomy covers not only published approaches but also some commercial approaches that are sufficiently documented to be analyzed. Along with classification, we provide representative examples of existing mechanisms.

We do not claim that these taxonomies are as detailed as possible. Many classes could be divided into several deeper levels. Also, new attack and defense mechanisms are likely to appear, thus adding new classes to the ones we propose. Our goal was to select several important features of attack and defense mechanisms that might help researchers design innovative solutions, and to use these features as classification criteria. It was also important not to confuse the reader with a too elaborate and detailed classification. It is our hope that our work will be further extended by other researchers.

We also do not claim that classes divide attacks and defenses in an exclusive manner, i.e., that an instance of an attack or a particular defense system must be classified into a single class based on a given criterion. It is possible for an attack

to be comprised of several mechanisms, each of them belonging to a different class.

The depth and width of the proposed taxonomies are not suitable for a traditional numbering of headings — numbers would quickly become too elaborate to follow. We therefore introduce a customized marking (numbering) of subsection headings in Sections 4.1 and 4.2. Each classification criterion is marked by abbreviating its name. Attack classes under this criterion are further marked by criterion abbreviation and an arabic number, connected by a dash. To indicate depth of a specific criterion or a class in the taxonomy, the complete mark of a subsection is generated by traversing the taxonomies depicted in Figure 4.1 and Figure 4.2, from root to the object in question, concatenating levels with a colon. For example: if an attack classification criterion is *degree of automation*, it will bear the mark *DA*. The second attack class under this criterion, *semi-automatic attacks*, will bear the mark *DA-2*. One level below, semi-automatic attacks are divided according to communication mechanism (heading mark *DA-2:CM*) into attacks with direct communication (heading mark *DA-2:CM-1*) and attacks with indirect communication (heading mark *DA-2:CM-2*). To keep the heading names short, some words are omitted. In the previous example, the subsection describing division by *degree of automation* will bear the heading *DA: Degree of Automation*, whereas the complete heading should be *DA: Attack Classification by Degree of Automation*. The subsection describing attacks with indirect communication will bear the heading *DA-2:CM-2: Indirect Communication*, whereas the complete heading should be *DA-2:CM-2: Semi-Automatic Attacks with Indirect Communication*.

4.1 Taxonomy of DDoS Attacks

In order to devise a taxonomy of distributed denial-of-service attacks, we observe the means used to prepare and perform the attack (recruit, exploit and infect phases), the characteristics of the attack itself (use phase) and the effect it has on the victim. Figure 4.1 summarizes the taxonomy. In the remainder of this section we discuss each of the proposed criteria and classes.

DA: Degree of Automation

Each of the recruit, exploit, infect and use phases can be performed manually or can be automated. Based on the degree of automation, we differentiate between *manual*, *semi-automatic* and *automatic* DDoS attacks.

DA-1: Manual

Only the early DDoS attacks belonged to the manual category. The attacker scanned remote machines for vulnerabilities, broke into them, installed attack code, and then commanded the onset of the attack. All of these actions were soon automated.

DA-2: Semi-Automatic

In semi-automatic attacks, the DDoS network consists of handler (master) and agent (slave, daemon) machines. The recruit, exploit and infect phases are automated. In the use phase, the attacker specifies the attack type, onset, duration and the victim via the handler to agents, who send packets to the victim.

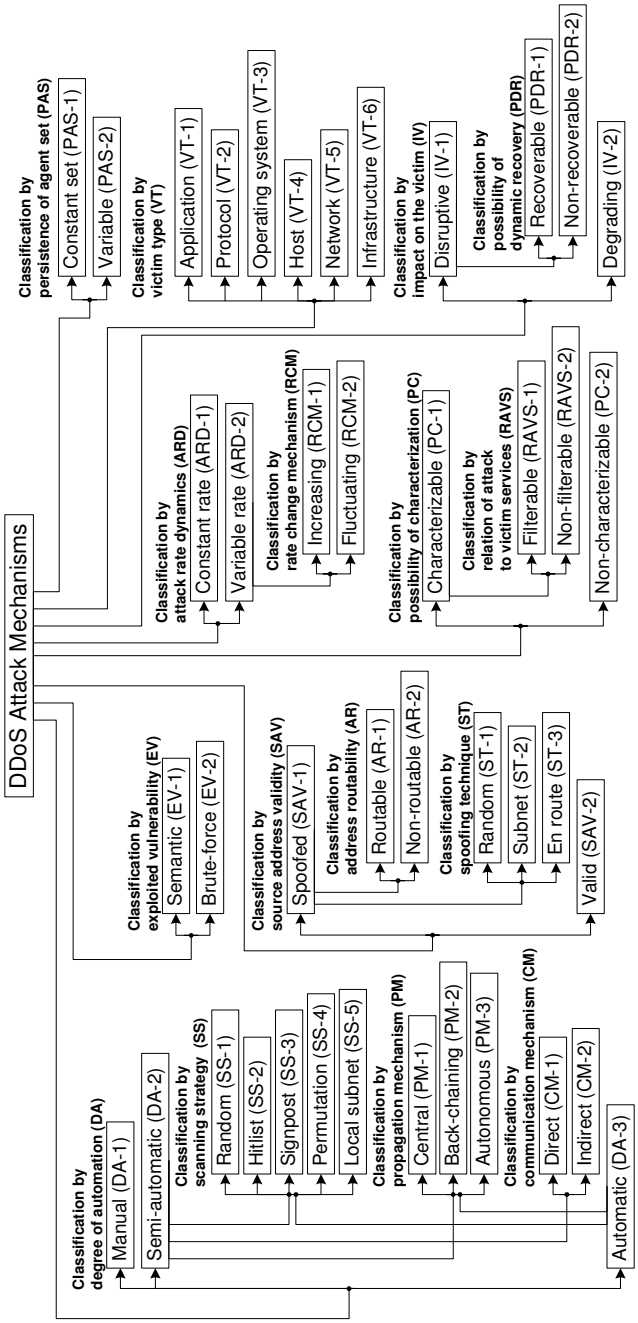

Figure 4.1: Taxonomy of DDoS attack mechanisms

DA-2:CM: Communication Mechanism

Based on the communication mechanism deployed between agent and handler machines, we divide semi-automatic attacks into *attacks with direct communication* and *attacks with indirect communication*.

DA-2:CM-1: Direct Communication

During attacks with direct communication, the agent and handler machines need to know each other's identity in order to communicate. This is usually achieved by hard-coding the IP address of the handler machines in the attack code that is later installed at the agent machine. Each agent then reports its readiness to the handlers, who store its IP address for later communication. The obvious drawback of this approach is that discovery of one compromised machine can expose the whole DDoS network. Also, since agents and handlers listen to network connections, they are identifiable by network scanners.

DA-2:CM-2: Indirect Communication

Attacks with indirect communication deploy a level of indirection to increase the survivability of a DDoS network. Recent attacks provide the example of using IRC channels [CER01] for agent/handler communication. Further, the attack code can be changed over time. For instance, the W32/leaves worm [Nat01] used for automatic propagation can receive and interpret commands through an IRC service which enables dynamic updates of the attack code. The use of IRC services replaces the function of a handler, since the IRC channel offers sufficient anonymity to the attacker. Since DDoS agents establish outbound connections to a standard service port used by a legitimate network service,

agent communications to the control point may not be easily differentiated from legitimate network traffic. The agents do not incorporate a listening port that is easily detected by network scanners. An attacker controls the agents using IRC communications channels. Thus, discovery of a single agent may lead no further than the identification of one or more IRC servers and channel names used by the DDoS network. From there, identification of the DDoS network depends on the ability to track agents currently connected to the IRC server. To avoid discovery, attackers frequently deploy channel-hopping, using any given IRC channel for short periods of time. The IRC service is maintained in a distributed manner, and the IRC server hosting a particular IRC channel may be located on a home computer or in a different country. This makes it hard to prevent inappropriate use of IRC functionality. Although the IRC service is the only current example of indirect communication, there is nothing to prevent attackers from subverting other legitimate services for similar purposes.

DA-3: Automatic

Automatic DDoS attacks automate the use phase in addition to the recruit, exploit and infect phases, and thus avoid the need for communication between attacker and agent machines. The start time of the attack, attack type, duration and victim are preprogrammed in the attack code. Deployment mechanisms of this attack class offer minimal exposure to the attacker, since he is only involved in issuing a single command – the start of the attack script. The hardcoded attack specification suggests a single-purpose use of the DDoS network, or the inflexible nature of the system. However, the propagation mechanisms usually leave a backdoor to the compromised machine open, enabling easy future access and modification of the attack code. Further, the code that controls all phases can

be arbitrarily complex and adaptive, checking for updates at prearranged places. The drawback of automated attacks is that all flexibility must be designed in advance and built into the code.

DA-2 and DA-3:SS: Scanning Strategy

Both semi-automatic and automatic attacks recruit the agent machines by deploying automatic scanning and propagation techniques, usually through use of worms. The goal of the scanning strategy is to locate as many vulnerable machines as possible while creating a low traffic volume to escape detection. Based on the scanning strategy, we differentiate between attacks that deploy *random scanning*, *hitlist scanning*, *signpost scanning*, *permutation scanning* and *local subnet scanning*. We give a brief description of these scanning techniques here and refer the reader to [Wea] for a detailed description and performance comparison. Attackers usually combine the scanning and exploit phases, thus gaining a larger agent population, and our description of scanning techniques relates to this model.

DA-2 and DA-3:SS-1: Random Scanning

During random scanning, each compromised host probes random addresses in the IP address space, using a different seed. Code Red (CRv2) performed random scanning [Moo]. Random scanning potentially creates a high traffic volume since many machines are likely to probe the same addresses. The probability for collision increases as a larger portion of total address space gets infected. The high traffic volume can lead to attack detection.

DA-2 and DA-3:SS-2: Hitlist Scanning

A machine performing hitlist scanning probes all addresses from an externally supplied list. When it detects a vulnerable machine, it sends a portion of the initial hitlist to the recipient and keeps the rest. Hitlist scanning allows for great propagation speed and no collisions during the scanning phase. The disadvantage is that the hitlist needs to be assembled in advance. The information contained in the list is not likely to be gathered through scanning (since it would duplicate the effort) but rather collected over time through some less conspicuous techniques. For instance, the hitlist could be assembled using information published at netscan.org related to domains that still support directed IP broadcast and can thus be used for a Smurf attack [CERj]. The hitlist also needs to be transmitted to machines that are being infected. If the list is too large, this traffic might be of high volume and lead to attack detection; if it is too small, it will generate a small agent population.

DA-2 and DA-3:SS-3: Signpost Scanning

Signpost scanning (also called topological scanning in [Wea]) uses information on the compromised host to select new targets. E-mail worms use signpost scanning, exploiting the information from address books of compromised machines for their spread. A Web-server-based worm could spread by infecting each vulnerable Web browser of clients that click on the server's Web page, and then further infect servers of subsequent Web pages visited by these clients. Signpost scanning does not generate a high traffic load and thus reduces chances of attack detection. The drawback is that the spreading speed depends on agent machines and their user behavior, i.e. it is not controllable by the attacker. The agent mobilization may be slower and less complete than with other scanning techniques.

DA-2 and DA-3:SS-4: Permutation Scanning

During permutation scanning, all compromised machines share a common pseudo-random permutation of the IP address space; each IP address is mapped to an index in this permutation. Permutation scanning is preceded by small hitlist scanning during which an initial population of agents is formed. A machine infected during this initial phase begins scanning through the permutation by using the index computed from its IP address as a starting point. Whenever it sees an already-infected machine, it chooses a new random start point. A machine infected by permutation scanning always starts from a random point in the permutation. Permutation scanning has the effect of providing a semi-coordinated, comprehensive scan while maintaining the benefits of random probing. This technique is described in [Wea] as not yet deployed. The analysis provided there shows that the spreading speed could be on the order of several minutes, while a small number of collisions should not lead to attack detection.

DA-2 and DA-3:SS-5: Local Subnet Scanning

Local subnet scanning can be added to any of the previously described techniques to preferentially scan for targets that reside on the same subnet as the compromised host. Using this technique, a single copy of the scanning program can compromise many vulnerable machines behind a firewall. Code Red II [CERc] and Nimda Worm [CERh] used local subnet scanning.

DA-2 and DA-3:PM: Propagation Mechanism

After the recruit and exploit phases, the agent machine is infected with the attack code. Based on the attack code propagation mechanism during the infect phase,

we differentiate between attacks that deploy *central source propagation*, *back-chaining propagation* and *autonomous propagation*, building on the propagation models described in [CER01].

DA-2 and DA-3:PM-1: Central Source Propagation

During central source propagation, the attack code resides on a central server or set of servers. After compromise of the agent machine, the code is downloaded from the central source through a file transfer mechanism. The 1i0n [CERf] worm operated in this manner. Central source propagation imposes a large burden on a central server, generating high traffic and possibly leading to attack discovery. The central server is also a single point of failure; its removal prohibits further agent mobilization.

DA-2 and DA-3:PM-2: Back-Chaining Propagation

During back-chaining propagation, the attack code is downloaded from the machine that was used to exploit the system. The infected machine then becomes the source for the next propagation step. The Ramen worm [CERi] and Morris Worm [HM91] used back-chaining propagation. Back-chaining propagation is more survivable than central-source propagation since it avoids a single point of failure (central server).

DA-2 and DA-3:PM-3: Autonomous Propagation

Autonomous propagation avoids the file retrieval step by injecting attack instructions directly into the target host during the exploit phase. Code Red [CERa], Warhol Worm [Wea] and numerous E-mail worms use autonomous propagation.

Autonomous propagation reduces the frequency of network traffic needed for agent mobilization, and thus further reduces chances of attack discovery.

Note that one could easily imagine an attack that would not fall into any of the proposed manual, semi-automatic and automatic classes. For instance, just the recruit and use phases of the attack could be automated, and the exploit and infect phases could be performed manually. Generating classes to accommodate all combinations of automated and non-automated phases would introduce unnecessary complexity since most of these attacks are not likely to occur. We therefore limited our attention to known and probable combinations.

EV: Exploited Vulnerability to Deny Service

Distributed denial-of-service attacks exploit different strategies to deny the service of the victim to its clients. Based on the vulnerability that is exploited to deny service, we differentiate between *semantic* and *brute-force* attacks.

EV-1: Semantic

Semantic attacks exploit a specific feature or implementation bug of some protocol or application installed at the victim in order to consume excess amounts of its resources. For example, in the TCP SYN attack, the exploited feature is the allocation of substantial space in a connection queue immediately upon receipt of a TCP SYN request. The attacker initiates multiple connections that are never completed, thus filling up the connection queue. In the CGI request attack, the attacker consumes the CPU time of the victim by issuing multiple CGI requests. In the authentication server attack, the attacker exploits the fact that the signature verification process consumes significantly more resources than bogus signature generation. He sends numerous bogus authentication requests

to the server, tying up its resources. The NAPTHA [SAN00] attack is an especially powerful attack on the TCP protocol. It initiates and establishes numerous TCP connections that consume the connection queue at the victim. NAPTHA bypasses the TCP protocol stack on the agent machine, not keeping the state for connections it originates. Instead it participates in the connection, inferring its attributes from received packets. Thus a single agent machine can easily deplete the resources of any victim.

EV-2: Brute-Force

Brute-force attacks (or, as they are frequently called, *flooding attacks*) are performed by initiating a vast amount of seemingly legitimate transactions. Since an upstream network can usually deliver higher traffic volume than the victim network can handle, a high number of attack packets can exhaust the victim's resources.

There is a thin line between semantic and brute force attacks. Semantic attacks also overwhelm a victim's resources with excess traffic, and badly designed protocol features at remote hosts are frequently used to perform "reflector" brute-force attacks, such as the DNS request attack [CERe] or the Smurf attack [CERj]. The difference is that the victim can usually substantially mitigate the effect of semantic attacks by modifying the misused protocols or deploying proxies. However, it is helpless against brute-force attacks due to their misuse of legitimate services (filtering of the attack packets would also mean filtering legitimate requests for service) or due to its own limited resources (a victim cannot handle an attack that swamps its network bandwidth). Countering semantic attacks by modifying the deployed protocol or application pushes the corresponding attack mechanism into the brute-force category. For example, if the victim deploys TCP

SYN cookies [CERk] to combat TCP SYN attacks, it will still be vulnerable to TCP SYN attacks that generate more requests than its network can accommodate. Classification of the specific attack needs to take into account both the attack mechanisms used, and the victim's configuration and deployed protocols. It should be noted that brute-force attacks need to generate a much higher volume of attack packets than semantic attacks to inflict damage on the victim. So by modifying the deployed protocols, the victim pushes its vulnerability limit higher. It is interesting to note that the variability of attack packet contents is determined by the exploited vulnerability. Packets comprising semantic and some brute force attacks must specify some valid header fields and possibly some valid contents. For example TCP SYN attack packets cannot vary the protocol or SYN flag field, and HTTP flood packets must belong to an established TCP connection and therefore cannot spoof source addresses.

SAV: Source Address Validity

Source address spoofing plays a crucial role in denial-of-service, since malicious packets cannot be traced to the source, and responsibility for actions cannot be assigned. If source address spoofing were eliminated, many denial-of-service attacks could be thwarted through resource management techniques – giving the fair share of host or network resources to each source IP address. Based on the source address validity, we distinguish between *spoofed source address* and *valid source address* attacks.

SAV-1: Spoofed Source Address

This is the prevalent type of attack since it is always to the attacker's advantage to spoof the source address, avoid accountability, and possibly create more noise

for detection.

SAV-1:AR: Address Routability

We further divide spoofed source address attacks based on the address routability into *routable source address* and *non-routable source address* attacks.

SAV-1:AR-1: Routable Source Address

Attacks that spoof routable addresses take over the IP address of another machine. This is sometimes done, not to avoid accountability, but to perform a reflection attack on the machine whose address was hijacked. During a reflection attack many requests for some service are made using the spoofed address of the victim machine, and multiple replies are then sent back to the victim, overwhelming it. One example of a reflection attack is a Smurf attack.

SAV-1:AR-2: Non-Routable Source Address

Attackers can spoof non-routable source addresses, some of which can belong to a reserved set of addresses (such as 192.168.0.0/16) or be part of an assigned but not used address space of some network. Attack packets carrying reserved addresses can be easily detected and discarded, while those packets carrying non-used addresses are significantly harder to detect.

SAV-1:ST: Spoofing Technique

Spoofing technique defines how the attacker chooses the spoofed source address in its attack packets. Based on the spoofing technique, we divide spoofing attacks into *random*, *subnet* and *en route* spoofed source address attacks.

SAV-1:ST-1: Random Spoofed Source Address

Many attacks spoof random source addresses in the attack packets, since this can simply be achieved by generating random 32-bit numbers and stamping packets with them. Recent attempts to prevent spoofing using ingress filtering [FS00] and route-based filtering [LMW02, PL01] force attackers to devise more sophisticated techniques, such as subnet and en route spoofing that can avoid current defense approaches.

SAV-1:ST-2: Subnet Spoofed Source Address

In subnet spoofing, the attacker spoofs a random address from the address space assigned to the agent machine's subnet. Since machines at a subnet share the medium (Ethernet) to reach the exit router (first hop en route to the outside world), spoofing can be detected by this router (e.g., if the attacker does not use ARP spoofing, the router can detect conflicting associations of MAC and IP addresses on the Ethernet). It is impossible to detect it anywhere between the exit router and the victim.

SAV-1:ST-3: En Route Spoofed Source Address

An en route spoofed source address attack would spoof the address of a machine or subnet that is en route from the agent machine to the victim. There are no known instances of attacks that use en route spoofing, but this potential spoofing technique could affect route-based filtering [LMW02, PL01] and is thus discussed here.

SAV-2: Valid Source Address

Attackers benefit from source address spoofing and are likely to deploy it whenever possible. Valid source address attacks frequently originate from agent machines running Windows, since all Windows versions prior to XP do not export user-level functions for packet header modification. Those attacks that target specific applications or protocol features must use valid source addresses if the attack strategy requires several request/reply exchanges between an agent and the victim machine. One example of such an attack is NAPTHA [SAN00]. While spoofing is desirable for the attacker, effective attacks are generally possible without spoofing.

ARD: Attack Rate Dynamics

During the attack, each participating agent machine sends a stream of packets to the victim. Depending on the attack rate dynamics of an agent machine, we differentiate between *constant rate* and *variable rate* attacks.

ARD-1: Constant Rate

The majority of known attacks deploy a constant rate mechanism. After the onset is commanded, agent machines generate attack packets at a steady rate, usually as many as their resources permit. The sudden packet flood disrupts the victim's services quickly. This approach provides the best cost-effectiveness to the attacker since he can deploy a minimal number of agents to inflict the damage. On the other hand, the large, continuous traffic stream can be detected as anomalous and arouse suspicion in the network hosting an agent machine, thus provoking attack discovery.

ARD-2: Variable Rate

Variable rate attacks vary the attack rate of an agent machine to delay or avoid detection and response.

ARD-2:RCM: Rate Change Mechanism

Based on the rate change mechanism, we differentiate between *increasing rate* and *fluctuating rate* attacks.

ARD-2:RCM-1: Increasing Rate

Attacks that have a gradually increasing rate lead to a slow exhaustion of the victim's resources. A victim's services could degrade slowly over a long time period, thus substantially delaying detection of the attack.

ARD-2:RCM-2: Fluctuating Rate

Attacks that have a fluctuating rate adjust the attack rate based on the victim's behavior or preprogrammed timing, occasionally relieving the effect to avoid detection. A pulsing attack provides one example of a fluctuating rate attack. During a pulsing attack, agent hosts periodically abort the attack and resume it at a later time. If this behavior is simultaneous for all agents, the victim experiences periodic service disruptions. If, however, agents are divided into groups that coordinate so that one group is always active, then the victim experiences continuous denial-of-service while the network hosting agent machine may not notice any anomalous traffic.

PC: Possibility of Characterization

Looking at the content and header fields of attack packets, it is sometimes possible to characterize the attack. Based on the possibility of characterization, we differentiate between *characterizable* and *non-characterizable* attacks.

PC-1: Characterizable

Characterizable attacks are those that target specific protocols or applications at the victim and can be identified by a combination of IP header and protocol header values, or maybe even packet contents. Examples include the TCP SYN attack (only packets with SYN bit set in the TCP header can potentially be part of the attack), ICMP ECHO attack, DNS request attack, etc.

PC-1:RAVS: Relation of Attack to Victim Services

Characterizable attacks are further divided, based on the relation of attack to victim services, into *filterable* and *non-filterable* attacks.

PC-1:RAVS-1: Filterable

Filterable attacks are those that use malformed packets or packets for non-critical services of the victim's operation. These can thus be filtered by a firewall. Examples of such attacks are a UDP flood attack or an ICMP ECHO flood attack on a Web server. Since a Web server only needs TCP traffic and some DNS traffic (which can be characterized as permitting only those inbound UDP packets that are DNS replies to previous outbound DNS requests), it can easily block all other inbound UDP traffic and all ICMP traffic, and still operate correctly.

PC-1:RAVS-2: Non-Filterable

Non-filterable attacks use well-formed packets that request legitimate services from the victim. Thus, filtering all packets that match the attack characterization would lead to an immediate denial of the specified service to both attackers and legitimate clients. Examples are HTTP requests flooding a Web server or a DNS request flood targeting a name server. In the case of non-filterable attacks, the contents of an attack packet are indistinguishable from the contents of packets originating from a legitimate client.

PC-2: Non-Characterizable

Non-characterizable attacks attempt to consume network bandwidth using a variety of packets that engage different applications and protocols. Sometimes packets will even be randomly generated using reserved protocol numbers.

Note that classification of attack as characterizable or not depends strongly on the resources that can be dedicated to characterization and the level of characterization. For instance, an attack using a mixture of TCP SYN, TCP ACK, ICMP ECHO, ICMP ECHO REPLY and UDP packets would probably be characterizable, but only after considerable effort and time, and only if one had access to a sophisticated characterization tool. Also, an attack using a mixture of TCP packets with various combinations of TCP header fields can be characterized as a TCP attack, but finer characterization would probably fail. So, when performing classification of attacks into characterizable or non-characterizable, a lot is left to interpretation, and ease of characterization should be taken into account.

PAS: Persistence of Agent Set

Attacks have been known to vary different features: type of traffic and attack packets' headers and contents can be varied during the attack, decoy packets can be interleaved with attack packets, attack rate can be adjusted dynamically, etc. All these techniques hinder attack detection. Recently there were occurrences of attacks that varied the set of agent machines active at any one time, further avoiding detection and hindering traceback. We regard this technique as important since it invalidates assumptions underlying many defense mechanisms — that agents are active throughout the attack and can thus be traced back following the path of the attack traffic. We divide attacks, based on the persistence of the agent set, into attacks with *constant agent set* and attacks with *variable agent set*.

PAS-1: Constant Agent Set

During attacks with a constant agent set, all agent machines act in a similar manner, taking into account resource constraints. They receive the same set of commands and are engaged simultaneously during the attack. Examples are an attack in which all agents start sending attack traffic simultaneously,[1] or they engage in a pulsing attack, but the "on" and "off" periods for pulses match over all agent machines.

[1]The definition of a "simultaneous start" is somewhat relaxed in this context since the attacker's command travels to the agents with a variable delay. Further, because agent machines are under different loads, they do not start sending at the exact same moment.

PAS-2: Variable Agent Set

During attacks with a variable agent set, the attacker divides all available agents into several groups and engages only one group of agents at any one time — like the army general who deploys his battallions at different times and places. A machine could belong to more than one group, and groups could be engaged again after a period of inactivity. One example attack of the variable agent set type is an attack in which several agent groups take turns pulsing, thus flooding the victim with a constant flow of packets.

VT: Victim Type

Attacks need not be perpetrated against a single host machine. Depending on the type of victim, we differentiate between *application*, *protocol*, *operating system*, *host*, *network* and *infrastructure* attacks.

VT-1: Application

Application attacks exploit some feature of a specific application on the victim host, thus disabling legitimate client use of that application and possibly tying up resources of the host machine. If the shared resources of the host machine are not completely consumed, other applications and services should still be accessible to the users. For example, a bogus signature attack on an authentication server ties up resources of the signature verification application, but the target machine will still reply to ICMP ECHO requests, and other applications that do not require authenticated access should still work.[2]

Detection of application attacks is challenging because other applications on

[2]This example assumes that CPU time is shared in a fair manner between all active applications.

the attacked host continue their operations undisturbed, and the attack volume is usually small enough not to appear anomalous. The attack packets are virtually indistinguishable from legitimate packets at the transport level (and frequently at the application level), and the semantics of the targeted application must be heavily used for detection. Since there are typically many applications on a host machine, each application would have to be modelled in the defense system and then its operation monitored to account for possible attacks. Once detection is performed, the host machine has sufficient resources to defend against these small volume attacks, provided that it can separate packets that are legitimate from those that are part of the attack.

VT-2: Protocol

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

VT-3: Operating System

Operating system attacks misuse a vulnerability in a specific version of an operating system installed at the target machine. One example of such an attack would be a flood of incoming packets that causes fast process forking until the process table is overloaded. Although such flooding would misuse an application or a protocol to invoke process forking, the targeted vulnerability (e.g., small process table) is part of the operating system. Operating system attacks completely freeze or slow down the operation of the target machine.

VT-4: Host

Host attacks disable access to the target machine completely by overloading or disabling its communication mechanism. Examples of this attack are a TCP SYN attack [CERk] and attacks that overload the network interface or network link of the target machine. All attack packets carry the destination address of the target host. If protocols running on the host are properly patched, the host attacks likely to be perpetrated against it are reduced to attacks that consume network resources. The high packet volume of such attacks facilitates detection. Since its network resources are consumed, the host cannot defend against these attacks alone, but can usually request help from some upstream machine (e.g., an upstream firewall).

VT-5: Network

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

VT-6: Infrastructure

Infrastructure attacks target some distributed service that is crucial for global Internet operation or operation of a subnetwork. Examples include the recent attacks on domain name servers [Nar02], large core routers, routing protocols, certificate servers, etc. The key feature of these attacks is not the mechanism they

deploy to disable the target (e.g., from the point of view of a single attacked core router, the attack can still be regarded as a host attack), but the simultaneity of the attack on multiple instances of a critical service in the Internet infrastructure. Infrastructure attacks can only be countered through the coordinated action of multiple Internet participants.

IV: Impact on the Victim

Depending on the impact of a DDoS attack on the victim, we differentiate between *disruptive* and *degrading* attacks.

IV-1: Disruptive

The goal of disruptive attacks is to deny the victim's service to its clients. All currently known attacks belong to this category.

IV-1:PDR: Possibility of Dynamic Recovery

Depending on the possibility of dynamic recovery during or after the attack, we differentiate between *recoverable* and *non-recoverable* attacks.

IV-1:PDR-1: Recoverable

In the case of recoverable attacks, the victim recovers as soon as the influx of attack packets is stopped, or shortly afterwards. For example, if the attack is a UDP flooding attack, tying up the victim's network resources, the victim will be able to use these resources as soon as the attack is stopped. In the case of a TCP SYN flooding attack, the connection table space is freed shortly after the attack has stopped, when the half-open connection records expire.

IV-1:PDR-2: Non-Recoverable

A victim of a non-recoverable attack cannot automatically recover after the attack is stopped, but requires some human intervention (e.g., rebooting the victim machine or reconfiguring it). For example, an attack that causes the victim machine to crash, freeze or reboot would be classified as a non-recoverable attack.

IV-2: Degrading

The goal of degrading attacks is to consume some (presumably constant) portion of a victim's resources. Since these attacks do not lead to total service disruption, they could remain undetected for a significant time period. On the other hand, damage inflicted on the victim's business could be immense. For example, an attack that effectively ties up 30% of the victim's resources would lead to a denial-of-service to some percentage of customers during high load periods, and possibly slower average service. Some customers, dissatisfied with the quality, would consequently change their service provider, and the attack victim would lose income. Alternately, the false load could result in the victim spending money to upgrade its servers and networks. The addition of new resources would easily be countered by the attacker through more powerful attacks. Almost all existing proposals to counter DDoS attacks would fail to address degrading attacks.

4.2 Taxonomy of DDoS Defenses

Some DDoS defense mechanisms address a specific kind of DDoS attack — such as attacks on Web servers or authentication servers. Other approaches attempt to be effective against a wider range of attacks. Most of the proposed approaches require certain features to achieve peak performance, and will perform quite dif-

ferently if deployed in an environment where these requirements are not met. As is frequently pointed out, there is no "silver bullet" against DDoS attacks. Therefore we need to understand not only each existing DDoS defense approach, but also how those approaches might be combined together to effectively and completely solve the problem. The proposed taxonomy, shown in Figure 4.2 should help us reach this goal. The remainder of this section will discuss each of the proposed classes of defense mechanisms.

AL: Activity Level

Based on the activity level of DDoS defense mechanisms, we differentiate between *preventive* and *reactive* mechanisms.

AL-1: Preventive

Preventive mechanisms attempt either to eliminate the possibility of DDoS attacks altogether or to enable potential victims to endure the attack without denying services to legitimate clients.

AL-1:PG: Prevention Goal

According to the prevention goal, we further divide preventive mechanisms into *attack prevention* and *denial-of-service prevention* mechanisms.

AL-1:PG-1: Attack Prevention

Attack prevention mechanisms modify systems and protocols on the Internet to eliminate the possibility of a DDoS attack. The history of computer security suggests that a prevention approach can never be 100% effective, since global

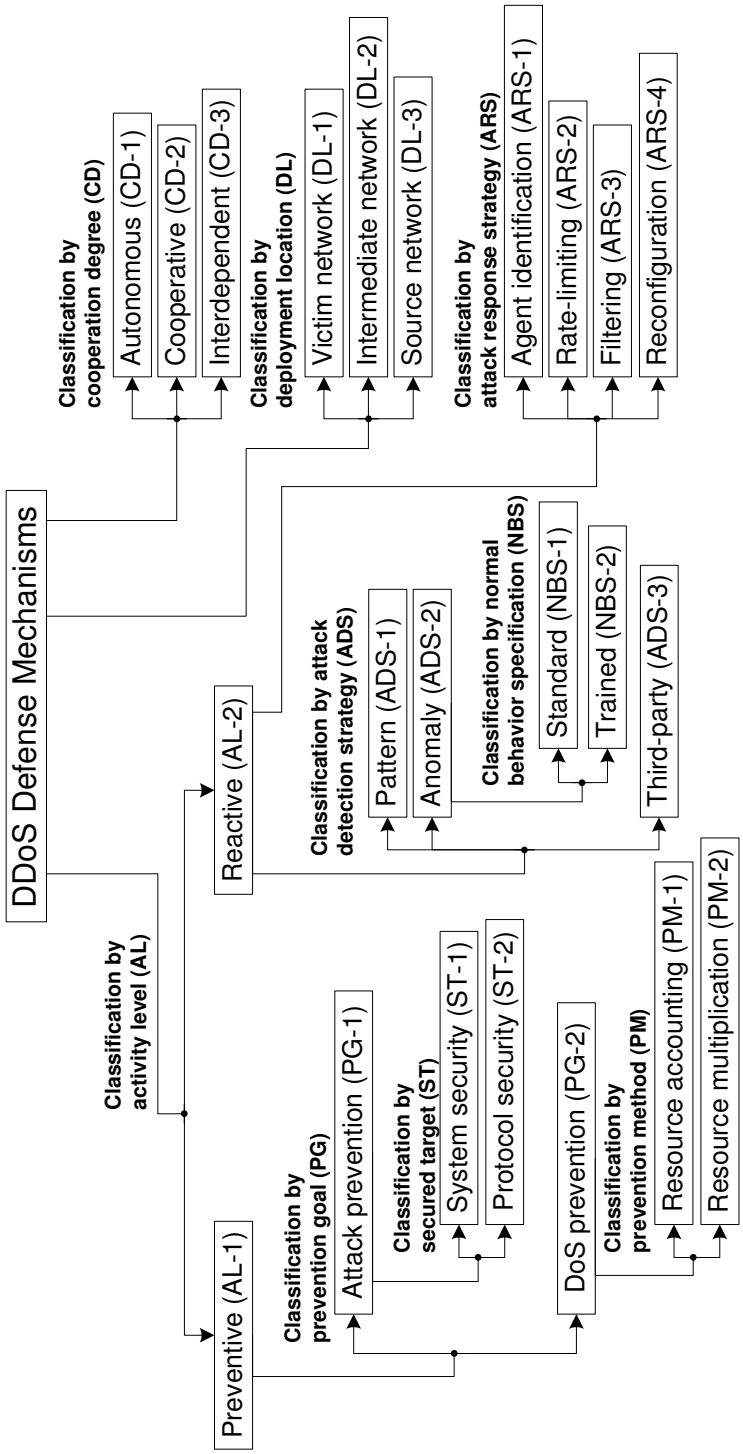

Figure 4.2: Taxonomy of DDoS defense mechanisms

deployment cannot be guaranteed. However, doing a good job here will certainly decrease the frequency and strength of DDoS attacks. Deploying comprehensive prevention mechanisms can make a host completely resilient to protocol attacks. Also, these approaches are inherently compatible with and complementary to all other defense approaches.

AL-1:PG-1:ST: Secured Target

Based on the secured target, we further divide attack prevention mechanisms into *system security* and *protocol security* mechanisms.

AL-1:PG-1:ST-1: System Security

System security mechanisms increase the overall security of Internet hosts and routers, guarding against illegitimate accesses to a machine, removing application bugs and updating protocol installations to prevent intrusions and misuse of the system. DDoS attacks owe their power to large numbers of subverted machines that cooperatively generate attack streams. If these machines were secured, the attackers would lose their army, and the DDoS threat would then disappear. On the other hand, systems vulnerable to intrusions can themselves become victims of denial-of-service attacks in which the attacker, having gained unlimited access to the machine, deletes or alters its contents. Potential victims of DDoS attacks can be easily overwhelmed if they deploy vulnerable protocols. Examples of system security mechanisms include monitored access to the machine [Tri], applications that download and install security patches, firewall systems [McAa], virus scanners [McAb], intrusion detection systems [Axe00], access lists for critical resources [Cisb], capability-based systems [SH02] and client-legitimacy-based systems [OB].

AL-1:PG-1:ST-2: Protocol Security

Protocol security mechanisms address the problem of a bad protocol design. For example, many protocols contain operations that are cheap for the client but expensive for the server. Such protocols can be misused to exhaust the resources of a server by initiating large numbers of simultaneous transactions. Classic misuse examples are the TCP SYN attack, the authentication server attack, and the fragmented packet attack (in which the attacker bombards the victim with malformed packet fragments, forcing it to waste its resources on reassemble attempts). IP source address spoofing is another important example. Examples of protocol security mechanisms include guidelines for a safe protocol design in which resources are committed to the client only after sufficient authentication is done [LNA00, Mea99] or the client has paid a sufficient price [ANL01], deployment of a powerful proxy server that completes TCP connections [SKK97], protocol scrubbing that removes ambiguities from protocols that can be misused for attacks [MWJ00], or the approaches that eliminate spoofing [PL01, LMW02, FS00], etc.

AL-1:PG-2: DoS Prevention

Denial-of-service prevention mechanisms enable the victim to endure attack attempts without denying service to legitimate clients. This is done either by enforcing policies for resource consumption or by ensuring that abundant resources exist so that legitimate clients will not be affected by the attack.

AL-1:PG-2:PM: Prevention Method

Based on the prevention method, we divide DoS prevention mechanisms into *resource accounting* and *resource multiplication* mechanisms.

AL-1:PG-2:PM-1: Resource Accounting

Resource accounting mechanisms police the access of each user to resources based on the privileges of the user and his behavior. The user in this case might be a process, a person, an IP address, or a set of IP addresses having something in common. Resource accounting mechanisms guarantee fair service to legitimate well-behaved users. In order to avoid user identity theft, they are usually coupled with legitimacy-based access mechanisms that verify the user's identity. Approaches proposed in [JB99, ZL97, SP99, GR02, LRS00] illustrate resource accounting mechanisms.

AL-1:PG-2:PM-2: Resource Multiplication

Resource multiplication mechanisms provide an abundance of resources to counter DDoS threats. The straightforward example is a system that deploys a pool of servers with a load balancer and installs high bandwidth links between itself and upstream routers. This approach essentially raises the bar on how many machines must participate in an attack to be effective. While not providing perfect protection, for those who can afford the costs this approach has often proved sufficient. For example, Microsoft has used it to weather large DDoS attacks. Another approach is the use of Akamai services for distributed Web site hosting. User requests for a Web page hosted in such a manner are redirected to an Akamai name server, which then distributes the load among multiple, geographically distributed Web servers hosting replicas of the requested page.

AL-2: Reactive

Reactive mechanisms strive to alleviate the impact of an attack on the victim. To attain this goal they need to detect the attack and respond to it. The goal of attack detection is to detect every attempted DDoS attack as early as possible and to have a low degree of false positives. Upon attack detection, steps can be taken to characterize the packets belonging to the attack stream and provide this characterization to the response mechanism.

AL-2:ADS: Attack Detection Strategy

We classify reactive mechanisms based on the attack detection strategy into mechanisms that deploy *pattern detection*, *anomaly detection*, and *third-party detection*.

AL-2:ADS-1: Pattern Detection

Mechanisms that deploy pattern detection store the signatures of known attacks in a database. Each communication is monitored and compared with database entries to discover occurrences of DDoS attacks. Occasionally the database is updated with new attack signatures. The obvious drawback of this detection mechanism is that it can only detect known attacks, and it is usually helpless against new attacks or even slight variations of old attacks that cannot be matched to the stored signature. Also, the attacks generating legitimate-like packets with sufficient randomness of packet fields do not trigger this type of detection. On the other hand, known attacks are easily and reliably detected, and no false positives are encountered. Snort [Sou] provides one example of a DDoS defense system that uses pattern attack detection. A similar approach has been helpful in controlling

computer viruses. As in virus detection programs, signature databases must be regularly updated to account for new attacks.

AL-2:ADS-2: Anomaly Detection

Mechanisms that deploy anomaly detection have a model of normal system behavior, such as normal traffic dynamics or expected system performance. The current state of the system is periodically compared with the models to detect anomalies. Approaches presented in [YEA00, MBF02, Infb, GP01, Cs3, Maz, Arb, BBNa, BBNb, MPR02] provide examples of mechanisms that use anomaly detection. The advantage of anomaly detection over pattern detection is that previously unknown attacks can be discovered. The caveat is that anomaly detectors must trade off their ability to detect all attacks against their tendency to misidentify normal behavior as an attack.

AL-2:ADS-2:NBS: Normal Behavior Specification

Based on a normal behavior specification, we divide anomaly detection mechanisms into *standard* and *trained* mechanisms.

AL-2:ADS-2:NBS-1: Standard

Mechanisms that use standard specifications of normal behavior rely on some protocol standard or a set of rules to specify this behavior. For example, the TCP protocol specification describes a three-way handshake that has to be performed for TCP connection setup. An attack detection mechanism can make use of this specification to detect half-open TCP connections and delete them from the queue, or it can use TCP SYN cookies to defend against TCP SYN attacks. The

advantage of a standard-based specification is that it generates no false positives; all legitimate traffic must comply to the specified behavior. The disadvantage is that attackers can still perform sophisticated attacks which, on the surface, seem compliant to the standard and thus pass undetected.

AL-2:ADS-2:NBS-2: Trained

Mechanisms that use trained specifications of normal behavior monitor network traffic and system behavior and generate threshold values for different traffic parameters. All traffic exceeding these values is regarded as anomalous. This approach catches a broad range of attacks, but it has following disadvantages:

1. **Threshold setting.** Anomalies are detected when the current system state differs from the model by a certain threshold. The setting of a low threshold leads to many false positives, while a high threshold reduces the sensitivity of the detection mechanism.

2. **Model update.** Systems and communication patterns evolve with time, and models need to be updated to reflect this change. Trained specification systems usually perform automatic model update using statistics gathered at a time when no attack was detected. This approach makes the detection mechanism vulnerable to slowly increasing rate attacks that can, over a long period of time, mistrain models and delay or even avoid attack detection.

AL-2:ADS-3: Third-Party Detection

Mechanisms that deploy third-party detection do not handle the detection process themselves, but rely on an external message that signals the occurrence of the attack and provides attack characterization. Examples of mechanisms

that use third-party detection are easily found among traceback mechanisms [BLT01, SWK00, DFS01, SP01, SPS01].

AL-2:ARS: Attack Response Strategy

The goal of the attack response is to relieve the impact of the attack on the victim while imposing minimal collateral damage to legitimate clients. We classify reactive mechanisms, based on the response strategy, into mechanisms that deploy *agent identification*, *rate-limiting*, *filtering* and *reconfiguration*.[3]

AL-2:ARS-1: Agent Identification

Agent identification mechanisms provide the victim with information about the identity of the machines that are performing the attack. This information can then be combined with other approaches to alleviate the impact of the attack. Agent identification examples include numerous traceback techniques [BLT01, SWK00, DFS01, SP01, SPS01]. One frequently mentioned motivation for deployment of defense mechanisms by intermediate and source networks is possible enforcement of liability for attack traffic. A successful mechanism for reliable agent identification would be necessary for liability enforcement.

AL-2:ARS-2: Rate-Limiting

Rate-limiting mechanisms impose a rate limit on a stream that has been characterized as malicious by the detection mechanism. Examples of rate-limiting mechanisms are found in [MBF02, GP01, Cs3, MPR02, Arb]. Rate-limiting is a

[3]One possible defense approach not discussed here is *retribution*. The defense system may attempt to locate agent machines and launch counterattacks against them. However, this action is illegal and is therefore out of this dissertation's scope.

lenient response technique that is usually deployed when the detection mechanism has a high level of false positives or cannot precisely characterize the attack stream. The disadvantage is that such an approach will allow some attack traffic through, so extremely high-scale attacks might still be effective even if all traffic streams are rate-limited.

AL-2:ARS-3: Filtering

Filtering mechanisms use the characterization provided by detection mechanisms to filter out the attack streams completely. Examples include dynamically deployed firewalls [DO], and also a commercial system, TrafficMaster [Maz]. Unless the detection strategy is very reliable, filtering mechanisms run the risk of accidentally denying service to legitimate traffic. Worse, clever attackers might leverage them as denial-of-service tools.

AL-2:ARS-4: Reconfiguration

Reconfiguration mechanisms change the topology of the victim or the intermediate network to either add more resources to the victim or to isolate the attack machines. Examples include reconfigurable overlay networks [ABK01, Infb], resource replication services [YEA00], attack isolation strategies [Ast, BBNa], etc.

CD: Cooperation Degree

DDoS defense mechanisms can perform defensive measures either alone or in cooperation with other entities in the Internet. Based on the cooperation degree, we differentiate between *autonomous*, *cooperative* and *interdependent* mechanisms.

CD-1: Autonomous

Autonomous mechanisms perform independent defense at the point where they are deployed (a host or a network). Firewalls and intrusion detection systems provide easy examples of autonomous mechanisms. Even if a defense system performs its function in a distributed manner, it would still be considered autonomous if it can be completely deployed within the network it protects (like a network intrusion detection system).

CD-2: Cooperative

Cooperative mechanisms are capable of autonomous detection and response, but can achieve significantly better performance through cooperation with other entities. The aggregate congestion control (ACC) system [MBF02] deploying a pushback mechanism [IB02] provides an example. ACC detects the occurrence of a DDoS attack by observing congestion in a router's buffer, characterizes the traffic that creates the congestion, and acts locally to impose a rate limit on that traffic. However, it achieves significantly better performance if the rate-limit requests can be propagated to upstream routers that otherwise may be unaware of the attack.

CD-3: Interdependent

Interdependent mechanisms cannot operate autonomously at the deployment point; they rely on other entities either for attack prevention, attack detection or for efficient response. Traceback mechanisms [BLT01, SWK00, DFS01, SP01, SPS01] provide examples of interdependent mechanisms. A traceback mechanism deployed at a victim site would provide no benefit. Secure overlay services

[KMR02] are another example of an interdependent mechanism. They provide successful protection to the victim, rerouting legitimate traffic through the Internet, but only if victim's clients are aware of and cooperate with the mechanism.

DL: Deployment Location

With regard to deployment location, we differentiate between mechanisms deployed at the *victim*, *intermediate*, or *source network*.

DL-1: Victim Network

DDoS defense mechanisms deployed at the victim network protect this network from DDoS attacks and respond to detected attacks by alleviating the impact on the victim. Historically, most defense systems were located at the victim since it suffered the greatest impact of the attack and was therefore the most motivated to dedicate some resources to security mechanisms. Resource accounting [JB99, ZL97, SP99, GR02, LRS00] and protocol security mechanisms [LNA00, Mea99, ANL01, SKK97] provide examples of these systems.

DL-2: Intermediate Network

DDoS defense mechanisms deployed at the intermediate network provide infrastructural protection service to a large number of Internet hosts. Victims of DDoS attacks can contact the infrastructure and request the service, possibly providing adequate compensation. Pushback [MBF02] and traceback [BLT01, SWK00, DFS01, SP01, SPS01] techniques are examples of intermediate network mechanisms. Such mechanisms are not yet widely deployed, and many of them can only be effective in wide deployment.

DL-3: Source Network

The goal of DDoS defense mechanisms deployed at the source network is to prevent network customers from generating DDoS attacks. Such mechanisms are necessary and desirable, but motivation for their deployment is low since it is unclear who would pay the expenses associated with this service. Mechanisms proposed in [GP01, Cs3, MPR02] provide examples of source network mechanisms.

4.3 Using the Taxonomies

In designing the above taxonomies, we selected those features of attack and defense mechanisms that, in our opinion, offer critical information regarding seriousness and type of threats, and effectiveness and cost of defenses. Some attack features, such as damage inflicted, duration, number of agents involved, etc., were not included as criteria. Although these are critical when investigating or understanding the incident, there is currently no publicly available information base that would allow us to design meaningful classifications. A standardized incident-reporting system would greatly improve that. Some defense mechanism characteristics, such as timeliness of response, level of false positives, collateral damage, etc., were also not included as criteria. We believe that these are important, but they must be strictly measured in a controlled and realistic environment using a widely accepted benchmark suite. Without meeting these requirements, we felt that any classification on these criteria that we could design would be uninformed and likely unjust to some mechanisms.

In attack taxonomy design, the selected criteria cover various preparatory phases that preceed the attack (*degree of automation, scanning and propagation*

strategy, communication mechanism), the organization of agent machines (*persistence of the agent set*), the way the attack is perpetrated (*exploited vulnerability*), the attack packet contents (*source address validity, address routability, spoofing technique, possibility of characterization, relation of attack to victim services*), behavior of the individual agent streams (*attack rate dynamics, rate change mechanism*), and the victim (*victim type, impact on the victim, possibility of dynamic recovery*). In defense taxonomy design, the selected criteria covers the defense goal (*activity level*), how it is achieved (*prevention goal, secured target, prevention method, attack detection strategy, normal behavior specification, attack response strategy*), where the system should be deployed (*deployment location*), and what the requirements are for deployment scope (*cooperation degree*).

We provide here an example case analysis of how these taxonomies could be used. Let us assume that we want to protect our medium-size Web server from attacks that deplete server resources only, and do this in a manner that guarantees continued good service to legitimate clients. Based on an analysis of attacks suffered so far, we are convinced that we will not be the subject of attacks that deplete network resources and we decide not to protect against those. First, using the attack taxonomy, we conclude that the attacks from which we want protection can be both semantic (EV-1, e.g., malformed packets or misuse of faulty server protocols) and brute force (EV-2, e.g., too many legitimate-like requests). These attacks are likely to be characterizable (PC-1) but non-filterable (RAVS-2, since we host a Web server and are likely to receive many legitimate requests that obscure the attack). They are application (VT-1, Web server) and host (VT-2, machine hosting the server) attacks, and they are likely to be disruptive recoverable (IV-1, PDR-1) and degrading (IV-2) attacks. We have no information about degree of automation (DA), source address validity (SAV), attack rate dynamics (ARD) or persistence of agent set (PAS), so we assume that

the attack can belong to any of corresponding classes.

Next, using the defense taxonomy, we would like to choose effective protection measures. To prevent semantic attacks we need to apply attack prevention measures (PG-1) which include system and protocol security measures (ST-1 and ST-2). Semantic attacks are likely to target Web server software, the TCP implementation and the HTTP/CGI protocol. As defense measures, we need to update our software regularly and deploy TCP SYN cookies [CERk]. Additionally, we will close all unused ports to prevent intrusions and install a firewall that protects from semantic attacks that use malformed packets. To defend against brute force attacks that consume more resources than a Web server has (once all its protocols have been updated and protected), we can either use DoS prevention measures (PG-2) to help us sustain the attack (PM-1 and PM-2), or deploy reactive defense systems (AL-2) that detect the attack and recognize and preferentially serve legitimate requests. Since the attack is recoverable, a reactive defense should lead to continued good service to legitimate clients. However, since the attack is likely to be non-filterable, differentiating the legitimate from the attack packets may be impossible. Our best option is to resort to DoS prevention measures: deploy resource accounting (PM-1) and purchase resource multiplication services from another organization (PM-2).

4.4 Summary

Distributed denial-of-service attacks are a complex and serious problem, and consequently numerous approaches have been proposed to counter them. However, the multitude of current attack and defense mechanisms obscures a global view of the DDoS problem. These taxonomies are a first attempt to cut through the obscurity and achieve a clear view of the problem and the existing solutions.

They are intended to help the community think about the threats we face and the measures we can use to counter those threats.

One benefit we foresee from the development of DDoS taxonomies is that of fostering easier cooperation among researchers developing DDoS defense mechanisms. Attackers cooperate to exchange attack code and information about vulnerable machines, and to organize their agents into coordinated networks to achieve immense power and survivability. The Internet community must be equally cooperative within itself to counter the DDoS threat. Good taxonomies for DDoS attack and defense mechanisms will facilitate communications and offer the community a common language for discussing solutions. They will also clarify how different mechanisms are likely to work in concert, and identify areas of remaining weaknesses that require additional mechanisms. Similarly, the research community needs to develop common metrics and benchmarks to evaluate the efficacy of DDoS defense mechanisms, and these taxonomies can be helpful in shaping those tasks, as well.

We do not claim that these taxonomies are complete and all-encompassing. We must not be deceived by the simplicity of the current attacks. For the attackers, this simplicity arises more from convenience than necessity. As defense mechanisms are deployed to counter simple attacks, we are likely to see more complex attack scenarios. Many more attack possibilities exist and must be addressed before we can completely handle the DDoS threat; some of these are likely to be outside the current boundaries of the taxonomies presented here. Thus, these taxonomies are likely to require expansion and refinement as new threats and defense mechanisms are discovered. The DDoS attack taxonomy and DDoS defense taxonomy outlined in this chapter are useful to the extent that they clarify our thinking and guide us to more effective solutions to the problem

of distributed denial-of-service. The ultimate value of the work described here will thus lie in the degree of discussion and future research that it provokes.

CHAPTER 5

Source-End Defense

Ideally, distributed denial-of-service attacks should be stopped as close to the sources as possible. Source-end DDoS defense has several advantages over intermediate-network and victim-end defense approaches:

- **Congestion avoidance.** Restraining attack streams near the source preserves Internet resources that are usually overwhelmed by the attack traffic. This reduces overall congestion and increases resources available to legitimate users.

- **Small collateral damage.** Many DDoS defense systems respond to the attack by filtering or rate-limiting all traffic to the victim. Legitimate traffic thus suffers collateral damage. Moving DDoS defense closer to the sources reduces the range of legitimate traffic adversely affected by the response, as the traffic from uncompromised source networks proceeds to the victim unhampered.

- **Easier traceback.** Being close to the source facilitates easier attack traceback and investigation. Administrators can receive attack alerts from source-end defense systems and examine all the machines in the protected source network in order to detect those that are compromised.

- **Sophisticated detection strategies.** Routers closer to the sources are likely to relay less traffic than intermediate routers and can dedicate more of their resources to DDoS defense. This facilitates use of more complex detection and response strategies.

However, source-end defense faces also faces hard challenges with regard to detection and response selectiveness.

Attack detection at the source-end is hindered by attack distribution. In a highly distributed attack, each source network may observe only a small portion of the attack, which is unlikely to generate anomalous statistics. An attack can be conducted through legitimate requests; thus, observation of outgoing packets will not raise suspicion. Alternately, the source-end defense may try to detect the denial-of-service effect on the victim by observing its behavior. Under autonomous operation, the source-end defense is not permitted to contact the victim and inquire about its state. Therefore it can only speculate about a victim's condition based on partial observation.

Response selectiveness is an imperative for source-end defense. Networks hosting source-end defense do not experience a direct benefit from the defense system's operation and are thus poorly motivated to deploy it. In order to provide deployment incentive, a system must not only successfully detect and restrain many attacks but also: (a) provide good service to legitimate traffic between the deploying network and the victim, (b) have a low level of false positives, and (c) have a low deployment cost.

5.1 Source-End Detection

There are several approaches that can be used either separately or jointly for source-end DDoS detection:

- **Source-end firewall.** Firewalls perform attack detection using known attack signatures. This detection has a low level of false positives and can successfully be used to filter out malformed packets that are used for protocol attacks.

- **Threshold anomaly detection.** Each source-end network can define a set of thresholds for various traffic types, describing expected values for a set of parameters, such as average packet rate per connection, average number of outgoing UDP packets per destination, outgoing packet size distribution given time of the day, etc. These threshold values can be obtained through extensive training, guaranteeing low levels of false positives, and can detect a wide range of attacks. However, traffic patterns change and models need to be retrained. If the retraining is automatic (i.e. system adjusts its models to slow-changing traffic trends), attackers can misuse this to avoid detection.

- **Two-way traffic dynamics.** Since outgoing attack streams appear legitimate at the source network, just observation of the outgoing traffic alone cannot provide sufficient information for detecting anomalous behavior and raising the alarm. However, it is generally the case that attack streams are unresponsive to congestion signals—i.e., the outgoing attack stream to the victim will not reduce its rate if notified of congestion through reduction in the number of peer responses. A source-end system can use this observation to detect DDoS attacks misusing inherently responsive transport protocols

such as TCP.

- **Spoofing detection.** Attack packets frequently spoof source IP addresses to avoid detection and accountability. Therefore, the occurrence of aggressive spoofing in the outgoing flow can be a likely sign of a DDoS attack. In order to detect aggressive spoofing, the system can: (a) enforce ingress filtering on all outgoing packets, thus preventing random spoofing, and (b) restrict the number of outgoing connections for a single destination. These steps help detect those attacks that are performed through inherently non-responsive transport protocols, such as UDP, in those cases where aggressive spoofing is employed.

- **Connection semantics.** Sophisticated attacks can be performed while preserving correct two-way traffic dynamics and avoiding aggressive spoofing. It is much more difficult to preserve correct connection semantics, since this requires an attacker to store state per connection, thus putting more burden on the attacking machine. If system can afford to monitor, or even sample, per-connection state, this will enhance its ability to detect subtle attacks that preserve two-way dynamics, such as degrading attacks, slowly increasing rate attacks, small rate attacks and pulsing attacks.

5.2 Source-End Response

The source-end response to a detected DDoS attack must take into account the complexity of source-end detection which results in a low level of confidence in the attack signal. Since detection is unreliable, response must be **liberal** to minimize damage to legitimate traffic inflicted by false detections. Response must also be **selective**, i.e., the system must be able to detect and preserve

legitimate traffic to the alleged victim. The selectiveness of the response plays a crucial role in defining deployment incentive. If the system were not selective in constraining outgoing traffic to the victim, legitimate packets from a deploying network would be regularly dropped whenever the attack was detected. In this scenario, a network would be worse off deploying a source-end DDoS defense, because its legitimate packets, that otherwise might get some responses, would be dropped. This negative feature, along with the fact that the victim, not the source network deploying the defense system, harvests the benefit of DDoS defense, seems to be a strong argument against the source-end defense. On the other hand, a selective response that favors legitimate traffic will provide better service to legitimate clients of the source network during the attack, as their packets will not compete with the attack packets for the limited bandwidth. Instead, the legitimate packets receive preferential treatment and are sent promptly to the victim. Finally, response must be **dynamic**, it must adjust based on the attack detection signal and attack force, providing a stricter response to high-rate attacks, and a more lenient response to low-rate ones, and aborting all response promptly after the attack has subsided.

5.3 Deployment Incentive

A source-end defense system faces a hard deployment incentive challenge. Its cost is sustained by the deploying network, which does not receive substantial benefit from its operation. Furthermore, a victim cannot be asked to compensate for this cost, as being payed to not participate in an attack would create an opportunity for extortion. On the other hand, source-end defenses are necessary to stop Internet misuse, as all other points of defense (intermediate and victim networks) do not prevent attack traffic from congesting shared Internet resources.

Historically, similar problems have been solved by legislation. It is possible that in the future, through contracted or legislative action, those who do not take reasonable steps to secure their system will be held liable for damages inflicted by attackers misusing their machines. In that case, a source-end defense system would become part of an established security practice, and therefore a network deploying the system could not be held liable if its machines were misused for an attack.

Many people have concluded that stopping attacks completely is impossible, since there is a vast number of machines whose owners are unaware of security holes or are unwilling to fix them. A single source-end defense system installed at the network's exit router would prevent DDoS attacks originating from anywhere in the network, in spite of unsecured machines within.

Good performance characteristics, such a low false alarm rate, high effectiveness of attack response, a service guarantee to legitimate traffic, and low deployment cost would further strengthen the motivation for deployment of source-end DDoS defenses.

5.4 Summary

A source-end DDoS defense has many advantages over defenses deployed at other places. It provides effective response, facilitates traceback, and may deploy sophisticated detection and traffic profiling due to low volume of traffic. On the other hand, a source-end defense faces many challenges in order to be widely deployed. Attack detection is difficult and unreliable, and response has to be flexible and selective to compensate for poor detection and to offer deployment incentive. A successful source-end defense system must therefore provide accurate detection

and a selective, dynamic response.

CHAPTER 6

D-WARD

D-WARD (DDoS Network Attack Recognition and Defense) is a DDoS defense system deployed at a source-end network. Its goal is twofold:

1. Detect outgoing DDoS attacks and stop them by controlling outgoing traffic to the victim.

2. Provide good service to legitimate transactions between the deploying network and the victim while the attack is on-going.

D-WARD can operate either as an autonomous system or as a participant in a distributed defense system. In autonomous operation, D-WARD detects attacks and responds to them without communication with any other entity. In distributed cooperative operation, D-WARD enhances its detection by receiving attack alerts from other participants. D-WARD can also accept and grant some response requests from others. More details about cooperative operation are provided in Sections 10.1 and 10.2.

D-WARD polices only the outgoing traffic from its network. Any traffic originating in other domains being relayed by D-WARD is not policed.

6.1 Terminology and Assumptions

The D-WARD system is installed at the *source router* that serves as a gateway between the deploying network (*source network*) and the rest of the Internet. Figure 6.1 depicts D-WARD deployment. In basic deployment, the source router is assumed to be the only connection point between the source network and the rest of the Internet. D-WARD can thus observe every packet exchanged between source network and the outside world. In another scenario, there are many gateway routers on the source network border, but each gateway is a default exit and entry point for a set of foreign addresses. Thus, an instance of the D-WARD system can be deployed at each gateway and observe every packet exchanged between the source network and its assigned set of foreign addresses. Alternately, some gateway routers host D-WARD systems, and partially police the source network's outgoing traffic only to certain destinations. The last scenario, in which multiple gateway routers exist and there is asymmetric routing of incoming and outgoing packets (i.e., it may happen that an outgoing packet to destination X traverses a different gateway than an incoming packet from source X) is unfavorable for D-WARD since it cannot form complete traffic observations. Asymmetric routing is discussed in more detail in Section 6.9.

D-WARD is configured with a set of local addresses whose outgoing traffic it polices — its *police address set*. This set identifies, for example, all machines in the stub network or all customers of an ISP. We assume that D-WARD is able to identify the police address set, either through some protocol or through manual configuration.

D-WARD observes total traffic between its police address set and the rest of the Internet, at *flow* and *connection* granularity. A flow is defined as the aggregate traffic between the police address set and one foreign host (i.e., one

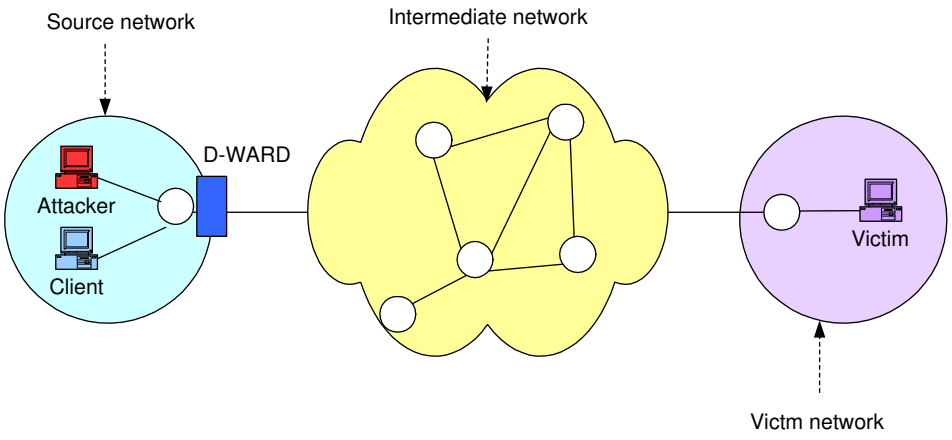

Figure 6.1: D-WARD deployment

foreign IP address). A connection is defined as the aggregate traffic between a pair of IP addresses and port numbers, where one address belongs to the police address set and the other is a foreign address. Figure 6.2 illustrates the notions of a flow and a connection.

D-WARD applies ingress filtering to the outgoing traffic, thus combating random spoofing. However, no assumptions are made with regard to subnet spoofing, and D-WARD is capable of handling attacks that use subnet spoofing.

6.2 Philosophy

Due to the similarity of attack to legitimate traffic, it is unwise to base any defensive action on per-packet observations. D-WARD bases most of its decisions on flow and connection monitoring over time. Rather than establishing the legitimacy of individual packets, D-WARD observes flow and connection behavior and classifies the complete flow or connection as legitimate or attack. It adjusts its operation dynamically to match the classifications. D-WARD further observes

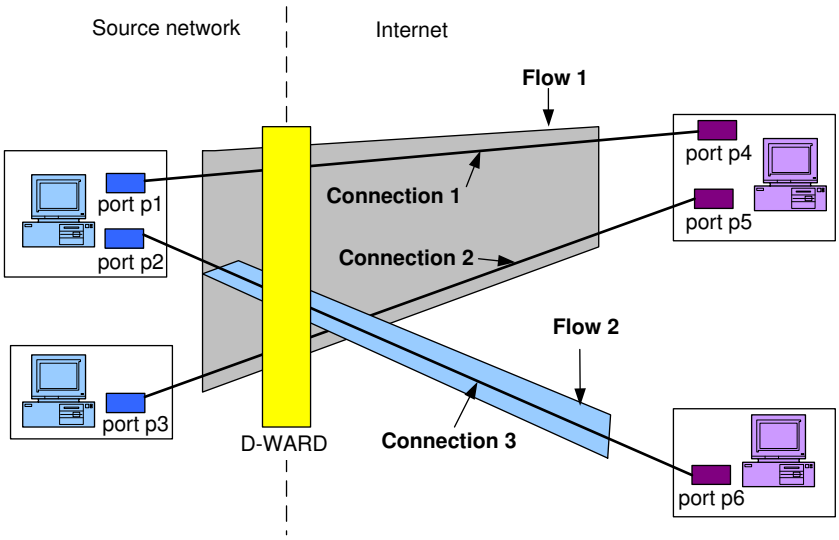

Figure 6.2: Flows and connections

how its actions affect flows and connections and uses this additional information for future classifications.

D-WARD detects outgoing DDoS attacks by monitoring two-way traffic between the source network and the rest of the Internet. The system looks for the following anomalies in traffic dynamics that may be signs of a DDoS attack:

1. **Non-responsive foreign host: Aggressive sending rate coupled with low response rate.** This anomaly pertains only to two-way communications that follow a request/response paradigm such as TCP, some types of ICMP traffic, DNS traffic, NTP traffic, etc. In these communications, one party sends one or several packets to the other party, and waits for a reply (either confirmation of receipt or a response) before sending any more packets. For such communications it is anomalous to observe an aggressive sending rate coupled with a low response rate. A low response rate is perceived by D-WARD as an indication that the foreign host may be overwhelmed by the attack and cannot reply, while an aggressive sending rate

indicates that local hosts are likely participants in the attack. By detecting non-responsive foreign hosts, D-WARD actually aims to detect the occurrence of the denial-of-service effect. This is a very reliable attack signal, as an attacker must create the denial-of-service effect to inflict damage on the victim. On the other hand, coupling detection to denial-of-service may lead to "after-the-fact" detection, once damage has been done. It would be better if detection could be performed in the early stages of the attack, thus preserving more of the victim's resources. We discuss techniques for early attack detection in Section 6.7.

2. **Presence of IP spoofing.** D-WARD deploys ingress filtering and discards, at all times, outgoing packets that do not carry local addresses. In addition, D-WARD monitors the number of simultaneous connections between the source network and each foreign host. Those foreign hosts that are engaged in a suspiciously high number of connections with the victim are deemed to be part of a subnet spoofing attack (where the agent machine spoofs addresses local to its own subnetwork). This form of IP spoofing detection is not quite accurate, since it may happen that a destination becomes so popular that numerous local hosts initiate legitimate connections with it (thus forming a "flash crowd" effect). Therefore, an incidence of a large number of connections per foreign address is perceived as an attack signal only if there are no better anomaly models. This, for instance, is done for UDP-based attack detection but not for TCP and ICMP attacks.

Once the attack has been detected, D-WARD responds by rate-limiting the total outgoing flow from the source network to the victim, and thus relieves the victim of a heavy traffic volume. As a liberal response, rate-limiting is chosen instead of filtering. Since the attack detection may be inaccurate, rate-limiting

allows some packets to reach the victim and possibly correct the future detection. Instead of deploying a fixed rate limit, D-WARD attempts to determine (guess) the maximum sending rate that the foreign host can handle. The problem of regulating the sending rate of a one-way flow to the level manageable by the receiver (or the route to the receiver) has been recognized and addressed by the TCP congestion control mechanism.[1] D-WARD strives to solve a similar problem at a more aggregated scale. It needs to control the total flow to the foreign host, or the portion of that flow that has been characterized as troublesome, and it infers the foreign host's state from the attack detection signal.

D-WARD's rate-limiting strategy applies modified TCP congestion control ideas to this problem. A fast exponential decrease of the sending rate is performed when the attack is detected to quickly relieve the victim of high-volume traffic. Once the attack subsides, D-WARD performs a slow recovery of rate-limited flows, linearly increasing the sending rate. This is done to probe the state of the receiver, and to reevaluate its ability to handle traffic. After a while, if the attack is not repeated, D-WARD performs a fast recovery of rate-limited flows. The sending rate is increased exponentially to provide fast recovery and subsequent removal.

In addition to the attack signal, D-WARD also bases its rate-limit settings on the observed flow behavior. If the flow is compliant and does not continue to send aggressively, its rate limit will be decreased more slowly and increased more rapidly than if it is misbehaving. This policy facilitates fast recovery of misclassified legitimate flows while severely limiting ill-behaved aggressive flows that are likely part of the attack.

While imposing a rate limit on total outgoing flow to the victim, D-WARD

[1] In TCP terminology, a flow has the same meaning as a connection in D-WARD.

attempts to detect those connections within the flow that are likely to be legitimate. Packets belonging to legitimate connections will be forwarded to their destination regardless of the rate limit, thus providing good service to legitimate traffic during the attack. Differentiation of legitimate from attack connections is a very difficult problem in DDoS defense. D-WARD performs this differentiation in two ways:

1. It monitors connections at all times, and uses legitimate connection models to detect legitimate connections. Once detected, those connections are recorded in the *Legitimate Connection List*. When the attack is detected, packets belonging to connections from the *Legitimate Connection List* are deemed legitimate, and are not subject to rate-limiting. As the connection state is reevaluated periodically, malicious connections that wiggle their way onto the *Legitimate Connection List* gain little advantage. They will be promptly detected and removed from the list.

2. During the attack, D-WARD uses a set of models to evaluate legitimacy of packets initiating new connections. Those packets that pass the match are subject to separate rate-limiting. This technique increases the chances of successful connection origination during the attack, while limiting the amount of damage that a stealthy attacker can do (an attacker that can generate packets that pass the legitimacy test). This is the only per-packet classification that D-WARD performs, and it is described in Section 6.4.3.

In order to detect both attack flows and legitimate connections, D-WARD uses a set of **legitimate traffic models.** In the flow classification case, those flows that clearly mismatch the corresponding models are deemed attack flows. In the connection classification case, those connections that clearly match the

model are deemed legitimate. This technique minimizes the possibility of errors for the following reasons:

- It is difficult to model malicious behavior as there is high variability in an attack's features. On the other hand, legitimate traffic models can be constructed by observing protocol specifications and common traffic patterns.

- In the flow classification case, legitimate traffic models are designed to ensure that no legitimate flow is likely to violate them. A mismatch is then a likely sign of an attack. On the other hand, a model match may not guarantee the absence of the attack. To address this, D-WARD uses traffic models that severely constrain the attack variability (e.g., spoofing, dynamics, or sending rate). If the attacker generates malicious traffic to match the model, he may avoid source-end detection. However, constrained malicious traffic will either not be very successful in performing the attack (small rate, infrequent traffic), will be subject to conventional traffic control approaches (such as fair resource sharing, in the case of limited spoofing) or will facilitate traceback (in the case of no spoofing).

- In the connection classification case, legitimate connection models are designed to ensure that no legitimate connection is likely to violate them. However, there may be situations when D-WARD does not have enough information to classify a connection, e.g., when a connection has just been initiated and only one packet has been sent. If this connection is deemed legitimate, then all malicious connections in the spoofed attack case will be deemed legitimate. If it is deemed an attack, then many legitimate connections will be misclassified. To address this, D-WARD introduces the notion of *transient* classification, indicating that there is not enough data

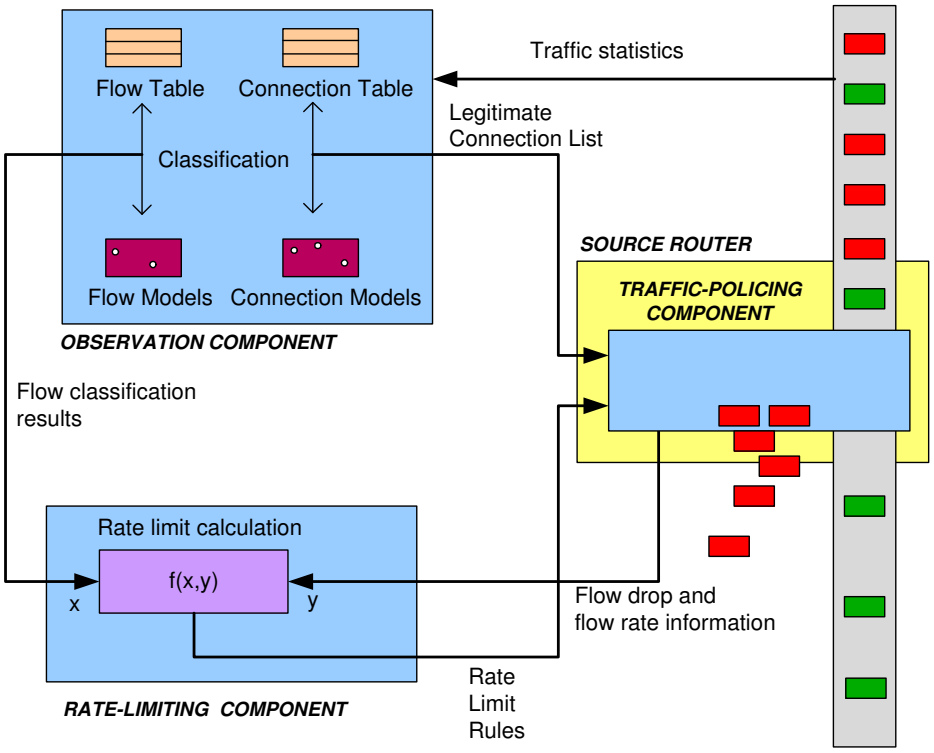

Figure 6.3: D-WARD architecture

to perform classification. Transient connection traffic will be subject to per-packet classification and rate-limiting.

6.3 Architecture

D-WARD is a self-regulating reverse-feedback system. It consists of *observation*,*rate-limiting* and *traffic-policing* components. The traffic-policing component must be part of the source router, while the observation and rate-limiting components can be part of a self-contained unit that interacts with the source router to obtain traffic statistics and install rate-limiting rules. Figure 6.3 depicts the D-WARD architecture.

The observation component monitors all packets passing through the source router and gathers statistics on two-way communications between the police address set and the rest of the Internet, recording the flow and connection granularity. This monitoring can be performed, for example, by sniffing the traffic at the source router interfaces. Periodically, statistics are compared to models of legitimate traffic, and flows and connections are classified. Classification results are passed to the rate-limiting component, which adjusts the rate limit rules. Both the *Legitimate Connection List* and rate limit rules are communicated to the traffic-policing component, which then enforces rate limits and ensures forwarding of legitimate packets. The imposed rate limits modify associated traffic flows and thus affect future observations, closing the feedback loop.

6.4 Observation Component

The observation component gathers statistics and monitors them at flow and connection granularity. Flow statistics are stored in the *Flow Table* while connection statistics are stored in the *Connection Table*. As spoofed attacks may generate a large number of records in these tables, table size is limited to avoid excessive memory consumption. To accommodate all relevant information in limited-size tables, the observation component cleans the tables: (1) periodically expelling all records that are stale, and (2) on overflow, expelling those records that are deemed less useful than others.

The observation component periodically classifies flows and connections. Flow classification is used to detect occurrences of a DDoS attack, while connection classification is used to identify legitimate connections that should receive good service in case the associated flow becomes rate-limited. The connection profiling is performed continuously (as opposed to on-demand profiling once the attack

is detected). The advantage of this approach is that the Legitimate Connection Table is already populated when the attack occurs. D-WARD uses this knowledge to provide continued good service to connections in the table, thus they suffer no damage because of the attack. If D-WARD were using on-demand profiling, a delay would exist between the attack detection signal and the populating of the Legitimate Connection Table. During this time, legitimate connection traffic would be damaged by the attack.

6.4.1 Flow Statistics and Flow Classification

Each outgoing and incoming packet modifies one record in the *Flow Table*. As an outgoing flow may contain a mixture of many transport protocols and applications, the flow record consists of several fields dedicated to specific protocol statistics. D-WARD currently keeps separate statistics on three types of traffic within the flow: TCP, UDP, and ICMP. Which statistics will be kept on protocol traffic depends on the legitimate flow models. Section 7.2 contains more details on statistics content.

Flow classification is performed each *Flow Observation Interval* seconds. During classification, D-WARD compares flow statistics for each protocol field with corresponding legitimate flow models. A flow will be classified as:

1. **Attack** if at least one field's statistics do not match the corresponding model.

2. **Suspicious** if the statistics or all fields match their corresponding models, but the flow has recently been classified as attack.

3. **Normal** if if the statistics or all fields match their corresponding models, and the flow has not been recently classified as attack.

The above list is given in chronological order. When an attack occurs, a change in traffic statistics will first lead to attack classification of the associated flow. The attack classification lasts as long as at least one of two conditions is met:

1. The attack-detected signal is positive.

2. There are packet drops on the flow, due to rate-limiting

The reason for the dual condition is the following: the attack-detected signal is generated based solely on the flow's traffic forwarded by D-WARD to the victim, and thus expresses the measure of "how well can the victim handle this flow's traffic." It may well be the case that D-WARD is severely dropping a flow's packets, thus producing a low outgoing rate which results in a negative attack-detected signal. This indicates that D-WARD is doing a good job of controlling the traffic and need not decrease the rate limit further, but does not mean that the rate limit should be lifted. Therefore, D-WARD will maintain the attack classification for as long as a flow is generating packet drops, even if the attack-detected signal is negative.

Once the attack has subsided, the flow will be first classified as suspicious for *Compliance Period* seconds. Suspicious classification leads to a cautious, slow increase of the rate limit. If the attack repeats before *Compliance Period* has expired, the flow will be classified as an attack again. Otherwise, the flow will be classified as normal. The distinction between suspicious and normal flows is made to minimize damage from recurring attacks. Attacks that repeat with an interval shorter than *Compliance Period* will achieve their full force only in the first attempt. Since their repetitions will happen during suspicious classification, the imposed rate limit will constrain the outgoing flow, preventing further damage

to the victim. Attacks repeating with an interval longer than *Compliance Period* are likely to achieve their full force for a few seconds in every cycle (until they have been detected and stopped). Larger *Compliance Period* values prevent more recurring attacks, but also constrain traffic from transient connections longer.[2]

The following models are used in flow classification.

Legitimate TCP flow model. A large percentage of traffic in the Internet (about 90%) is TCP traffic. The TCP protocol [Ins81] uses a two-way communication paradigm to achieve reliable delivery. During a TCP session, the data flow from the source to destination host is controlled by the constant flow of acknowledgments in the reverse direction. If the flow of acknowledgments subsides, this is regarded as sign of congestion [Jac88] and the sending rate is promptly reduced. Thus, normal TCP communication can be modelled by the ratio of the number of packets sent to and received from a specific destination. Ideally, this ratio should be one, but network congestion and different TCP implementations using delayed and selective acknowledgments push it to slightly higher values. D-WARD's legitimate TCP flow model defines TCP_{rto} as the maximum allowed ratio of the number of packets sent and received on the flow.[3] The flow is classified as an attack flow if its packet ratio is above the threshold. The observed packet ratio is smoothed before classification to avoid noise in the measurements.

Legitimate ICMP flow model. ICMP messages are used as out-of-band control messages to report some network error or to assist network troubleshooting. Unfortunately, they have also been successfully used for various attacks. The ICMP protocol [Pos81] specifies many different message types. During normal operation the "timestamp," "information request," and "echo" messages should

[2]Legitimate connection traffic is not subject to rate-limiting and is thus not affected by the *Compliance Period* value.

[3]D-WARD uses 3 for the TCP_{rto} value.

be paired with the corresponding reply. Using this observation, the normal ICMP flow model defines $ICMP_{rto}$ as the maximum allowed ratio of the number of echo, timestamp, and information request and reply packets sent and received on the flow. The frequency of other ICMP messages, such as destination unreachable, source quench, redirect, etc., is expected to be so small that a predefined rate limit can be used to control that portion of the traffic.

Legitimate UDP flow model. The UDP protocol [Pos80] is used for unreliable message delivery and in general does not require any reverse packets for its proper operation. Many applications that use UDP generate a relatively constant packet rate, but the maximum rate depends heavily on the underlying application. UDP-based applications exhibit different dynamics, ranging from the low-frequency request/response of Network Time Protocol, to the steady rate of Real Audio, to the high-rate satellite data streaming. Additionally, users may write custom applications that use UDP traffic, which further hinders definition of UDP traffic models. D-WARD thus defines a very broad legitimate UDP flow model, attempting to detect only those UDP attacks that use heavy subnet spoofing or that send at a very high rate. An alternate approach is to build a legitimate flow model for each application that uses UDP, and detect attacks when unknown UDP traffic is observed. This approach is partially used for generation of legitimate UDP connection models. It would likely detect more UDP-based attacks, but it would be prohibitively expensive as many more statistics would have to be stored. Additionally, as users may define new UDP applications, models would have to be constantly updated. On the other hand, it is fairly simple to defend against UDP-based attacks close to the victim. Many Internet hosts do not normally receive heavy inbound UDP traffic and may place appropriate filters at upstream ISPs to completely deny UDP traffic. Clients that receive streaming media can install firewall rules that permit only inbound UDP traffic

on connections initiated by the client.

The legitimate UDP flow model is defined as two thresholds: n_{conn} — an upper bound on the number of allowed connections per destination, and p_{conn} — a lower bound on the number of allowed packets per connection. The thresholds help identify a UDP attack through spoofed connections, detecting the occurrence of many connections with very few packets per connection. The model classifies a flow as an attack when both of these thresholds have been breached.

6.4.2 Connection Statistics and Connection Classification

Each outgoing and incoming packet modifies one record in the *Connection Table*. A connection can only carry traffic from one protocol and one application. Legitimate connection models define statistics contents. Section 7.2 contains more details on connection statistics contents.

Connection classification is performed each *Connection Observation Interval* seconds. During classification, D-WARD compares connection statistics with corresponding legitimate connection models. A connection will be classified as:

1. **Good** if its statistics match the corresponding model.
2. **Bad** if its statistics do not match the corresponding model.
3. **Transient** if there is not enough data to perform classification.

Good connections receive guaranteed good service during the rate-limit phase, while bad and transient connections have to compete for the rate-limited bandwidth. The difference in handling bad and transient connections is present only in connection table record handling. Bad connection records never expire,[4] while

[4] A bad connection record will be deleted only after the connection has stopped sending traffic and is classified as good.

transient connection records expire after a very short interval.

Legitimate TCP connection model. D-WARD's legitimate TCP connection model is similar to its legitimate TCP flow model. It uses the same value of TCP_{rto} — the maximum allowed ratio of the number of packets sent and received on the connection. The connection is classified as good if its packet ratio is below the threshold. The observed packet ratio is smoothed before classification to avoid noise in the measurements.

Legitimate ICMP connection model. D-WARD does not deploy legitimate ICMP connection models.[5] ICMP traffic rarely has connection semantics — it is issued infrequently to diagnose network problems, contains very few clustered packets, and comprises a very small percentage of total Internet traffic. Legitimate ICMP connection models would be of little use to legitimate ICMP traffic, as short, infrequent connections would not be able to take advantage of priority status — they would terminate shortly after being validated. On the other hand, it is likely that dropping legitimate ICMP traffic during an attack will not produce great damage to legitimate clients.

A set of legitimate ICMP connection models resembling the legitimate ICMP flow models was deployed in D-WARD, and the above claims were proved correct in experiments. Further, spoofed ICMP attacks generated an abundance of records in the *Legitimate Connection List*, resulting in deletion of legitimate connection records and worse service to legitimate traffic. This effect was noticed only at very high spoofed packet rates, and did not generate a noticeable decrease in the service level perceived by legitimate clients. However, the number of legitimate packet drops by D-WARD was increased (due to missing records in the

[5]ICMP is a connectionless protocol, and it does not use port numbers. D-WARD observes all ICMP communication between two hosts as one connection, assigning zero values to port numbers.

Legitimate Connection List), resulting in retransmissions and connection delays. Hence, the attempt to detect legitimate ICMP connections was abandoned.

Legitimate UDP connection model. D-WARD builds per-application models for UDP traffic. We identified the main categories of applications that use UDP, which are:

1. Domain Name Service (DNS)
2. Network Time Protocol (NTP)
3. multimedia streaming
4. voice over IP (VoIP)
5. Internet multi-player games
6. Network File System (NFS)
7. chat applications

We compared the selected categories with statistics from the Cooperative Association for Internet Data Analysis (CAIDA) Web site [CAI] which shows the top 25 applications (regardless of underlying transport protocol). There are only a few UDP applications among the top 25 since UDP traffic occupies a small proportion of all network traffic (around 3%). All of these are on our list of selected UDP applications.

The following models were designed (and a subset of these was implemented):

- Domain Name Service (DNS). DNS can be implemented over TCP or UDP, but UDP is more common. Under normal operation, DNS connection traffic will exhibit a 1:1 ratio of packets sent to packets received. If the response

packet is lost, the DNS client typically repeats its request to another server before retrying the same server. The retransmission interval is between 2 to 5 seconds, and the message size is 46 to 512 bytes (not including UDP and IP headers). DNS packets are identified according to the application field in the IP header which carries the value 17, and according to a port number 53 in the UDP header. Request and response packets are identified according to the first bit of the third byte of the DNS header, which is 0 for a query packet, 1 for a response packet. A legitimate DNS connection model is defined by a finite state machine shown in Figure 6.4. The connection starts from `NO_STATE`. When a DNS request is sent to a foreign host the connection goes to the `DNS_REQ` state and when a reply is received the connection goes to the `DNS_REP` state. The connection can accommodate repeated requests only if their frequency is lower than `DNS_EXPIRY`, which is set to 1 second. Any violation of the model brings the connection into the `ERROR` state, which leads to bad classification.

- Network Time Protocol (NTP). In NTP there are three main methods or association modes in which a system can synchronize with other hosts: client mode, broadcast or multicast client mode, and symmetric active mode. In client mode, a host polls one or more hosts to get the current time, and processes replies as received. In broadcast and multicast modes, a host does not poll. Instead, it listens for NTP packets that are broadcast or multicast over the local network. In symmetric active mode, a host polls other hosts and responds to polls from those hosts. In addition, hosts retain time-related information about the hosts with which they are communicating. Hence, there are three possible UDP connection models for NTP. Under normal operation NTP connection traffic will exhibit a 1:1 ratio of packets sent to

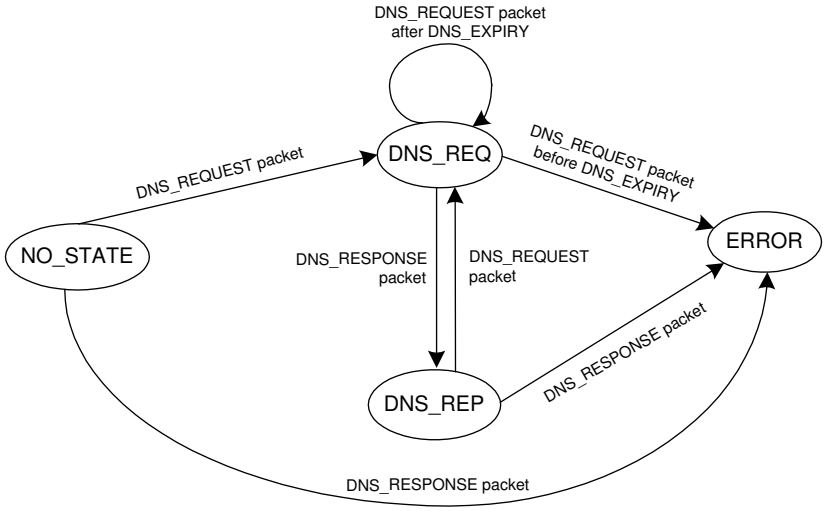

Figure 6.4: DNS finite state machine

packets received. If the response packet is lost, the NTP client typically repeats its request to another server before retrying the same server. The polling interval ranges from 64 to 1024 seconds. Message sizes range from 44 to 56 bytes. D-WARD only designs and implements client NTP connection models. A legitimate NTP connection model is defined by a finite state machine shown in Figure 6.5. The connection starts from `NO_STATE`. When an NTP request is sent to a foreign host, the connection goes to the `NTP_SENT` state; and when a response is received, the connection goes to the `NTP_REC` state. The connection can accommodate repeated requests only if their frequency is lower than `NTP_EXPIRY`, which is set to 60 seconds. Any violation of the model brings the connection into the `ERROR` state, which leads to bad classification.

- Multimedia streaming. The most common application programs used for audio and video streaming are RealPlayer, Windows Media Player, Quick-

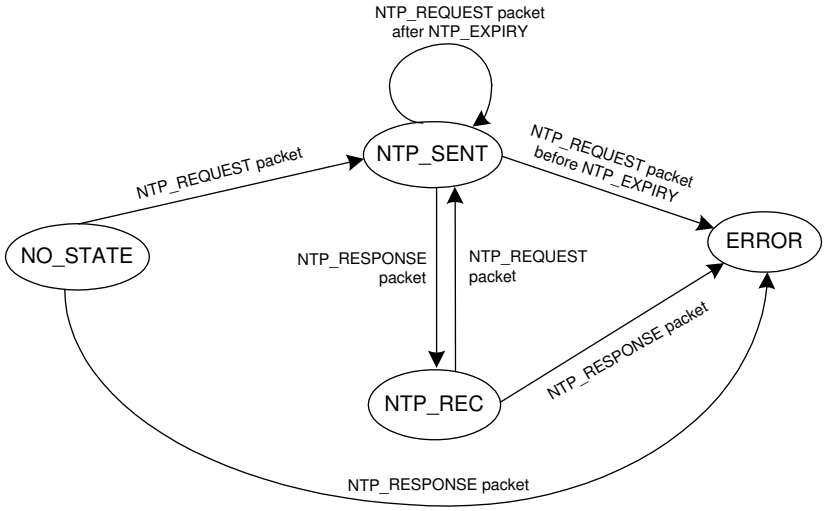

Figure 6.5: NTP finite state machine

time, and Shoutcast mp3 streaming. Quicktime and Real Player use the Real-time Transport Protocol (RTP) over UDP for data delivery and the Real-time Streaming Protocol (RTSP) over TCP for control. Windows Media Player uses the Microsoft Media Server (MMS) protocol — Microsoft's proprietary network streaming protocol over TCP and UDP which contains both data delivery and control mechanisms. D-WARD only provides models for multimedia streaming applications that use RTP and RTSP (because MMS is a proprietary protocol, applications using this protocol cannot be modelled). The streaming data is sent from server to client in small, equally spaced RTP packets, ranging in size from 12 to 72 bytes. For every few received RTP packets, the client sends one RTP packet back to the server. RTSP traffic is sent via TCP at the beginning of the session and every 1 to 2 seconds to deliver reports about the session condition. D-WARD models legitimate streaming media connections by correlating the behavior of RTP and RTSP connection traffic, and by using a streaming media finite state

machine shown in Figure 6.6. When an RTP packet is sent or received, D-WARD checks to see if the related RTSP connection exists. The RTSP connection is located by searching for a connection with the same source and destination IP address as the RTP connection, and the foreign port number 554, which is the RTSP port number. If the RTSP connection exists and it is not stale (it has been active in last `RTSP_ACTIVE` seconds, currently set to value 5), the RTP connection goes to `STREAMING` state. Otherwise it goes to `ERROR` state. When the connection classification is performed, RTP connections in the `STREAMING` state will also be checked for high values of RTP sent to the received packet ratio. If this ratio is lower than RTP_{rto} (currently set to 20), the RTP connection will be classified as good. If the ratio is higher than RTP_{rto} or the connection is in `ERROR` state, then it will be classified as bad. By performing live tests we have observed that streaming media servers will continue to generate RTP traffic toward the client in the absence of reverse RTP and RTCP packets (as might occur in the case of DDoS attack). Such behavior will be tagged by D-WARD as illegitimate, and the connection will be classified as bad.

Models for voice over IP, Internet multi-player games, Network File System (NFS) and chat applications' traffic are not provided in current D-WARD version but they could be built similar to the DNS, NTP or streaming media models.

As new uses of UDP become popular, new connection models will need to be generated and added to D-WARD. Past experience suggests that this will happen infrequently.

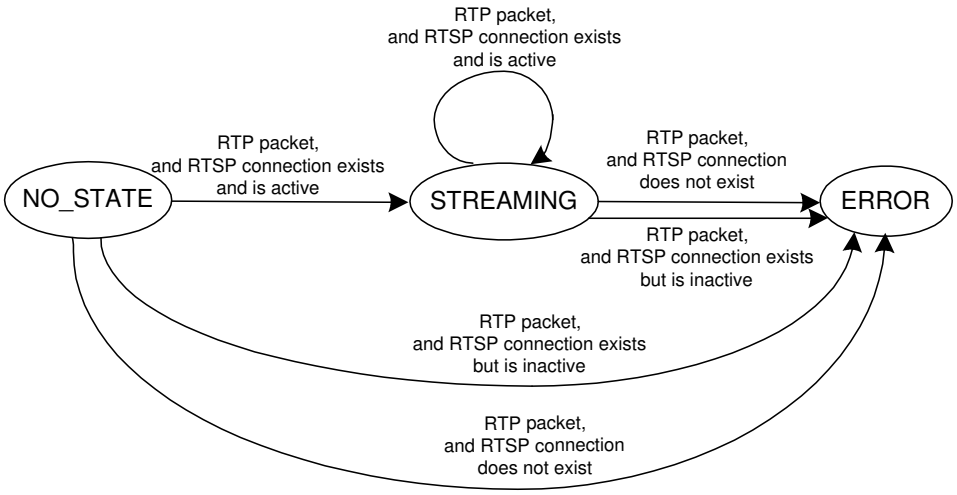

Figure 6.6: Streaming media finite state machine

6.4.3 First Packet Classification

A difficult problem that D-WARD faces in connection separation is the problem of classifying connections based on the first outgoing packet. This problem occurs when a connection attempts to start during the attack. The first outgoing packet seen by D-WARD cannot be recognized as legitimate, since legitimate connection models require more data to perform classification. It is thus likely that this connection will be regarded as transient and its traffic will be subjected to rate-limiting. During rate-limiting, these first packets will likely lose the competition against aggressive attack traffic, and will be dropped. Dropping those first packets on a connection may severely discourage connection setup (this is especially the case with TCP connections) and thus will result in poor service to new connections during the attack. The likely outcome is that no new connection traffic will be allowed to pass during the attack.

The following example, depicted in Figure 6.7, illustrates an especially severe

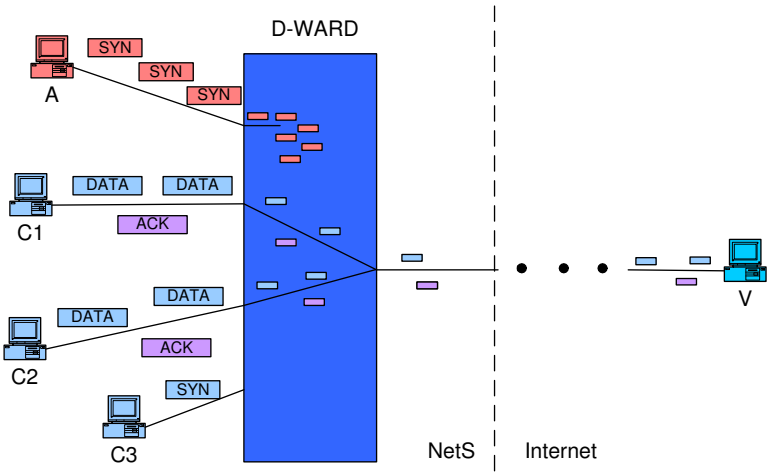

Figure 6.7: Example of first packet classification problem

case of the first packet classification problem: new TCP connection setup during a TCP SYN flood attack.

Source network NetS is protected by D-WARD. A TCP SYN flood using subnet spoofing is generated from subverted host A to victim V. D-WARD detects this attack and imposes a rate limit on the outgoing flow from NetS to V. Legitimate clients C1 and C2 have previously established connections to V, and these connections will be identified as legitimate and protected from rate-limiting. During the attack, legitimate client C3 attempts to connect to victim V. D-WARD observes a multitude of attack TCP SYN packets and one legitimate TCP SYN packet. There is no information in the packet that the attacker cannot spoof. The only difference between those two packet classes is in future behavior: the legitimate client will complete the three-way handshake and continue to generate traffic on the connection, while the attacker will not. Therefore, those two packet classes are indistinguishable by D-WARD. Both a legitimate connection and all attack connections will be classified as transient and will be subject to

rate-limiting. During rate-limiting, aggressive attack traffic is more likely to win, resulting in legitimate TCP SYN packet drops. The problem is actually more severe than this, as not only the first packets but all packets on new connections will likely be dropped until the *Connection Observation Interval* expires and the connection can be validated.

While persistent packet drops will hurt any connection performance, they are especially detrimental to TCP connections. The TCP protocol perceives packet loss as a sign of congestion and slows down its sending rate. Each consecutive packet loss leads to the exponential decrease in control window size (the parameter determining the amount of traffic to be sent between acknowledgments). As losses start occurring from the very beginning of the connection, when the control window is already quite small, further decreases quickly bring it to its minimum value. The parallel adverse effect is the exponential increase of the retransmission interval when retransmitted packets are also lost. Jointly, these two mechanisms are very successful in relieving and adapting to temporary congestion, but they also make TCP non-competitive against aggressive attack traffic. A new connection will send less traffic and do so more reluctantly as each packet is lost. Over time, it will either either terminate or be severely delayed by D-WARD's action.

There are several possible strategies to ameliorate this situation:

1. **Assume TCP SYN cookie deployment by the victim.** The TCP SYN flood attack can be successfully handled up to very high rates by using TCP SYN cookies [SKK97]. This approach handles the half-open connection problem by replacing server state with cryptographic information. SYN cookies are particular choices of initial TCP sequence numbers by TCP servers. In response to a received TCP SYN packet, the server chooses its initial sequence number as a function of: time when request was received,

the server's maximum segment size (MSS), a server-selected secret function of the client IP address and port number, the server IP address and port number, and time. This choice of sequence number complies with the basic TCP requirement that sequence numbers increase slowly; the server's initial sequence number increases slightly faster than the client's initial sequence number. A server then sends a SYN ACK with a chosen initial sequence number, but does not reserve a record in its connection queue. When the server receives an ACK completing the three-way handshake, it checks that the secret function works for a recent value of time, and then rebuilds the connection queue entry from the encoded MSS. D-WARD can assume use of TCP SYN cookies by the victim and always forward TCP SYN packets. Unfortunately, many hosts do not deploy TCP SYN cookies, so D-WARD would fail to protect such hosts from the TCP SYN flood attack. Also, this approach would not solve the problem of subsequent outgoing packets on this connection that may arrive before the connection has been validated. These packets will be subject to rate-limiting and will likely be dropped.

2. **Proxy TCP connections.** D-WARD can choose to handle TCP SYN flood attacks by deploying TCP SYN cookies itself and performing the three-way handshake instead of the victim. This approach is taken by numerous defense systems. A defense system intercepts TCP SYN packets and replies, instead of the server, using TCP SYN cookies. Once the client completes the three-way handshake, the defense system replays the TCP SYN to the server and acts as the client to complete connection setup. The problem arises because each party must independently choose an initial sequence number in its packets. Thus, the server's initial sequence number which is chosen first by defense system (in the server's name) will

not match the initial sequence number chosen by the server. The defense system cannot change sequence number information at either party. It is therefore left with two choices: (1) proxy the complete connection, rewriting sequence numbers appropriately, or (2) drop the connection by sending a RST packet to the client, remembering that it was legitimate so that on retry its TCP SYN packets will be allowed to reach the server directly. As a connection retry must be initiated by the application, a connection drop is a bad choice. Thus, the majority of defense systems decide to proxy the complete TCP connection. This choice would impose a heavy burden on D-WARD since it would have to keep even more connection state and modify each packet on legitimate connections during the attack.

3. **Introduce modifications to TCP to specify desired ISN.** One possible approach to proxy only the three-way handshake and not the complete connection would be to add a new option to the TCP protocol — *desired ISN*. This option would be used by the defense system in the replayed TCP SYN packet, requiring the server to choose a specific value of its initial sequence number. This modification would require TCP protocol changes at all servers in the source network and would likely introduce many security problems.

4. **Reset half-open connections.** D-WARD can initially forward all TCP SYN packets, but keep track of half-open connections. If they are not completed within a certain time interval, D-WARD can issue a RST packet to the server, freeing its resources. This approach is also taken by many defense systems. Its drawback is that D-WARD would have to keep state on half-open connections, thus becoming a potential victim of denial-of-service attack itself.

5. **Reject randomly spoofed packets.** The *value prediction technique* assumes that the defense system can know or accurately predict a range of valid values for certain TCP or IP header fields in the first connection packets (the *predicted value range*). The prediction function must be accurate enough so that legitimate packets will always be detected and validated. The predicted value range must also be relatively small so that packets that randomly spoof header values have a low probability of being validated as legitimate. D-WARD uses the value prediction technique to recognize legitimate packets at TCP connection startup. A stealthy attacker can still learn the predicted value range and adequately spoof his packets to guarantee that they will be identified as legitimate. To counter this, D-WARD introduces a special fixed *Early Packet Rate Limit* that applies to packets that match the predicted value range. The overall outcome is that legitimate clients' traffic will receive good service during a randomly spoofed TCP SYN flood attack, while it will still likely be dropped during a stealthy TCP SYN flood attack. Since the stealthy attack traffic is still subject to rate-limiting with *Early Packet Rate Limit*, the attacker cannot misuse the value prediction technique to perform successful attacks. Thus, the value prediction technique significantly ameliorates the situation in a randomly spoofed attack case, and does not make it any worse in a stealthy attack case.

D-WARD uses the value prediction technique to predict sequence number values in TCP packets on new connections. As a TCP connection is set up, each party chooses an initial sequence number (ISN) for this connection. This number is used by the other party to keep count of received bytes, to perform reordering of packets that come out of order and to request retransmissions of lost content.

Each outgoing TCP packet is marked by a sequence number value that is derived from ISN in the following manner: for packets carrying bytes $[N, N+1..N+K]$ on a connection, the sequence number value will be $ISN + N + K$. So the sequence number value increases slowly within the connection.

The ISN value was initially designed as a time-driven, linearly increasing value [Ins81]. It was generated based on a clock that was incremented each 4 microseconds, so that $ISN = T$, where T is the clock value at connection setup time. This approach generated a security hole and the opportunity for TCP connection hijacking attacks as explained in [Bel96]. Let us assume that machines A and B have formed a trusted relationship so that users from A can execute commands at B without supplying a password, through *rsh* (remote shell). In a connection hijacking attack, the attacker impersonates A to execute malicious commands at B. He first contacts B by sending a SYN packet and receiving a SYN ACK, with B's ISN. As ISN is increased linearly, the attacker can now predict all future ISN values. He then impersonates A and sends a TCP SYN packet with A's spoofed address to B. B replies with a SYN ACK packet to A. The attacker cannot see this packet but, as he can guess B's ISN, he has enough information to generate an ACK and complete the three-way handshake in A's name. The attacker then has unlimited access to B and can execute malicious commands.

To counter this attack, the ISN generation process was altered to include a small amount of randomness so that the ISN value cannot be guessed by the attacker. It is recommended in [Bel96] that the ISN be generated as a sum of timer value and a cryptographic function on the connection identifier. Therefore, ISN is calculated as: $ISN = T + f(localIP, localport, foreignIP, foreignport)$.

In order to devise an ISN prediction function, we first examine how different operating systems choose their initial sequence numbers. We initiate TCP

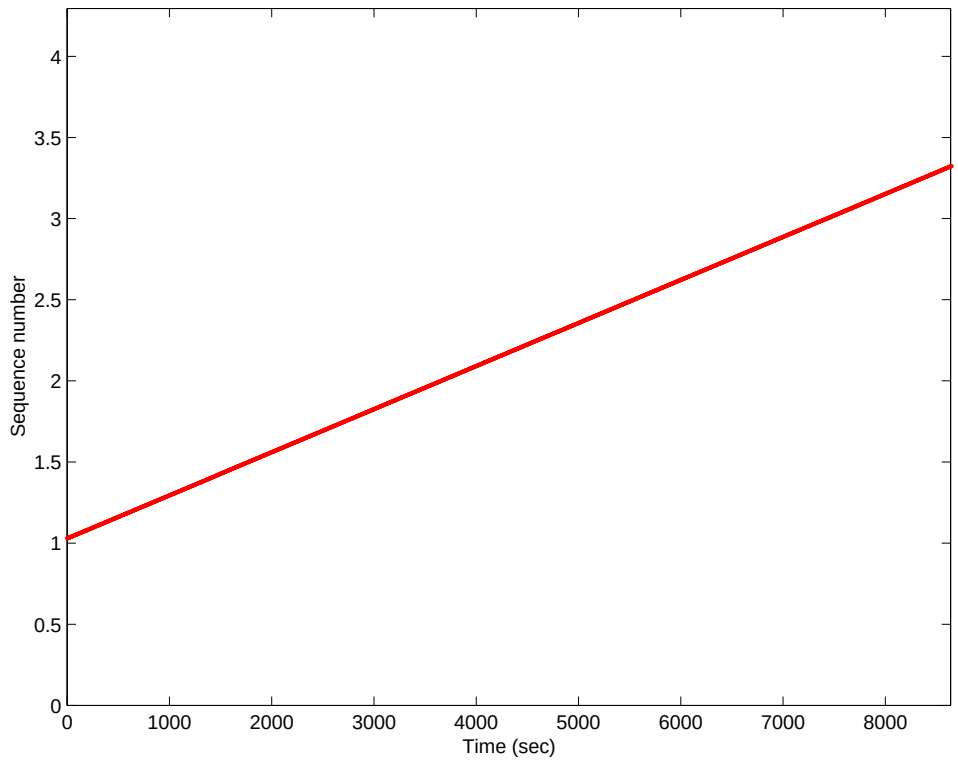

Figure 6.8: Sequence number space of Windows 2000

connections sequentially to three machines, one running Windows 2000, another running Linux 2.4.9, and the third running FreeBSD 4.8. We then plot the initial sequence number space in Figures 6.8, 6.9 and 6.10, respectively.

As we can see, FreeBSD generates purely random initial sequence numbers[6] while Windows and Linux generate initial sequence numbers that slowly increase with time. On closer observation, as depicted in Figures 6.11 and 6.12 that are zoomed-in segments of Figures 6.8 and 6.9, we conclude that Windows chooses its initial sequence numbers purely as a function of time ($ISN = T$) while Linux introduces a small amount of randomness ($ISN = T + rnd$) as suggested in

[6]This approach is not compliant with [Bel96], but the complete randomization of the sequence number space minimizes the possibility of connection hijacking attacks.

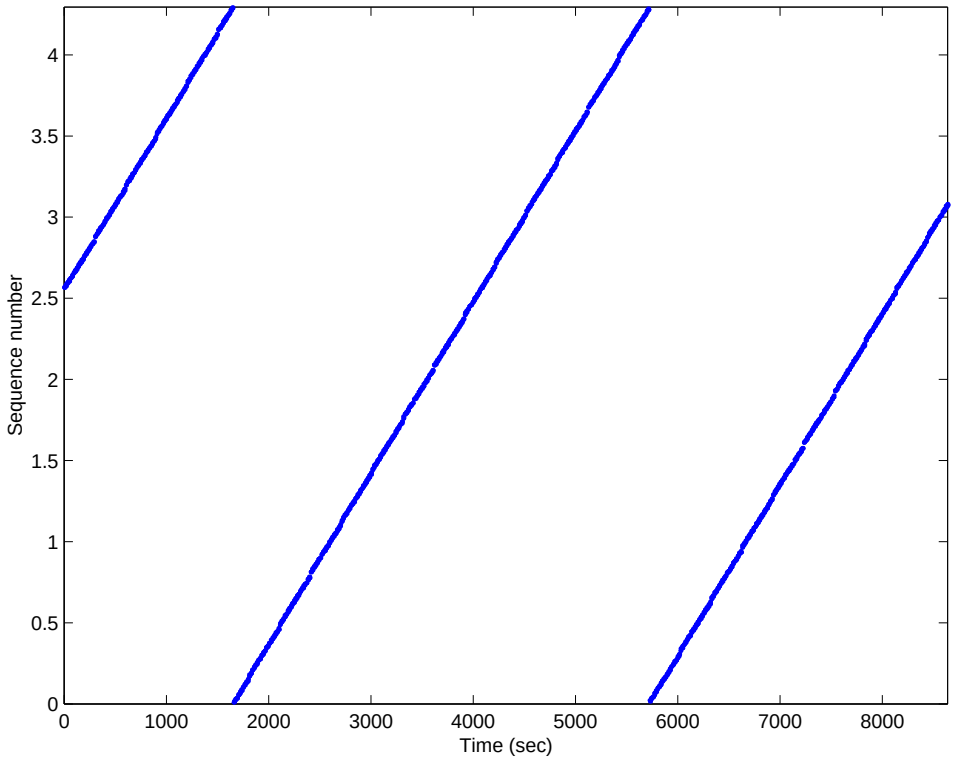

Figure 6.9: Sequence number space of Linux 2.4.9

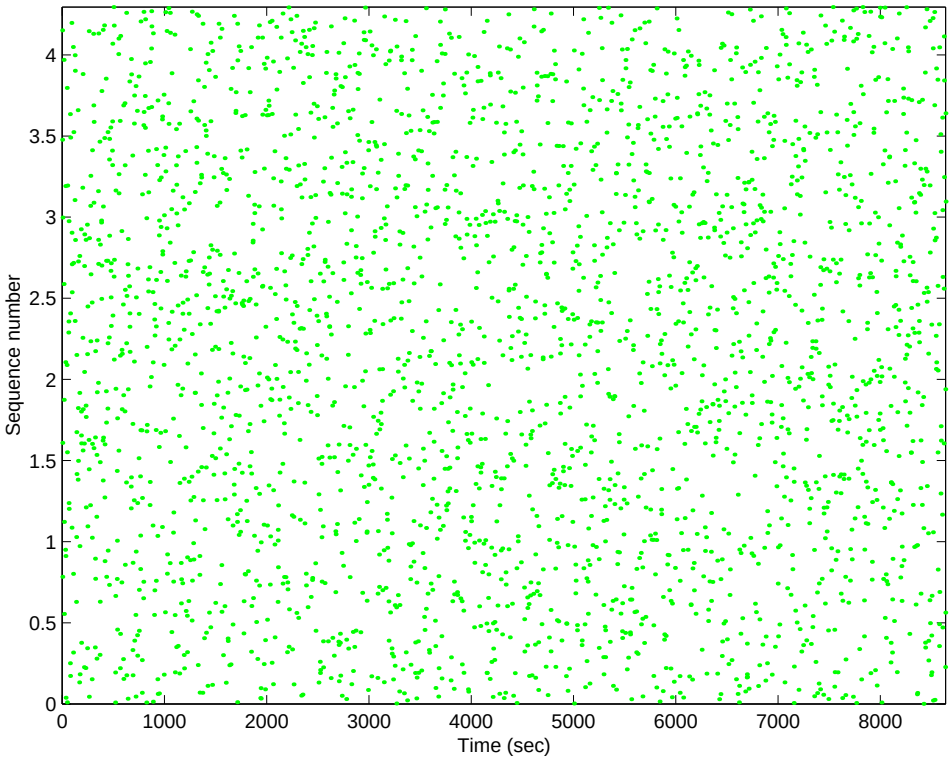

Figure 6.10: Sequence number space of FreeBSD 4.8

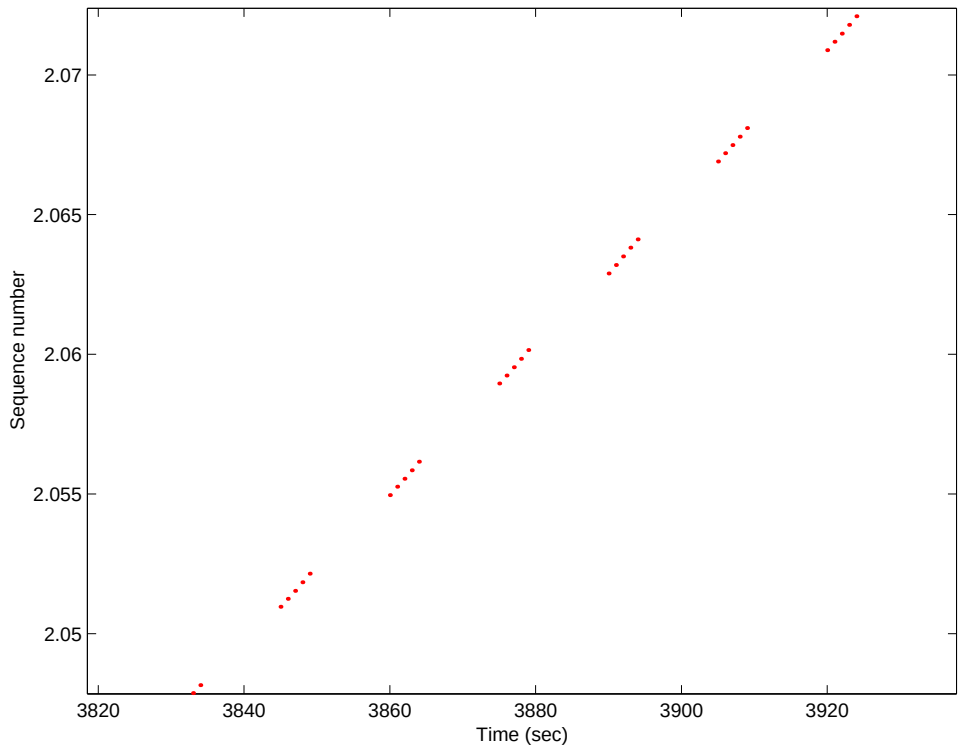

Figure 6.11: A closer look at sequence number space of Windows 2000

[Bel96].

D-WARD aims to predict a range of possible initial sequence number values in TCP packets from a given source address. In order to do so, D-WARD builds an *Initial Sequence Number Model* for each source address. The model records the last sequence number and the time it was seen for the two most recently validated connections. The choice of last sequence number instead of initial sequence number is necessary, since the connection is validated after the setup phase when only its current sequence number can be observed. This will introduce a small error in prediction. However, as D-WARD predicts only a range of legitimate values, and not the exact value, this error is well within the margin.[7]

[7]Note that the degree of the ISN prediction that is sufficient for D-WARD purposes is not

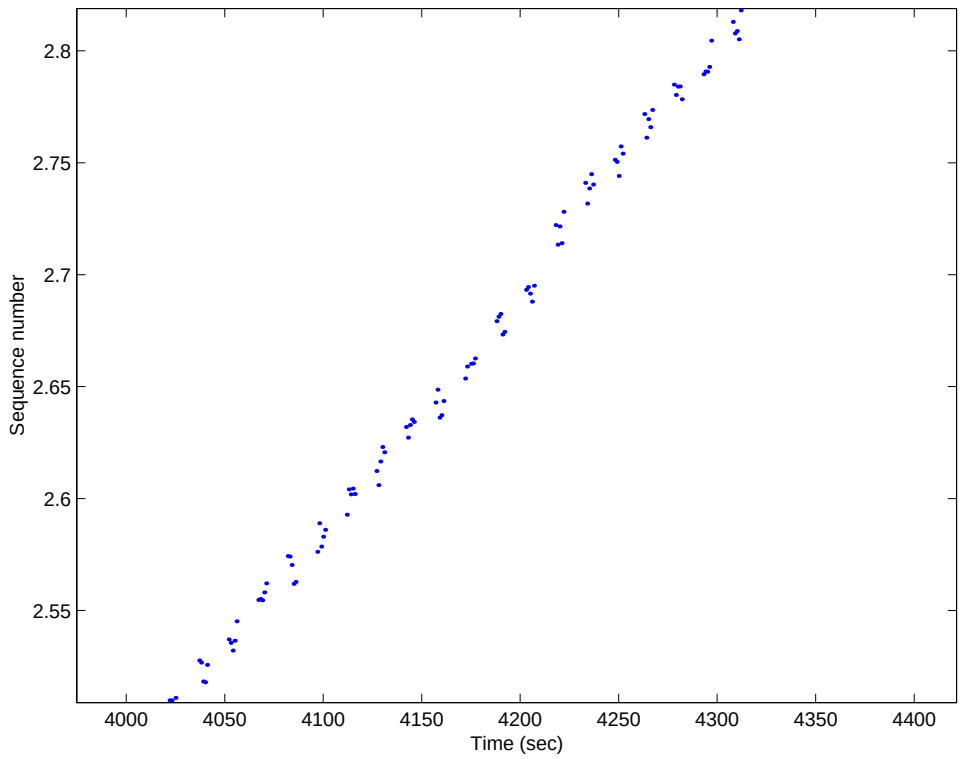

Figure 6.12: A closer look at sequence number space of Linux 2.4.9

D-WARD periodically examines the two recorded sequence numbers and classifies machine as Windows, Linux or FreeBSD. More details about this classification are given in Section 7.4.2. During an attack, D-WARD will attempt to validate each outgoing TCP packet from transient connections using the *Initial Sequence Number Model* for its source address. If the sequence number from the packet is within the error margin of the predicted value, the packet will be subject to *Early Packer Rate Limit*; otherwise it will be subject to regular rate-limiting.

6.5 Rate-Limiting Component

The rate-limiting component adjusts rate limit values every *Flow Observation Interval*. It receives classification results from the observation component and flow behavior history from the traffic policing component and devises a rate-limit value for each active flow.

Flow behavior history is expressed through two metrics: the byte amount of flow traffic forwarded to the victim — B_{sent}, and the byte amount of flow traffic dropped due to rate limiting — $B_{dropped}$. Both values are measured within the *Flow Observation Interval*. Together they define the *Flow Compliance Factor* — a measure of how well a flow complies to the imposed rate limit. The *Flow Compliance Factor* is calculated as:

$$fcf = \frac{B_{sent}}{B_{sent} + B_{dropped}} \tag{6.1}$$

Flow Compliance Factor values range from 0 to 1, where higher values indicate better compliance with the imposed rate limit.

sufficient to carry out connection hijacking attacks.

6.5.1 Exponential Decrease

When the flow is classified as an attack flow for the first time after a long period of being normal, its rate is limited to a fraction of the offending sending rate. The size of the fraction is specified by the configuration parameter f_{dec}.

$$rl = \frac{B_{sent}}{Flow\ Observation\ Interval} * f_{dec} \tag{6.2}$$

Subsequent classification of a flow as an attack restricts the rate limit further, applying exponential decrease according to the formula:

$$rl = min(rl, \frac{B_{sent}}{Flow\ Observation\ Interval}) * f_{dec} * fcf \tag{6.3}$$

where rl is the current rate-limit. The *Flow Compliance Factor* — fcf — describes the degree of misbehavior of the flow, and defines the restrictiveness of the rate-limit. Flows that have worse behavior ($fcf \ll 1$) are quickly restricted to very low rates, whereas this restriction is more gradual for better-behaving flows ($fcf \sim 1$). The lowest rate limit that can be imposed is defined by the $MinRate$ configuration parameter, so that at least some packets can reach the destination and trigger a recovery phase.

6.5.2 Linear Increase

When the attack-detected signal becomes negative, the associated flow will be classified as suspicious, at which point the recovery mechanism is triggered. The recovery phase is divided into slow-recovery and fast-recovery. During the slow-recovery phase, a flow is penalized for having been classified as an attack flow by a linear increase in the allowed rate according to the formula:

$$rl = rl + rate_{inc} * fcf \tag{6.4}$$

The speed of the recovery is defined by the $rate_{inc}$ parameter and the duration of the slow-recovery phase is defined by the *Compliance Period*.

6.5.3 Exponential Increase

When the flow has been classified as normal, the fast-recovery phase is triggered. During the fast-recovery phase the rate is increased exponentially according to the formula:

$$rl = rl * (1 + f_{inc} * fcf) \tag{6.5}$$

The speed of the recovery is defined by the f_{inc} parameter, and the rate increase is limited by the $MaxRate$ configuration parameter. As soon as the rate limit becomes greater than $MaxRate$, the recovery phase is finished, and the rate limit is removed.

Figure 6.13 depicts rate-limit values and classification results for a sample flow.

6.6 Traffic-Policing Component

The traffic-policing component periodically receives rate-limited flow information from the rate-limiting component (every *Flow Observation Interval*) and connection classification information from the observation component (every *Connection Observation Interval*). It uses this information to reach a decision whether to forward or drop each outgoing packet in the following manner:

- **If** the packet belongs to a non-limited flow, forward it, **else**
- **If** the packet belongs to a good connection, forward it, **else**

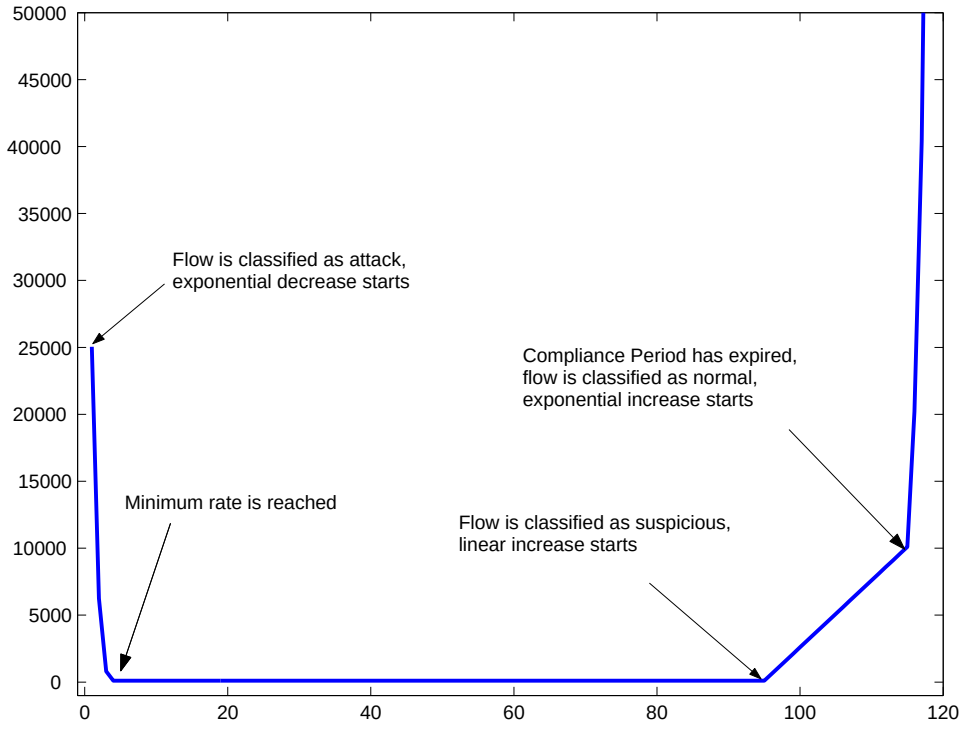

Figure 6.13: Rate limit values and classification results for a sample flow

- **If** the packet is TCP **and** its sequence number matches the predicted value **and** the *Early Packet Rate Limit* for the flow is not exhausted, forward it, **else**
- **If** the flow rate limit is not exhausted, forward the packet, **else**
- Drop the packet.

6.7 Stealthy Attackers

The attacker may resort to stealthy techniques to avoid attack detection or to trick D-WARD into classifying malicious connections as legitimate. In the following sections we discuss each of these strategies and offer possible approaches to detect and counter them.

6.7.1 Small-Rate Attacks

The legitimate TCP and ICMP flow models will work well for high-volume attacks, when the attack creates a large disturbance in the flow statistics. However, a stealthy attacker may try to avoid detection by distributing attack traffic over multiple source networks so that each network observes only a small portion of the attack. If the amount of attack traffic is much smaller than the amount of legitimate traffic sent to the victim, then the victim responses to legitimate traffic may maintain a low sent-to-received packet ratio, hindering detection. In the case of small-rate attacks that aim merely to degrade the victim's resources, detection will be completely avoided. In the case of TCP attacks that severely damage the victim's operation, the stealthy attacker will not be able to completely avoid detection. Once the victim is severely overloaded, it will stop generating a suffi-

cient amount of replies. Legitimate TCP traffic will then back off (and become smaller than the amount of attack traffic), thus exposing the attack and inducing detection. However, this detection comes after the fact. A significant portion of legitimate traffic has already been dropped and legitimate connections have likely suffered a large delay.

In order to improve detection time and accuracy for small-rate attacks, D-WARD introduces the *non-validated residue* detection method. This method amends flow classification and attempts to detect attacks based on the steady backlog of transient connection traffic. The flow statistics are extended to keep track of the amount of traffic belonging to transient connections at each *Flow Observation Interval*. We shall call this value *non-validated traffic*. Under normal system operation the amount of non-validated traffic is zero, since all connections within the outgoing traffic are classified as good. When new connections are initiated, they create short-lived peaks of non-validated traffic that are soon brought to zero in subsequent classification steps. Under normal load these peaks are far apart. Under the attack, assuming that the connection classification mechanism has a low level of false negatives, the amount of non-validated traffic will exhibit prolonged bursts.

Figure 6.14 depicts the amount of non-validated traffic during normal operation and during the attack in a sample experiment. Each peak represents the initiation of a new connection to the victim. The tests involved intensive and frequent file transfers between our test network and the victim, thus the frequency of peaks is much higher than should be expected in the real network. It can be observed that under normal load the width of the peaks is small (2 to 3 seconds). Between peaks the amount of non-validated traffic returns to zero. In the attack case, the amount of non-validated traffic is positive for the whole duration of the

attack, following the shape of the attack traffic.

D-WARD detects prolonged bursts by calculating the minimum amount nv_{min} of observed non-validated traffic in the previous N_{burst} classification intervals. Attack detection is triggered if nv_{min} is greater than zero. From the above description, it is evident that successful detection does not depend on the volume of the attack but purely on whether its duration is longer than N_{burst} intervals. We call this method *continuous non-validated residue detection*. Detection of small-rate attacks is tested in experiments, and the results are presented in Section 9.5.

6.7.2 Small-Rate, Infrequent, Pulsing Attacks

A sophisticated attacker could further attempt to avoid detection by performing pulsing attacks. He would send the attack traffic during T_{on} intervals (where $T_{on} < N_{burst}$) to avoid *continuous non-validated residue detection*), and pause during the next T_{off} intervals, then repeat the cycle. Note that the rate of the pulsing attack must be small to avoid detection by legitimate flow models. One sample pulsing attack and the disturbances it invokes in non-validated traffic statistics are shown in Figure 6.15.

To counter this type of attack, D-WARD samples the amount of non-validated traffic in each interval with probability p_{sample}. Collected samples are stored in a first-in-first-out (FIFO) queue of size N_q. Attack detection is triggered if at any observation interval the minimum element in this queue has a non-zero value. The size of the queue N_q determines the probability that the attack will be detected; a shorter queue increases the chances of detection, but also may increase the number of false positives. Sampling probability p_{sample} defines the speed of filling the queue and thus affects the detection. We call this detection method

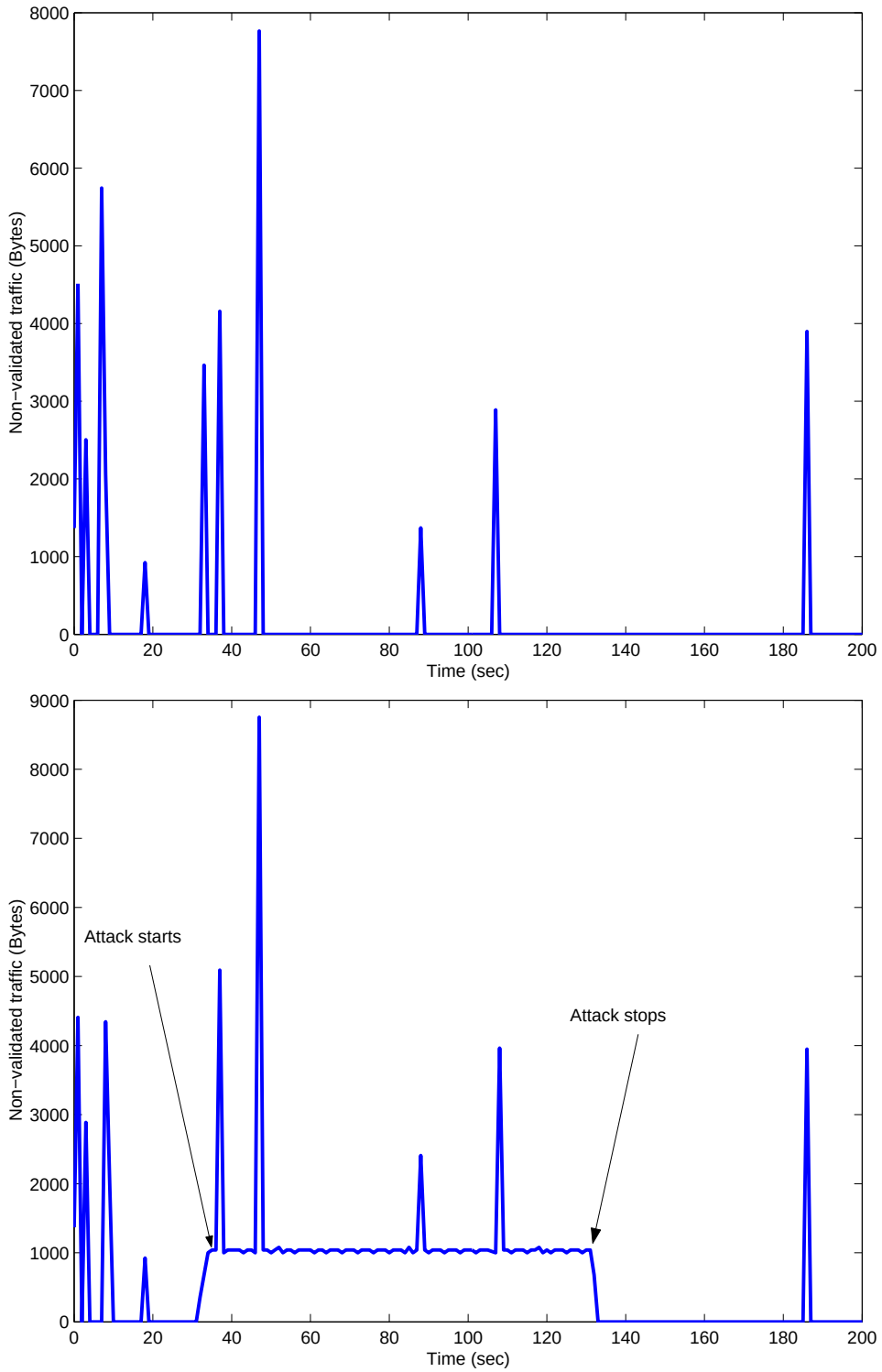

Figure 6.14: The amount of non-validated traffic during normal operation and attack

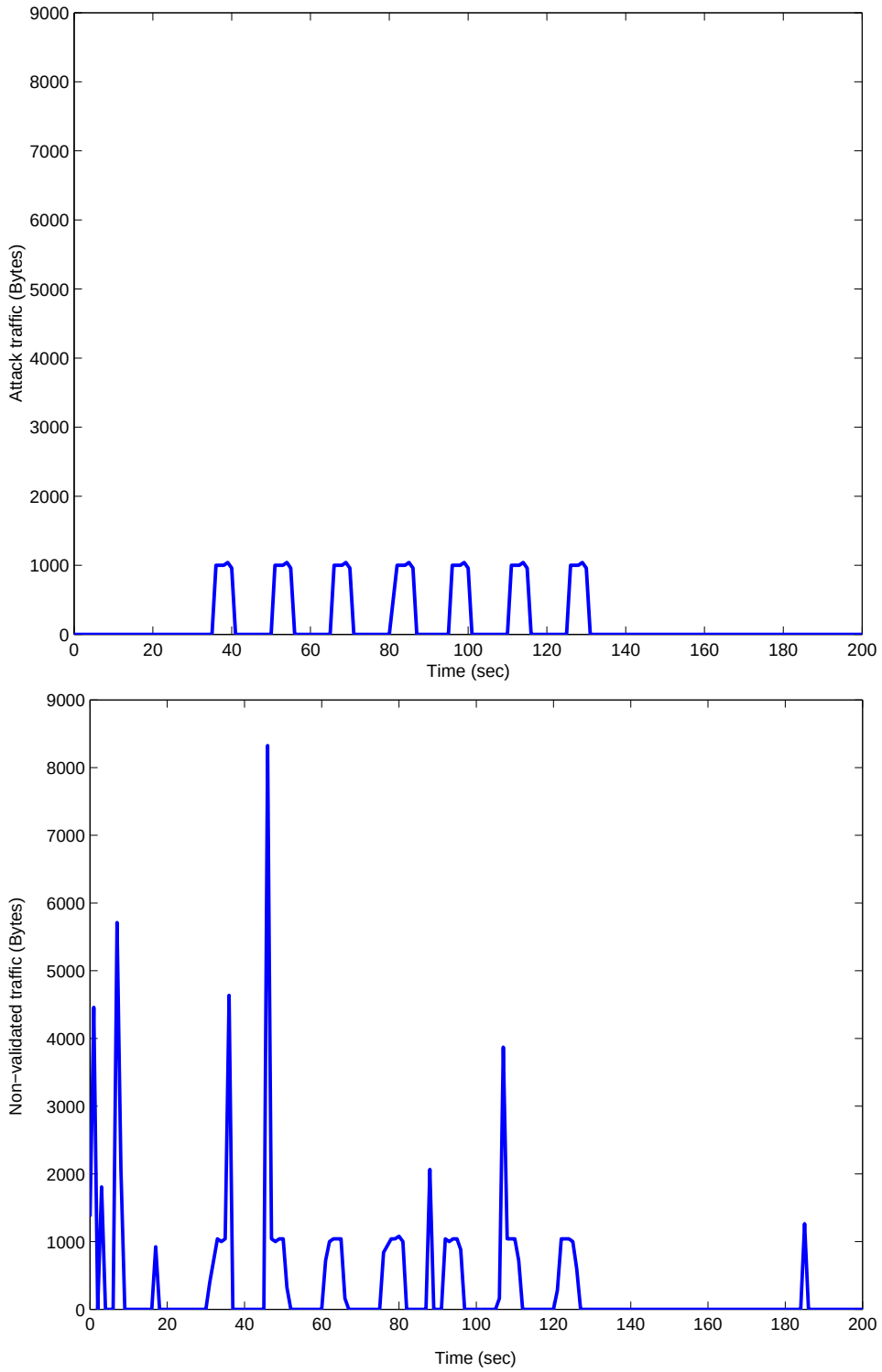

Figure 6.15: Pulsing attack and the amount of non-validated traffic that it produces

pulsing attack detection. Detection of pulsing attacks is tested in experiments and the results are presented in Section 9.5.

6.7.3 Spoofing Acknowledgments

When classifying TCP and ICMP connections and flows, D-WARD relies on return packets from the destination to determine whether this host is under attack. If an attacker could spoof these reply packets, then D-WARD would believe that it was seeing legitimate traffic. The attacker would have to gather twice the number of slave machines, half within the D-WARD network for the actual attack and half on the outside for spoofed reply generation. He would further have to carefully plan the attack and synchronize output of inside and outside slaves, in order to generate appropriate acknowledgements for current outgoing packets.

One possible solution for detecting this attack would be to allow communication between the victim and D-WARD. Since the victim would issue explicit notifications of the attack false replies could not delay detection. However, attack connections would still be falsely classified as legitimate.

The other approach is to have D-WARD allocate a small set of records and delay some outgoing TCP packets, storing their contents in these records. Each outgoing and incoming TCP packet is then matched against these records. If an outgoing match is found (i.e., connection data and sequence numbers match those in the records), the duplicate packet is dropped. If an incoming match is found, (i.e., connection data and acknowledgment numbers match the connection data and sequence numbers from the records), attack detection is triggered and the associated connection is invalidated.

The current D-WARD implementation does not contain any of the above methods for detection of spoofing acknowledgements. However, the communica-

tion of D-WARD system with the victim was enabled and tested in the COSSACK and DefCOM integration efforts, described in Sections 10.1 and 10.2.

6.8 Security

Security protocols are sought-after targets because their compromise often gives a good deal of power and prestige to the attacker. Special care must be taken to secure D-WARD against intrusion, misuse and attacks that deny service to legitimate clients.

Compromising a D-WARD router would allow an attacker to apply rate limits on any packets flowing out of the domain. However, the attacker can do much more damage with full control of the router, so adding D-WARD functionality makes the situation no worse.

D-WARD is unlikely to make it easier for the attacker to break into the router, since it exchanges no messages with other entities. D-WARD nodes were specifically designed to operate autonomously from other D-WARD nodes. In general, more complex systems are less secure than simple systems, so removing communication is a benefit. The autonomous operation removes the problems associated with securing communication sessions among a large number of participants and relying on potentially subverted hosts. Furthermore, in the unlikely event that a D-WARD node is compromised, it cannot be used to compromise other nodes.

Clever attackers will try to disguise attack traffic as normal traffic so that D-WARD will not filter it. The attackers do not have much room to maneuver because the attack must mimic the congestion control mechanisms found in the protocols. Since the attackers want to degrade performance through congestion, obeying congestion-avoidance mechanisms is contradictory. The attacker would

be forced to employ many more machines to get the same effect at the victim.

UDP traffic poses a special problem for D-WARD. D-WARD recognizes only UDP attacks that use spoofed connections. An attacker could choose to consistently spoof a few addresses and thus avoid detection while still getting a sufficiently high volume of packets to the victim. This problem is discussed at length in Section 6.4.1. Extending D-WARD with a resource-sharing mechanism would enable detection and control of a few high-volume UDP connections, and counter this attack. The other approach is to incorporate D-WARD into a distributed DDoS defense system and thus complement its detection with victim-end detection. Two projects that combine D-WARD with a distributed defense system are described in Sections 10.1 and 10.2.

The attacker could aim its attack, not at a particular target machine, but at an incoming link to a target network. He would divide the attack so that a stream to each individual machine seems normal and is not detected by D-WARD, but the aggregate flow to the target network overwhelms its input link. D-WARD bases the detection of TCP and ICMP attacks on the ratio of number of sent and received packets, rather than on the stream size. We expect that in the case of incoming link overload, some target machines would not generate sufficient responses and their streams would be tagged as attacks. The current implementation of D-WARD could not correlate these attacks, but such a feature could be added. Communication with the victim network would also be helpful in detecting of this type of attack.

D-WARD is a DDoS defense system, and it would be highly undesirable if attackers were able to leverage it to deny service to legitimate traffic. Since D-WARD examines traffic on a connection granularity, an attacker who can spoof a currently active connection or soon-to-be-active connection can:

- "Smuggle" its packets among legitimate packets, if he is not sending aggressively, or
- Deny service to legitimate packets if his aggressive traffic leads to rate-limiting and classification of a given connection as bad.

TCP and ICMP connections could not be spoofed before they were set up because the victim would not respond correctly. UDP connections are more problematic, because a source can decide to send large amounts of data without a destination requesting it. An attacker could hijack legitimate connections from the source network to a given destination port running a UDP application by spoofing traffic to this port with a source address of a legitimate client. If the attacker generates an excess amount of UDP traffic, the attack would be detected and the connection would be classified as bad. Thus, the legitimate traffic to the victim on this connection would be subject to rate-limiting and some legitimate packets would be dropped. The hijacking of connections is possible in networks today and D-WARD does not offer any feature to make this easier for the attacker. Additionally, the hijacked connection is likely to lead to interrupted communication between the legitimate client and the attacker, since inserted bogus packets confuse the end hosts and lead one side to close the connection. The possible legitimate packet drops due to rate-limiting only speed up this process.

D-WARD operation is based on statistics and classifications stored in flow and connection tables. Since these tables are of limited capacity, they could overflow if the attacker generates a large amount of spoofed packets. D-WARD performs emergency clean-up of tables in case of overflow. We expect that flow table overflow would not affect the system performance, whereas connection table overflow would lead to degraded service to those legitimate connections that have been expelled from the table. Better emergency cleanup strategies would amend

this.

An attacker could perform a denial-of-service attack on the source network by preventing the response packets from reaching the D-WARD system. Seeing the reduced number of response packets, D-WARD could reach the conclusion that the source network is generating a DDoS attack and place a rate limit on outgoing flows. Thus the attacker denies the outgoing bandwidth to legitimate clients from the source network. There are two aspects to be noted here:

- Well-behaved flows will back off themselves in the absence of response packets, which will lead to their classification as "good," so they will not be affected by the rate limit.

- In the case where most reverse traffic does not reach the source network, legitimate communication from the source network is difficult or impossible. The reverse traffic will also confuse most protocols, and they will probably reduce their sending rates or shut down the connections entirely. Fake replies cause the same problem in today's networks, so adding D-WARD preserves the status quo.

D-WARD's deployment should not introduce additional security problems into the network. While D-WARD operation could lead to a few legitimate packet drops during some attacks (hijacked connections, connection table overflow and DoS on the D-WARD router), its good detection and fast response should make the benefits outweigh the costs.

6.9 Deployment

D-WARD faces two deployment challenges: (1) motivation for deployment and (2) asymmetric traffic. We discuss each of those challenges in the following sections and offer possible solutions.

6.9.1 Deployment Motivation

In addition to stopping outgoing DDoS attacks, D-WARD provides better service to a source network's legitimate clients during an attack. Thus, D-WARD brings direct benefit to the source network, in addition to relieving the victim of the attack traffic and improving Internet security. We expect that this will be a major motivation for deployment.

Cooperation formed the basis of the original ARPANET, and this is still seen in the Internet protocols of today. Different sessions of TCP, for instance, will fairly share bandwidth on a link, and new protocols are often judged on whether they are "TCP friendly." D-WARD operates in much the same way by protecting against DDoS attacks originating from a D-WARD protected network. The site deploying D-WARD benefits by not losing bandwidth to outgoing attacks (for sites who pay by usage, this could represent a significant benefit) and not having to deal with the social implications of hosting DDoS slaves.

Currently, the only results of unknowingly hosting a DDoS attack are annoying calls to system administrators and misuse of resources. In the future, it is possible that contracted or legislative action will hold those who do not take reasonable steps to secure their system liable for damages inflicted by attacks that misuse their machines. In this case, a corporation deploying D-WARD could argue that it had followed current security practices and therefore cannot be held

liable if DDoS attacks originate on its networks.

Many people have concluded that stopping attacks completely is impossible, since there is a vast number of machines whose owners are unaware of security holes or unwilling to fix them. For example, despite the best efforts to eradicate worms like Code Red [CERc] and Nimda [CERb], these still control a massive number of machines on the Internet. D-WARD brings this problem to a level of ISP or stub networks. Their administrators are likely to be security-conscious, and a single D-WARD system installed at their exit router would prevent DDoS attacks from the whole network.

D-WARD is unlikely to completely handle the DDoS threat in sparse deployment. If an attacker compromises only machines from networks not protected by D-WARD, their attack traffic will reach the victim unimpeded. Legitimate clients from D-WARD protected networks will not receive any benefit from D-WARD's operation during this attack, but will not suffer any harm either. However, if D-WARD is incorporated into a distributed defense system, such as DefCOM, it will significantly improve the quality of service perceived by legitimate clients even in sparse deployment. More details are given in Section 10.2.

6.9.2 Asymmetric Traffic

If a source network has several gateway routers (some or all of which are coupled with D-WARD) it may happen that some flow and connection traffic will exhibit asymmetric behavior, traversing different gateways in incoming and outgoing directions. This will cause problems to D-WARD's statistic gathering, creating incomplete observations and resulting in misclassifications. There are two possible solution to this problem:

1. D-WARD instances at different gateways communicate to exchange traffic statistics prior to classification. This would enable each D-WARD system to form a complete view of the traffic, but would create a lot of communication overhead as statistics would have to be exchanged prior to every classification. Alternately, classification periods (*Flow Observation Interval* and *Connection Observation Interval*) could be prolonged, thus reducing communication overhead but increasing detection delay and classification inaccuracy.

2. D-WARD instances could be installed within the source network at connection points between stub subnetworks and the rest of the source network. Thus D-WARD would have a complete view of the subnetwork traffic and police it appropriately. One such deployment is illustrated in Figure 6.16.

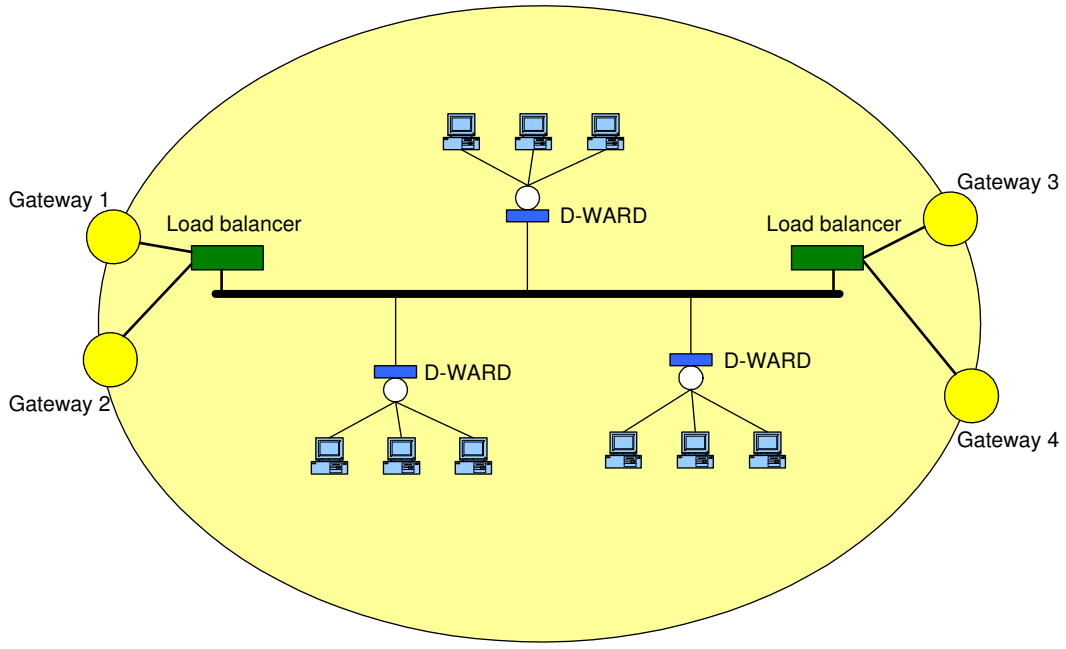

Figure 6.16: Deployment of D-WARD within source network in asymmetric traffic case

CHAPTER 7

Linux Router Implementation

D-WARD is implemented in a Linux router and in an IXP router. The Linux router implementation was done as a part of this thesis, whereas the IXP router implementation was done as a part of Gregory Prier's master's thesis [Pri03].

The Linux router implementation consists of two parts:

1. The user-level implementation of the monitoring and rate-limiting components
2. The loadable kernel module implementation of a traffic-policing component

This division of functionalities between user and kernel levels emerged after two prior versions of the D-WARD code had been modified to achieve better performance and greater ease of deployment.

The traffic-policing component decides whether to forward or drop packets in a unique manner. It could not have been implemented using Linux firewall functionality, so it was written from scratch. The customized implementation was also beneficial for evaluation of numerous traffic shaping approaches. The observation and rate-limiting components were also written from scratch.

D-WARD 1.0 was implemented purely at the user level. Since the traffic-policing component needs direct access to packets being relayed by the router to execute forwarding or dropping decisions, packets needed to be captured as they

pass through the kernel and copied to the user level. This was performed using the *IP_queue* functionality in the Linux kernel and userspace packet queuing.

Netfilter provides a mechanism for passing packets out of the stack for queueing to userspace, then receiving these packets back into the kernel with a verdict specifying what to do with the packets (such as ACCEPT or DROP). These packets may also be modified in userspace prior to reinjection back into the kernel. For each supported protocol, a kernel module called a queue handler may register with netfilter to perform the mechanics of passing packets to and from userspace. The standard queue handler for IPv4 is *ip_queue*. It is provided as an experimental module with 2.4 kernels, and uses a *Netlink* socket for kernel/userspace communication.

Once *ip_queue* is loaded, IP packets may be selected with *iptables* and queued for userspace processing via the `QUEUE` target. For example, running the following commands:

```
# modprobe iptable_filter
# modprobe ip_queue
# iptables -A OUTPUT -p icmp -j QUEUE
```

will cause any locally generated ICMP packets (e.g., ping output) to be sent to the textitip_queue module, which will then attempt to deliver the packets to a userspace application. If no userspace application is waiting, the packets will be dropped. An application may receive and process these packets via *libipq* library.

Once processed (examined and/or modified), packets are returned to the kernel space along with the decision: `NF_ACCEPT` to accept the packet or `NF_DROP` to silently drop the packet. More information on *IP_queue* functionality can be obtained by looking at *libipq* man pages.

IP_queue provides all the functionality necessary for the traffic-policing com-

ponent. However, copying packets (header and data) to userspace adds a large overhead that becomes critical as the packet rate increases. D-WARD 1.0 could only handle up to 1000 packets per second, thus prohibiting realistic DDoS attack tests. The decision was thus reached to move the traffic-policing functionality into the kernel space.

D-WARD 2.0 implemented the traffic-policing component inside the kernel and the observation and rate-limiting components at the user level. Communication between the two parts was achieved using Linux system calls.

The rate-limiting component was implemented as a loadable Kernel module. At the kernel level, packet capture was achieved using *netfilter hooks*. Netfilter [net] is a framework for packet mangling, outside the normal Berkeley socket interface. Each packet handing protocol (such as IPv4 or IPv6) defines "hooks" which are well-defined points in a packet's traversal of that protocol stack. As a packet reaches each of these points, the protocol will call the netfilter framework with the packet and the hook number. Kernel modules can register to listen to the different hooks for each protocol. When a packet is passed to the netfilter framework (upon encountering one of the hooks), netfilter checks to see if anyone has registered for that protocol and hook. If so, each registered kernel module gets a chance to examine (and possibly alter) the packet in the order in which they have registered the hook. Upon processing, the kernel module can discard the packet (returning NF_DROP to the framework), allow it to pass (returning `NF_ACCEPT`), tell netfilter to forget about the packet (returning `NF_STOLEN`), ask netfilter to queue the packet for userspace (returning `NF_QUEUE`) or repeat this hook (returning `NF_REPEAT`). Part of this functionality was used in D-WARD 1.0 through *iptables* rules to intercept packets and send them to userspace.

User level modules (observation and rate-limiting component) delivered the

Legitimate Connection List and *Rate Limit Rules* to the kernel using newly added system calls. The observation module received packet statistics using a modified *tcpdump* code. *tcpdump* uses the Berkeley Packet Filter (BPF) utility and *pcap* library to capture the header (and partially the content) of packets matching the given filter expression. The observation component modified the code so that the information is not printed but instead input into the statistics processing function.

D-WARD 2.0 could handle high packet rates (up to 10000 packets per second) but it had the following limitations. The *libpcap* utility copies packet header (and content) on a per-packet basis. This incurs overhead that becomes critical as packet rates increase. At packet rates higher than 1000 packets *libpcap* was unable to capture information on all passing packets. While D-WARD was still operational, there was a possibility that missing information would contain legitimate packet information and thus affect D-WARD performance. The other limitation was the ease of deployment. As D-WARD 2.0 added new system calls to the Linux kernel, it required kernel modification for deployment — a difficult undertaking that can frequently get very long and complicated. Furthermore, addition of new system calls or modification of old ones is a security hole in the Linux kernel and is forbidden in some releases.

D-WARD 3.0 implemented the traffic-policing component inside the kernel and the observation and rate-limiting components at the user level, just like D-WARD 2.0. However, communication between the two parts was achieved using *ioctl* calls. This enabled easy two-way communication between user and kernel processes and a seamless installation process. Installation requires that two devices be created:

1. The **dward** device is used for two-way communication between the obser-

vation and rate-limiting components on one side, and the traffic-policing component on the other, and

2. The **sniff** device is used for sniffing packet information and delivering it to the observation component.

The kernel processes then gather necessary information and store it, while the observation and rate-limiting components perform *ioctl* calls to pull this information when needed. The following sections will provide a detailed description of this architecture. The addition of customized packet-sniffing code enables efficient statistics gathering and copying to userspace, thus providing support for high packet rates.

D-WARD 3.1 adds two important features that affect system performance:

1. **Initial sequence number prediction.** None of versions **1.0**, **2.0** and **3.0** included *value prediction technique*. Those versions were thus dropping new connection packets during the attack, and their performance depended directly on frequency of connection initiation and connection length. To correct this problem, the *initial sequence number prediction technique* was developed and implemented in **D-WARD 3.1**, which is the current version of D-WARD code.

2. **Legitimate UDP connection models.** Legitimate UDP connection models are implemented only in **D-WARD 3.1**. The earlier versions may thus inflict collateral damage on legitimate UDP traffic.

The following sections provide more details about the **D-WARD 3.1** version.

7.1 Architecture

The architecture of the D-WARD 3.1 implementation is shown in Figure 7.1. The ovals framed with the solid line depict parallel processes in D-WARD. The ovals framed with the dotted line depict devices created for communication between user and kernel processes. Dashed lines with arrows illustrate *ioctl* calls that facilitate information exchange between user and kernel processes via created devices. Small squares at the bottom of ovals depicting kernel processes indicate the order in which those processes have registered to listen to the forwarding netfilter hook. Thus, the packets pass (and are handled) first by the **gst** module and then by the **rl** module. Dashed rectangles show grouping of processes and data within the three D-WARD components: observation, rate-limiting and traffic-policing.

7.2 Observation Component

The observation component stores flow and connection statistics in hash tables, for fast access.

7.2.1 Flow Hash Table

The flow hash table is indexed by the IP address of the foreign destination and contains the following fields:

- number of sent and received packets per protocol (TCP, ICMP, UDP, other and total)
- number of sent and received bytes per protocol (TCP, ICMP, UDP, other and total)

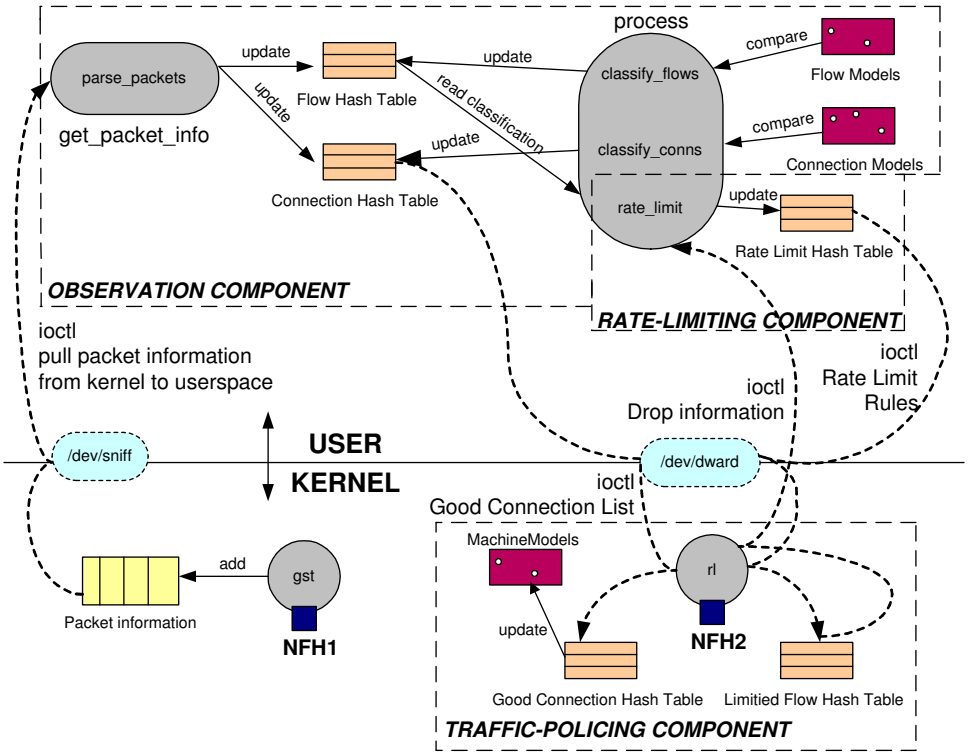

Figure 7.1: Architecture of D-WARD 3.1 implementation

- number of dropped bytes
- smoothed TCP packet ratio (number of sent divided by number of received packets)
- smoothed ICMP packet ratio (number of sent divided by number of received packets)
- timestamp of the last reset
- timestamp of the last activity (a packet sent or received)
- flow classification
- number of active connections for UDP protocol
- minimum value of non-validated traffic per protocol (TCP and ICMP only) in a given window
- number of recent consecutive suspicious classifications

These fields are depicted in Figure 7.2 (shaded fields are not present in the structure, e.g., active connection count is kept only for UDP statistics).

7.2.2 Connection Hash Table

The connection hash table is indexed by the key containing IP addresses and ports of the local and foreign hosts. A connection table record is depicted in Figure 7.3 and contains the following fields:

- number of sent and received packets
- number of sent and received bytes

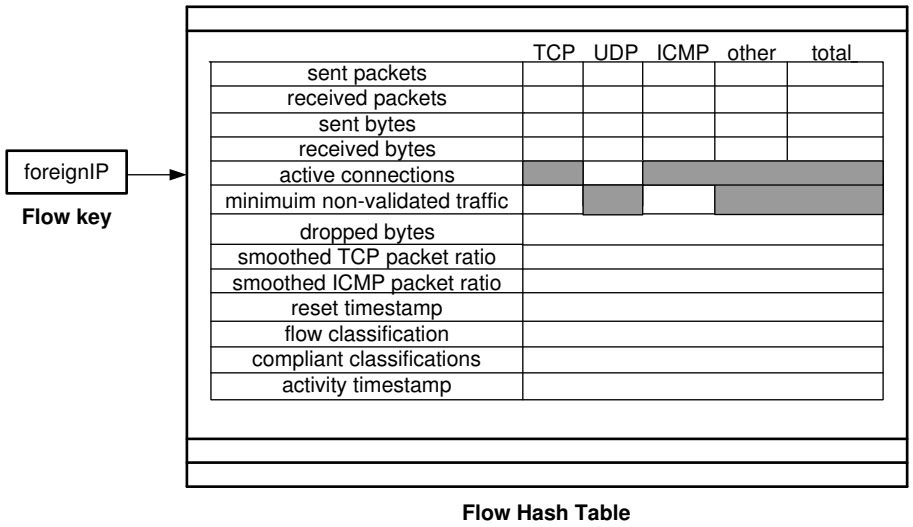

Figure 7.2: Flow table record

- connection protocol (TCP, UDP, ICMP or other)
- timestamp of the last reset
- timestamp of the last activity (a packet sent or received)
- connection classification
- application state (for UDP connections)
- application state expiry
- flag denoting whether this is a new connection

As the hash table size is much smaller than the number of possible entries, some entries will collide on insertion. The hash tables use *double hashing* to reduce the probability of collisions. Double hashing uses two functions: $h_1(x)$ and $h_2(x)$ to calculate the index for a given key x. The first index that will be accessed is calculated as: $index = h_1(x)$. If the collision occurs, subsequent

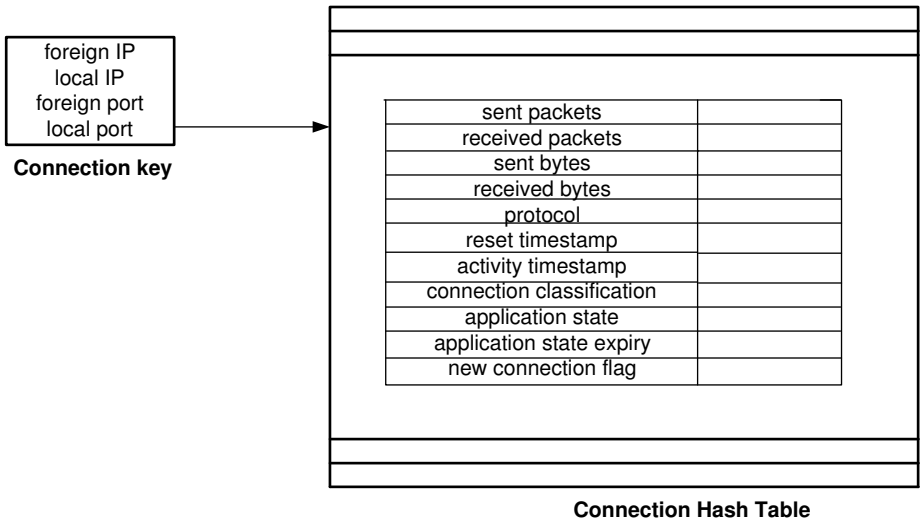

Figure 7.3: Connection table record

indexes will be calculated as $index = h_1(x) + trial * h_2(x)$, where trial denotes the number of collisions encountered so far. The idea of this approach is that even if two items hash to the same value of h_1, they will have different values of h_2, so that different probe sequences will be followed, resulting in better hash occupancy.

Flow hash indexes are calculated using the following hash functions:

$$\begin{aligned} k &= foreignIP \\ h_1(k) &= k \,\%\, size \\ h_2(k) &= 1 + (k \,\%\, (size - 1)) \end{aligned} \tag{7.1}$$

Connection hash indexes are calculated using the following hash functions:

$$\begin{aligned} k &= foreignIP + localIP + foreignport + localport \\ h_1(k) &= k \,\%\, size \\ h_2(k) &= 1 + (k \,\%\, (size - 1)) \end{aligned} \tag{7.2}$$

To avoid an extensive search for the item in the hash tables (or for the empty slot to insert the item) the number of trials was limited to three. If after three trials an item (or empty slot) is not found, the search (or insertion) is aborted. This optimization significantly increases efficiency while not inflicting large information loss. As hash accesses occur at each packet passing, reducing hash access overhead greatly improves performance.

Hash overflows are checked at record insertion time. If the hash is close to being full (90% full), an emergency cleanup will be invoked to free space for new records. During emergency cleanup, each record is examined to determine whether it can be deleted. If, after the first pass, the hash occupancy is not reduced below $Rehash\_load\_factor\_high$ (this is a configuration parameter, usually set at 75%), subsequent passes are initiated using laxer criteria for record deletion. The following criteria is used to decide whether a record should be deleted:

- For a flow record: delete the record if the number of sent packets is less than SP **and** the number of sent bytes is less than SB. SP and SB are doubled in every pass.
- For a connection record: delete the record if the number of sent packets is less than SP **and** the number of sent bytes is less than SB **and** the connection is classified as transient. SP and SB are doubled in every pass.

For performance reasons, no rehashing is allowed, and dynamic memory allocation is deployed only when necessary.

7.2.3 Obtaining Packet Information

The process **get_packet_info** continuously requests packet information from the kernel module **gst**. However, for performance reasons, the incidence of copying the data between kernel and userspace should be minimized. Therefore, the packet information is only copied when the kernel buffer is more than one-third full or after a number of copy requests (a value of 1000 has been chosen empirically) have been refused. When **get_packet_info** receives packet data it parses information in each packet and updates corresponding entries in the flow and connection hash tables.

7.2.4 Classifying Flows and Connections

The function **process** periodically classifies flows and connections and calls the rate limit function to determine the appropriate rate limits. A flow will be classified as an attack if at least one of the following conditions is met:

- Its smoothed TCP ratio is greater than TCP_{rto}.
- Its smoothed ICMP ratio is greater than $ICMP_{rto}$.
- The number of UDP connections is greater than n_{conn} **and** the ratio of number of UDP packets sent over number of UDP connections is lower than p_{conn}.
- Non-validated residue detection is used and the minimum value of non-validated TCP traffic is greater than zero.
- Non-validated residue detection is used and the minimum value of non-validated ICMP traffic is greater than zero.

If none of the above conditions is met, the flow will be classified as suspicious if its number of consecutive compliant classifications is lower than *Compliance Period* or the amount of dropped bytes is greater than zero. Otherwise the flow will be classified as normal.

A connection will be classified as transient if: (1) it is a TCP connection and less than 3 packets were sent from its setup, or (2) it is an ICMP connection and less than 2 packets were sent from its setup, or (3) it is a UDP connection and it has no application-level models. A connection will be classified as good if exactly one of the following conditions is met:

- It is a TCP connection and its TCP ratio (ratio of number of packets sent over number of packets received) is greater than TCP_{rto}.
- It is an ICMP connection and its ICMP ratio is greater than $ICMP_{rto}$. Note that legitimate ICMP connections will be used for non-validated residue detection but will not be inserted into *Legitimate Connection List*.
- It is a UDP connection **and** it has an application level model **and** it is in a valid (non-error) state.

Otherwise, a connection will be classified as bad. All TCP and UDP connections that have been classified as legitimate are inserted into the *Legitimate Connection List* that is delivered to kernel module **rl** through *ioctl* calls.

After connections have been classified, connection packet and byte statistics are reset. The connection is further checked for inactivity by comparing its last activity timestamp with $Good\_Inactive\_Period$ for good connections, or $Transient\_Inactive\_Period$ for transient connections. Bad connections are not tested for inactivity as they will be eventually classified as good (and shortly

thereafter inactive) once they have stopped participating in the attack. Inactive connections will be deleted.

7.3 Rate-Limiting Component

The rate-limiting component is invoked from the **process** function, after flows and connections have been classified. It obtains flow drop information from the **rl** module using *ioctl* calls and defines the appropriate flow rate limits according to formulas 6.2, 6.3, 6.4, and 6.5. Rate-limited flows are inserted into the **Rate limit hash table** which is indexed by the IP address of the foreign destination and only contains information about the current rate limit. The contents of this table are then delivered to the **rl** module using *ioctl* calls.

After the rate limit has been defined, flow packet and byte statistics are reset. A flow is further checked for inactivity by comparing the last activity timestamp with $Flow\_Inactive\_Period$ for normal flows. Information on attack and suspicious flows is not tested for inactivity as they will be eventually classified as normal after being inactive. Inactive flows will be deleted.

7.4 Traffic-Policing Component

Traffic-policing is implemented within the **rl** module, using netfilter hooks. The **rl** module stores information about rate-limited flows in its **Limited flow hash table** and information about good connections in its **Good connection hash table**. Those hash tables are organized similarly to the flow and connection tables within the observation component. A limited flow hash record is depicted in Figure 7.4 and contains the following fields:

sent Bytes	
sent good Bytes	
sent predicted Bytes	
estimated good traffic load	
dropped Bytes	
rate limit	

Figure 7.4: Limited flow table record

- number of bytes sent
- number of bytes sent within good connections
- number of bytes sent because they matched the predicted sequence number range
- number of bytes dropped
- current rate limit
- estimated good traffic load

A good connection hash record does not contain any additional data — just the connection key.

7.4.1 Traffic-Policing Process

Traffic-policing is performed on each outgoing packet in the following manner:

1. If packet is randomly spoofed (its address is not within *Police Address Set*), drop it.
2. If the associated flow is not in the *Limited Flow Hash Table*, forward the packet.

3. If the associated connection is in the *Good Connection Hash Table*, forward the packet and update *sent good bytes* within the associated flow record in *Limited Flow Hash Table*.

4. If the packet is TCP, it matches the predicted sequence number range and the sum of *sent predicted bytes*, and packet length is not greater than *Early Packet Rate Limit*, forward the packet and update *sent predicted bytes*.

5. If the sum of *sent bytes* and the packet length is not greater than the rate limit **and** there is still enough space for the estimated good traffic load within the rate limit, forward the packet and update *sent bytes*. The estimated good traffic load is calculated as the maximum value of *sent good bytes* on the associated flow, exponentially aged.

Packet statistics are reset every second. Figure 7.5 depicts the traffic policing process.

7.4.2 Machine Models

As discussed in Section 6.4.3, D-WARD performs sequence number range prediction for TCP connections. To predict the sequence number range, a model is built for each address from the police address set. This model contains the sequence number information from two recently established TCP connections — s_1 and s_2, along with their timestamps t_1 and t_2 and the machine classification. The classification is performed by calculating a prediction: $sp = predict(s_1, t_1, t_2)$ and validating sp against s_2 for different operating system models.

The **Windows OS model** expresses the sequence number dependency as:

$$sp = s_1 + (t_2 - t_1) * WIN\_FACTOR + WIN\_CONSTANT \qquad (7.3)$$

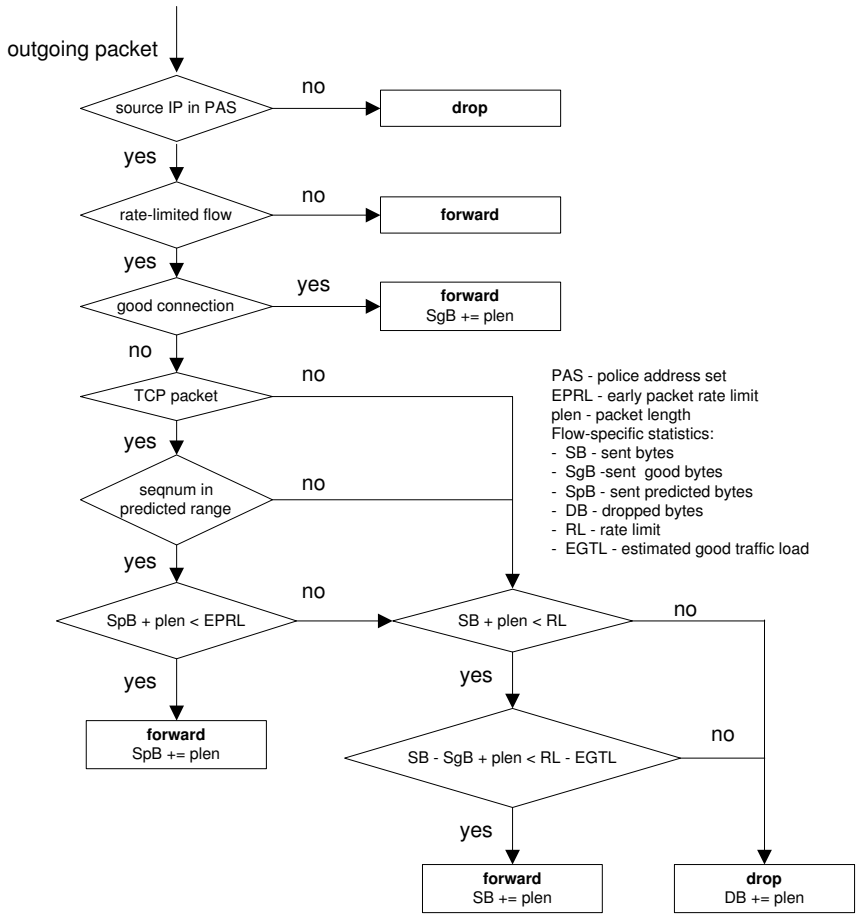

Figure 7.5: Traffic-policing process

If sp is within WIN_ERROR_MARGIN from s_2, the machine is classified as Windows. The Windows model is more stringent than the Linux model, so this classification is performed first.

The **Linux OS model** expresses the sequence number dependency as:

$$sp = s_1 + (t_2 - t_1) * LIN\_FACTOR + LIN\_CONSTANT \tag{7.4}$$

If sp is within LIN_ERROR_MARGIN from s_2, the machine is classified as Linux.

If the machine does not pass either the Windows or the Linux OS test, the machine is then classified as "other", and no prediction can be done.

The validation process for TCP packets during rate-limiting is the same as the validation process for machine classification. The recent sequence number information (s_1 and s_2) is updated only from connections inserted into the **good connection hash table**.

7.5 Traffic-Sniffing

Traffic-sniffing is implemented within the **gst** module, using netfilter hooks. The **gst** module simply collects a specified number of bytes from the packet (header and data) and stores them in a buffer. If the buffer overflows, the subsequent packets will be passed without information gathering.

CHAPTER 8

Experiment Setup

To capture the variety of possible attacks and to fairly evaluate the D-WARD system, we must allow variations in a number of different dimensions: (1) test topology, (2) background workload of legitimate traffic, and (3) attack characteristics. We also must define metrics that will be used to measure system performance. The following sections give more details about the test methodology and the chosen metrics.

8.1 Environment

The test topologies used in experiments must resemble realistic DDoS network topology. This requirement is hard to meet as the scale of a DDoS network is too large to be recreated in a testbed. In order to test D-WARD with topologies as large as possible, the Emulab [WLS02] test facility was chosen for the experiments.

Emulab started as a moderate-scale (just under 300) isolated testbed consisting of real machines at University of Utah that could be shared among researchers. It has recently been augmented to include simulated and wide-area nodes that are all seamlessly integrated into one test framework. D-WARD tests use only real machines.

The Emulab real-machine testbed is a universally-available time- and space-

shared network emulator. Several hundred PCs in racks, combined with secure, user-friendly, web-based tools, and driven by ns-compatible scripts or a Java GUI, allow users to remotely configure and control machines and links down to the hardware level. Packet loss, latency, bandwidth, queue sizes — all can be user-defined. Even the OS disk contents can be fully and securely replaced with custom images by any experimenter.

University of Utah's local installation currently features high-speed Cisco switches connecting five 100Mbit interfaces on each of 168 PCs. The University of Kentucky's installation contains 48 similarly networked PCs. The PC nodes can be used as edge nodes running arbitrary programs, simulated routers, traffic-shaping nodes, or traffic generators. While an "experiment" is running, the experiment (and its associated researchers) get exclusive use of the assigned machines, including root access.

8.2 Legitimate Traffic

The goal of DDoS defense mechanisms is to allow normal network operations to proceed despite DDoS attacks. Since the method of attack on those normal operations works on network traffic, proper evaluation of defense mechanisms requires capturing realistic characteristics of normal network traffic. Legitimate traffic during the experiments must be representative of legitimate traffic patterns one expects to encounter in real networks. To meet this requirement we designed a *trace reconstruction tool*, called *tracegen* that produces live legitimate traffic whose characteristics resemble those from a supplied *tcpdump* file. The trace reconstruction tool consists of a profiling program and a script generation program.

A profiling program is first invoked on the *tcpdump* file containing traffic traces, and supplied with the specification of local subnet prefix. The program profiles trace data on connection granularity (communication between two IP addresses and ports) observing only communications between hosts with addresses within local subnet prefix, and the outside world. The following information is extracted on each connection, and stored in the connection profile file:

- **Beginning time** — connection origination time, measured from the trace start
- **Duration**
- **Type** — TCP, UDP or ICMP
- **Application** — Telnet, FTP, DNS or NTP. TCP connections are observed as generating either Telnet-like or FTP-like traffic, based on their dynamics. During Telnet-like connections, connection data is exchanged in small-size packets over a long time period (more than 5 seconds), whereas during FTP-like connections, connection data is exchanged in bursts over a short time period (less than 5 seconds). UDP connections are classified as DNS, NTP or other. In the tests, only DNS and NTP traffic are reconstructed.
- **Total amount of traffic sent from the server to the client** — each connection is observed as communication between a server and a client. Only server-side data will be reconstructed, and the client will issue live requests to obtain it.
- **Direction** — within the connection, assignment of server/client roles is done based on the amount of traffic sent from one party to the other. The party that sends a larger amount of data to its peer is considered to be a

server. Based on its IP address and local subnet prefix, we can determine whether this is a local or a foreign server. If the server is a local host, the connection will have `IN` direction. This means that it will be initiated from the foreign client. Otherwise, the connection will have `OUT` direction, and it will be initiated from the local client.

In addition to TCP, ICMP, NTP and DNS traffic models, D-WARD also builds streaming media traffic models. These models cannot be verified at Emulab, since Emulab machines do not have sound cards and cannot generate or receive streaming media traffic. Thus a separate set of tests will be performed to verify D-WARD's behavior towards streaming media traffic. Those tests will be run in our small-scale testbed in the LASR group at UCLA.

A script generation program takes as input:

- the connection profile file generated by the profiling program
- two files containing IP addresses of local and foreign hosts that will perform live trace reconstruction
- frequency of reconstruction N — this option allows for the possibility that only each Nth connection will be reconstructed if we cannot obtain sufficient resources to reconstruct each connection
- frequency of additional DNS queries
- frequency of additional NTP queries
- frequency of additional ICMP messages

The script generation program then parses the connection profile file and, based on connection direction, assigns each connection to one of the local (or foreign) hosts that is this connection's client in round-robin fashion. The connection

server is randomly chosen from the set of foreign (or local) hosts. If frequencies of additional messages are non-zero, then additional DNS, NTP or ICMP messages will be generated. DNS messages can be generated both as outside queries (from the foreign client to the local server) and as inside queries (from the local client to the foreign server). NTP and ICMP messages are only generated as inside queries. The output of the script generation program is a set of *bash* scripts that will initiate connections assigned to each of the local and foreign machines. In addition to this, the duration of each connection will be measured using the `time` command.

For telnet connection reconstruction, a custom program, `run_telnet`, is written that takes as input connection duration and amount of data to be transferred. This program is invoked by the connection's client remotely at the connection's server. The program calculates a unit of data to be transfered each second as ratio of total amount of traffic and connection duration. The program then flips a coin each second, randomly selecting the value from 0 to 3 times the calculated unit, and transfers this amount of data to the client. This approach creates pseudo-uniform communication similar in dynamics to telnet traffic.

For FTP connection reconstruction, a file of a size specified in the connection profile is generated at the server using a custom-written `gen_ftp` program. This file is then copied to the client during the test, thus reconstructing the given connection.

For DNS query reconstruction, a simple DNS network is set up in which all foreign hosts are authoritative servers for one namespace, and all local hosts are authoritative servers for another namespace. The `nslookup` command is then invoked with the `set server` option specifying the server to be queried. The command requests the address from the local or foreign namespace, according to

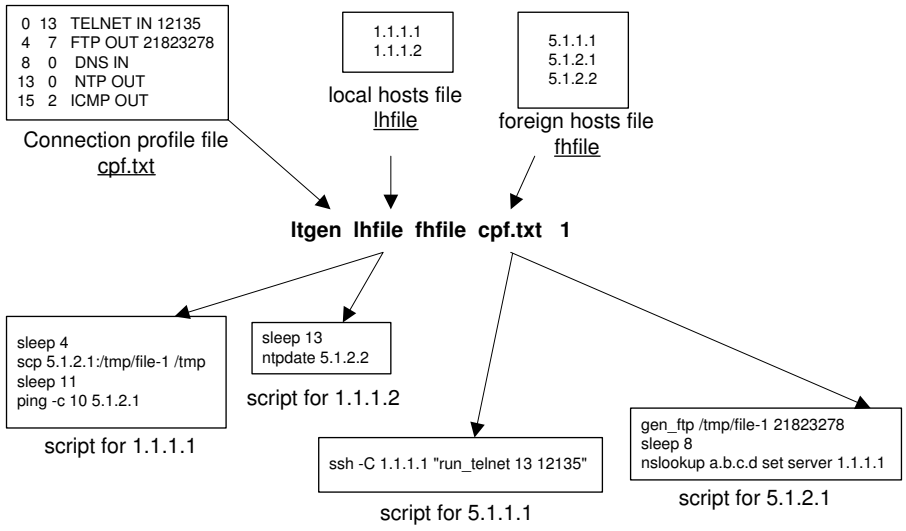

Figure 8.1: Sample connection profile and script files

connection direction.

For NTP query reconstruction, the `ntpdate` command is invoked at the local client specifying the server to be queried.

For ICMP query reconstruction, the `ping -c 10` command is invoked at the local client specifying the server to receive ten successive ICMP_ECHO requests.

Figure 8.1 depicts a sample connection profile file and script files.

We then select a five-minute long segment from a tcpdump file capturing traffic traces between the UCLA Computer Science Department and the rest of the Internet. Legitimate traffic from this segment will be reconstructed during test runs. One test run lasts for 300 seconds.

Figures 8.2 and 8.3 show the moving average of the amount of traffic sent and received (in packets and bits) in the chosen trace and in the reconstructed traffic. The solid line represents trace statistics and the dotted line represents the reconstructed traffic statistics. We see that all reconstructed traffic measures closely

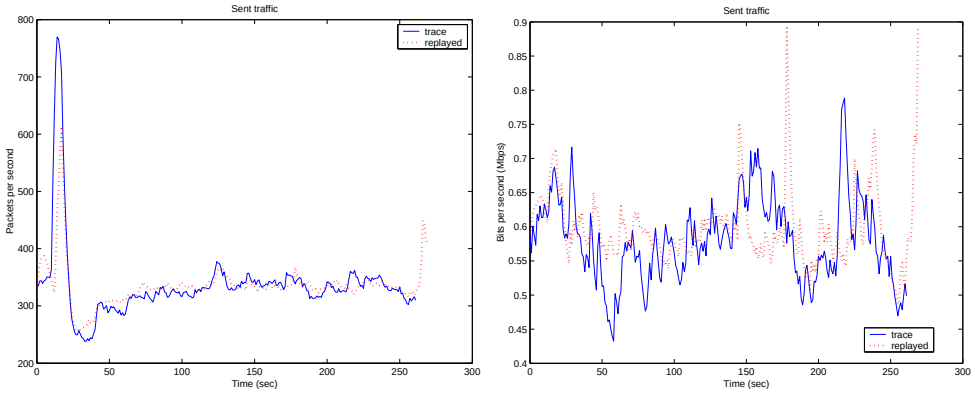

Figure 8.2: Sent traffic in packets and bytes

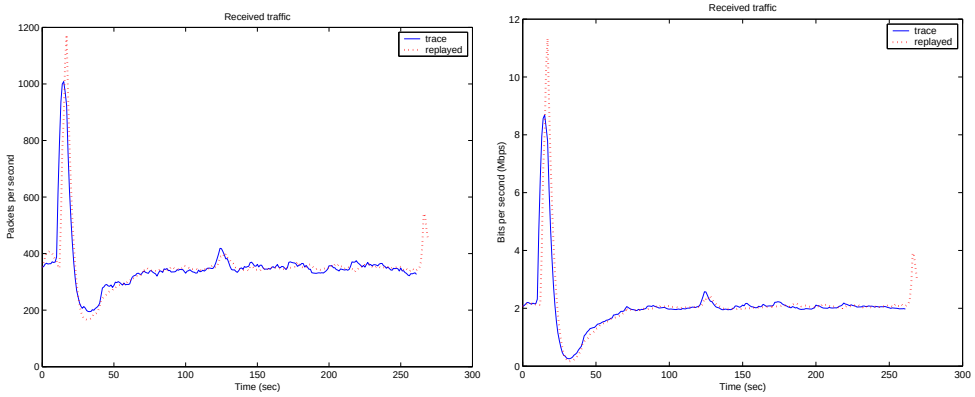

Figure 8.3: Received traffic in packets and bytes

resemble the measures in the real trace, which indicates that the reconstruction strategy generates realistic network traffic.

8.3 Attack Traffic

In order to test different attack scenarios, we developed a customizable DDoS attack tool, called **cleo**. It uses a master-slave architecture to coordinate attacks among multiple slaves. Attack traffic mixture (relative ratio of TCP SYN,

Parameter	Value	Description
Packet Type	*pt pu pi*	Mixture of TCP, UDP, and ICMP in relative ratios *pt*, *pu*, and *pi*
Dynamics	ON	Maximum rate
	ONOFF *po pf*	Maximum rate intervals of *po* seconds separated by intervals of *pf* seconds
	GON *go*	Maximum rate is achieved through a gradual increase during *go* seconds.
	GOFF *go po gf pf*	Maximum rate is achieved through a gradual increase during *go* seconds and it is held for *po* seconds. The rate is decreased gradually over *gf* seconds and remains zero for *pf* seconds.
Spoofing	NONE	No spoofing
	RANDOM	Random address is spoofed
	SUBNET prefix	Only addresses from specified prefix are spoofed
Packet size	*ps*	All attack packets have *ps* bytes
	VAR *pmin pmax*	Packet size is picked randomly between *pmin* and *pmax*
Target ports	RANDOM	Random ports are targeted
	FIX *portfile*	Only ports specified in *portfile* are targeted
Rate	*rate*	Maximum attack rate in Bps

Table 8.1: Attack tool customization options.

ICMP_ECHO and UDP packets), packet size, attack rate, target ports, spoofing techniques and attack dynamics can be customized. Table 8.1 gives the available choices for each of these parameters.

In the experiments, attacks will last 100 seconds, starting at 27 seconds and ending at 127 seconds from the beginning of the run.

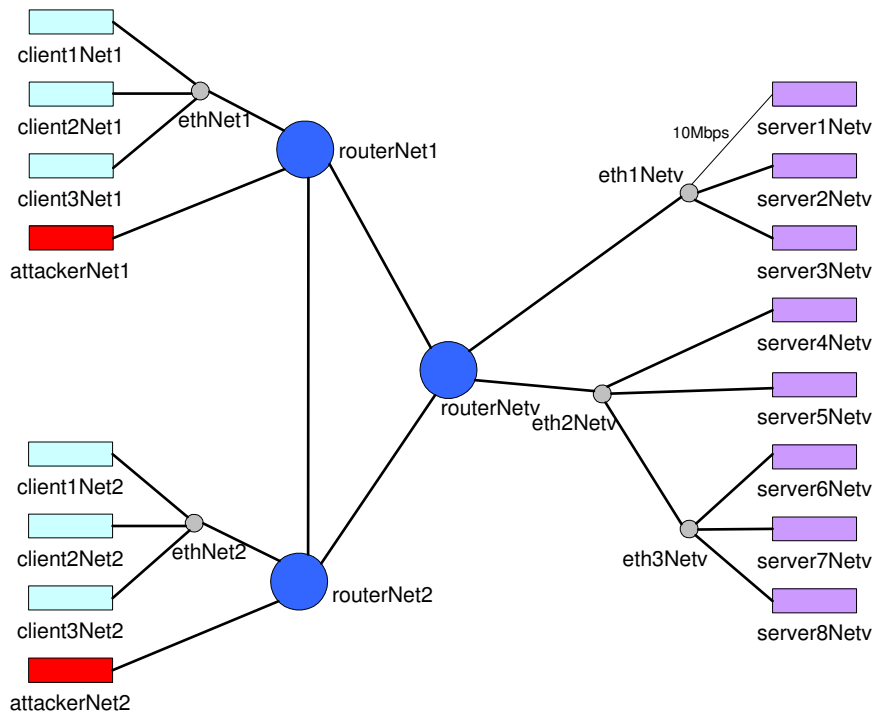

Figure 8.4: Topology 1 for basic D-WARD evaluation

8.4 Topologies

In order to obtain comparable results over various tests, we limited the variety of topologies used in the experiments. For basic D-WARD evaluation testing the performance of one D-WARD system, we use Topology 1, depicted in Figure 8.4.

The topology consists of 6 local hosts and 8 foreign hosts, which are sufficient to recreate the total traffic in the trace. Local hosts are divided into two source networks — *Net1* and *Net2* — for the following reasons:

1. In order to create a denial-of-service effect at the victim, a sufficiently strong (high volume) attack must be generated. If this attack is created only from one source network, the volume may be too large for the D-WARD system

to handle.

2. In order to reproduce the distributed nature of the attack, we must engage at least two attack machines

Each source network also hosts one attacker. Networks reach the outside world via routers. Source routers — *routerNet1* and *routerNet2* host D-WARD systems that are activated when necessary. As each machine has four available network interfaces (the fifth is used for control traffic), hosts are connected to routers in several stages using intermediate switches — *ethNet1*, *ethNet2*,*eth1Netv*, *eth2Netv* and *eth3Netv*. All links in the network are 100Mbps links, except the link between *eth1Netv* and *server1Netv* which is 10 Mbps. This link will be a critical resource that is targeted by some attacks.

For evaluation of stealthy attacks and interactions between D-WARD systems, we use Topology 2 depicted in Figure 8.5.

The topology consists of 10 local hosts and 10 attackers divided into 10 identical source networks — *Net1* through *Net10*. The victim (foreign) network is the same as in Topology 1. Source routers — *routerNet1* through *routerNet10*, host D-WARD systems that are activated when necessary. Intermediate switches — *hub1*, *hub2*, *hub3*, *hub4*, *eth1Netv*, *eth2Netv* and *eth3Netv*, are used to connect all nodes while satisfying the four network interface limit. All links in the network are 100Mbps links, except the link between *eth1Netv* and *server1Netv* which is 10 Mbps. This link will be a critical resource that is targeted by some attacks.

For evaluation of streaming media models, we use Topology 3 depicted in Figure 8.6.

As the Emulab machines do not have sound cards they cannot run streaming media applications and therefore cannot be used to test D-WARD streaming

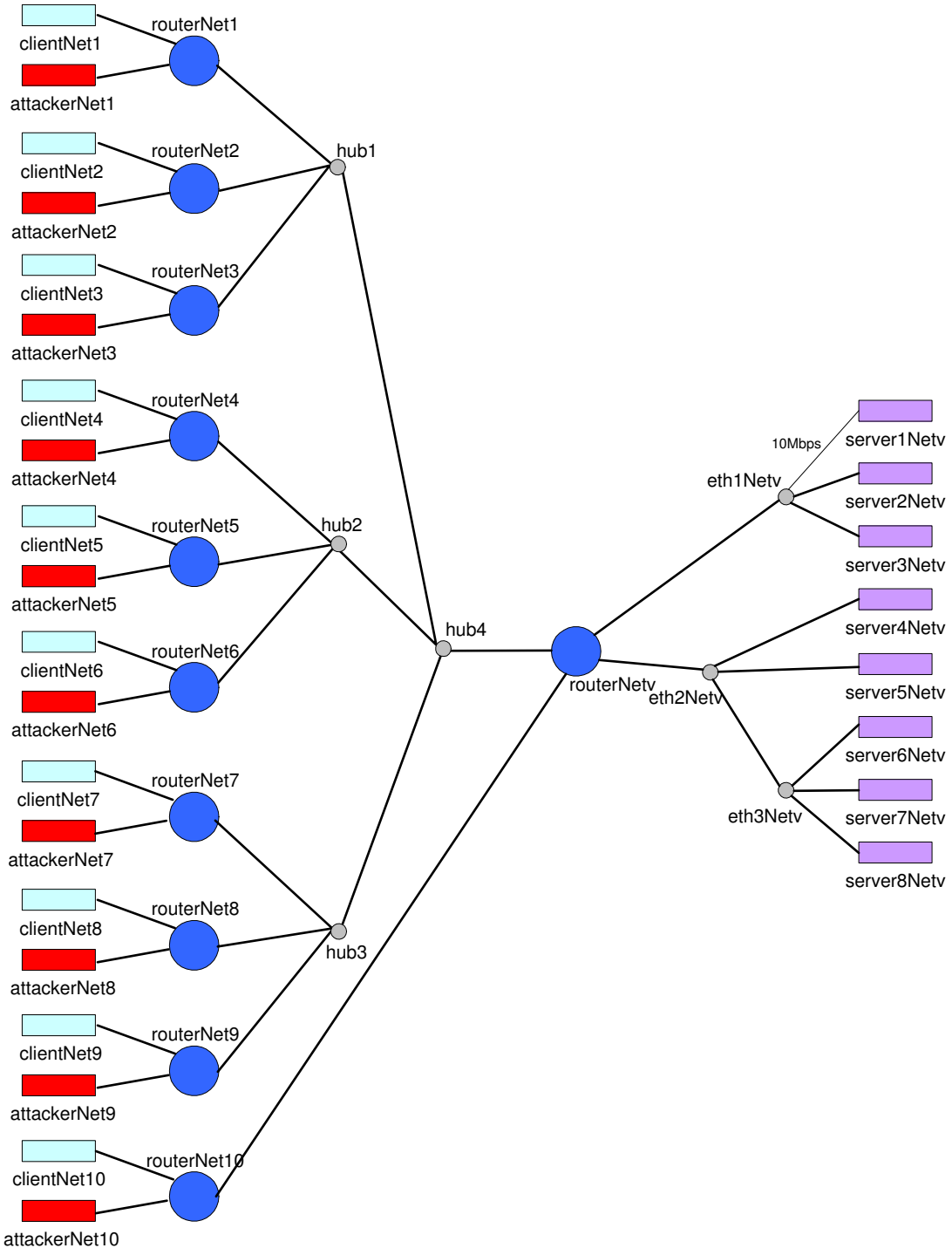

Figure 8.5: Topology 2 for D-WARD evaluation in large-scale network

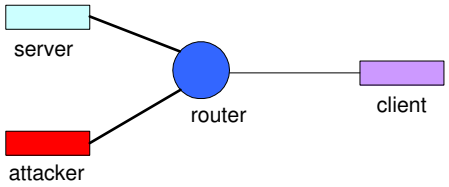

Figure 8.6: Topology 3 for evaluation of D-WARD streaming media models

media models. We tested these models in our testbed, at the UCLA Laboratory for Advanced Systems Research (LASR). Topology 3 consists of four machines: *client*, *attacker*, *router* and *server*. All links in the network are 100Mbps links, except the link between the *router* and the *client* which is 56 Kbps. This link will be a critical resource targeted by the attack. During the test run, legitimate traffic is generated by streaming a file in *Real Audio* format from the *server* (deploying a Helix Universal Server [Reaa] application) to the *client* (running Real Player 8[Reab]). During this communication, a high-rate, UDP attack is generated from the *attacker* to the *client*. The network hosting the *attacker* and the *server* is protected by D-WARD, residing on the *router*.

8.5 Metrics

Performance metrics used in evaluation of DDoS defense systems should clearly describe how well the system combats DDoS attacks. We use the following metrics to capture defense performance:

Legitimate Traffic Service Level. The ultimate goal of DDoS defense is to avoid the denial-of-service effect and provide good service to legitimate traffic during an attack. Thus the most important metric of success is the level of service that legitimate traffic receives during an attack. We calculate this level

as the total amount of legitimate traffic that reaches the victim while the attack is ongoing. This amount is expressed as a percentage of the amount received in the baseline case, when no attack is present.

Figure 8.7 depicts the meaning of this measure. Experiment time, in seconds, is given on x-axis while the total amount of legitimate traffic that the victim has received from the experiment's start until a given time is shown on the y-axis. The figure represents the baseline case, when only legitimate traffic is present and no defense is engaged, and the moderate-rate UDP attack case when both attack and legitimate traffic are present and no defense is engaged. The solid line represents the baseline case, and the dashed line represents a moderate-attack case. The denial-of-service effect manifests itself in the discrepancy between the dashed and the solid line, and is caused by the attack traffic taking over network resources from the legitimate traffic. When the attack subsides, legitimate clients' TCP mechanisms retransmit some quantity of the lost traffic, allowing for the dashed line to meet the solid line. Some connections will thus recover from the attack and experience the denial-of-service effect only as temporary loss of service and communication delay, while other connections will fail to complete their transfer until the experiment ends. In Figure 8.7 service level would be calculated as $\frac{a}{b}$.

Ideally the service level in the case where the defense is engaged should be 1. Care must be taken to include in the evaluation only those experiments that clearly create a denial-of-service effect when no defense is deployed, as only those represent a relevant baseline for comparison.

The denial-of-service effect is not deterministic, which produces a significant amount of variability in all measures. Figure 8.8 shows the amount of the legitimate traffic received by the victim in three experiment runs, featuring the same UDP flooding attack and same legitimate traffic. No defense is engaged in any of

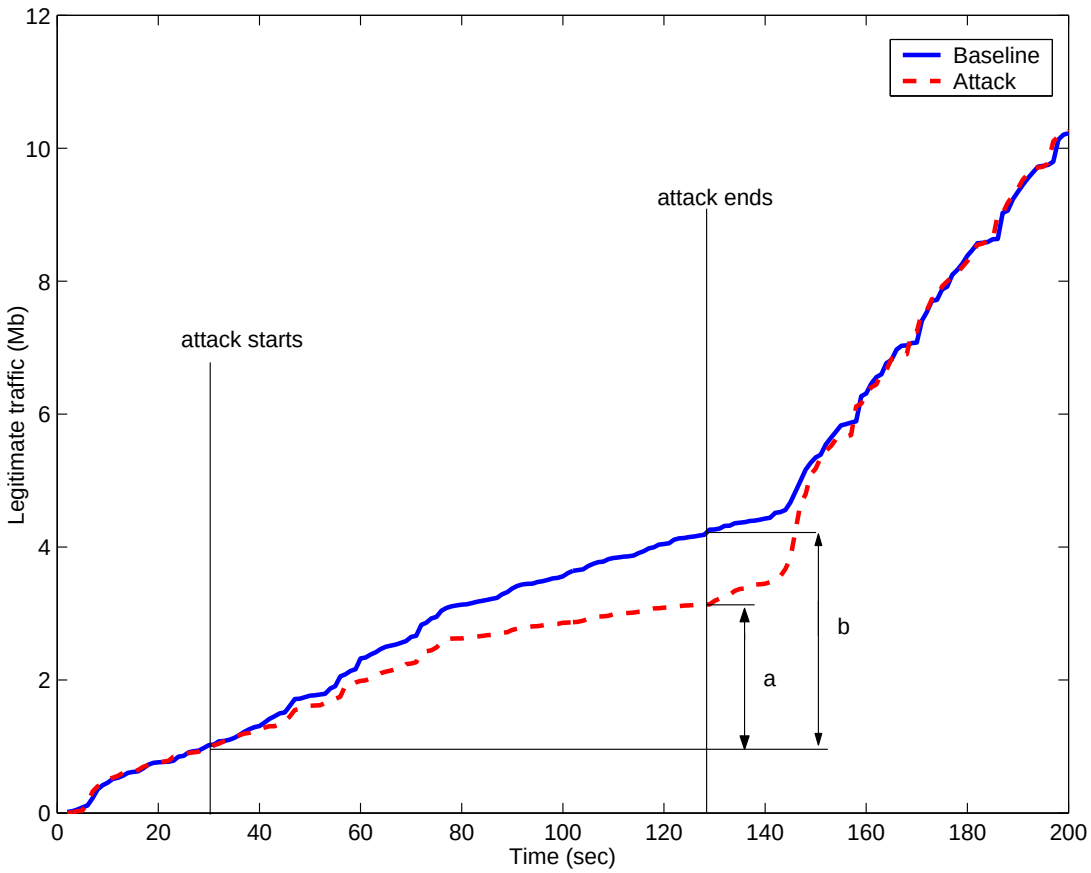

Figure 8.7: Service level calculation

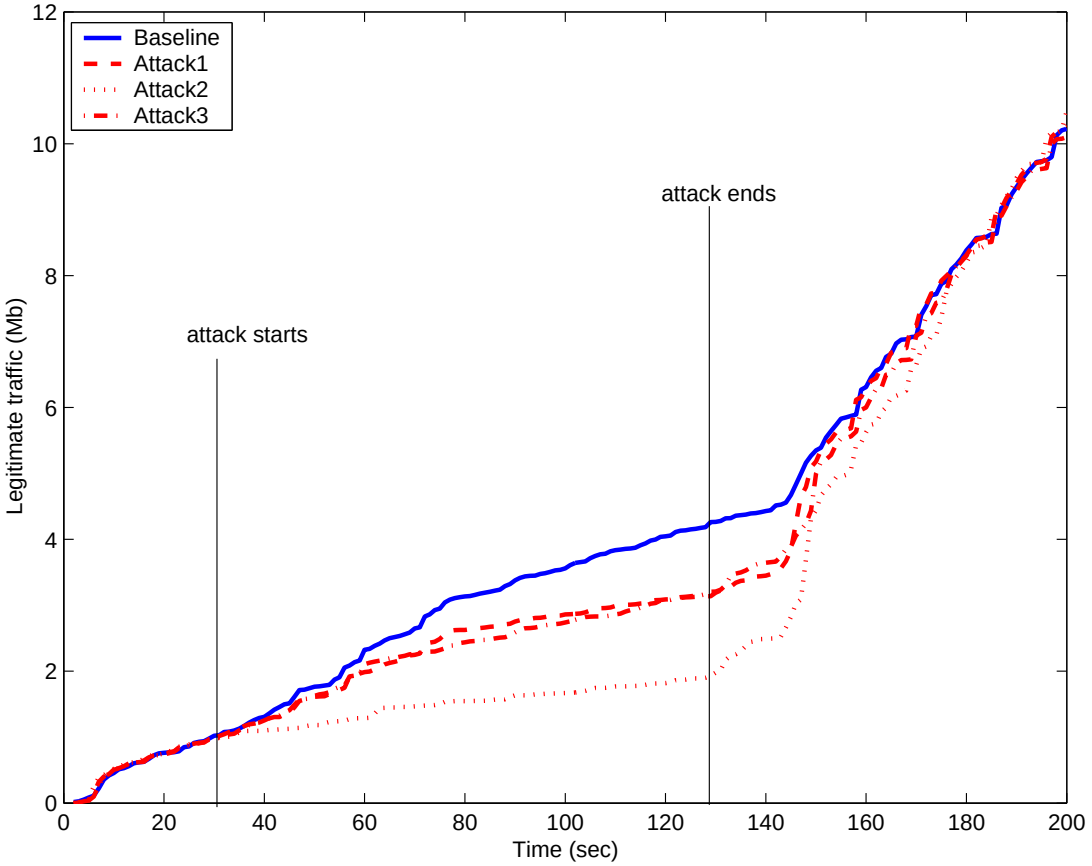

Figure 8.8: Variability of DoS effect

these runs. The variability stems from packet interactions in competition for a limited resource. Even though the same traffic is generated in each run, packets from various connections, interleaved with attack packets, arrive at the bottleneck link in different order. This leads to different packets being dropped in each run, thus affecting in different manners the TCP mechanism for each legitimate connection that has experienced packet drops.

To capture this variability, we ran each experiment ten times and presented each metric with boxplots. A boxplot produces a rectangular box encompassing the lower quartile, and upper quartile values. Median value is depicted with a

dot inside the box. Service level is shown on a 0 to 1 scale, where 0 means that no legitimate traffic is reaching the victim during the attack, and 1 means that all legitimate traffic is reaching the victim during the attack.

Connection Delay and Connection Failure. The amount of legitimate traffic delivered does not completely capture the attack effect on legitimate traffic, as some of the legitimate packets received during the attack may be retransmissions. Therefore, we complement the legitimate traffic service level measure with connection delay and connection failure metrics. These measures capture the user's experience of the attack.

Both the connection delay and connection failure measures are calculated only for those connections whose duration overlaps the attack. As connections originate independently from one another, only those connections that send their traffic while the attack is ongoing will experience adverse effects. Connection delay is given as set of three measures:

- **total delay,** representing the sum of all connection delays in the test run
- **maximum per connection delay,** representing the largest connection delay in the test run
- **median per connection delay,** representing the most-likely connection delay in the test run

Connection failure is expressed as number of failed connections in the test run. Each test run is repeated ten times and presented with boxplots.

In order to present comparable graphs over multiple experiments, we scale y-axis in the following manner. For total connection delay, y-axis runs from 0 to 5000 seconds. For maximum per connection delay, y-axis runs 0 to 300 seconds,

thus encompassing the duration of the test run. For median per connection delay y-axis runs 0 to 1 second. For number of failed connections, y-axis runs 0 to 100 connections.

Attack Detection Time. The important effectiveness measure is the time needed for attack detection. This time is measured from the start of the attack. Each test run is repeated ten times and presented with boxplots. We scale y-axis from 0 to 100 seconds, encompassing the duration of the attack.

Attack Response Time. Although the attack may be detected early, the defense becomes effective only when the attack traffic starts to be dropped. This metric is captured through attack response time, which is the time of the dropping of the first attack packet. This time is measured from the start of the attack. Each test run is repeated ten times and presented with boxplots. We scale y-axis from $0 - 100$ seconds encompassing the duration of the attack.

Attack Level. To express how well the defense controls the attack, we measure the total amount of attack traffic delivered to the victim. This amount is compared to the attack traffic amount that reaches the victim when defense is inactive, and is expressed as a percentage. Ideally this level should be 0. However, since typically the only bad effect of DDOS traffic is interfering with legitimate traffic, reducing the attack level to 0 is of secondary importance to increasing the legitimate traffic sevice level to 1. Each test run is repeated ten times and presented with boxplots. We scale y-axis from 0 to 100%.

Legitimate Drops. To express the selectiveness of the defense and measure collateral damage to legitimate traffic, we measure the total amount of legitimate traffic dropped by the defense system. This amount is compared to the legitimate traffic amount that reaches the victim during baseline runs and is expressed as a percentage. Ideally this measure should be 0. Each test run is repeated ten

times and presented with boxplots. We scale y-axis from 0 to 1%.

CHAPTER 9

Performance Results

D-WARD has many configurable parameters that guide its operation. Some of these parameters are set based on the examination of legitimate traffic seen in our network (such as tcp_{rto}), while others are set empirically to satisfy the trade-off between high effectiveness and low per-packet overhead (such as hash table sizes). Table 9.1 summarizes parameter values used in the following experiments.

In order to measure defense performance, we first examine attack rates that create a denial-of-service effect at the victim. Section 6.2 depicts the effect of these attacks on the victim's service and on legitimate connections. The following sections present defense performance in various settings.

9.1 Attack Effect

Different levels of UDP, TCP and ICMP attacks are tested in Topology 1 (shown in Figure 8.4) to determine the minimum attack rates that create the denial-of-service effect. In the case of UDP and ICMP attacks, the targeted resource is the 10Mbps link between *eth1Netv* and *server1Netv*. In the case of TCP attacks, the targeted resource is the victim's connection buffer.

Parameter	Value
Flow_Hash_Size	1003 records
Flow_Inactive_Period	360 seconds
Connection_Hash_Size	50003 records
Good_Inactive_Period	360 seconds
Transient_Inactive_Period	5 seconds
Rate_Limit_Hash_Size	1003 records
Limited_Flow_Hash_Size	103 records
Good_Connection_Hash_Size	2003 records
Flow_Observation_Interval	1 second
Connection_Observation_Interval	1 second
Compliance_Period	20 seconds
Min_Rate	16 Kbps
Max_Rate	80Mbps
$rate_{inc}$	4 Kbps
f_{dec}	0.5
f_{inc}	1
Early_Packet_Rate	800 Kbps
tcp_{rto}	3
$icmp_{rto}$	1.1
n_{conn}	100
p_{conn}	3
N_{burst}	5
N_{queue}	4
p_{sample}	0.6

Table 9.1: D-WARD configuration parameter values

9.1.1 UDP Flooding Attack

We generate UDP flooding attacks with 1KB-long packets, thus maximizing the chance of bandwidth consumption. Both source networks host attackers that produce the same attack rates.

9.1.1.1 Service Level

Figure 9.1 depicts the service level in a case of UDP attack. The x-axis shows the total amount of attack traffic flooding the critical link, and the y-axis shows the measured service level. Service starts degrading when the attack traffic exceeds link capacity — 10Mbps. It degrades severely as the attack rate increases. The service level measure exhibits significant variability for attack rates lower than 80Mbps. We believe that this occurs because the attack rate is not high enough to severely overwhelm the link, so ocassionaly some legitimate traffic succeeds in reaching the victim machine.

9.1.1.2 Connection Delay

Figures 9.2, 9.3 and 9.4 depict total delay for all connections and maximum and median per connection delays. We observe that total and maximum delay increase rapidly as soon as rate increases above 10Mbps. Meanwhile, median connection delay increases slightly due to attack, but remains quite small (at most 300ms). This indicates that delay distribution over impacted connections has a heavy tail — several connections are severely delayed (around 190 seconds) while the majority of connections suffer only a slight delay.

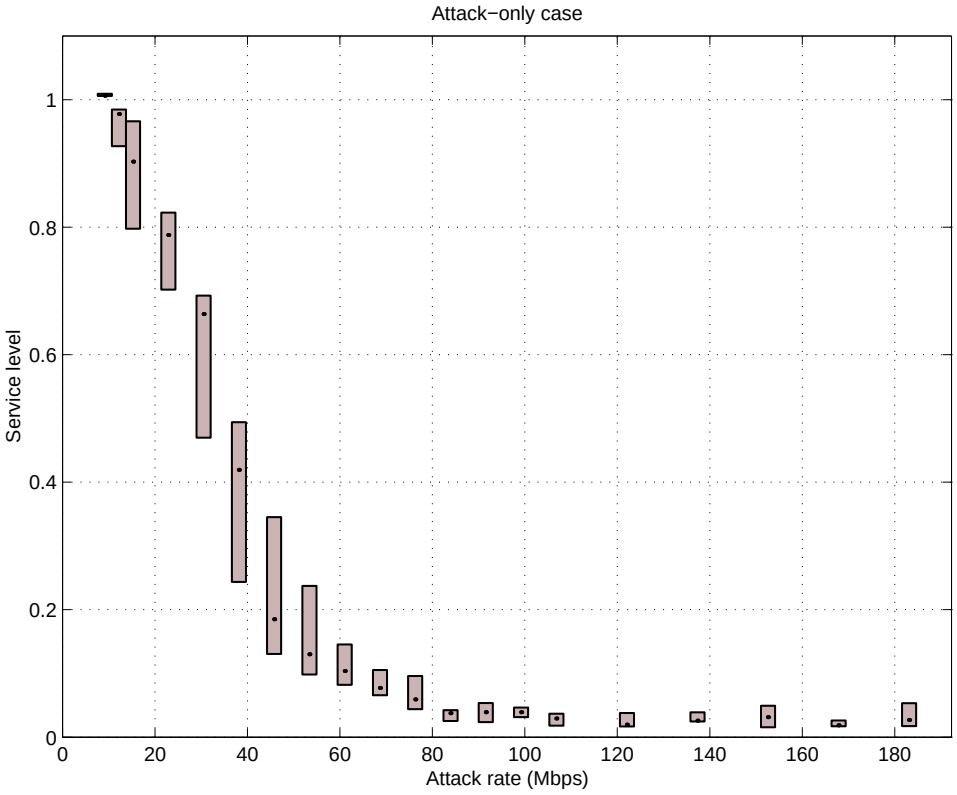

Figure 9.1: Service level in the case of a UDP flooding attack, no defense

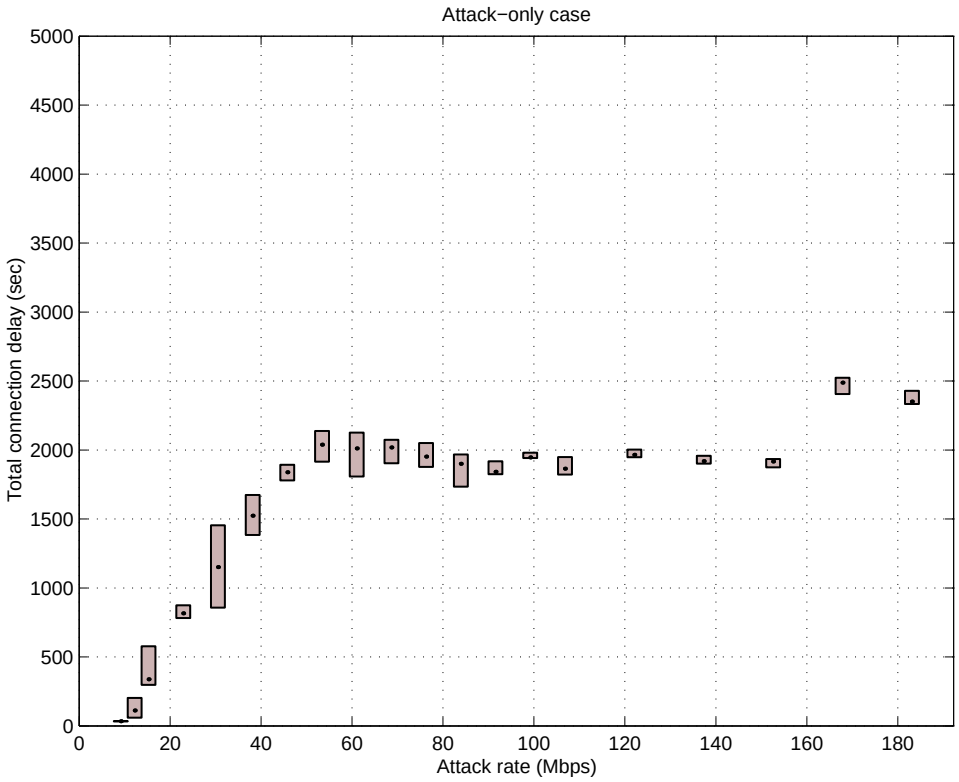

Figure 9.2: Total connection delay in the case of a UDP flooding attack, no defense

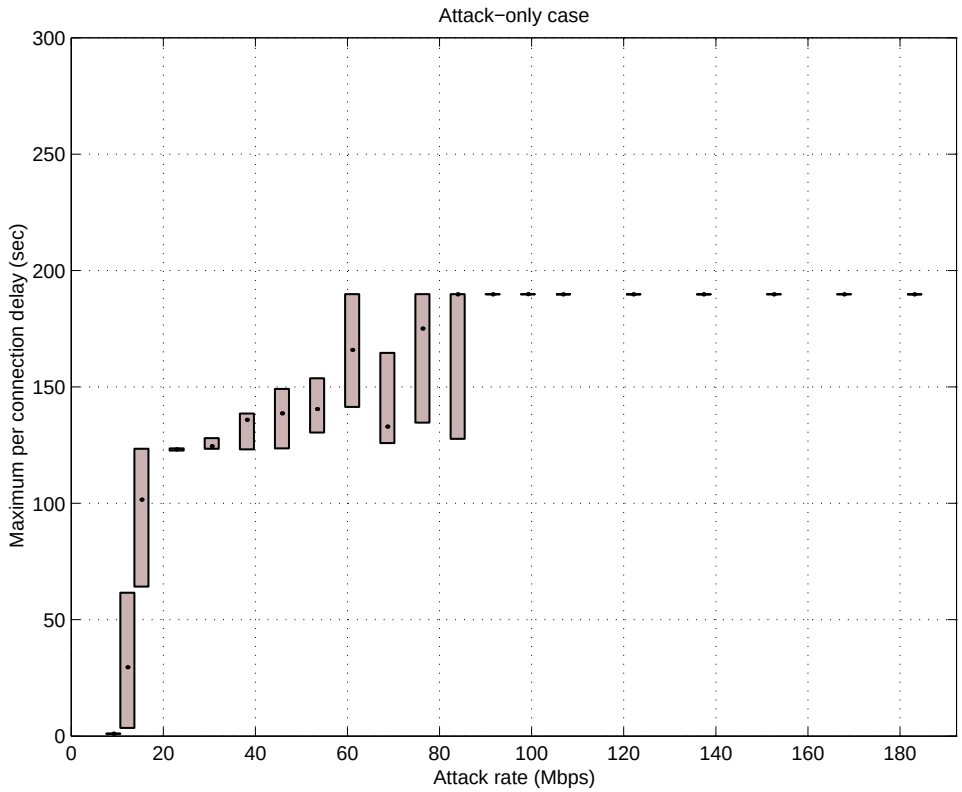

Figure 9.3: Maximum per-connection delay in the case of a UDP flooding attack, no defense

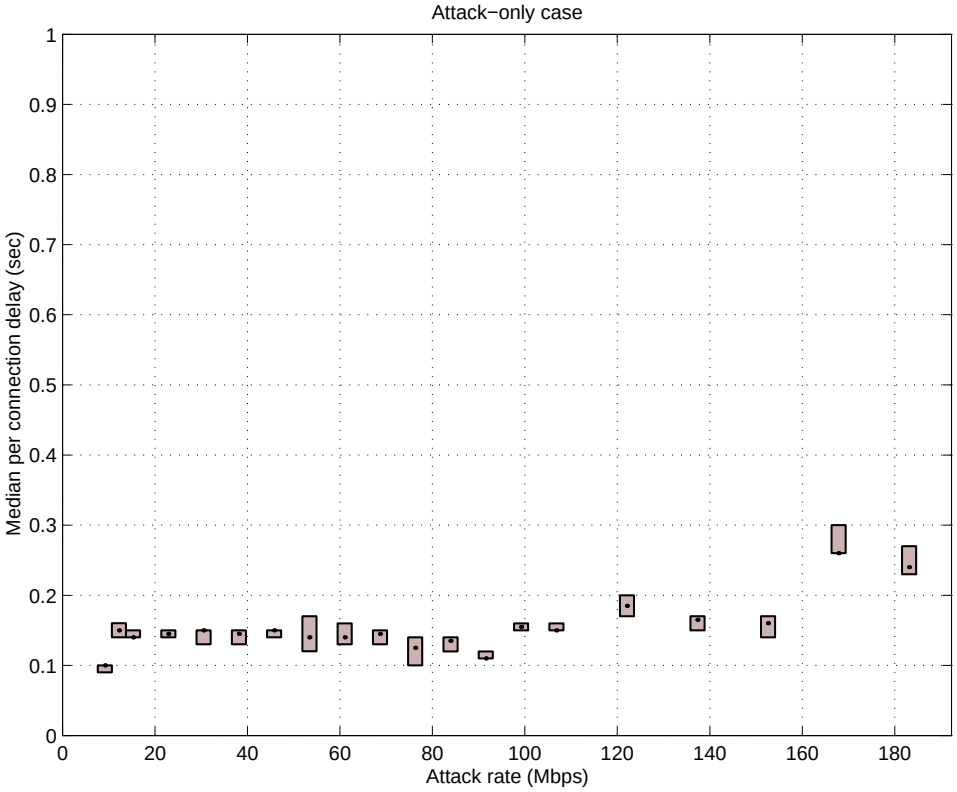

Figure 9.4: Median per-connection delay in the case of a UDP flooding attack, no defense

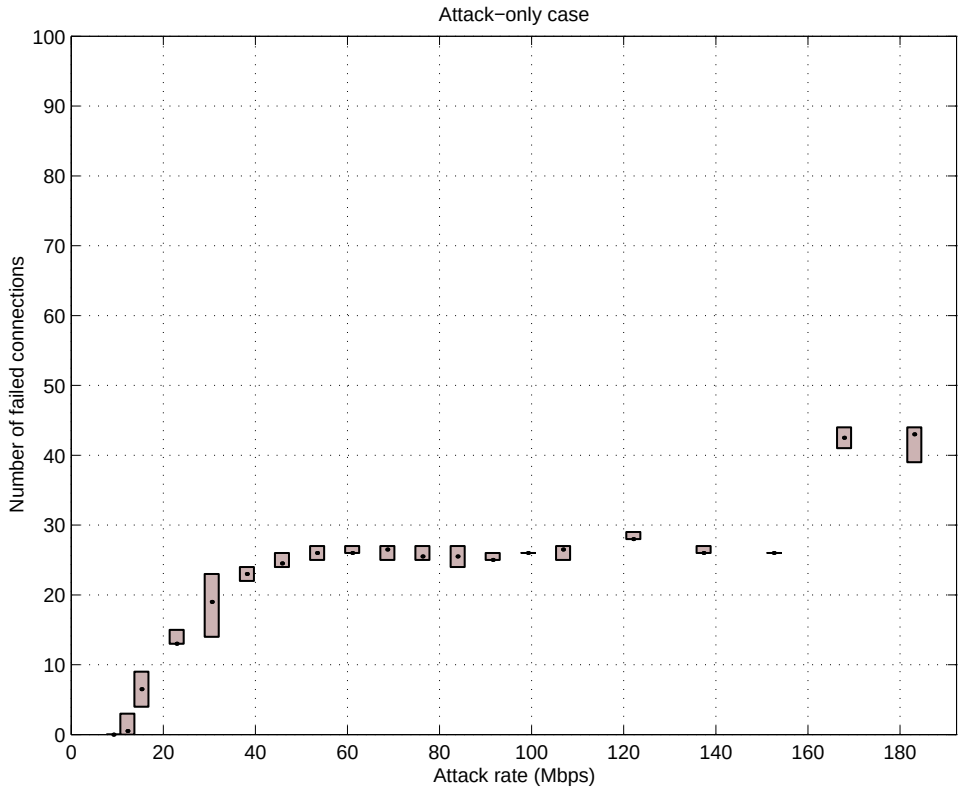

Figure 9.5: Number of failed connections in the case of a UDP flooding attack, no defense

9.1.1.3 Connection Failure

Figure 9.5 depicts the number of failed connections. We observe that this number increases rapidly around 10Mbps attack rate, and reaches values from 25 to 45 failed connections. Since there are 433 connections whose durations overlap the attack, this indicates a connection failure of around 10% .

9.1.2 ICMP Flooding Attack

We generate ICMP flooding attacks with 1KB-long packets, thus maximizing the chance of bandwidth consumption. Both source networks host attackers that

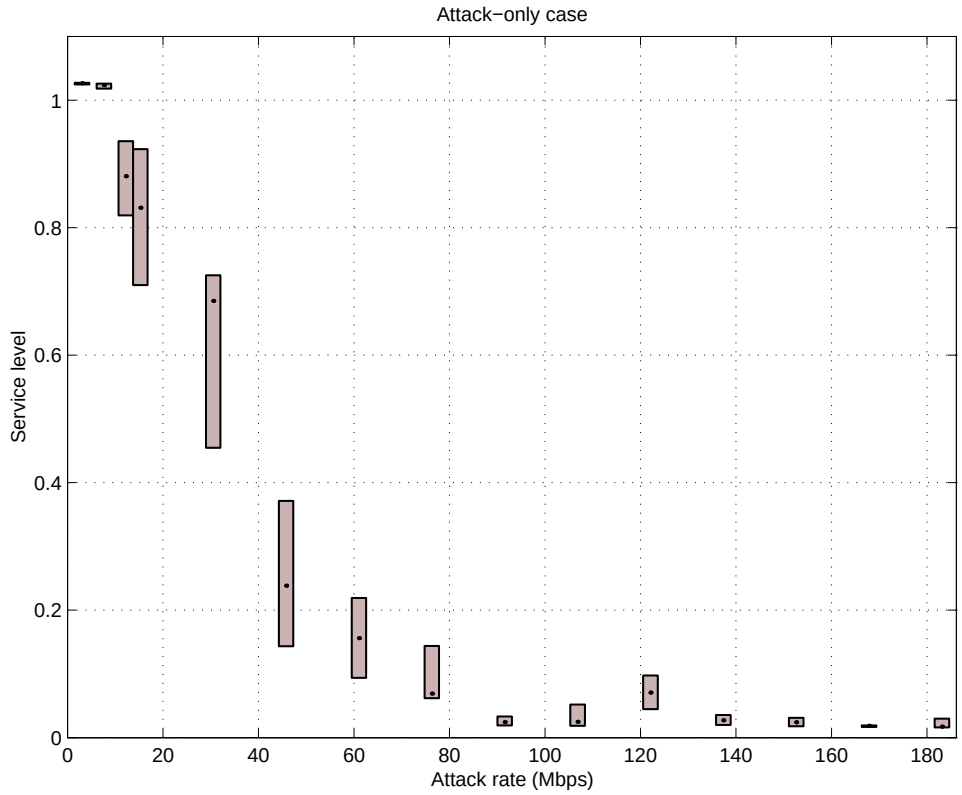

Figure 9.6: Service level in the case of an ICMP flooding attack, no defense

produce the same attack rates. Because the ICMP attack also targets network bandwidth just like the UDP attack, we expect similar results.

9.1.2.1 Service Level

Figure 9.6 depicts the service level in the case of an ICMP attack. Just like the case of a UDP flooding attack, service starts degrading when the attack traffic exceeds link capacity — 10Mbps. It degrades severely as the attack rate increases. The service level measure exhibits a large variability for attack rates lower than 80Mbps.

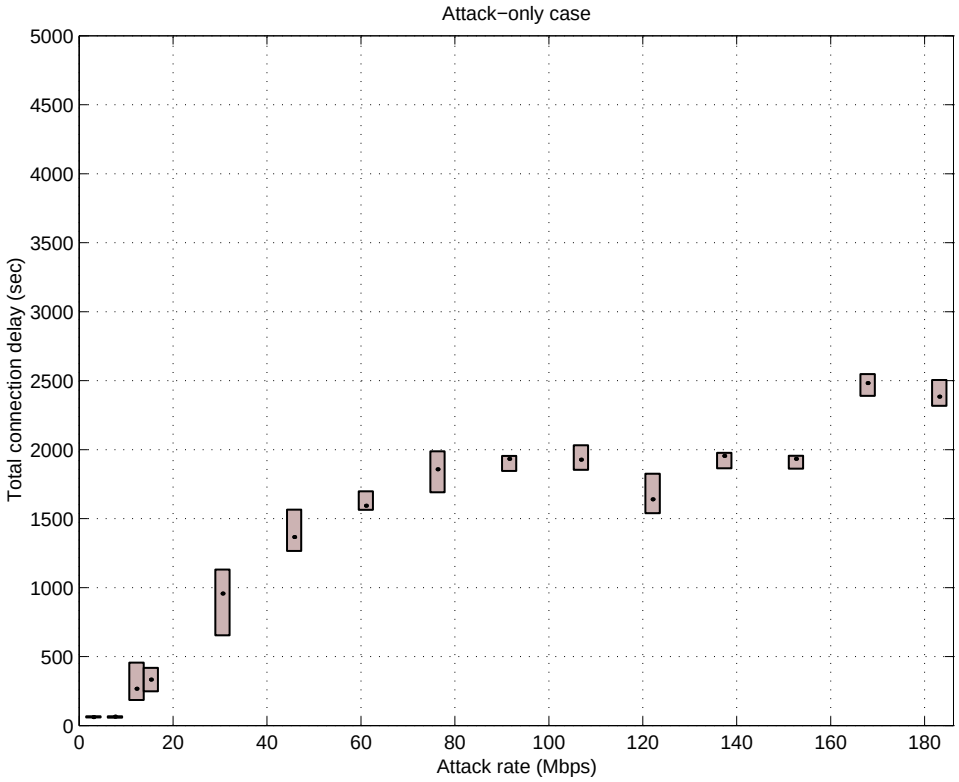

Figure 9.7: Total connection delay in the case of an ICMP flooding attack, no defense

9.1.2.2 Connection Delay

Figures 9.7, 9.8 and 9.9 depict total delay for all connections and maximum and median per-connection delays. We observe that all delays behave as they did in the UDP attack case. Total and maximum delays increase rapidly as the rate increases above the 10Mbps attack rate; the median connection delay increases slightly, but remains quite small (less than 300ms).

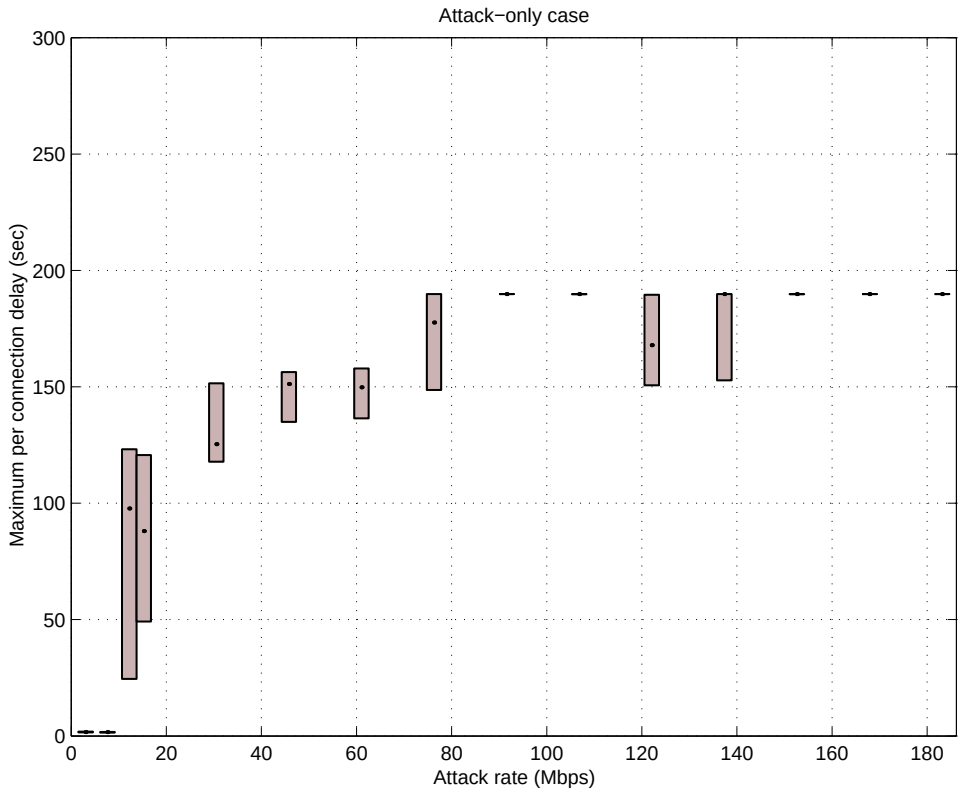

Figure 9.8: Maximum per-connection delay in the case of an ICMP flooding attack, no defense

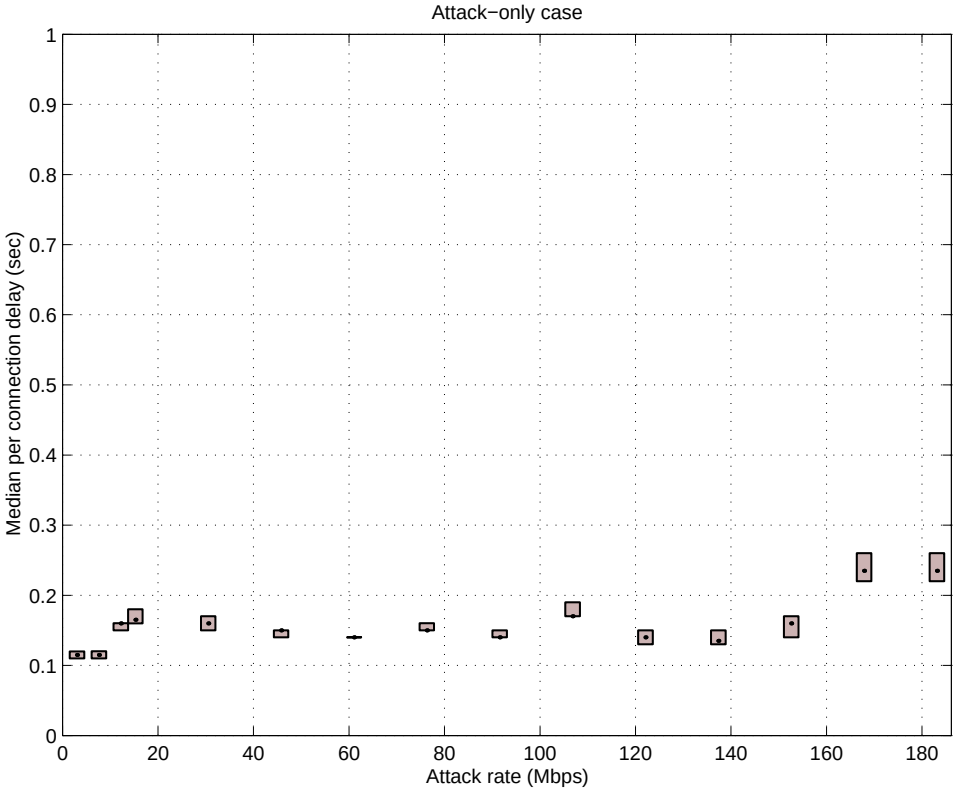

Figure 9.9: Median per-connection delay in the case of an ICMP flooding attack, no defense

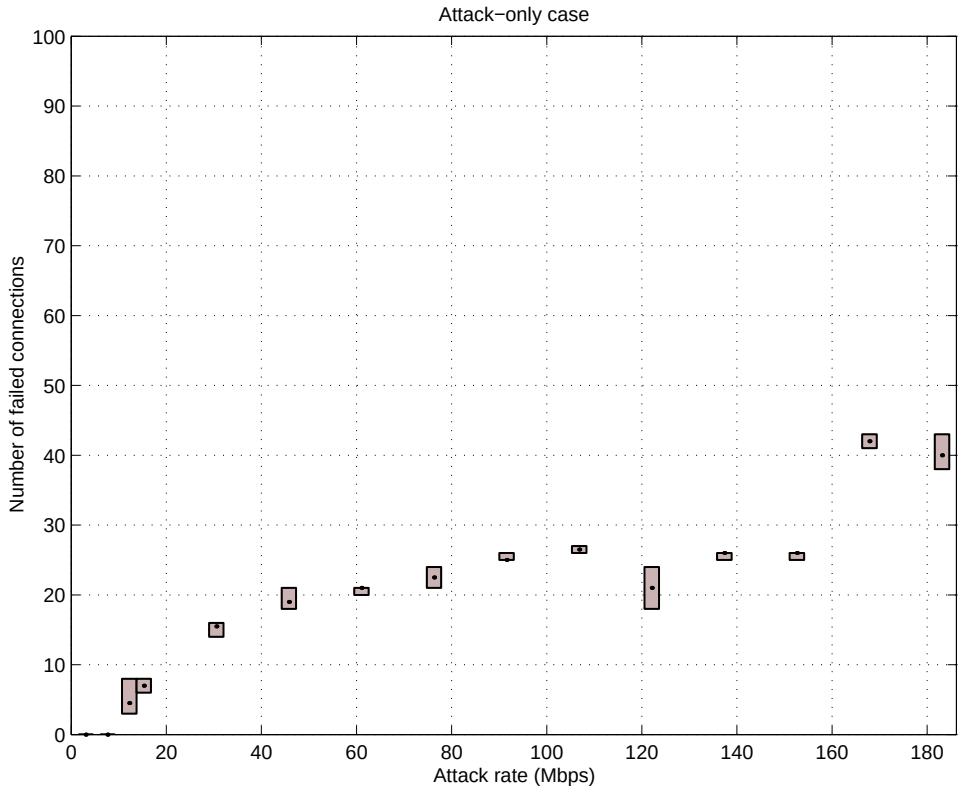

Figure 9.10: Number of failed connections in the case of an ICMP flooding attack, no defense

9.1.2.3 Connection Failure

Figure 9.10 depicts the number of failed connections. Again, the behavior is similar to that in UDP attack case. The number of failed connections increases rapidly for rates above 10Mbps, and reaches values from 20 to 42 failed connections.

9.1.3 TCP Flooding Attack

We generate TCP SYN flooding attacks, targeting SSH port 22. The critical resource targeted with a TCP attack is the connection buffer at the victim. The attack spoofs random source addresses and ports in order to generate many con-

nection records and quickly exhaust the buffer space. As the number of generated connections is proportional to the packet rate, this rate is shown at x-axis.

9.1.3.1 Service Level

Figure 9.11 depicts the service level in the case of a TCP attack. Service is quickly reduced to 90% when the attack traffic exceeds 100 packets per second. However, unlike previous attack cases, service level is not affected by a TCP attack rate increase. A TCP SYN attack exhausts connection buffer space, prohibiting setup of new connections, but it does not affect the already established connections. As soon as the connection buffer is exhausted (at 100 packets per second), the effect is visible. The service will be degraded by the amount of traffic belonging to new connections that attempt to be established while the attack is ongoing. In our tests, this amount is around 10%. Generally, the level of service degradation in the case of a TCP SYN attack will be a function of the amount of traffic belonging to new connections that originate during the attack.

9.1.3.2 Connection Delay

Figures 9.12, 9.13 and 9.14 depict the total delay for all connections and maximum and median per-connection delays. Total and maximum delay increase rapidly at the very beginning and they do not change with the attack rate. This behavior is expected based on the above analysis of the TCP SYN attack. Median connection delay remains quite small (below 200ms).

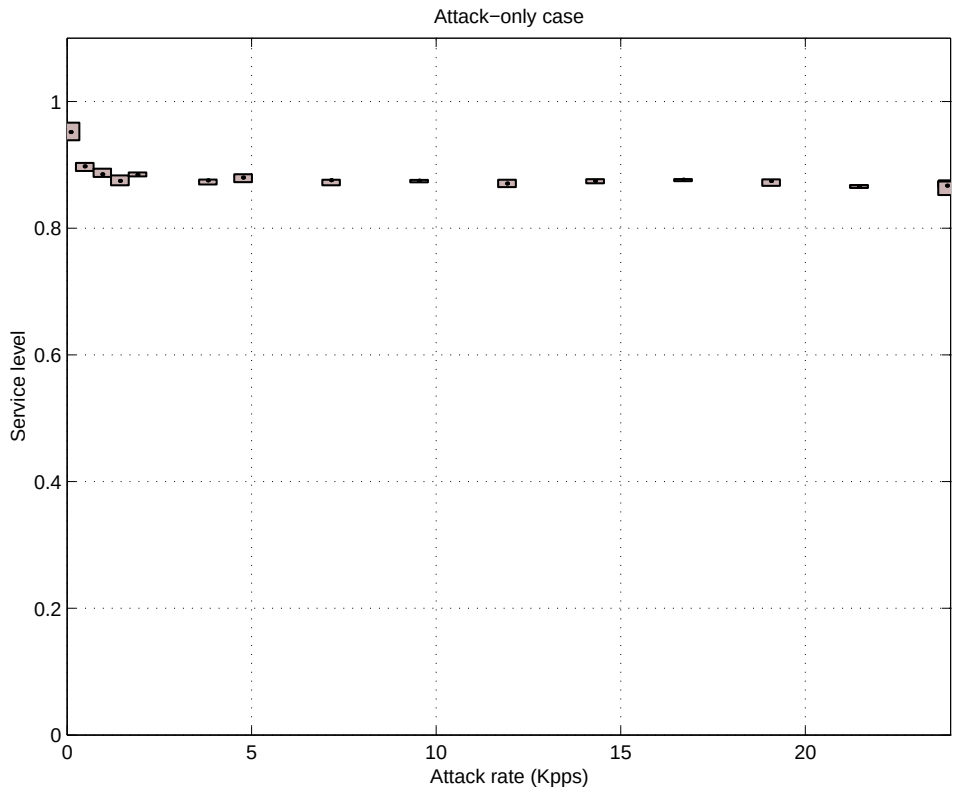

Figure 9.11: Service level in the case of a TCP flooding attack, no defense

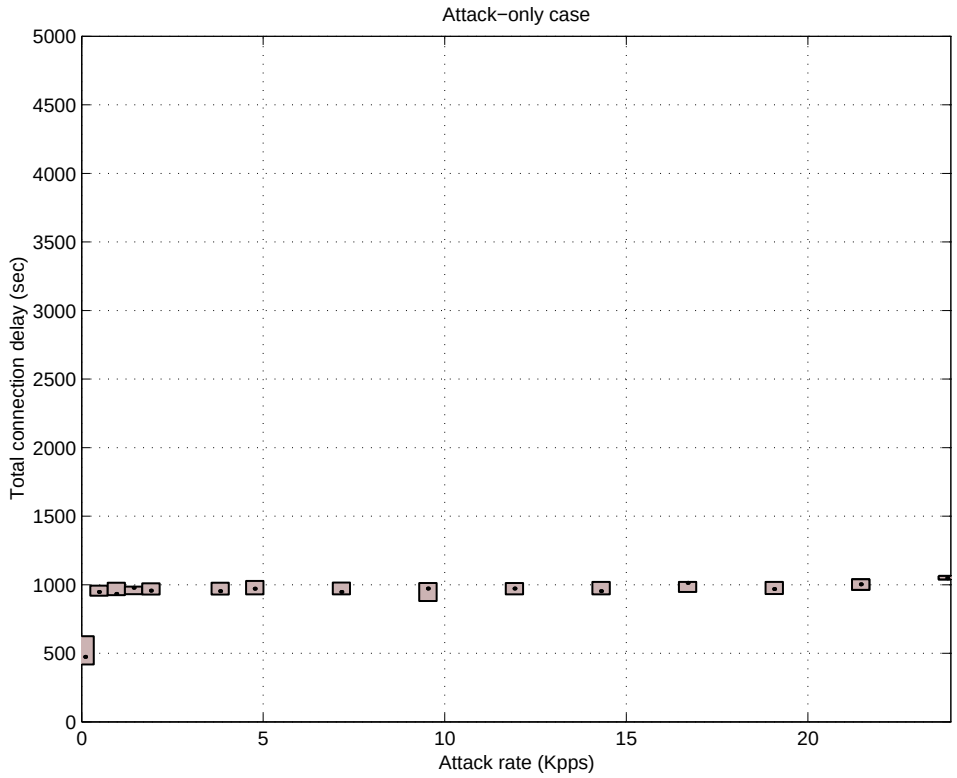

Figure 9.12: Total connection delay in the case of a TCP flooding attack, no defense

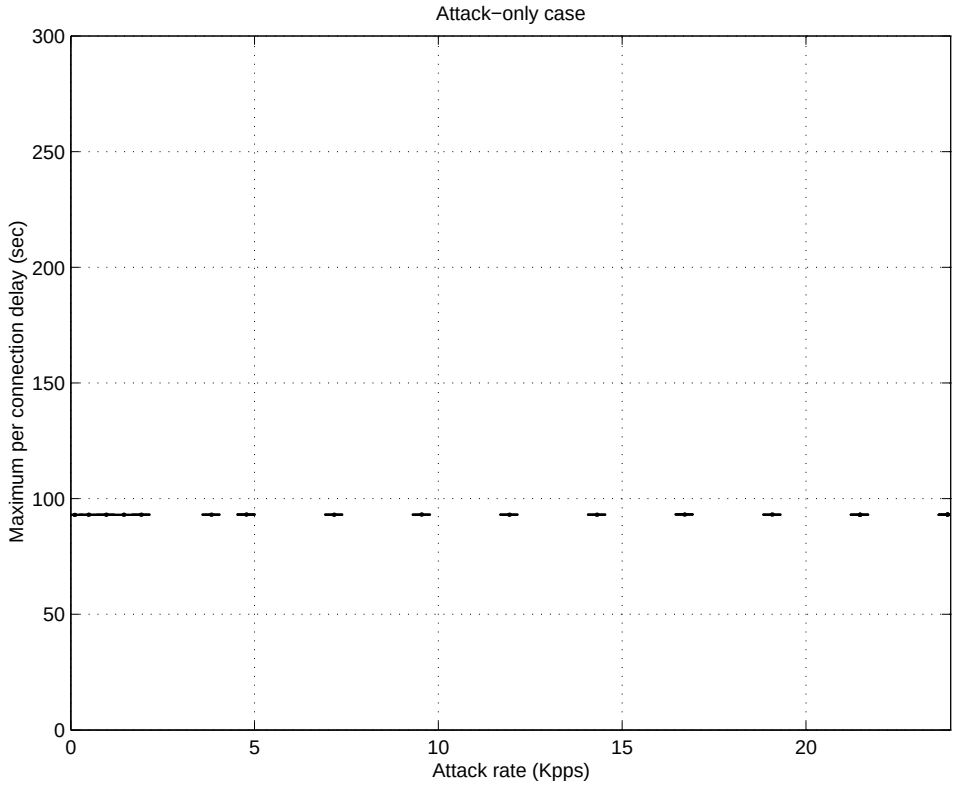

Figure 9.13: Maximum per-connection delay in the case of a TCP flooding attack, no defense

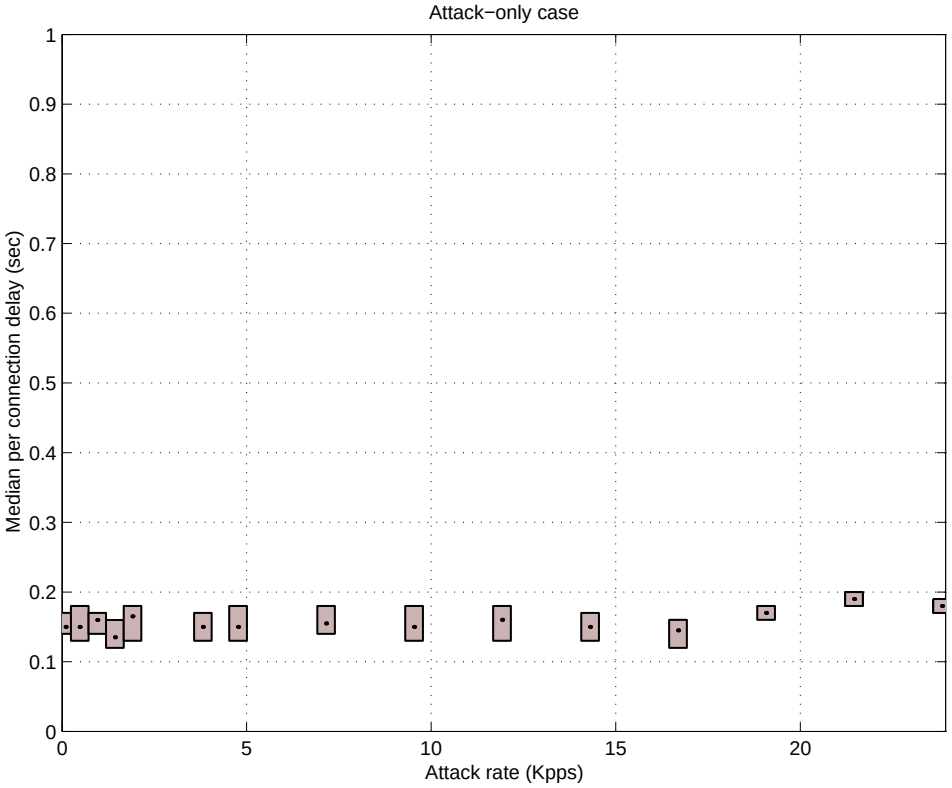

Figure 9.14: Median per-connection delay in the case of a TCP flooding attack, no defense

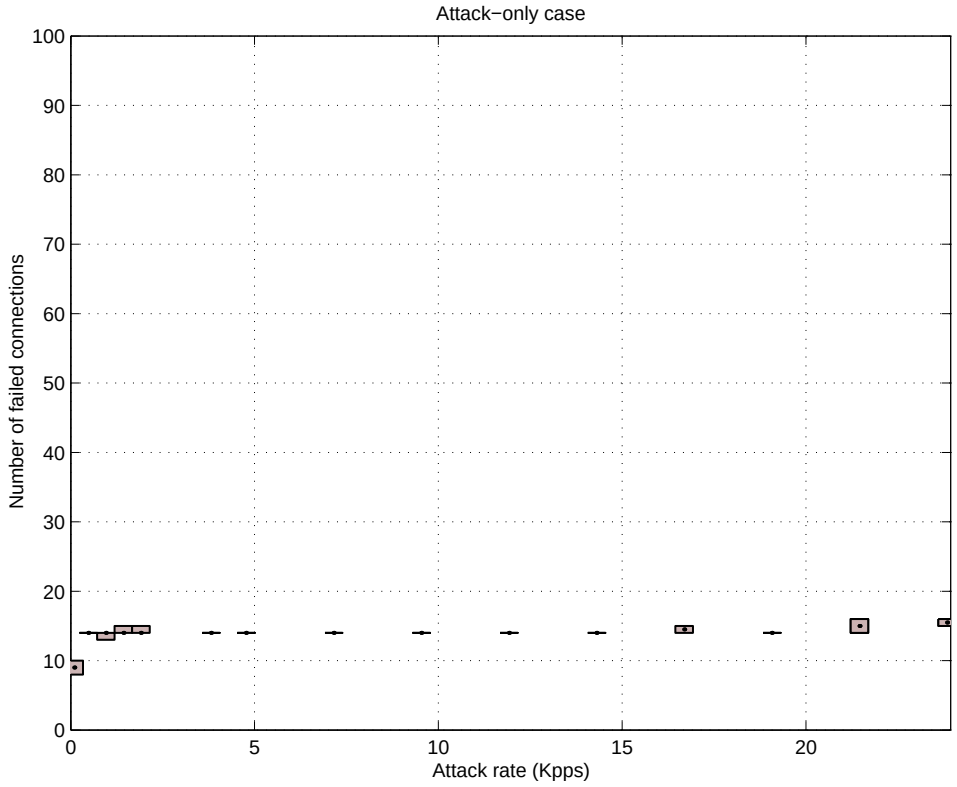

Figure 9.15: Number of failed connections in the case of a TCP flooding attack, no defense

9.1.3.3 Connection Failure

Figure 9.15 depicts the number of failed connections. This number rises promptly to values from 14 to 17 and does not change with the attack rate, as expected.

9.1.4 Maximum Attack Rate

D-WARD can handle up to 12,000 packets per second, assuming that all packets belong to spoofed connections. Since every packet in this case results in a hash traversal to find the appropriate record, this maximizes per-packet overhead. At higher rates, packet handling will interfere with forwarding, causing severe

Attack	Min. pps	Min. bps	Max. pps	Max. bps
UDP flood	600	4.5 M	12 000	90 M
ICMP flood	600	4.5 M	12 000	90 M
TCP flood	25	8000	12 000	3.6 M

Table 9.2: Relevant attack ranges

packet loss. This figure defines the maximum packet rate per one source network. Additionally, the link bandwidth can support up to 100 Mbps. Any higher attack rates will potentially create congestion upstream of D-WARD, at a point that is beyond the defense system's control. Thus in all tested attacks, the attack rate per source network will be kept below 100 Mbps and below 120 kpps.

Table 9.2 summarizes the relevant ranges of attack rates that will be used to evaluate D-WARD performance.

9.2 UDP Attacks

In order to test D-WARD's performance with UDP attacks, we generate a UDP flooding attack targeting random ports, using subnet spoofing of IP addresses and random spoofing of source ports. These spoofing techniques will generate a large number of UDP connections, thus exerting stress on D-WARD's connection hash table. All packets are 1KB long, maximizing the chance of bandwidth consumption. We test the whole range of attack rates that produce a denial-of-service effect. As there are two source networks generating the attack traffic, the tested range is from 9 Mbps to 183 Mbps.

9.2.1 Service Level

Figure 9.16 shows the service level perceived by legitimate clients during the attack. The case when the defense is engaged is compared to the case when the defense is not present. D-WARD successfully relieves the denial-of-service effect at the victim, providing a 100% service level to legitimate clients during the attack.

9.2.2 Connection Delays

Figure 9.17 depicts total legitimate connection delay during the attack. D-WARD successfully maintains total connection delay around 50 seconds, regardless of the attack rate. This is less than 3% of the total connection delay in the attack-only case.

Figure 9.18 depicts the maximum legitimate connection delay during the attack. We observe that D-WARD successfully controls connection delay, keeping maximum delay per connection below 40 seconds. While 40 seconds represent quite a long delay, this delay is much worse when the defense is not engaged (around 200 seconds). The reason why some connections experience such a severe delay even when D-WARD is present lies in the TCP congestion control mechanism. A few connections whose packets are dropped due to congestion in the few seconds after the start of the attack and before the defense is engaged, back off severely and reduce their sending rate. Even though these (and other legitimate) connections are treated favorably later on, this initial setback manifests itself in a long delay until these connections are completed.

Figure 9.19 depicts median legitimate connection delay during the attack. Median delay was already small in the attack-only case, so we do not expect

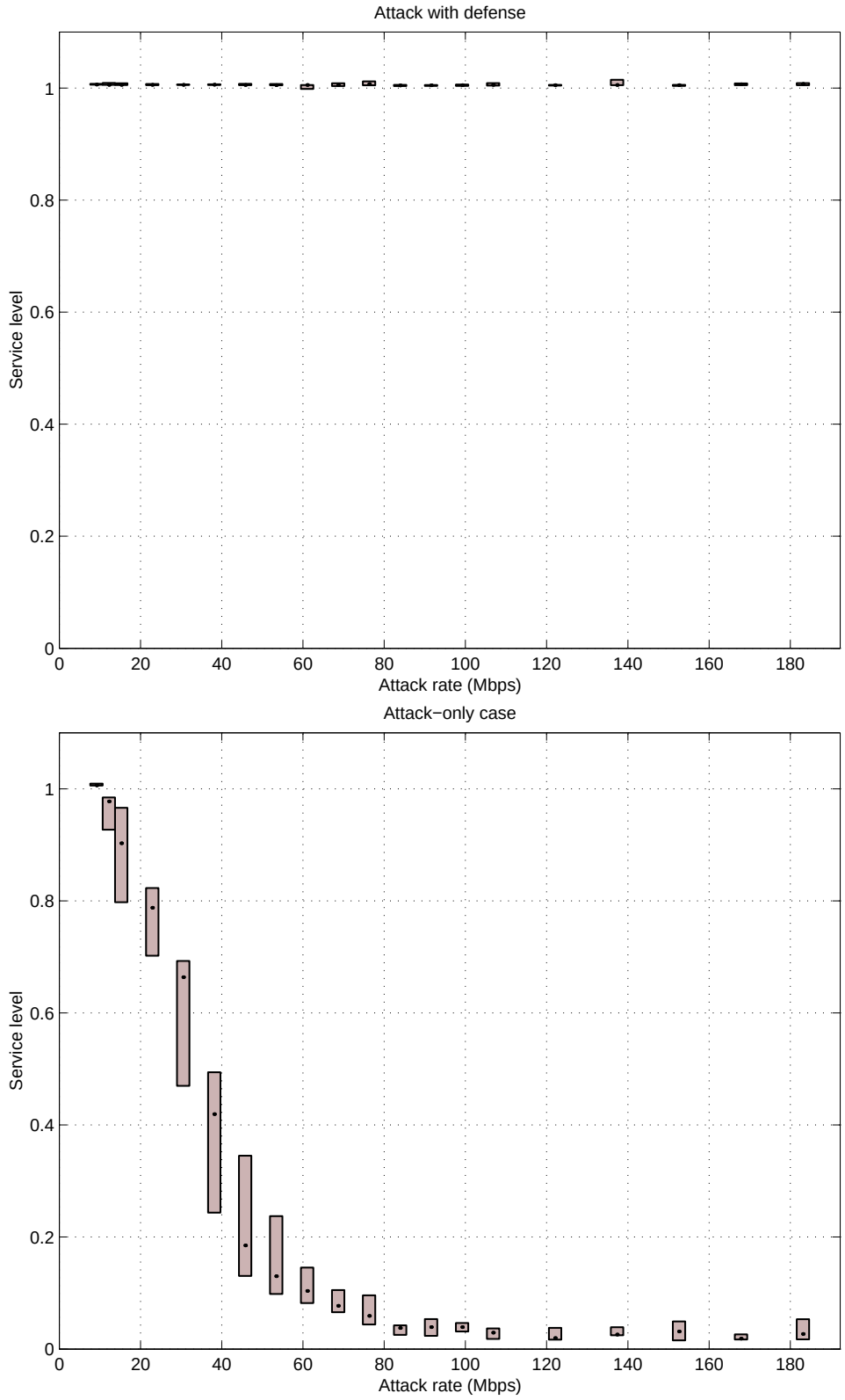

Figure 9.16: Service level in the case of a UDP flooding attack, with and without defense

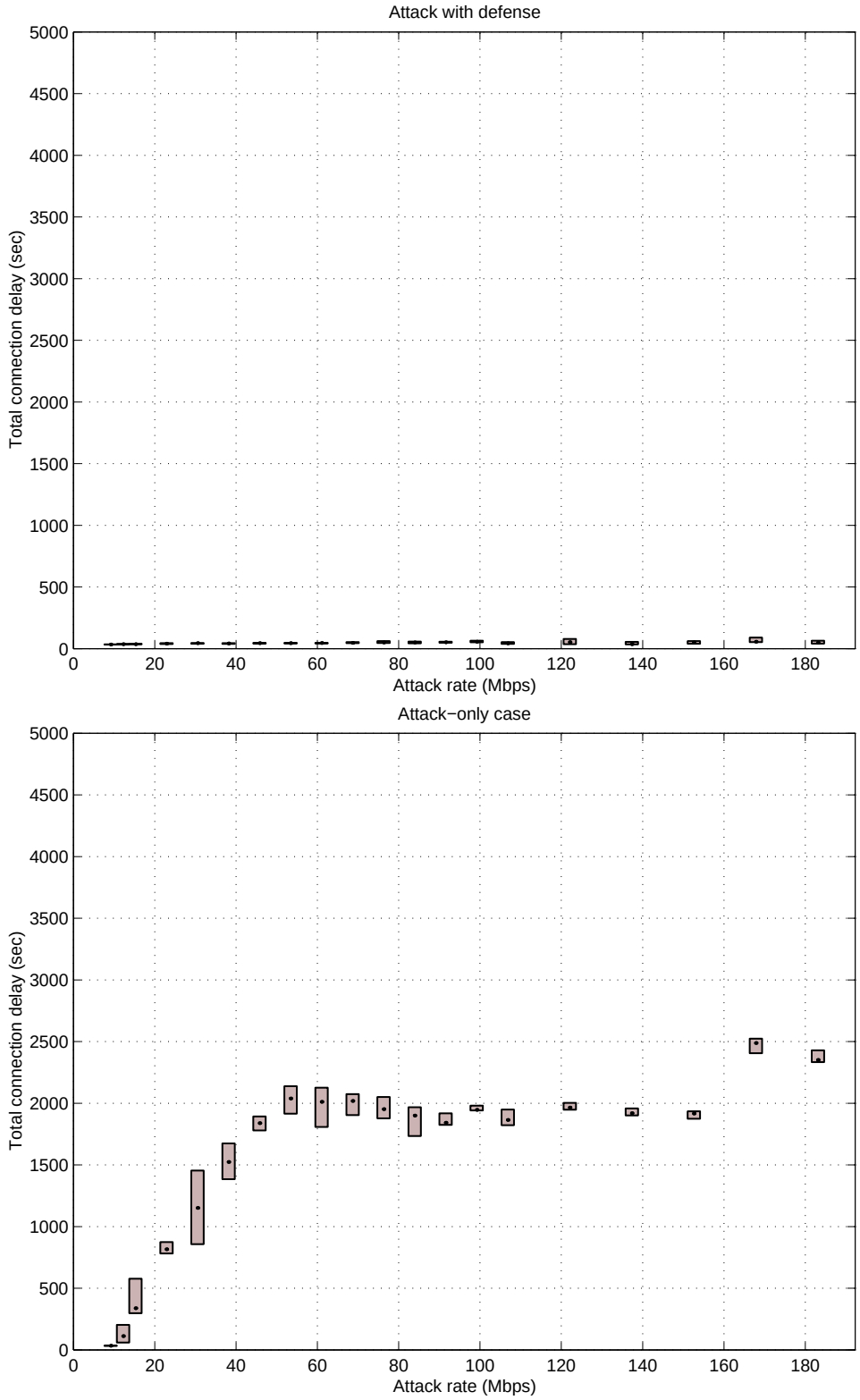

Figure 9.17: Total connection delay in the case of a UDP flooding attack, with and without defense

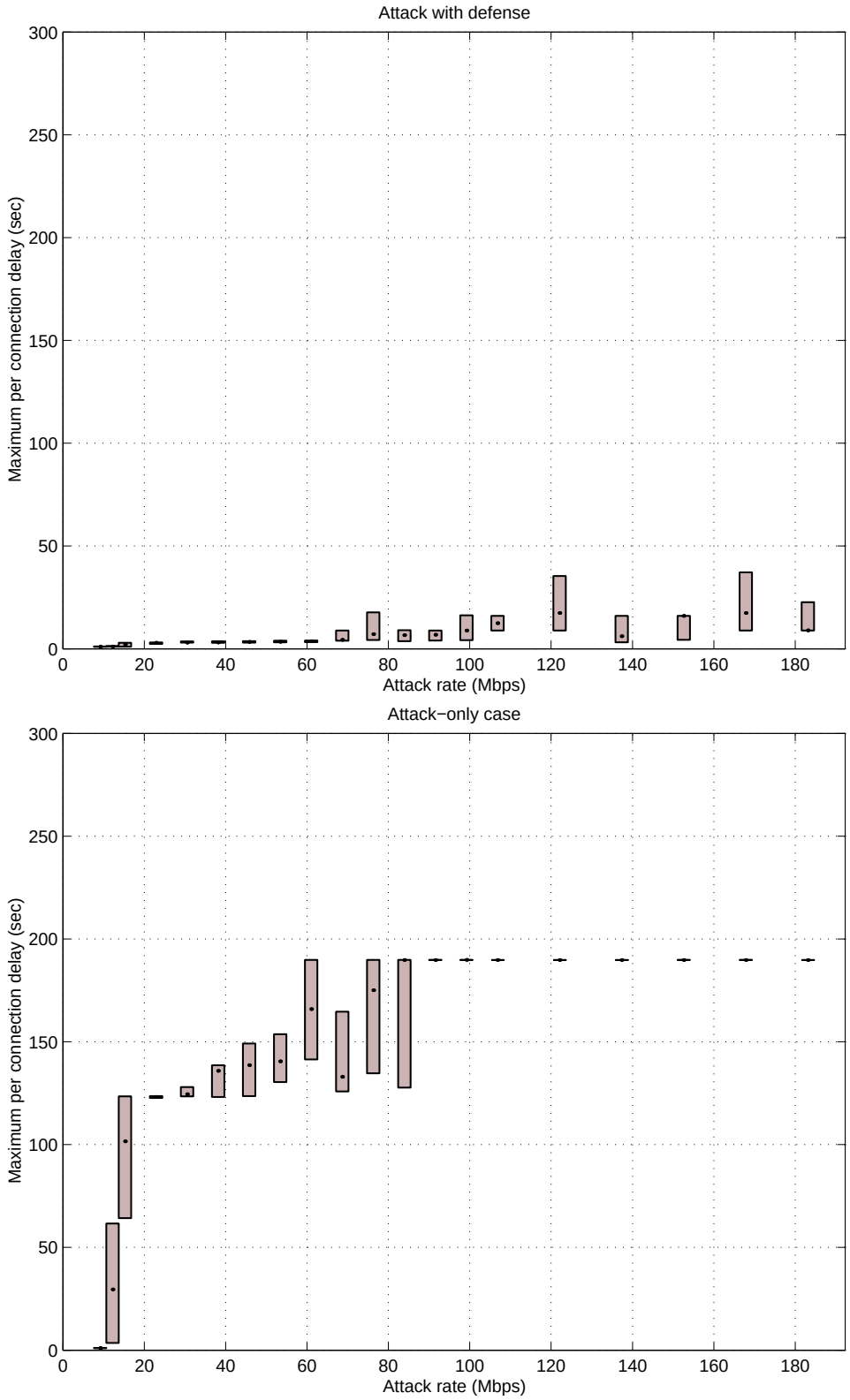

Figure 9.18: Maximum per-connection delay in the case of a UDP flooding attack, with and without defense

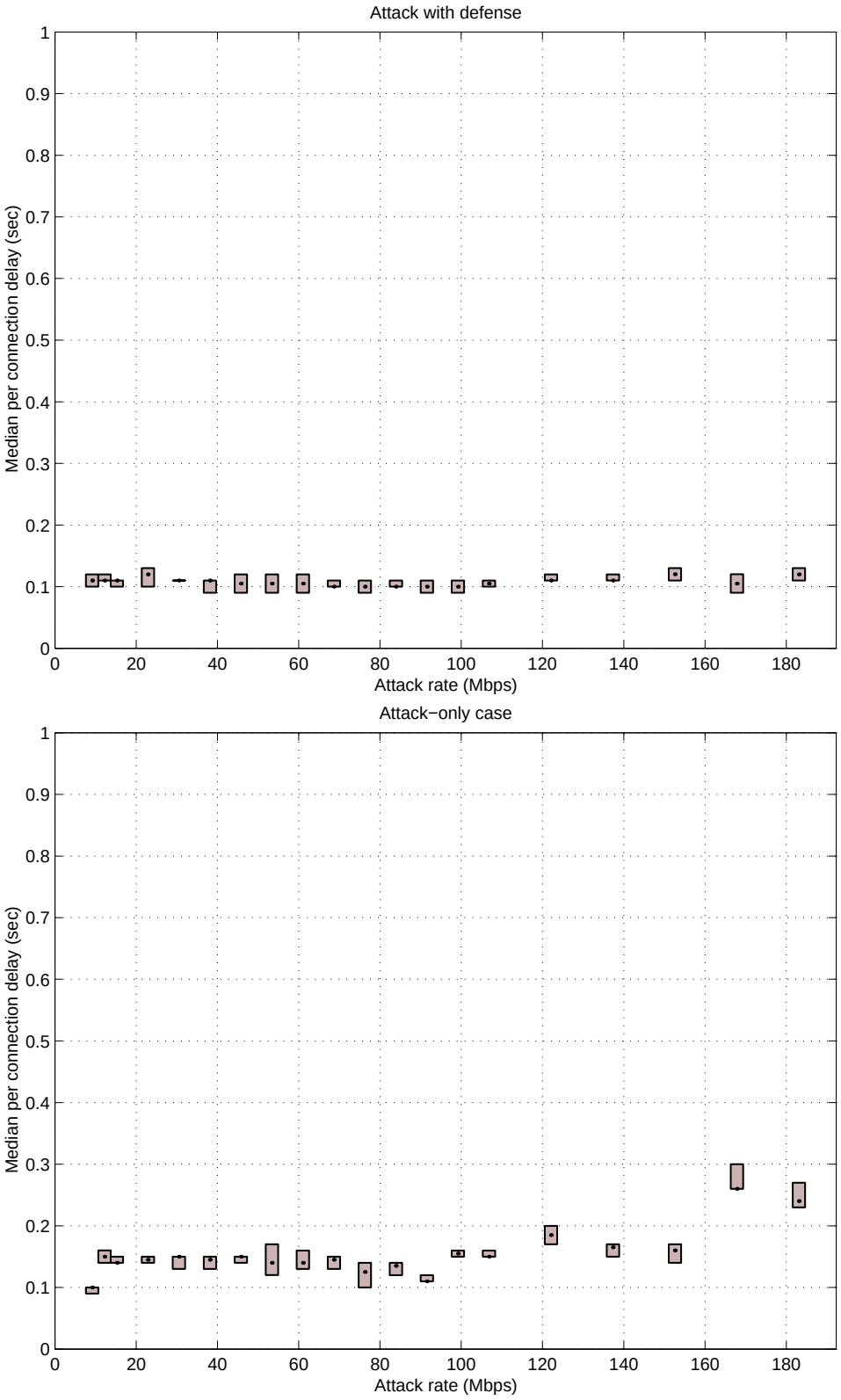

Figure 9.19: Median per-connection delay in the case of a UDP flooding attack, with and without defense

significant improvement from D-WARD deployment. Still, D-WARD manages to keep median delay at a very low level (around 100ms) and stable, in spite of the increasing attack rate.

9.2.3 Connection Failure

Figure 9.20 depicts the total number of failed connections during the attack. D-WARD maintains very low levels of failed connections — less than 4 regardless of the attack rate. This is around 1% of the total legitimate connections. Without D-WARD, the connection failure is much worse, around the values from 25 to 45 failed connections for strong attacks.

9.2.4 Defense Performance

Figures 9.21 and 9.22 depict D-WARD detection and response times for the UDP flooding attack. We observe that detection occurs in the first second of the attack, and the response starts 3 to 4 seconds later. This is a very prompt reaction to the attack.

Figure 9.23 depicts the percentage of legitimate traffic dropped by D-WARD. As we can observe, this value is around zero.[1] This indicates that no collateral damage is inflicted on legitimate traffic due to D-WARD operation.

Figure 9.24 depicts the percentage of attack traffic forwarded. This value is between 3% to 4% which agrees with our expectations. As D-WARD imposes very strict rate limits on detected attacks, the largest portion of the attack traffic reaches the victim in the short interval before the response has been activated. In this test response takes place after 3 to 4 seconds, and the attack lasts for

[1] Upon examining the raw data, we learned that occasionally, a maximum of 100 bytes will be dropped in the whole test run.

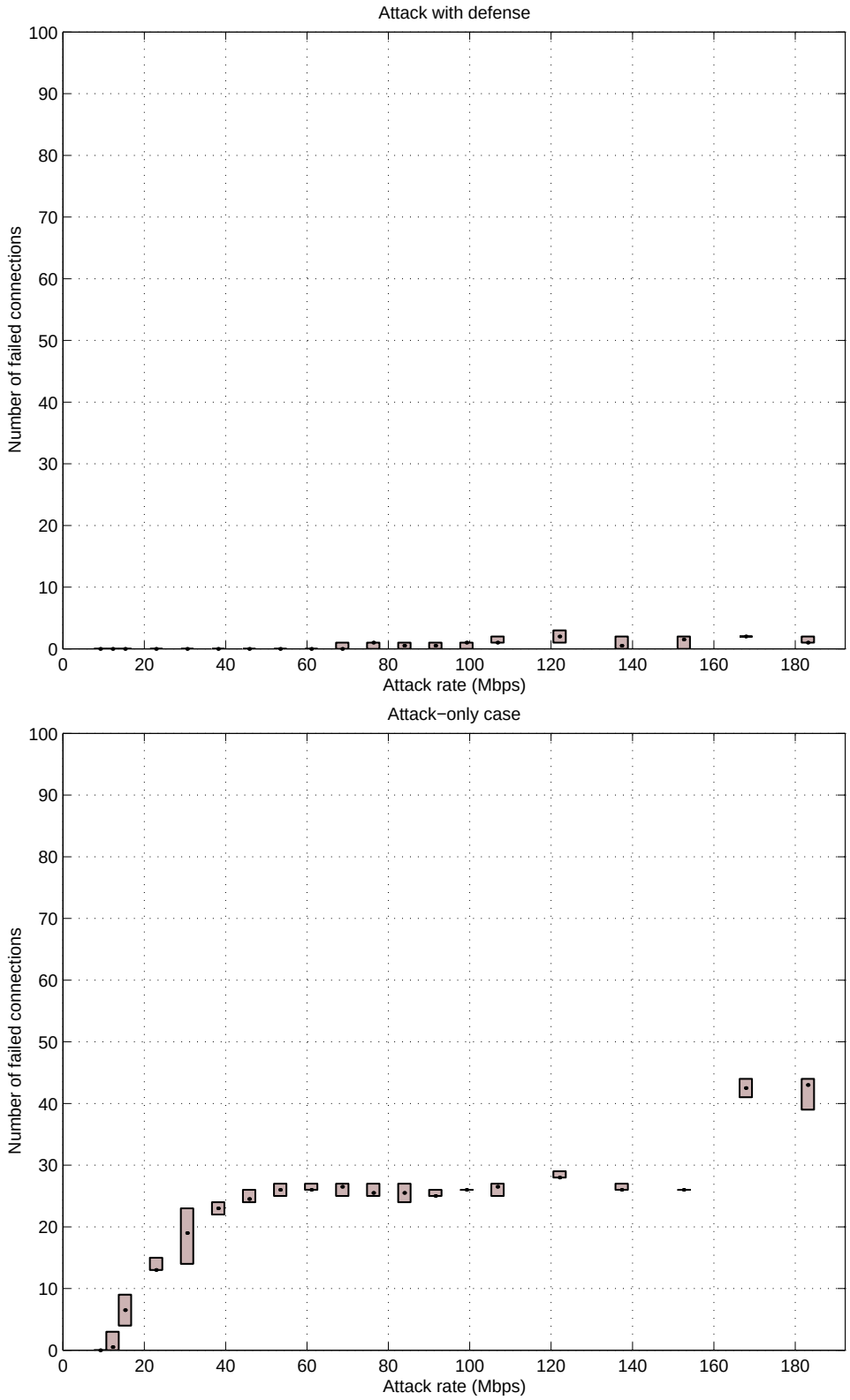

Figure 9.20: Number of failed connections in the case of a UDP flooding attack, with and without defense

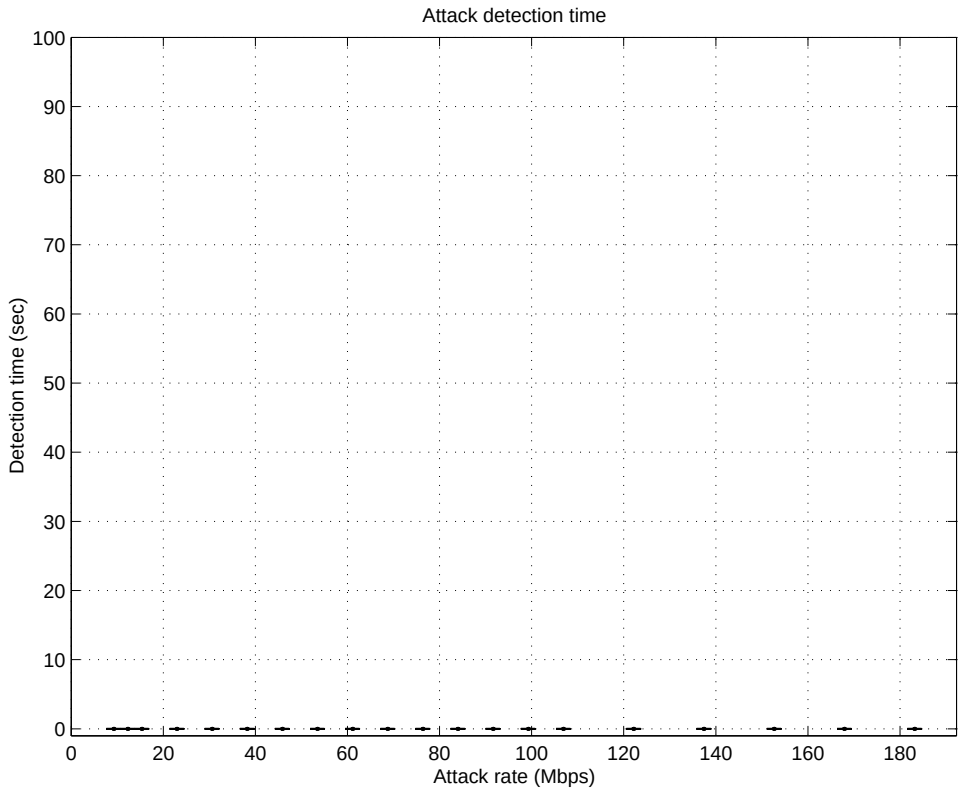

Figure 9.21: D-WARD attack detection time in the case of UDP flooding attack

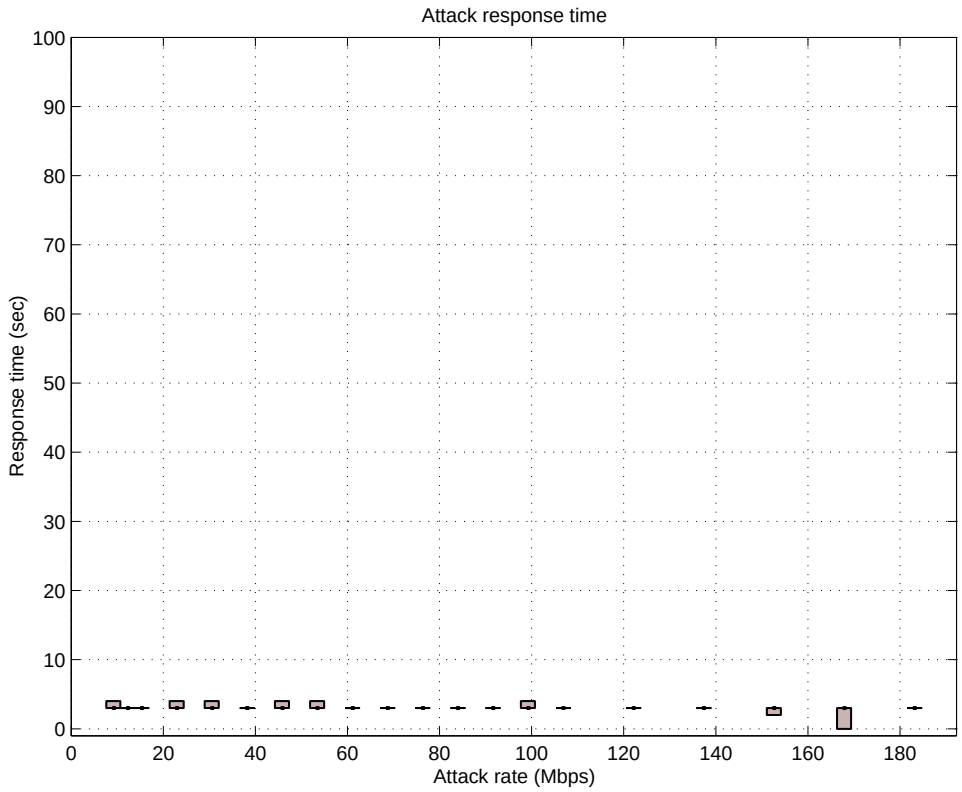

Figure 9.22: D-WARD response detection time in the case of UDP flooding attack

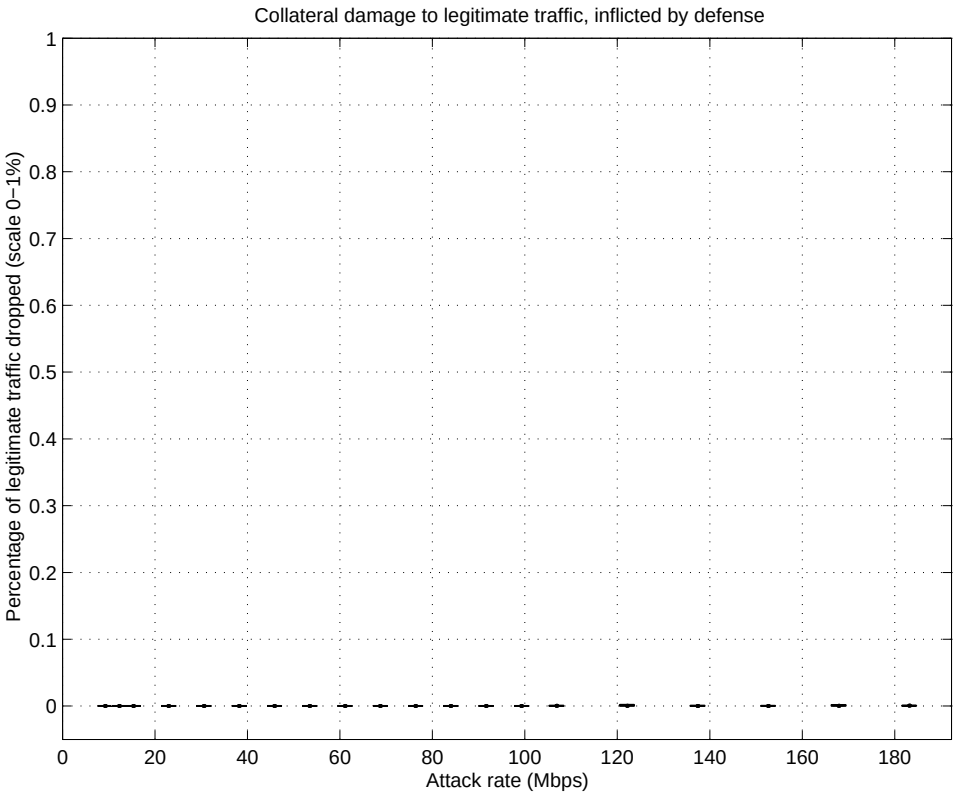

Figure 9.23: Percentage of legitimate traffic dropped by D-WARD in the case of UDP flooding attack

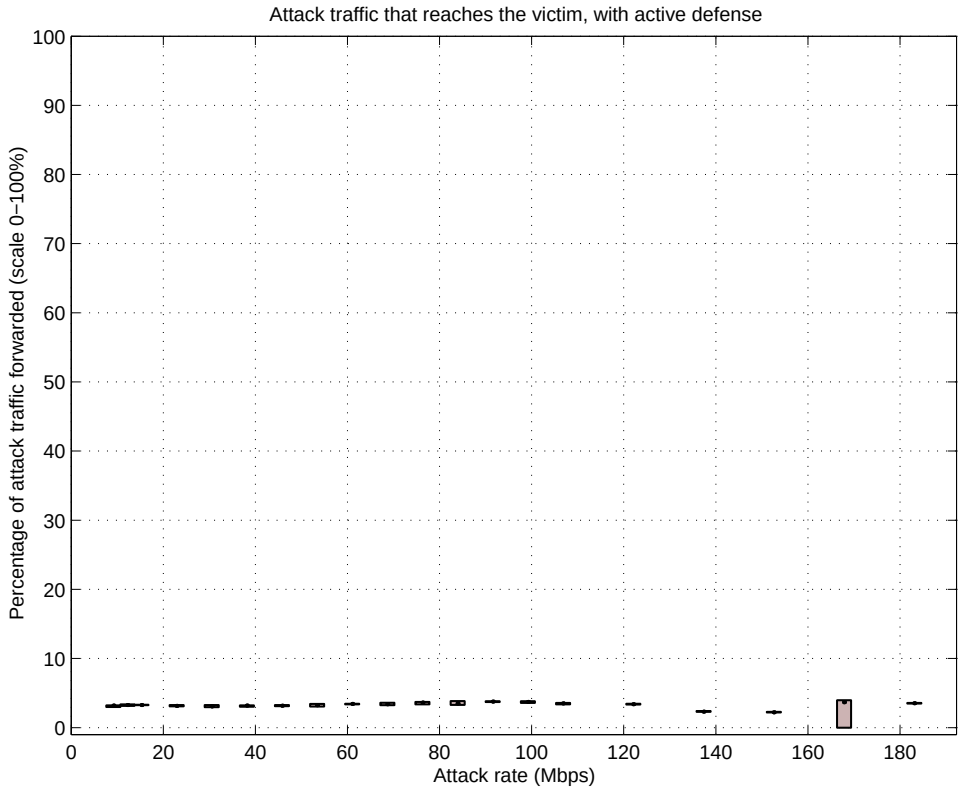

Figure 9.24: Percentage of attack traffic forwarded by D-WARD in the case of UDP flooding attack

100 seconds, so the expected percentage of attack traffic that reaches the victim should be 3% to 4%.

9.3 ICMP Attacks

In order to test D-WARD's performance with ICMP attacks, we generate an ICMP flooding attack using subnet spoofing of IP addresses. These spoofing techniques will generate a large number of ICMP connections, thus exerting stress on D-WARD's connection hash. All packets are 1KB long, maximizing the chance of bandwidth consumption. We test the whole range of attack rates that produce

a denial-of-service effect. As there are two source networks generating the attack traffic, the tested range is from 9Mbps to 180 Mbps.

9.3.1 Service Level

Figure 9.25 depicts service level perceived by legitimate clients during the attack. Just like the case of the UDP flooding attack, D-WARD successfully relieves the denial-of-service effect at the victim, providing a 100% service level to legitimate clients during the attack.

9.3.2 Connection Delays

Figure 9.26 depicts total legitimate connection delay during the attack. D-WARD successfully maintains a total connection delay of around 70 seconds, regardless of the attack rate. This value is less than 5% of the total connection delay in the attack-only case, for all but the lowest attack rates.

Figure 9.27 depicts maximum legitimate connection delay during the attack. We observe that D-WARD successfully controls connection delay, keeping the maximum delay per connection below 20 seconds. As explained in Section 9.2.2, the reason for such a long delay lies in the TCP congestion control mechanism that is activated by legitimate packet drops due to congestion in the short period (2 to 5 seconds) while D-WARD is being activated.

Figure 9.28 depicts median legitimate connection delay during the attack. Median delay was already small in the attack-only case. D-WARD manages to control median delay, maintaining the low level of around 100ms regardless of attack rate.

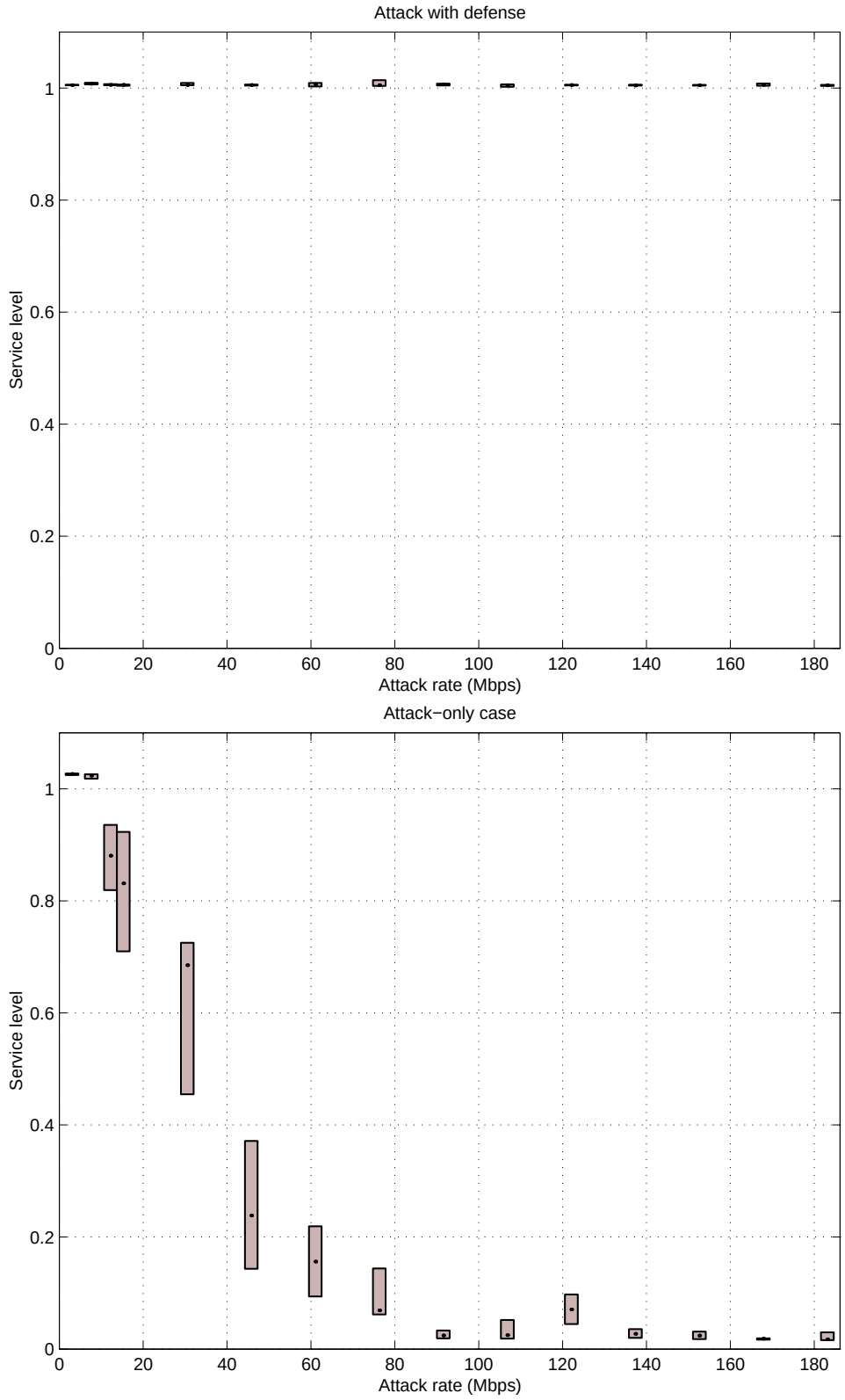

Figure 9.25: Service level in the case of ICMP flooding attack, with and without defense

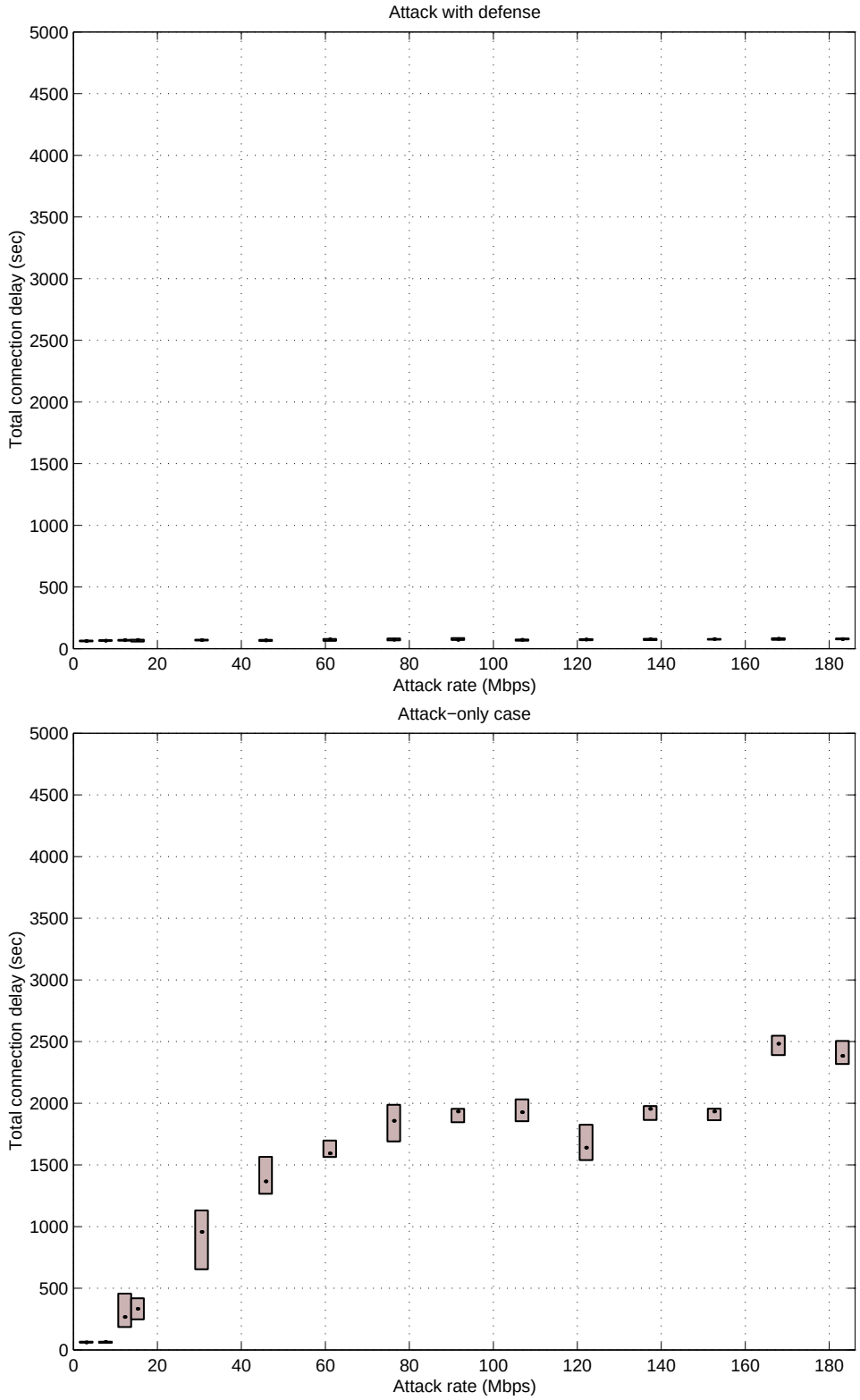

Figure 9.26: Total connection delay in the case of ICMP flooding attack, with and without defense

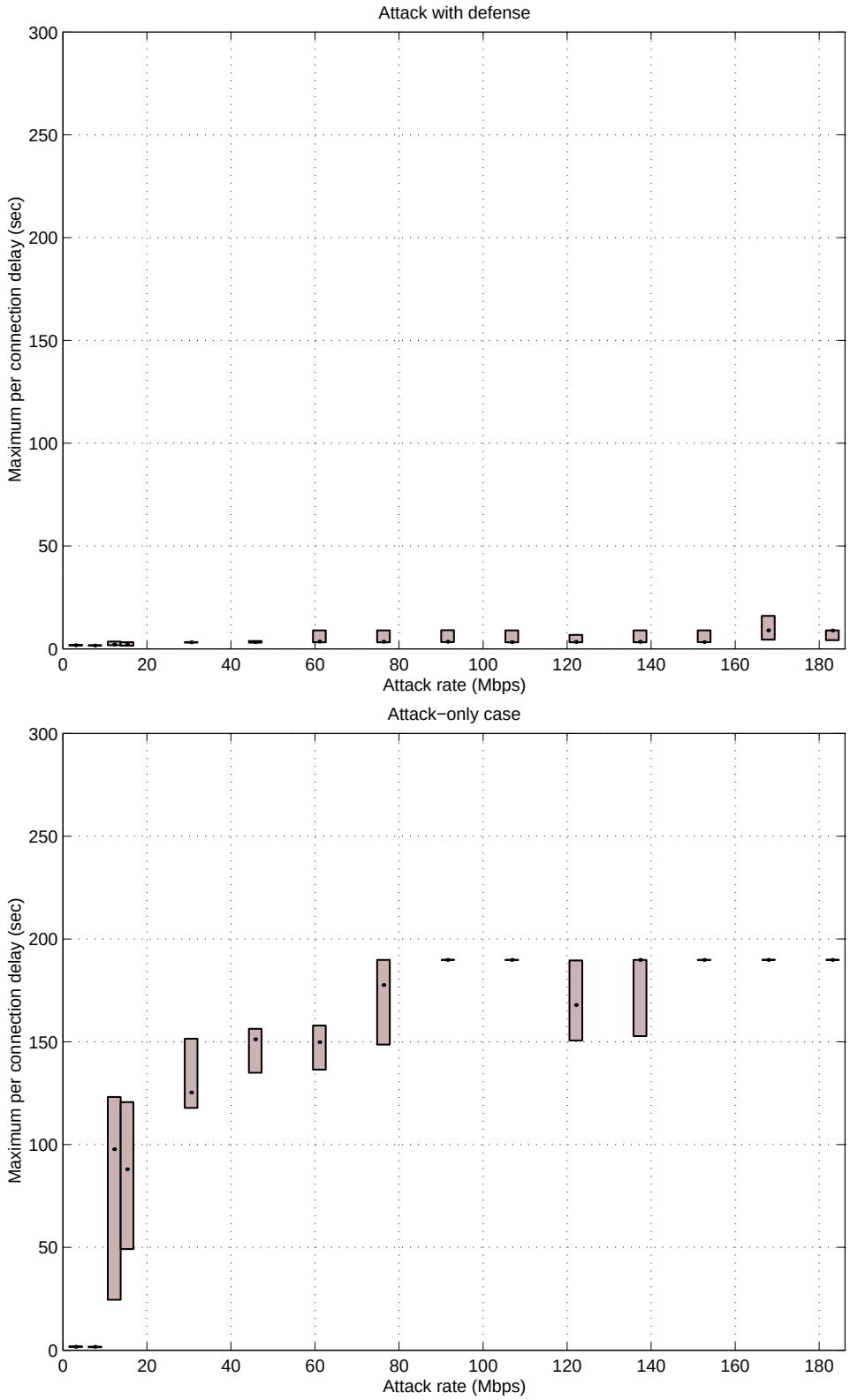

Figure 9.27: Maximum per-connection delay in the case of ICMP flooding attack, with and without defense

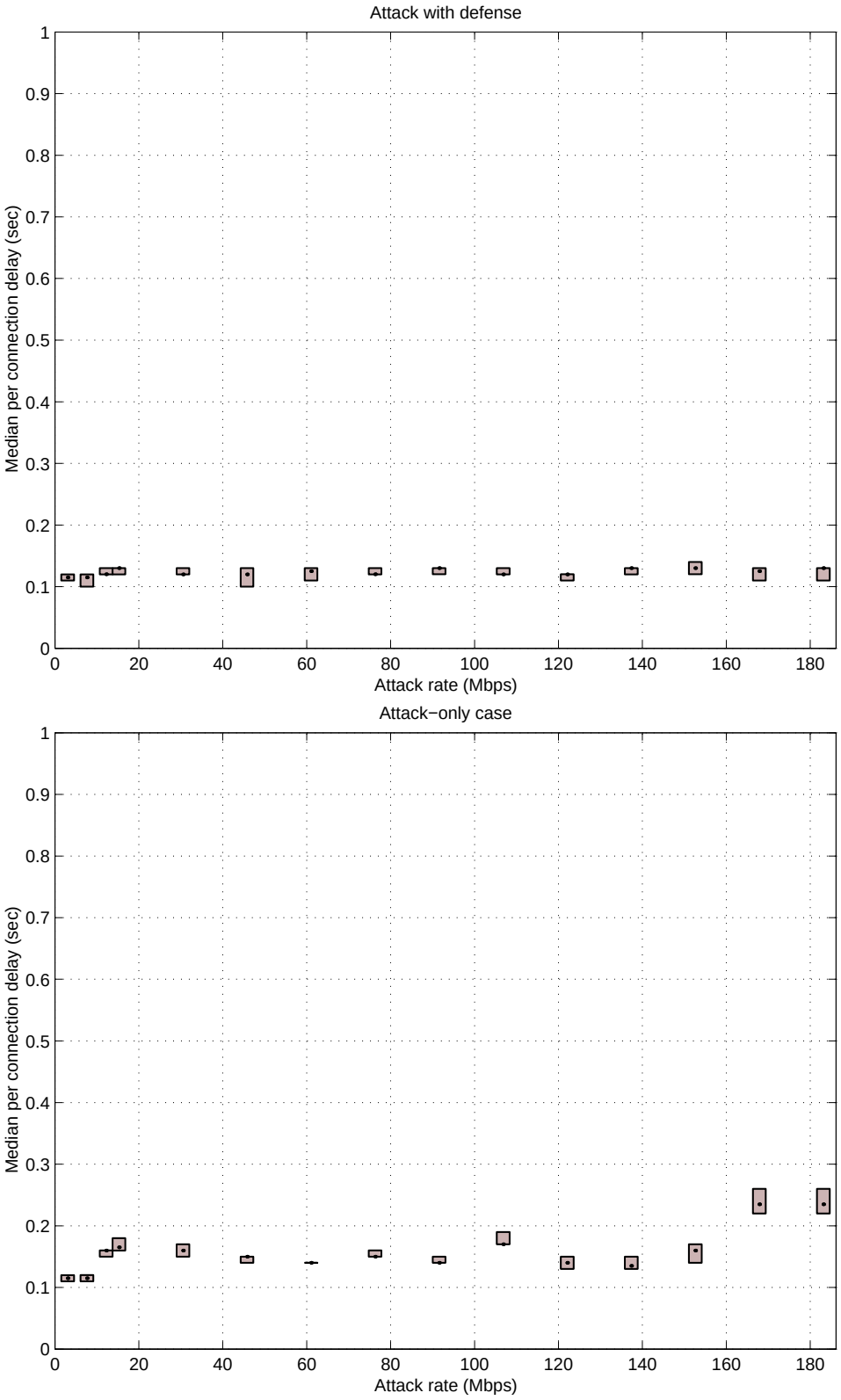

Figure 9.28: Median per-connection delay in the case of ICMP flooding attack, with and without defense

9.3.3 Connection Failure

Figure 9.29 depicts the total number of failed connections during the attack. D-WARD maintains very low levels of failed connections — less than 2 regardless of the attack rate. This is around 0.5% of the total legitimate connections.

9.3.4 Defense Performance

Figures 9.30 and 9.31 depict D-WARD detection and response times for an ICMP flooding attack. We observe that detection occurs within 1 to 2 seconds after the attack starts, and the response starts 2 to 3 seconds later.

Figure 9.32 shows the percentage of legitimate traffic dropped. As we can observe, this value is around zero.

Figure 9.33 depicts the percentage of attack traffic forwarded. For small attack rates, this value goes up to 100%. We attribute this to the fact that small attack rates do not produce a strong denial-of-service attack. Thus the response will not severely limit the attack traffic, reducing the attack's sending rate only so much as to alleviate the denial-of-service effect. At higher rates, the percentage of attack traffic forwarded drops to values from 3% to 4%, which agrees with our expectations.

9.4 TCP Attacks

In order to test D-WARD's performance with TCP attacks, we generate a TCP SYN flooding attack targeting the SSH port (22), using subnet spoofing of IP addresses and random spoofing of source ports. These spoofing techniques will generate a large number of TCP connections, thus exerting stress on D-WARD's

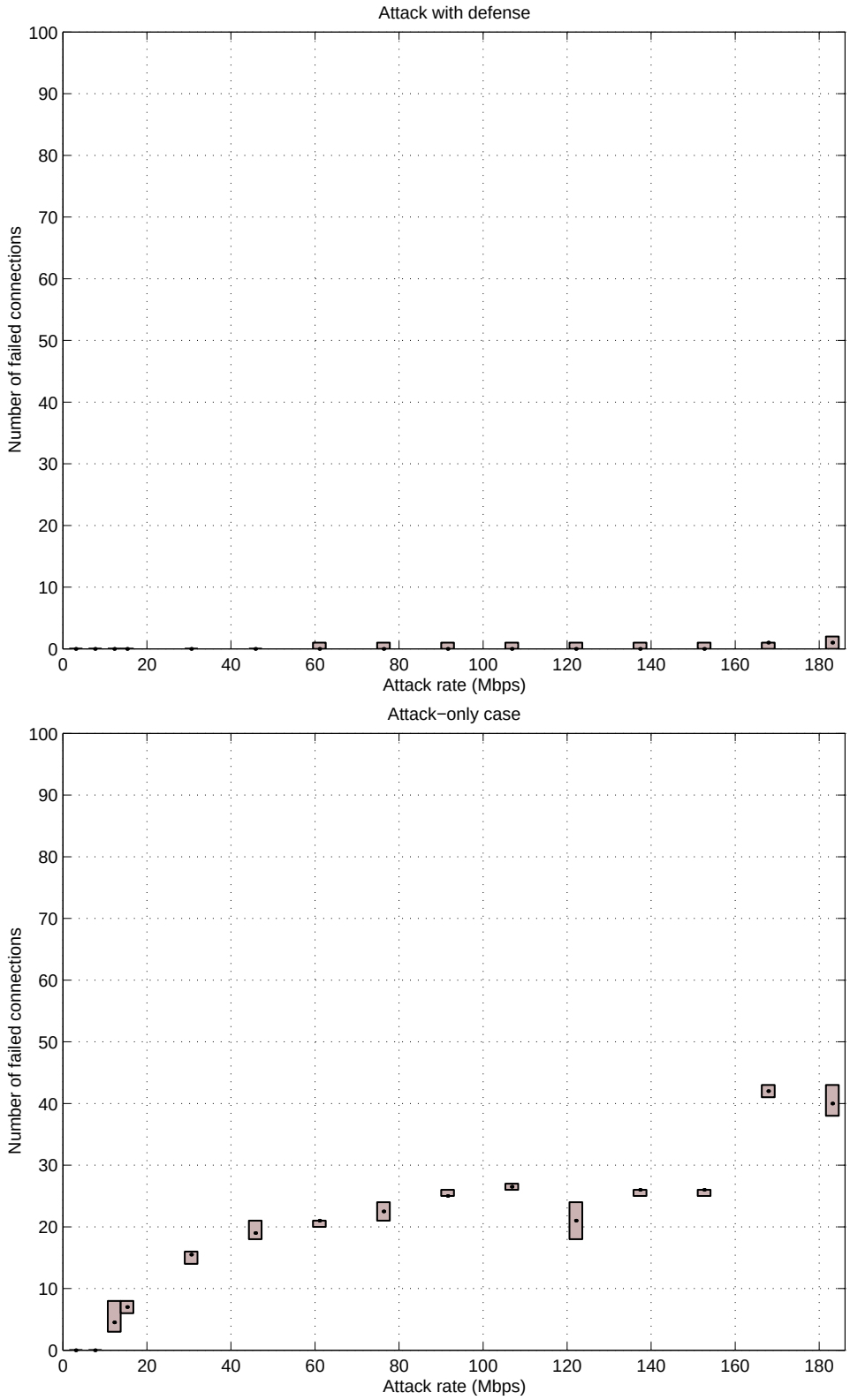

Figure 9.29: Number of failed connections in the case of ICMP flooding attack, with and without defense

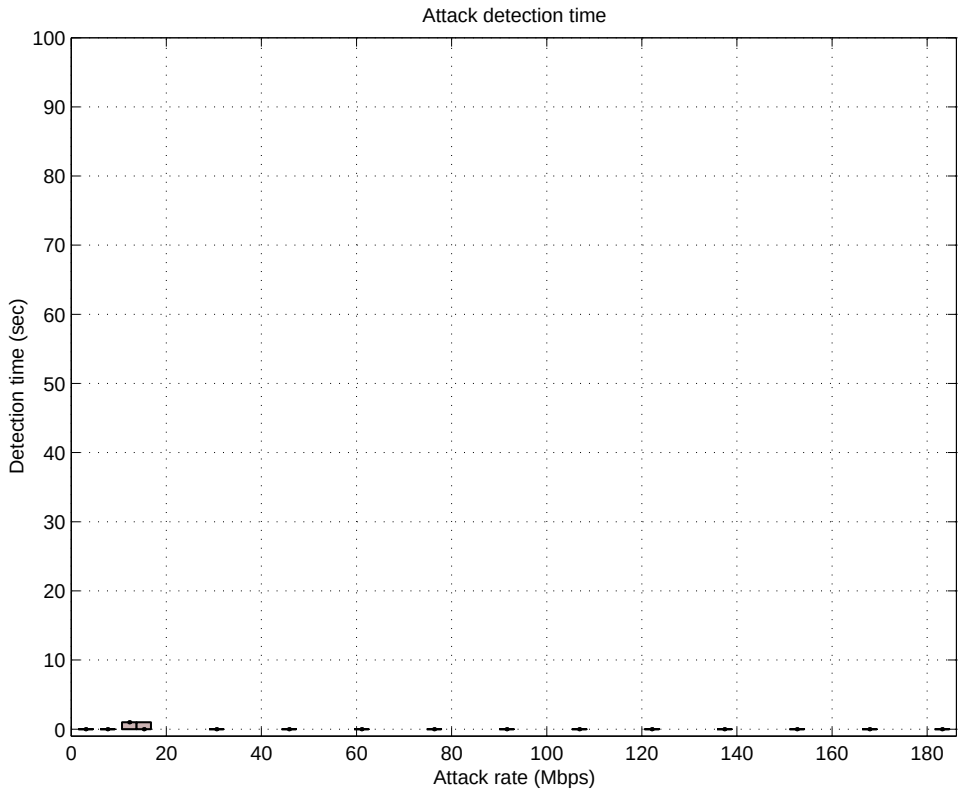

Figure 9.30: D-WARD attack detection time in the case of a ICMP flooding attack

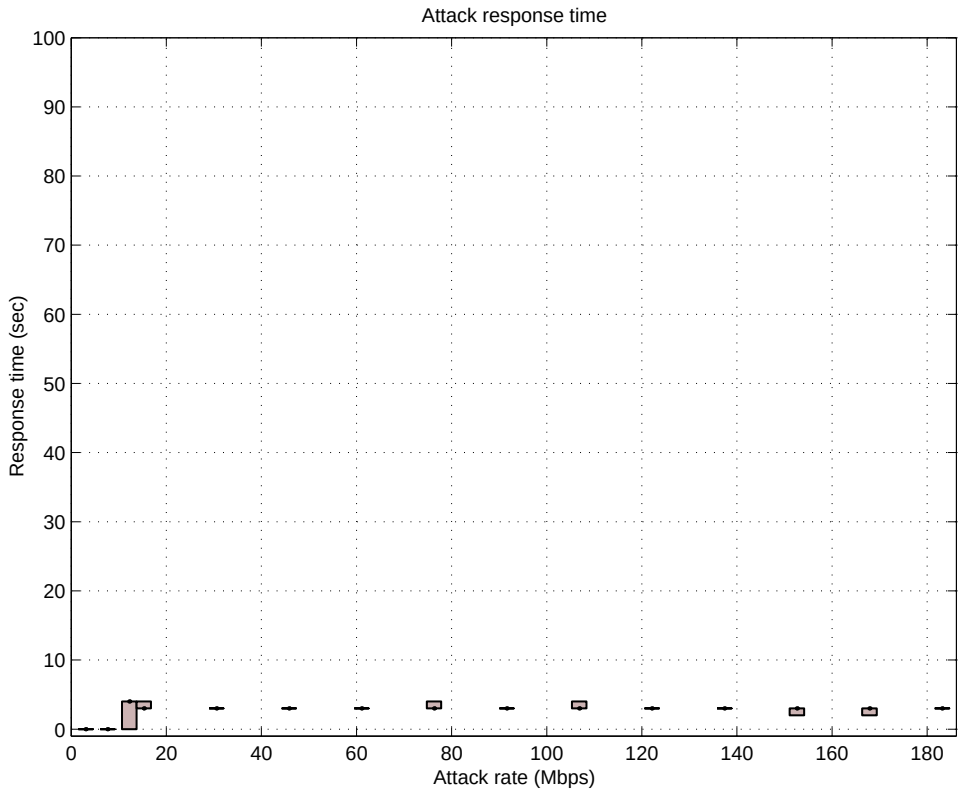

Figure 9.31: D-WARD response detection time in the case of ICMP flooding attack

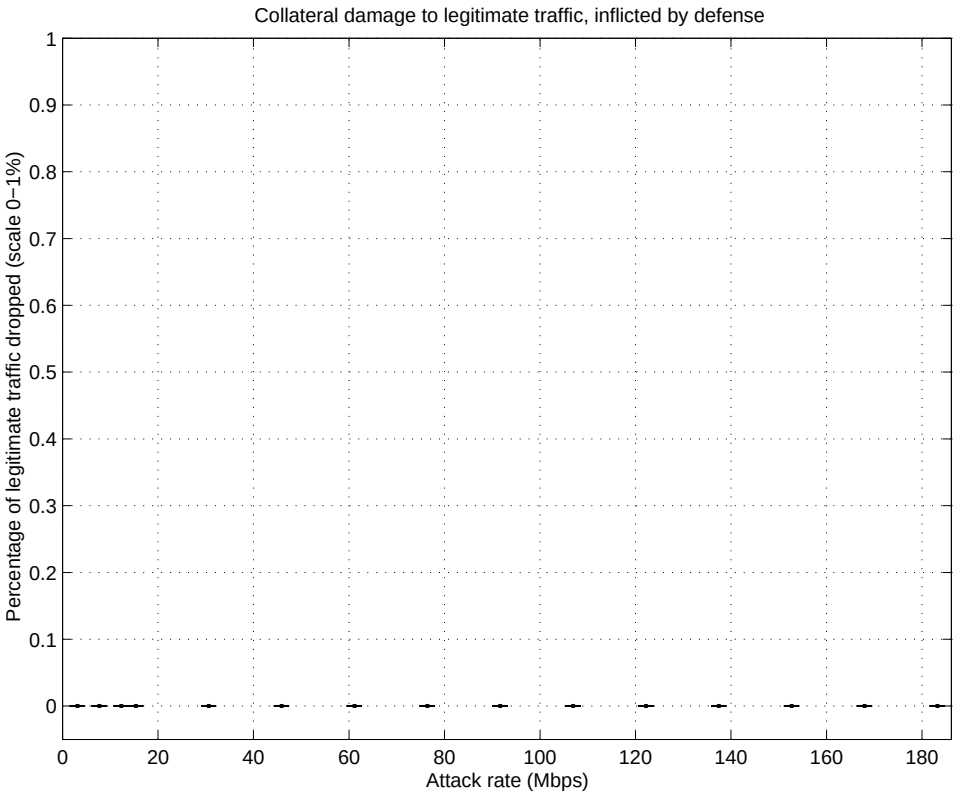

Figure 9.32: Percentage of legitimate traffic dropped by D-WARD in the case of ICMP flooding attack

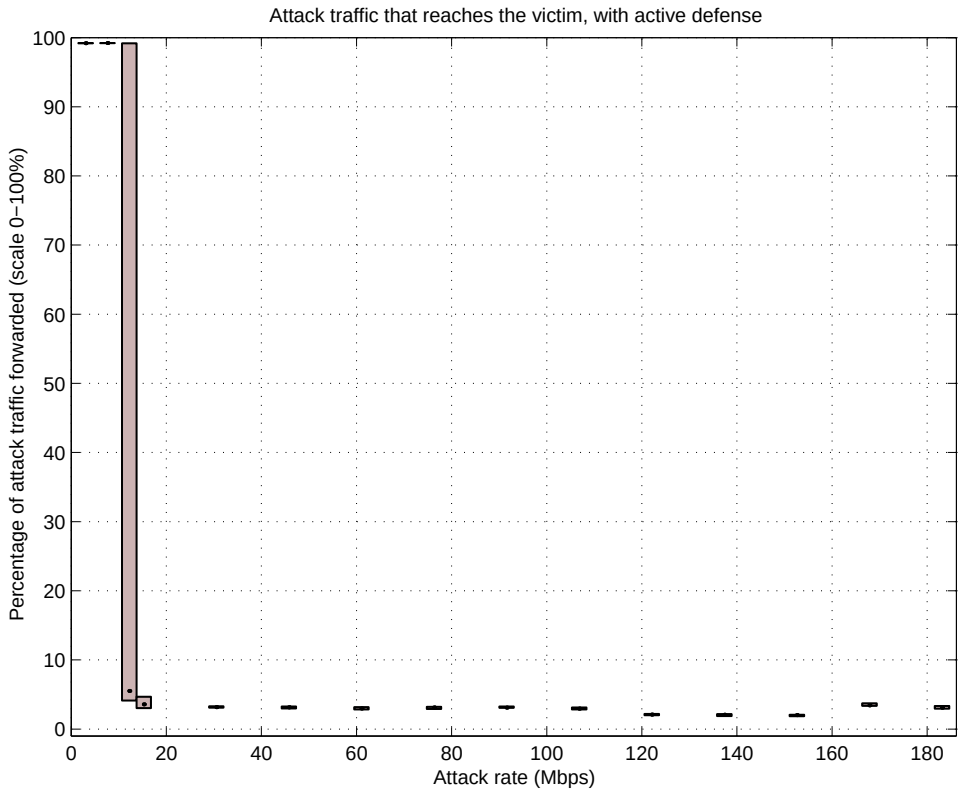

Figure 9.33: Percentage of attack traffic forwarded by D-WARD in the case of ICMP flooding attack

connection hash. Further, as all transactions use SSH and port 22, we maximize the attack effect on legitimate traffic. We test the whole range of attack rates that produce a denial-of-service effect. As there are two source networks generating the attack, the tested range is from 50 pps to 24Kpps. Both source networks host attackers that produce the same attack rates.

9.4.1 Service Level

Figure 9.34 depicts service level perceived by legitimate clients during the attack. As in the previous tests, D-WARD successfully relieves the denial-of-service effect at the victim, providing a 100% service level to legitimate clients during the attack.

9.4.2 Connection Delays

Figure 9.35 depicts total legitimate connection delay during the attack. D-WARD successfully maintains a total connection delay of around 200 seconds, regardless of the attack rate. This value is larger than the total connection delay in the UDP and ICMP attack cases, but it is still less than 20% of the total connection delay in the attack-only case.

Figure 9.36 depicts maximum legitimate connection delay during the attack. We observe that D-WARD successfully controls connection delay, keeping maximum delay per connection lower than 40 seconds. As explained in Section 9.2.2, the reason for such a long delay lies in the TCP congestion control mechanism that is activated by legitimate packet drops due to congestion in the short period (2 to 5 seconds) while D-WARD is being activated.

Figure 9.37 depicts median legitimate connection delay during the attack. Me-

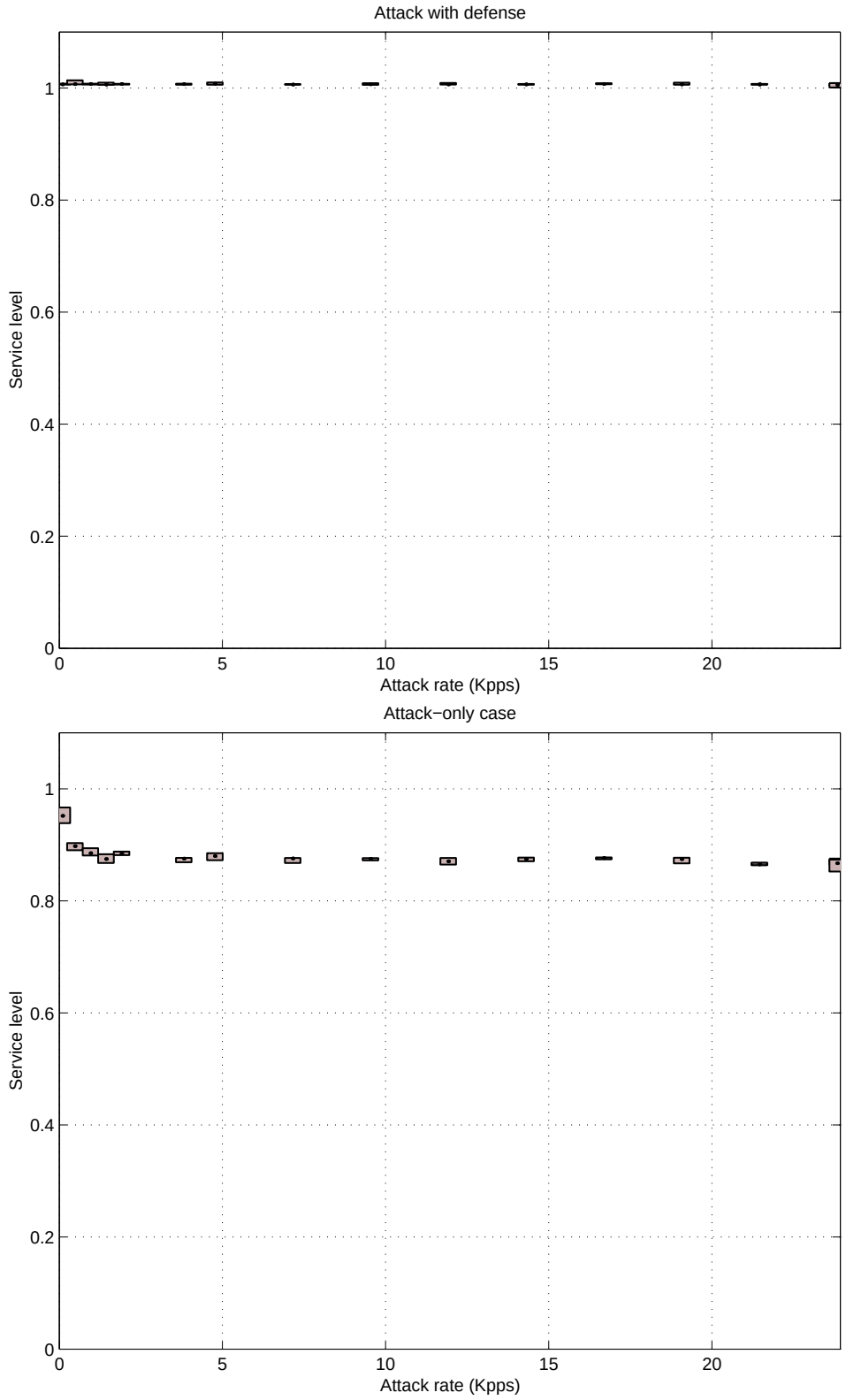

Figure 9.34: Service level in the case of a TCP SYN flooding attack, with and without defense

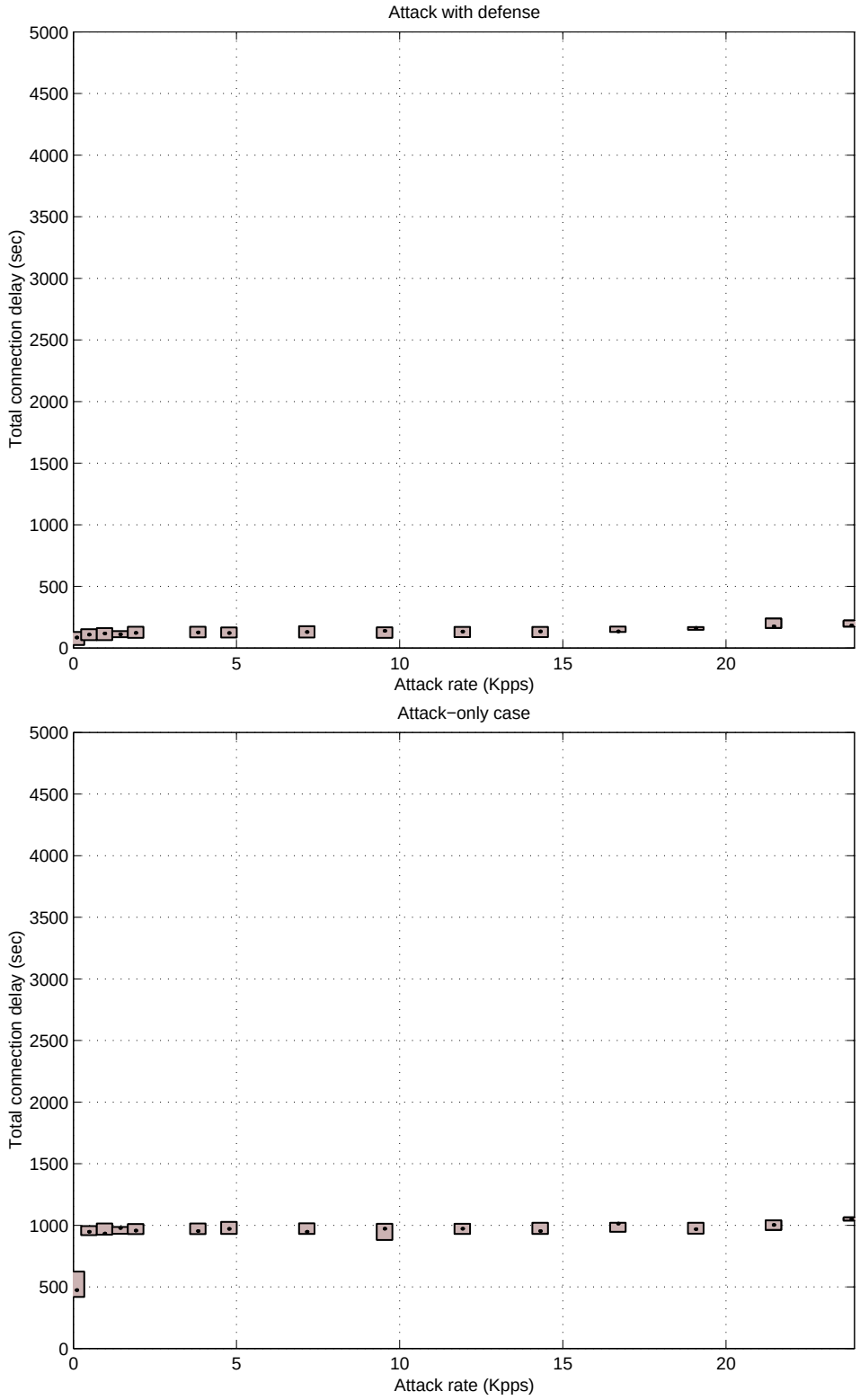

Figure 9.35: Total connection delay in the case of a TCP SYN flooding attack, with and without defense

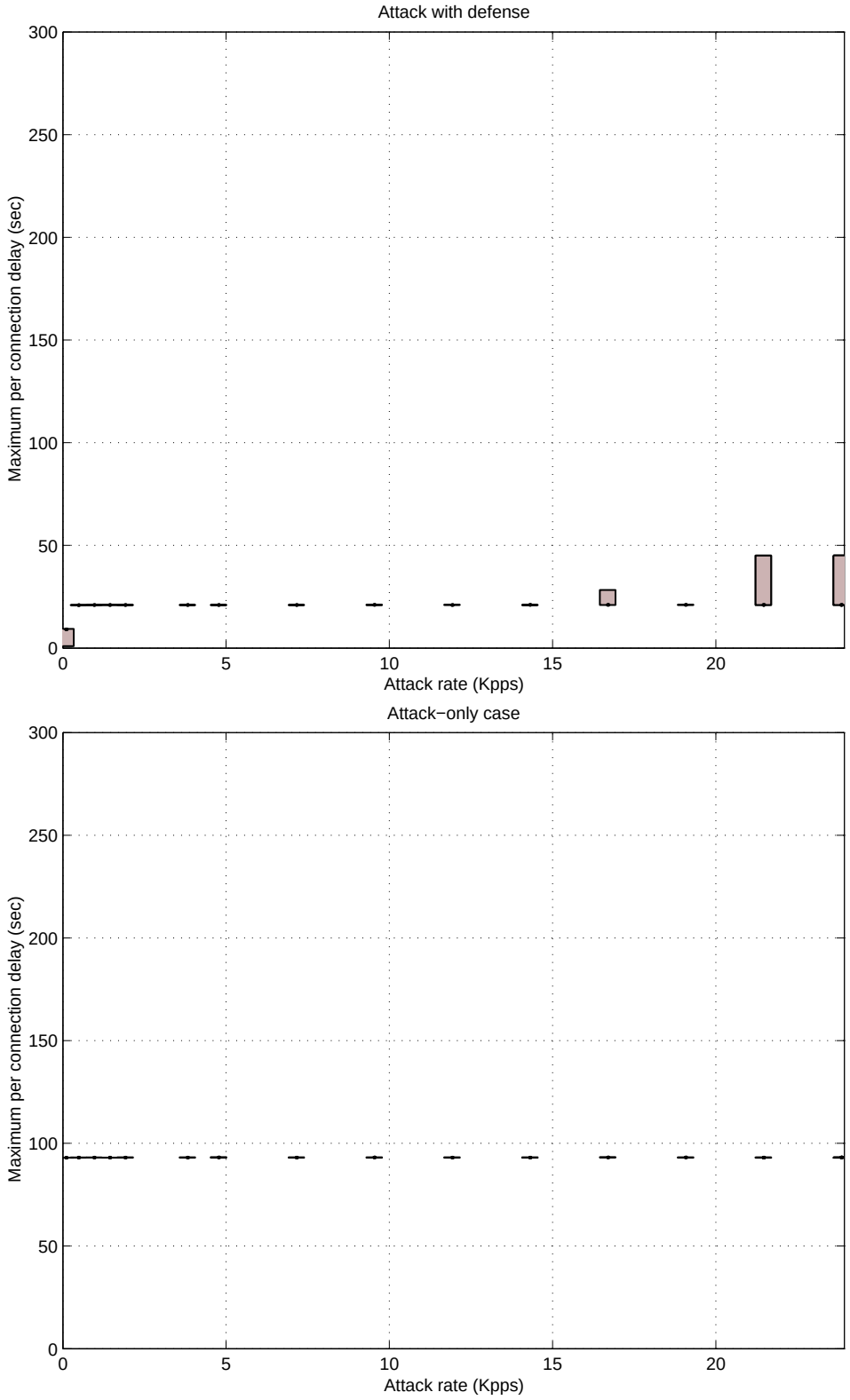

Figure 9.36: Maximum per-connection delay in the case of TCP SYN flooding attack, with and without defense

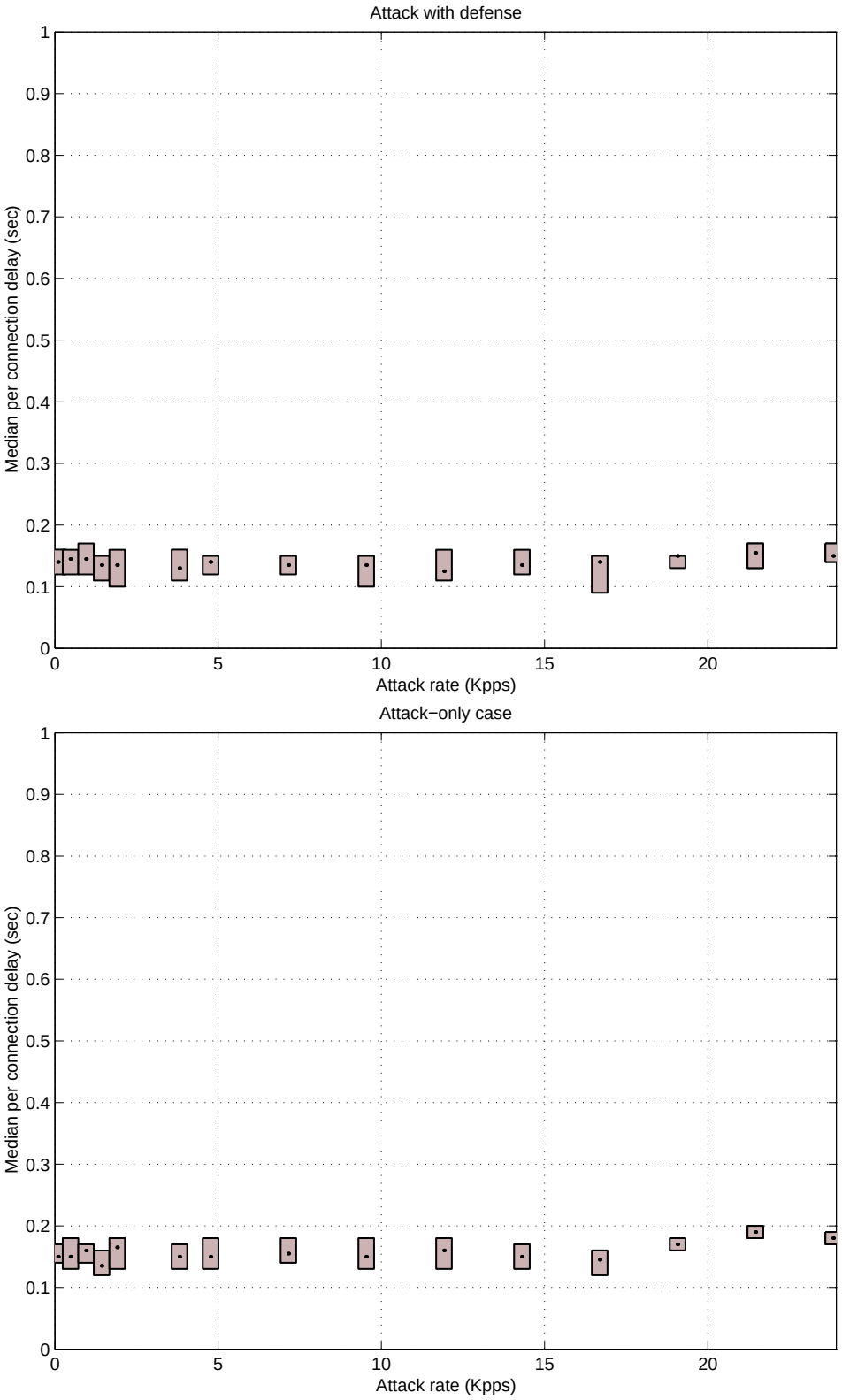

Figure 9.37: Median per-connection delay in the case of TCP SYN flooding attack, with and without defense

dian delay was already small in the attack-only case, and D-WARD just maintains this level.

9.4.3 Connection Failure

Figure 9.38 depicts total number of failed connections during the attack. D-WARD maintains low levels of failed connections — less than 7 regardless of the attack rate. This is less than 2% of the total legitimate connections.

9.4.4 Defense Performance

Figures 9.39 and 9.40 depict D-WARD detection and response times for UDP flooding attack. We observe that detection takes up to 20 seconds for the smallest attack rate (100 pps) but becomes prompt for higher attack rates (around 1 to 2 seconds). The response starts around 5 seconds after the detection. The detection delay for small attacks occurs because small attack rates do not create the denial-of-service effect promptly, but rather do so after some time. Thus detection cannot observe any anomalous events until connection buffer fills up; it therefore lags after the actual onset of the attack.

Figure 9.41 depicts percentage of legitimate traffic dropped. As we can observe, this value is around zero.

Figure 9.42 depicts percentage of attack traffic forwarded. For small attack rates, this value goes up to 20%. Small attack rates do not produce a strong denial-of-service attack, which delays detection and response. At higher rates, the percentage of attack traffic forwarded drops to values from 5% to 10% which agrees with our expectations, based on attack detection and response times.

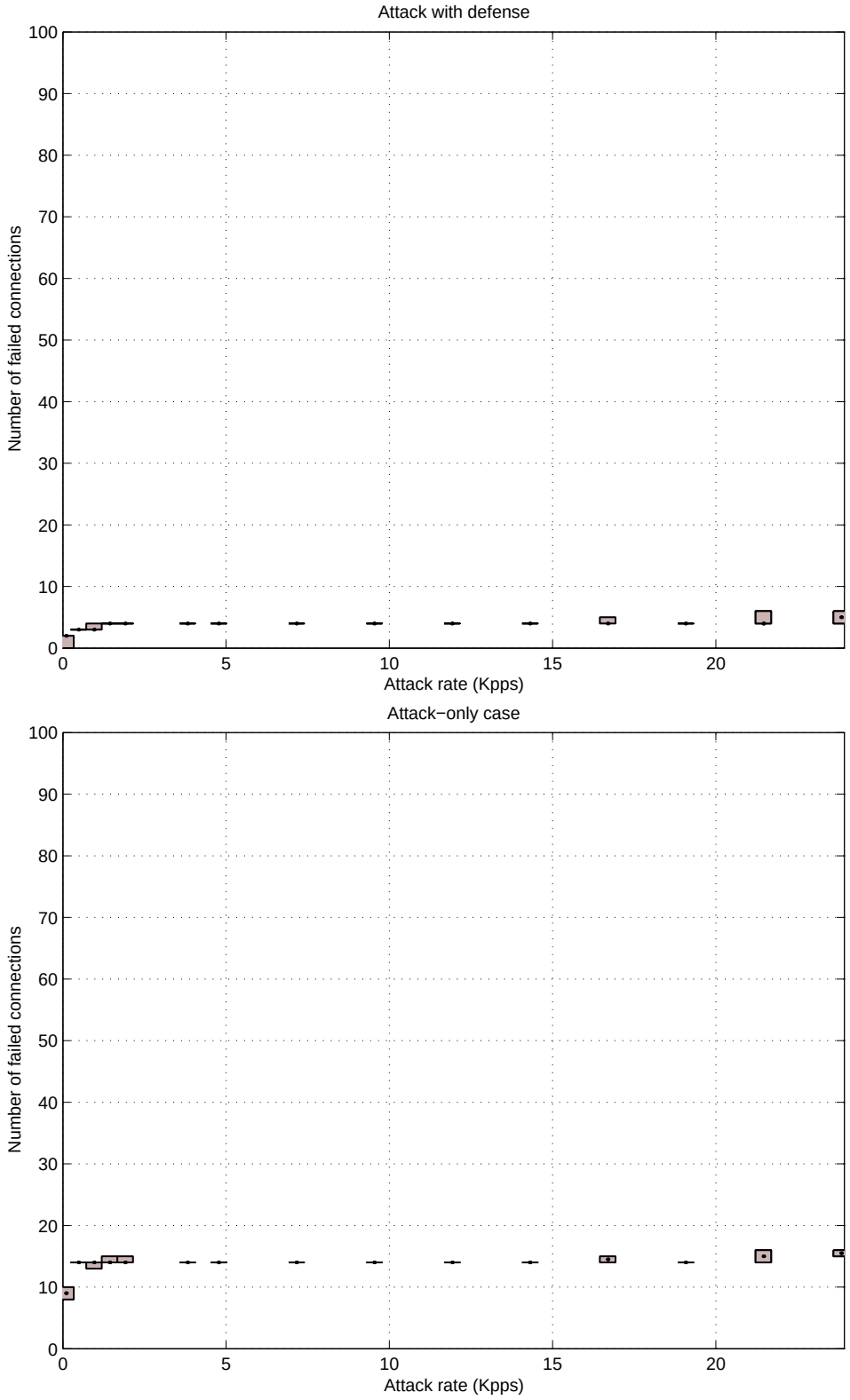

Figure 9.38: Number of failed connections in the case of a TCP SYN flooding attack, with and without defense

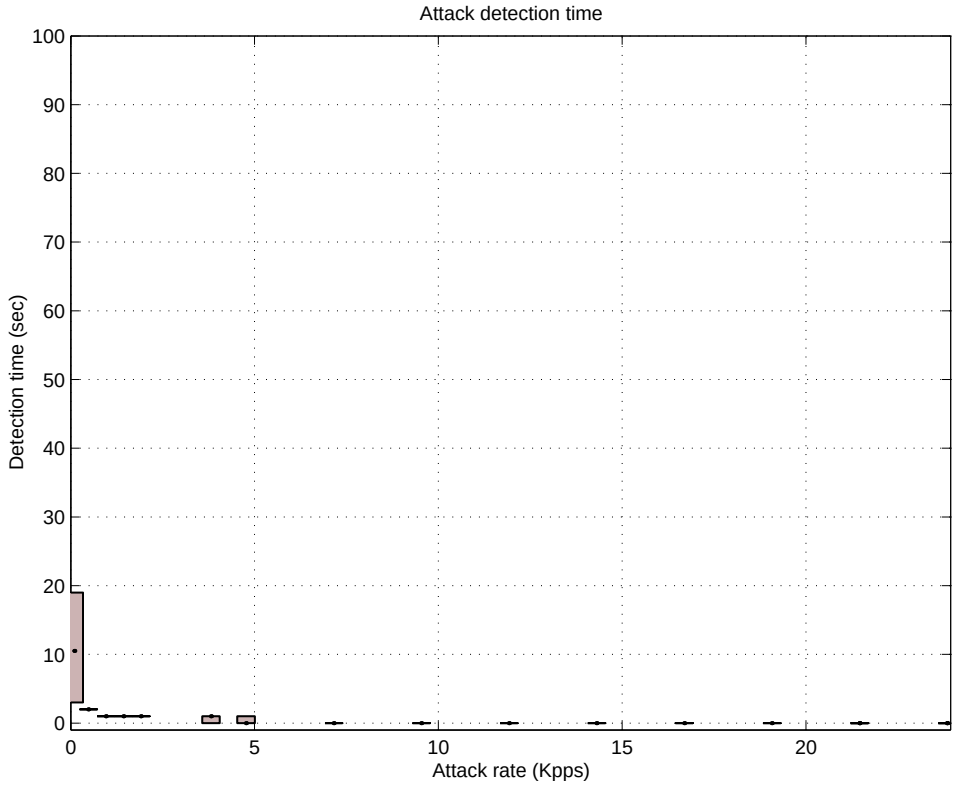

Figure 9.39: D-WARD attack detection time in the case of TCP SYN flooding attack

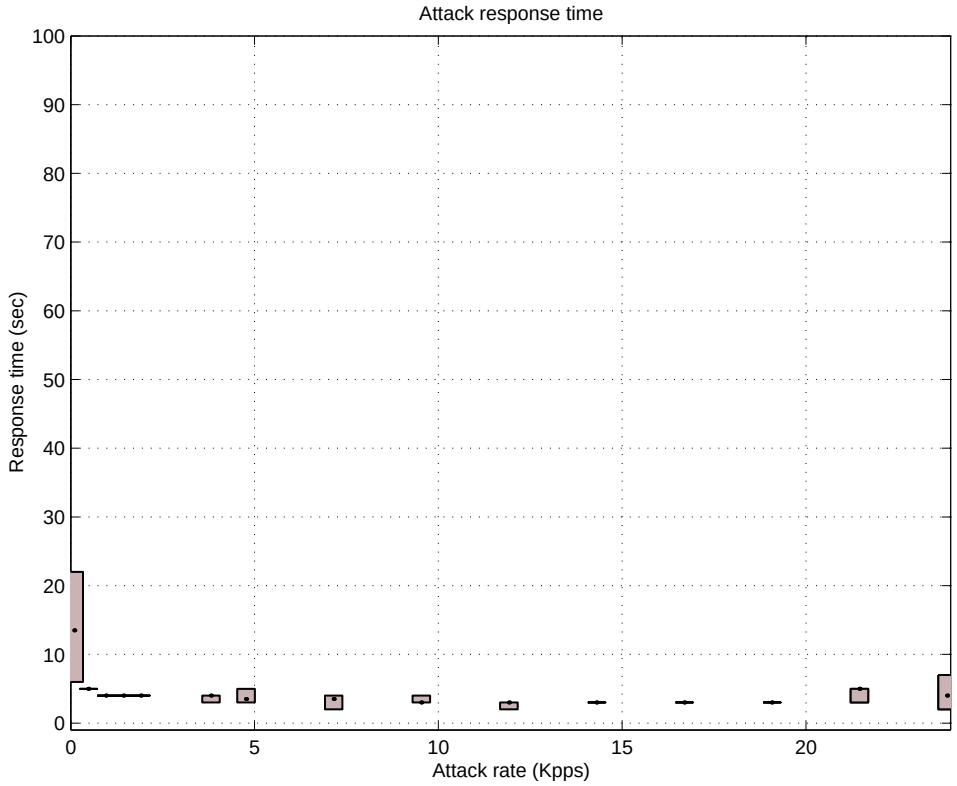

Figure 9.40: D-WARD response detection time in the case of TCP SYN flooding attack

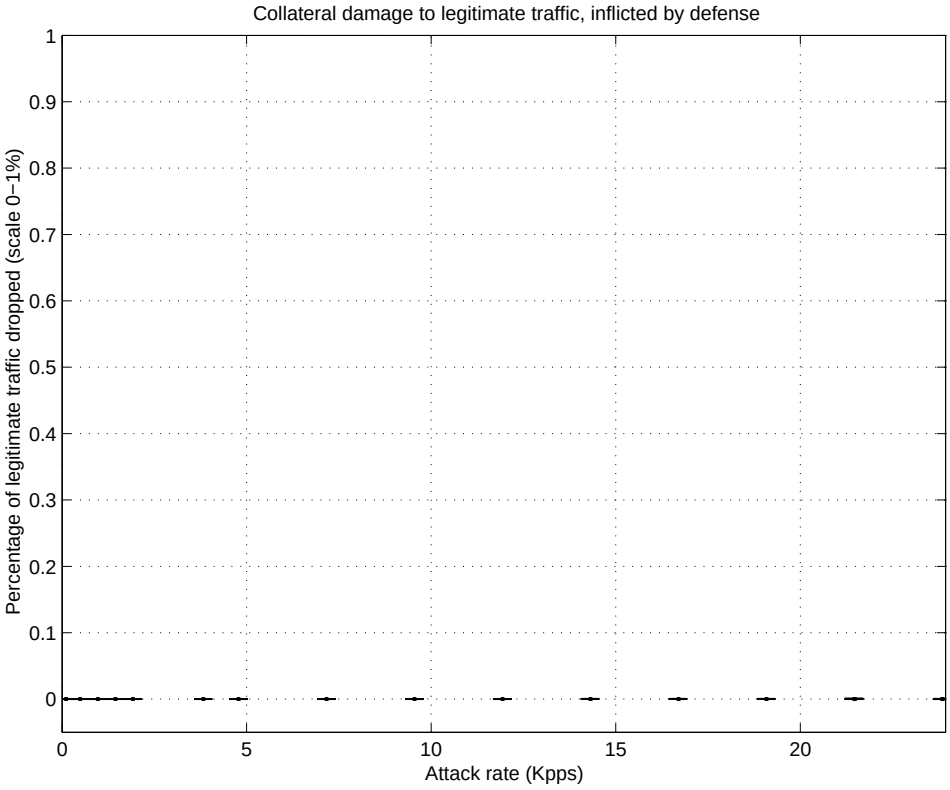

Figure 9.41: Percentage of legitimate traffic dropped by D-WARD in the case of TCP SYN flooding attack

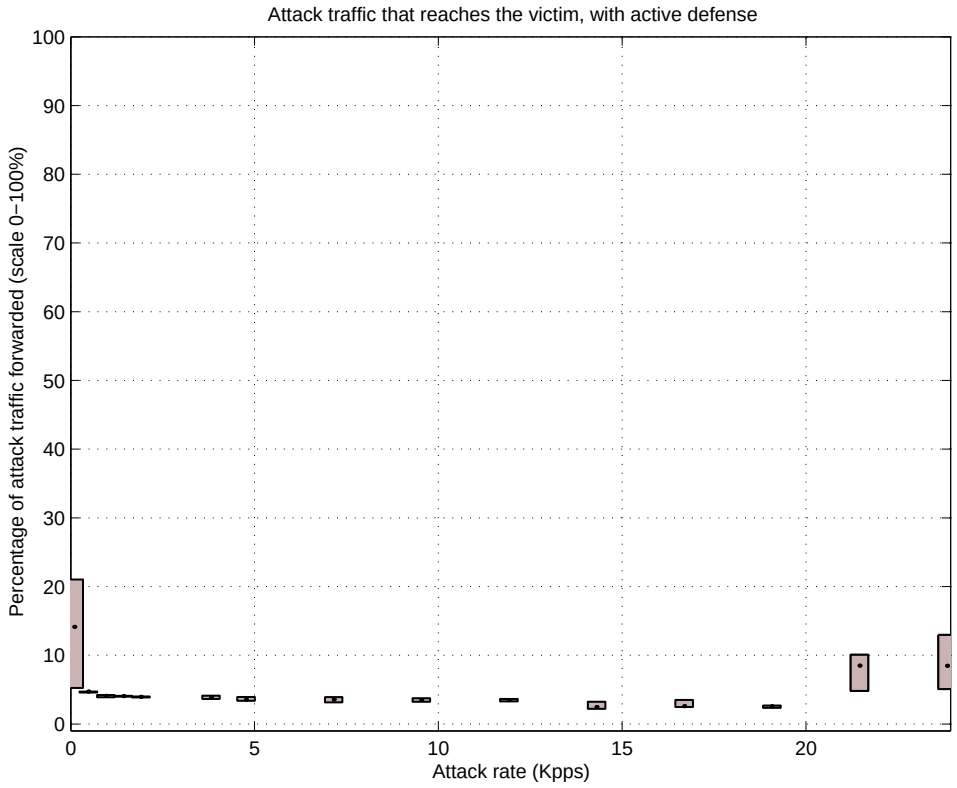

Figure 9.42: Percentage of attack traffic forwarded by D-WARD in the case of TCP SYN flooding attack

9.5 Detection Limits

In order to measure D-WARD's sensitivity for attack detection, we use Topology 1 (shown in Figure 8.4). We generate very small rate attacks from both source networks. We also turn on non-validated residue detection, thus enabling D-WARD to detect TCP and ICMP attacks early. While none of the generated attacks should cause a denial-of-service effect, our measurements show that we can still successfully detect and stop these attacks.

The smallest TCP attack detected by D-WARD is an one-packet-per-second attack. It is detected after 7 seconds. The smallest ICMP attack detected by D-WARD is also an one-packet-per-second attack. It is detected after 8 seconds. Both of these results are expected. As the non-validated residue detection only looks at the continued presence of non-validated traffic, it should detect a very small rate attack easily. The detection time depends only on the N_{bursts} configuration parameter and should be between N_{bursts} and $2 * N_{bursts}$ observation intervals. In our tests this produces a 5 to 10 second detection range.

The smallest UDP attack detected by D-WARD depends on the level of IP spoofing and new connection generation.[2] When the attacker deploys random spoofing, the smallest UDP attack detected by D-WARD is a 100-packet-per-second attack.

The attacker may also try to generate pulsing attacks to avoid detection. Large-rate pulsing attacks will be detected by observing resulting anomalies in TCP and ICMP flow statistics. Small-rate pulsing attacks with an active interval

[2]Recall that non-validated residue detection is not engaged for UDP traffic, as legitimate UDP connection models do not cover all possible UDP connections. Thus it is likely that non-validated traffic in the UDP case belongs to legitimate connections whose legitimate models do not exist in D-WARD. The UDP attack detection is only done based on the legitimate UDP flow models.

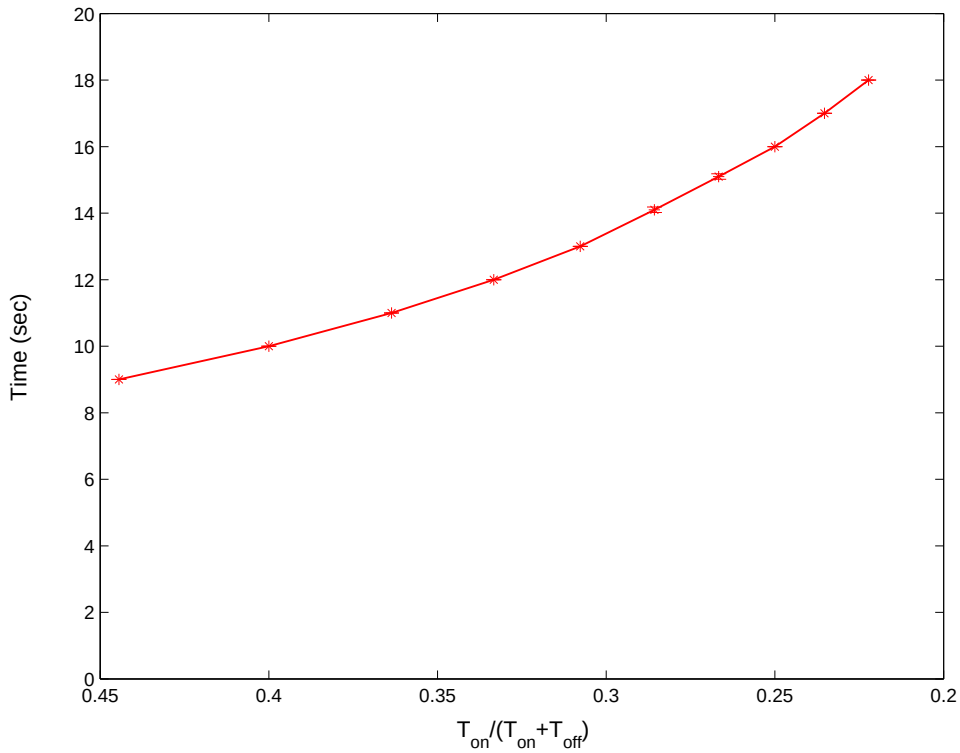

Figure 9.43: Detection time in pulsing attack case

larger than N_{bursts} will also be detected using continuous non-validated residue detection. Therefore, the attacker can only try to avoid detection by using small-rate pulsing attacks with an active interval shorter than N_{bursts}.

We ran a set of experiments to determine D-WARD detection performance for this type of attack. The duration of active periods, T_{on}, was fixed at $N_{bursts}-1$, i.e., 4 seconds, while the duration of inactive periods, T_{off}, was varied from 4 to 14, effectively changing the ratio $\frac{T_{on}}{T_{on}+T_{off}}$ from 0.5 to 0.2.

Figure 9.43 shows the detection time for pulsing attacks. The detection time depends exponentially on the ON/OFF period ratio but is reasonably small even for very small ratios (e.g., for an attack whose inactive intervals last five times its active intervals, the detection time was 18 seconds). This means that the attacker

who would like to avoid detection will have to perform small-rate, small-duration, repetitive attacks with large inactive periods, which forces him to subvert a much larger number of machines than before.

9.6 Synchronization Effects

In order to examine whether D-WARD encounters some synchronization issues when widely deployed, we constructed a large-scale experiment with the Topology 2 (shown in Figure 8.5). We now repeat the above experiments (UDP, ICMP and TCP SYN flood) with this topology. We generate the same amount of legitimate traffic, now spread over 10 source networks. We also generate the same sets of attacks as in Sections 9.2, 9.3 and 9.4, in the sense that the victim receives the same level of attack traffic as in the small-scale tests. The attack force is now spread over all ten networks, so that each network generates $\frac{1}{10}$ of the attack.

9.6.1 Large-Scale UDP Attacks

Figure 9.44 depicts service level values for the large-scale UDP attack case (the left graph). We also repeat the Figure 9.16 on the right, for comparison. As we can observe these two graphs are almost identical. The attack curve has a slightly different shape which we attribute to attack synchronization effects due to distribution. However, the defense curve is still the same, providing 100% service level to legitimate traffic.

Figures 9.45, 9.46 and 9.47 depict total connection delay and maximum and median per-connection delays for the large-scale UDP attack case. We also repeat Figures 9.17, 9.18 and 9.19 for comparison. Figures depicting total connection delay differ in the shape of the attack curve. Total delay is larger in the large-scale

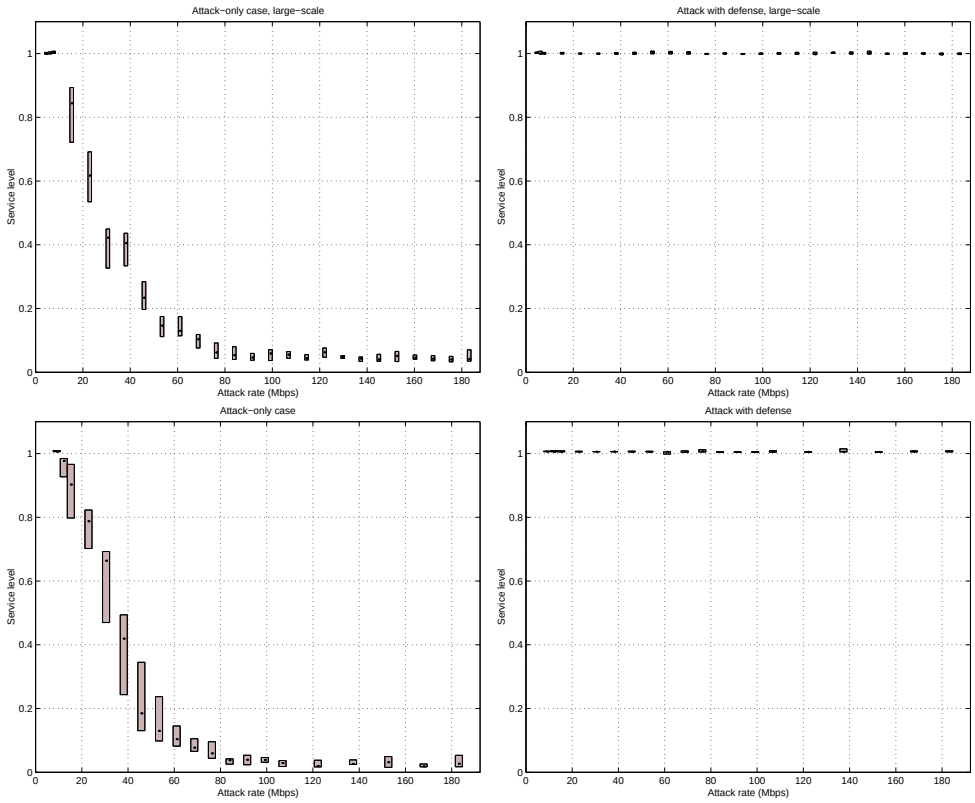

Figure 9.44: Service level in the case of large- and small-scale UDP flooding attack

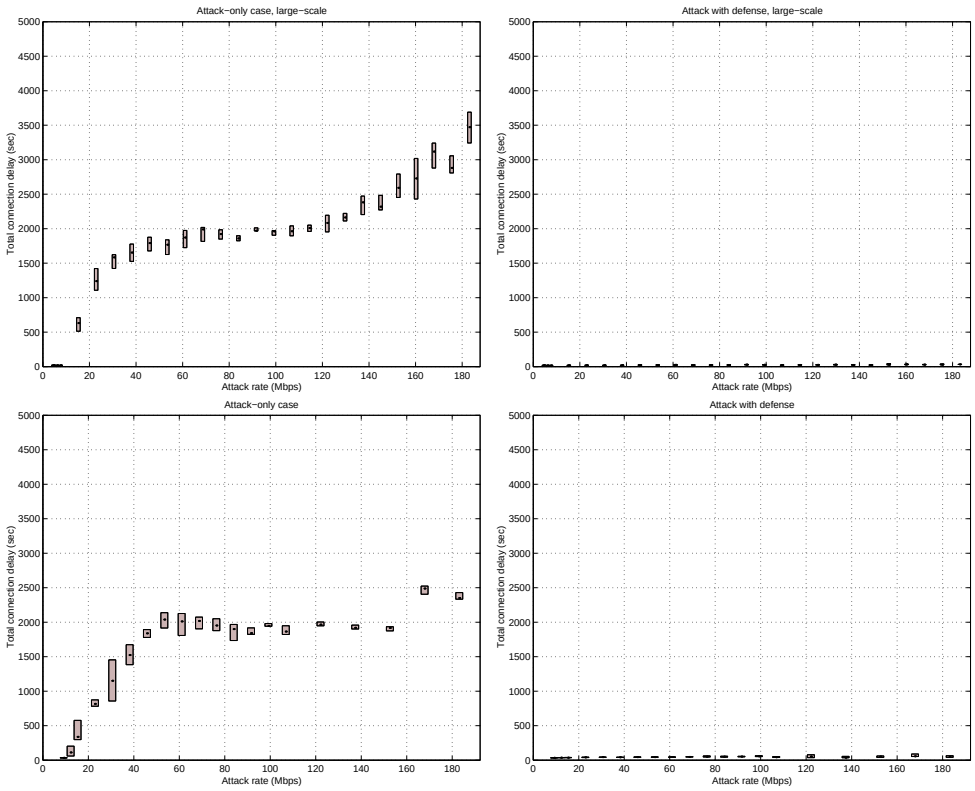

Figure 9.45: Total connection delay in the case of large- and small-scale UDP flooding attack

experiments. However, defense performance remains the same, keeping the total connection delay at a very low level. Figures depicting median and maximum per-connection delay are almost identical. Defense performance is even better in the large-scale experiment. We attribute this to the fact that each source network carries a smaller traffic load in the large-scale than in the small-scale experiment, thus D-WARD can better profile the traffic (legitimate connection records can be kept in hash tables) and control the attack.

Finally, Figure 9.48 depicts the number of failed connections for the large-scale UDP attack case. We also repeat the Figure 9.20 next to this for comparison. Again number of failed connections rises more rapidly for the attack-only case in

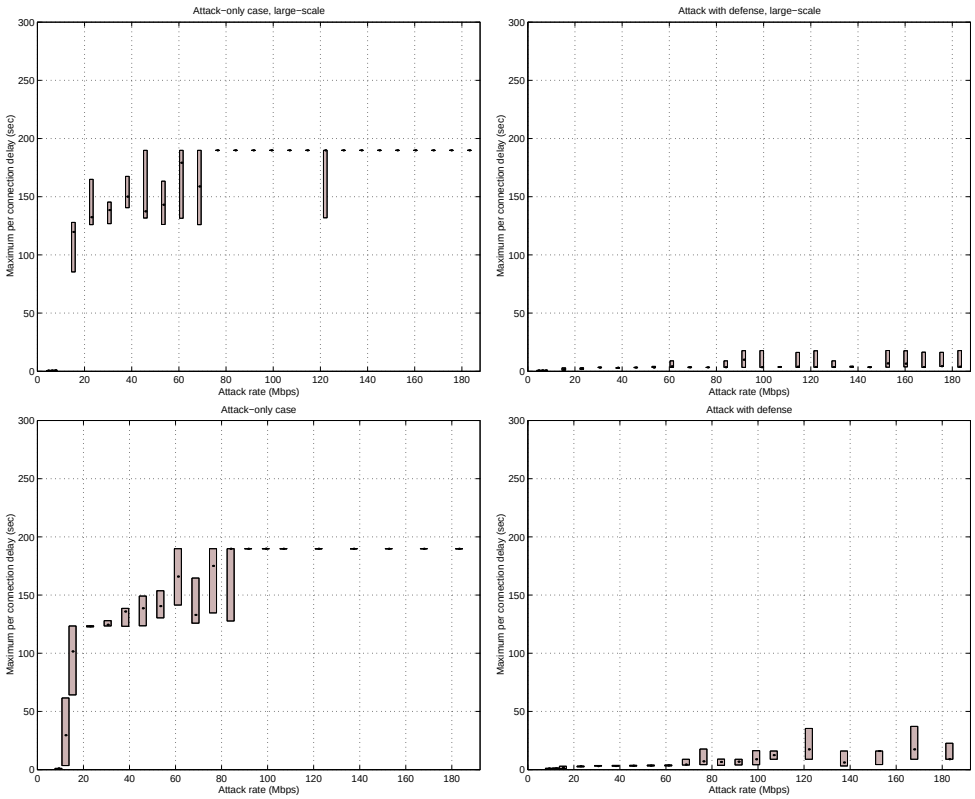

Figure 9.46: Maximum per-connection delay in the case of large- and small-scale UDP flooding attack

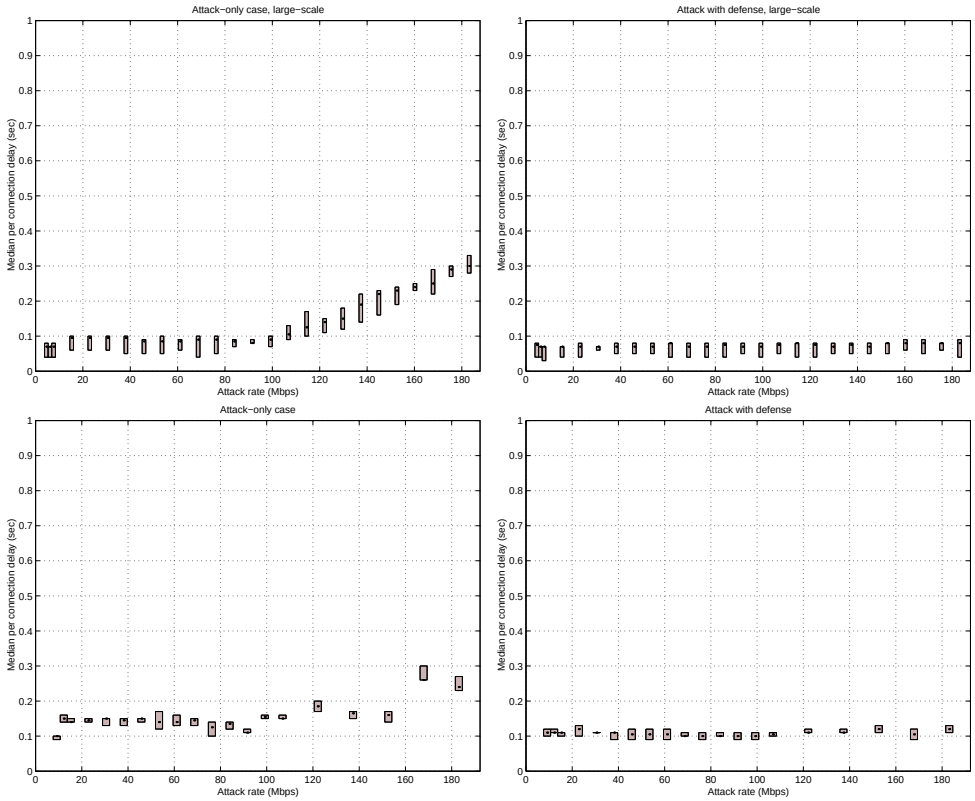

Figure 9.47: Median per-connection delay in the case of large- and small-scale UDP flooding attack

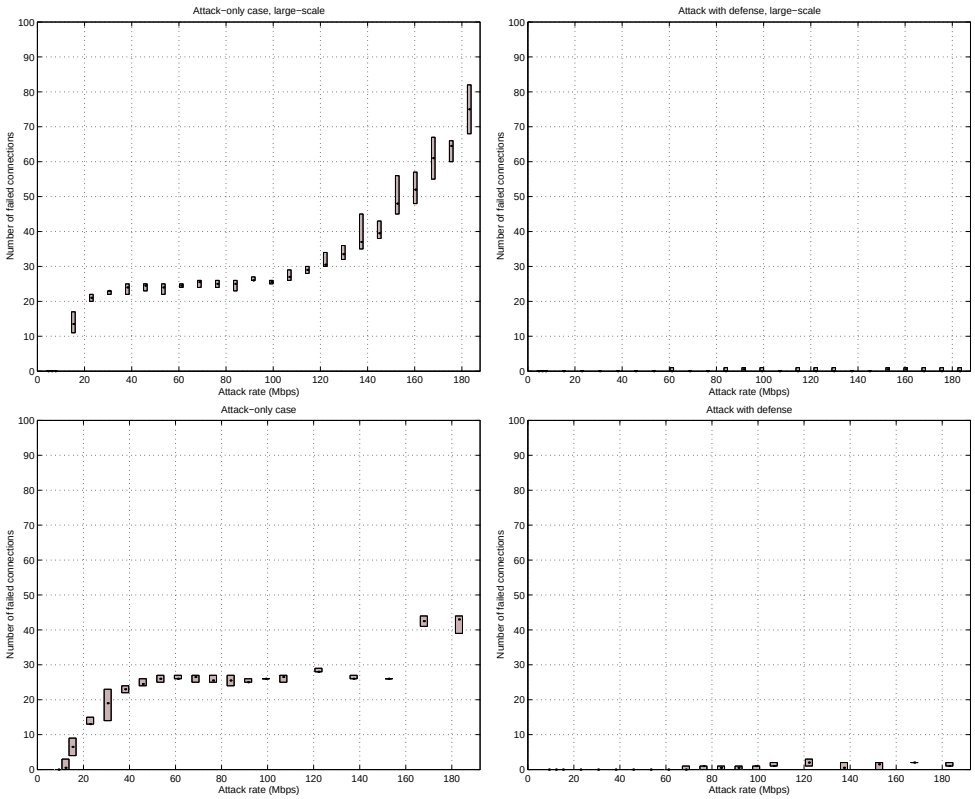

Figure 9.48: Number of failed connections in the case of large- and small-scale UDP flooding attack

the large-scale experiments. However, defense performance is comparable to, and even slightly better than performance in the small-scale experiments, keeping the number of failed connections at a very low level.

9.6.2 Large-Scale ICMP Attacks

Figure 9.49 depicts service level values for the large-scale ICMP attack case. We also repeat Figure 9.25 next to this for comparison. These two figures are almost identical. The shape of the attack curve again shows more variability in the large-scale experiments due to agent synchronization. The defense successfully

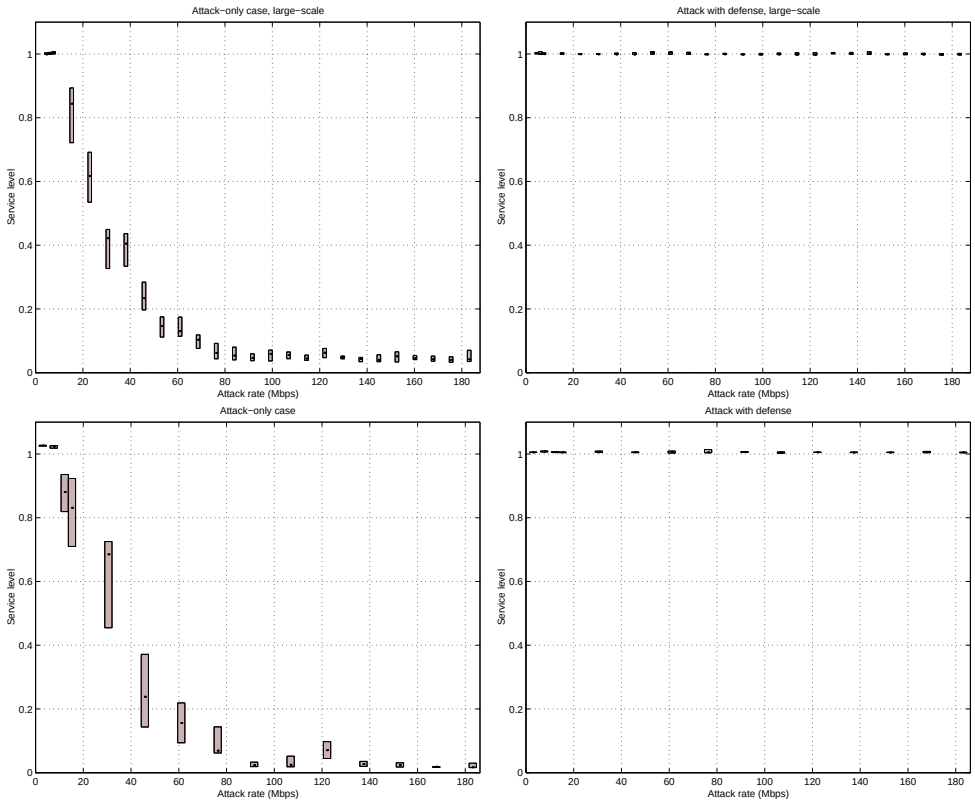

Figure 9.49: Service level in the case of large- and small-scale ICMP flooding attack

controls the attack, bringing a 100% service level to legitimate clients.

Figures 9.50, 9.51 and 9.52 depict the total connection delay and maximum and median per-connection delays for the large-scale ICMP attack case. We also repeat Figures 9.26, 9.28 and 9.27 for comparison. Figures depicting total connection delay show the same phenomenon as in large-scale UDP experiments — total delay in the attack-only case is larger in the large-scale than in the small-scale experiments. The defense performance is comparable, and even slightly better, in large-scale experiments. Figures depicting maximum and median per-connection delay are almost identical.

Finally, Figure 9.53 depicts number of failed connections for the large-scale

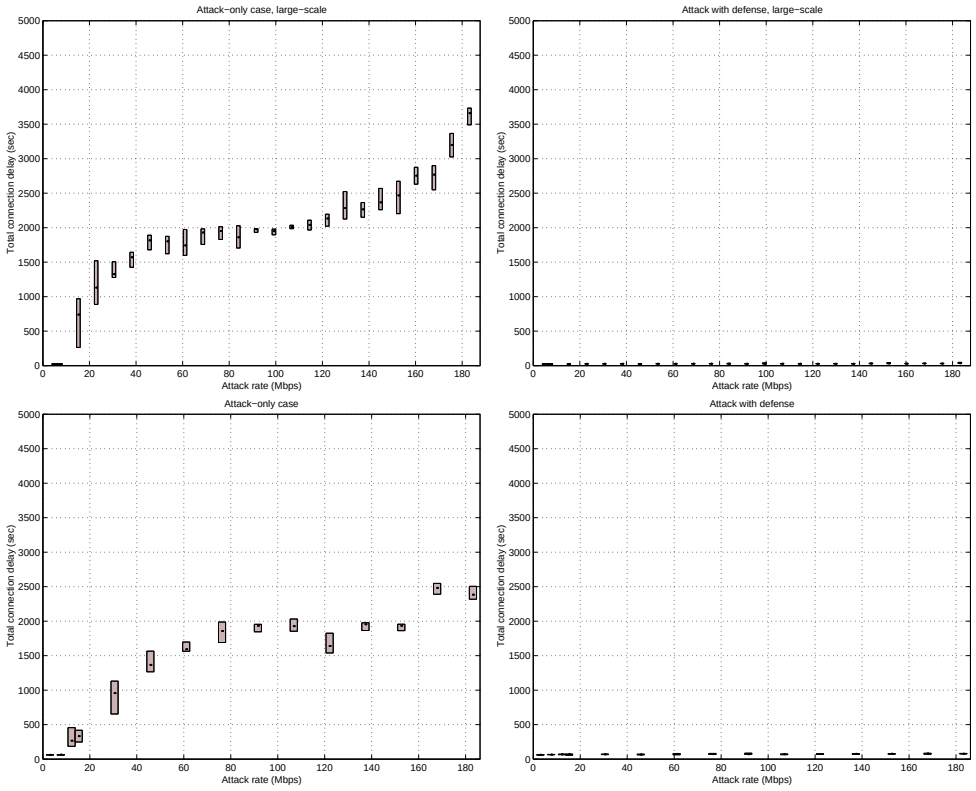

Figure 9.50: Total connection delay in the case of large- and small-scale ICMP flooding attack

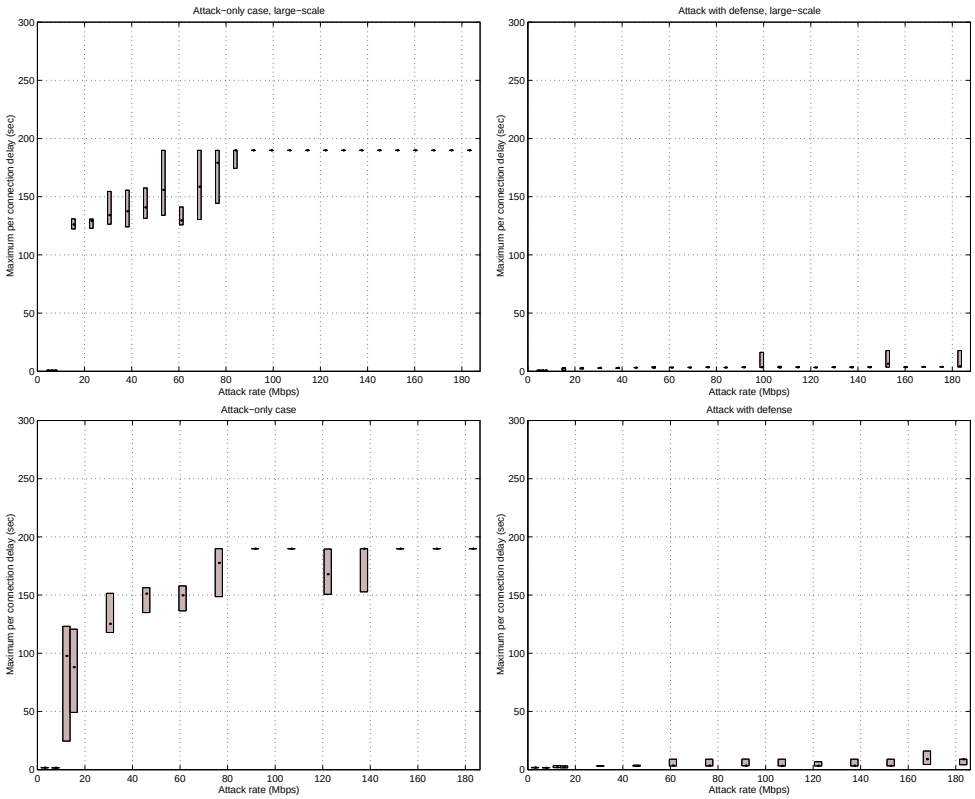

Figure 9.51: Maximum per-connection delay in the case of large- and small-scale ICMP flooding attack

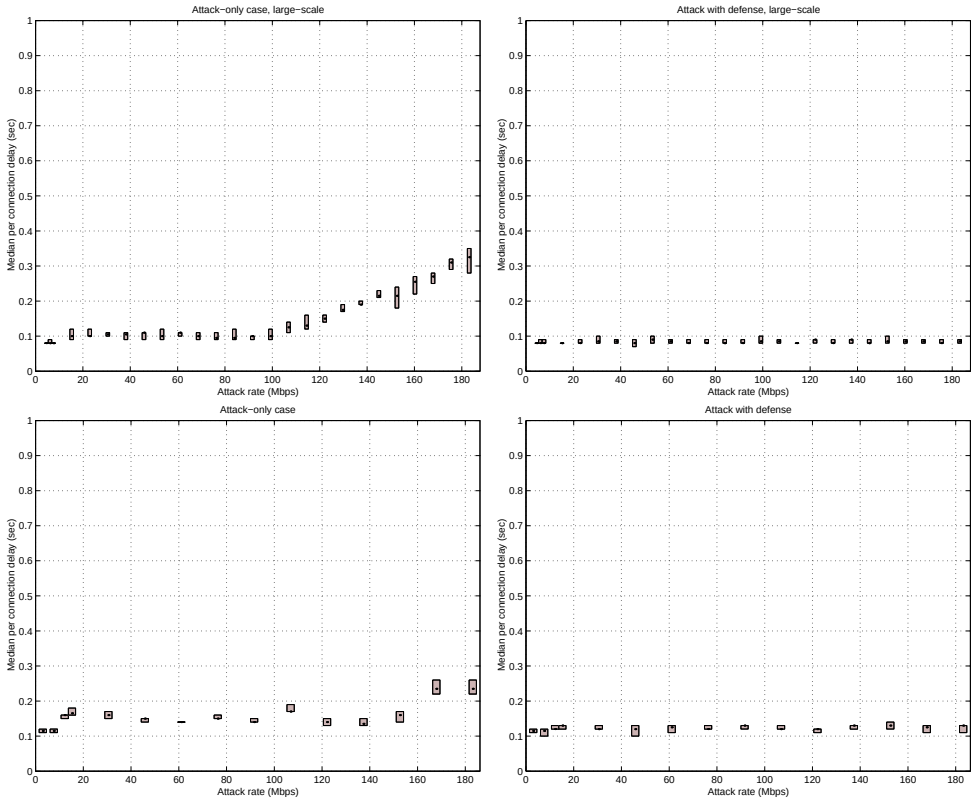

Figure 9.52: Median per-connection delay in the case of large- and small-scale ICMP flooding attack

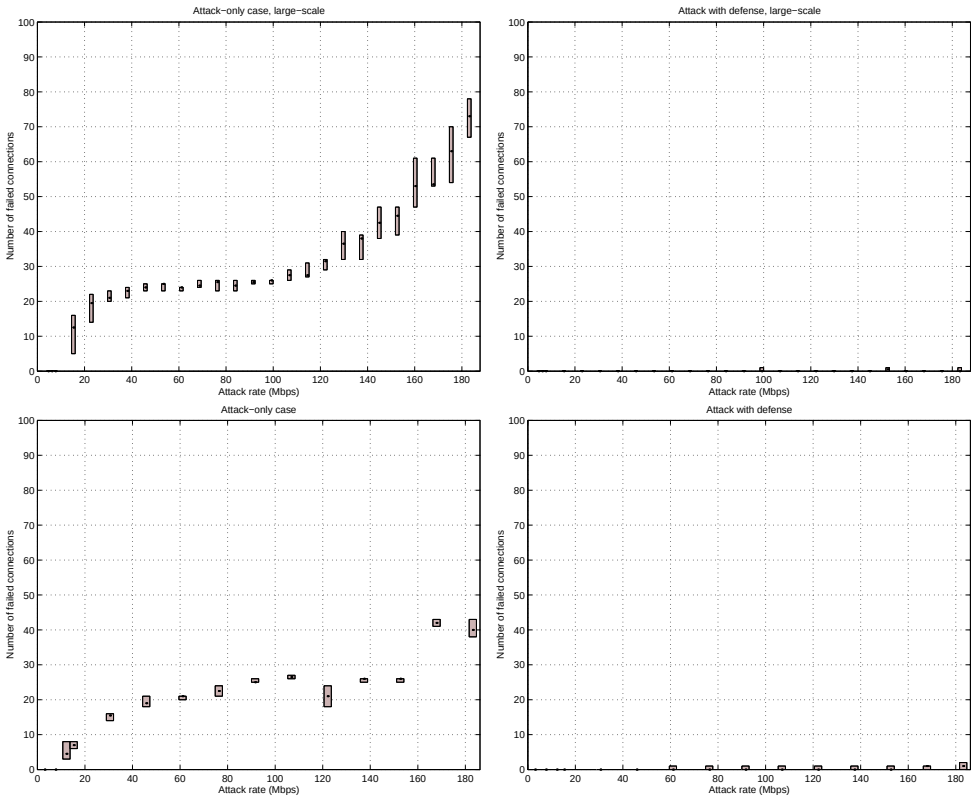

Figure 9.53: Number of failed connections in the case of large- and small-scale ICMP flooding attack

ICMP attack case. We also repeat the Figure 9.29 next to this for comparison. Just like the large-scale UDP attack case, the number of failed connections rises more rapidly for the attack-only case in the large-scale experiments. Defense performance remains comparable, keeping the number of failed connections at a very low level.

9.6.3 Large-Scale TCP Attacks

When testing large-scale TCP SYN attacks we observed that D-WARD does not manage to alleviate the denial-of-service effect even though it correctly detects

the attack and installs appropriate rate limits. Closer observation revealed that this effect is due to the high value of the Min_Rate parameter. Because D-WARD will never attempt to reduce the rate limit below Min_Rate, in the distributed attack case this may be sufficient for the attacker to get through and still perform a successful attack. To obtain meaningful results for the TCP SYN attack case in large-scale experiments we modify Min_Rate parameter to 160 Bps. Note that this parameter change does not affect all the results shown previously. As the denial-of-service effect in the previous experiments is alleviated using the higher Min_Rate value, lowering this value will not change measured performance results.

Figure 9.54 depicts the service level values for the large-scale TCP attack case. We also repeat the Figure 9.34 next to this for comparison. The attack curves differ due to agent synchronization effects. However, D-WARD still successfully provides a 100% service level to legitimate clients, both in the large-scale and in the small-scale experiments.

Figures 9.55, 9.56 and 9.57 depict the total connection delay and maximum and median per-connection delays for the large-scale TCP attack case. We also repeat Figures 9.35, 9.36 and 9.37 for comparison. Figures depicting total connection delay show the same phenomenon as in the large-scale UDP and ICMP experiments — total delay in the attack-only case is larger in the large-scale than in the small-scale experiments. Maximum and median per-connection delays are comparable in the attack-only case in both experiment sets.

Finally, Figure 9.58 depicts the number of failed connections for the large-scale TCP attack case. We also repeat the Figure 9.38 next to this for comparison. Just like the large-scale UDP and ICMP attack case, the number of failed connections rises more rapidly for the attack-only case in the large-scale experiments. Defense

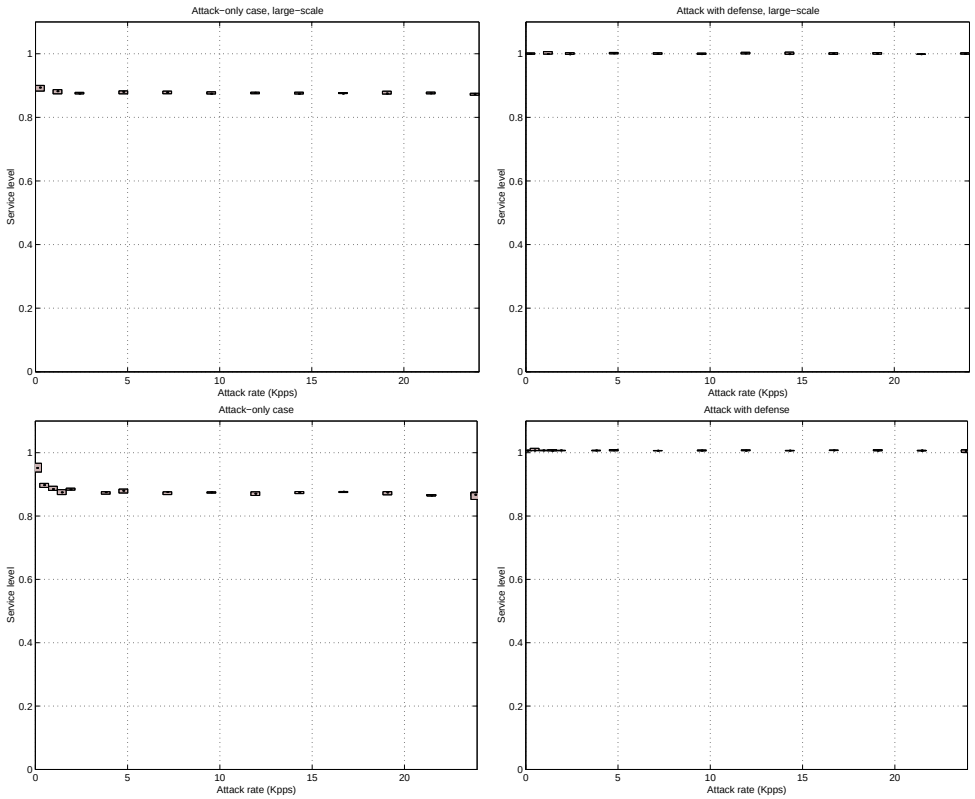

Figure 9.54: Service level in the case of large- and small-scale TCP flooding attack

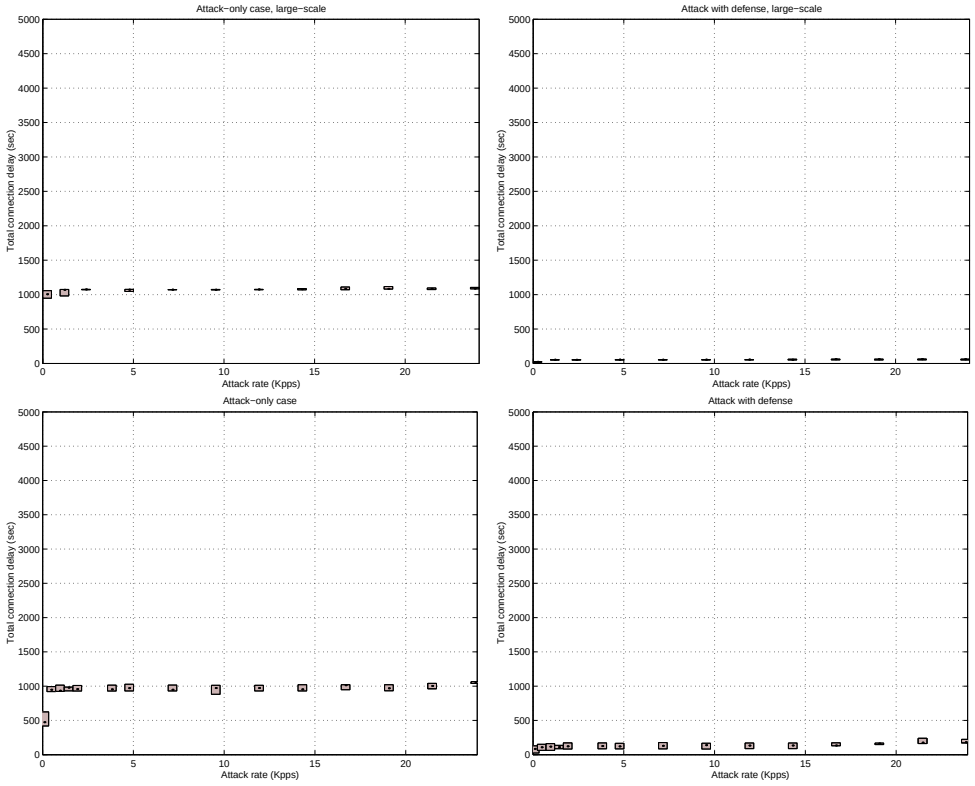

Figure 9.55: Total connection delay in the case of large- and small-scale TCP flooding attack

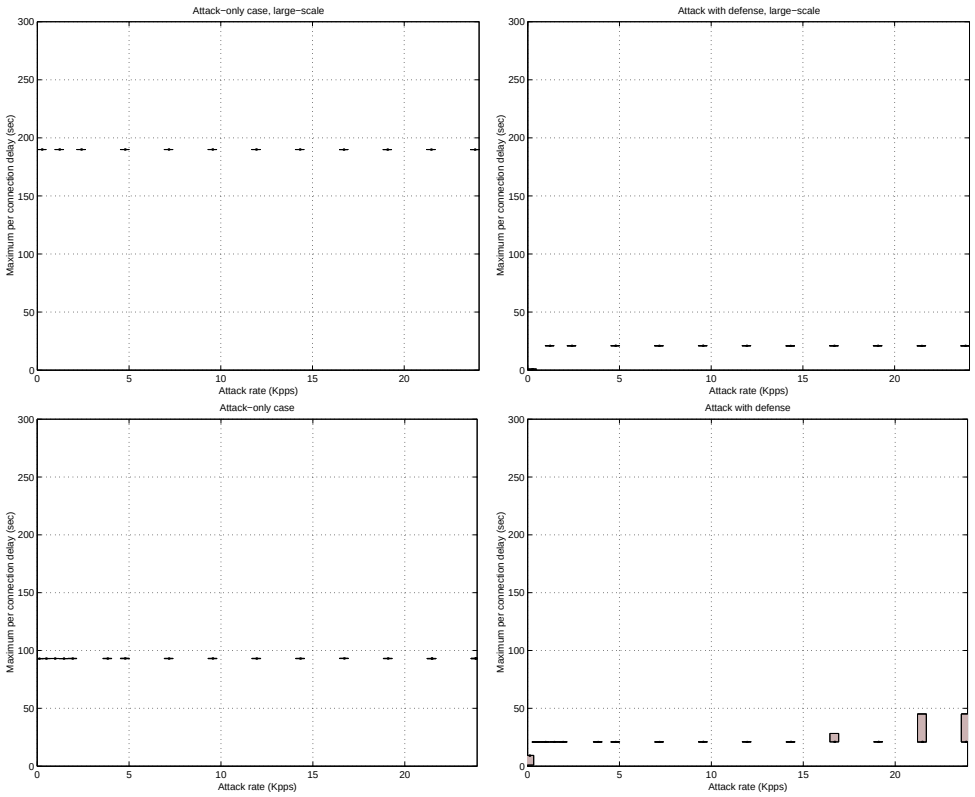

Figure 9.56: Maximum per-connection delay in the case of large- and small-scale TCP flooding attack

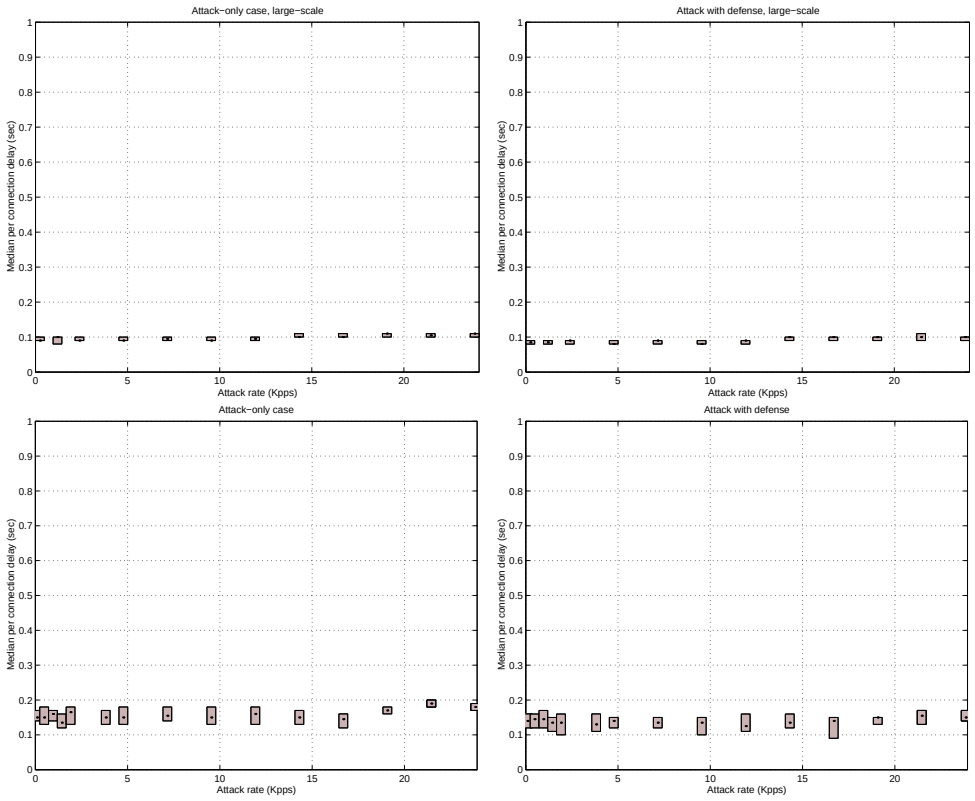

Figure 9.57: Median per-connection delay in the case of large- and small-scale TCP flooding attack

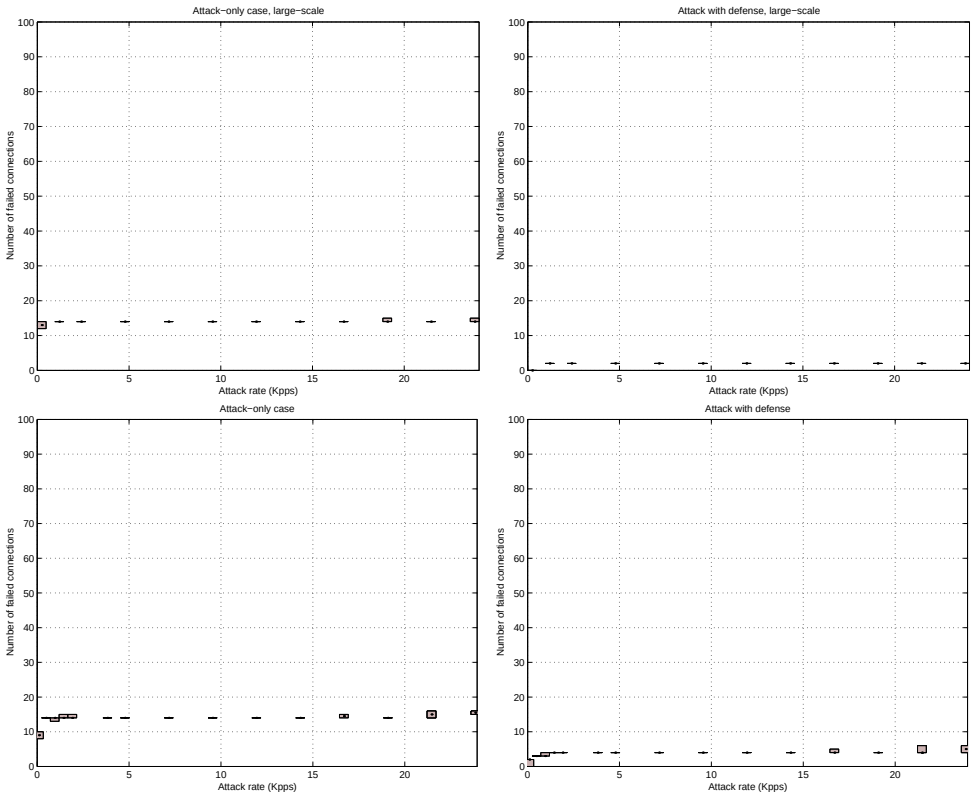

Figure 9.58: Number of failed connections in the case of large- and small-scale TCP flooding attack

performance remains comparable, and even better, keeping the number of failed connections at a very low level.

9.7 Pulsing Attacks

The attacker can attempt to avoid D-WARD detection and response by performing pulsing attacks. There are two possible strategies:

1. Generate small-rate pulsing attacks with an active interval shorter than N_{bursts} and a long inactive interval.

2. Generate large-rate pulsing attacks with an active interval shorter than 3 seconds to avoid response. Even though detection will be performed, the attacker still has 2 to 3 second window to pass the full force of the attack before response takes place.

The second strategy is much more fruitful to the attacker, especially if he interleaves his agent actions so that the victim experiences a continuous flow of the attack packets, while each agent remains active only for a short time. We generated a set of experiments to test D-WARD performance in this scenario. We used Topology 2 (shown in Figure 8.5) for these experiments and generated a pulsing TCP SYN attack. Each attacker is active for 3 seconds and inactive for a period varied in the experiments. The attackers get activated in sequential order, keeping the victim under a more-or-less constant flood of attack traffic.[3] The maximum attack traffic that reaches the victim ranges from 125 Kpps in the case of an one-second inactive interval to 19 Kpps in the case of inactive intervals of 27 second and higher.

Figure 9.59 depicts service level for this attack case. The x-axis shows the duration of the inactive period.

We observe that service level in the case when defenses are deployed remains around 1 until the inactive interval reaches 20 seconds and then declines to values between 92% and 95% This decline depends directly on the value of *Compliance_Period* parameter. As detection is performed in every attack period, the rate limit is successfully installed. However, after *Compliance_Period* intervals expire and no attack traffic is observed, the rate limit is removed, thus creating an opportunity for a new wave of attack traffic to go past D-WARD.

[3]For inactive periods longer than $9 * T_{on} = 27$ seconds, this flood will be interspersed with short pauses in attack flow at the victim.

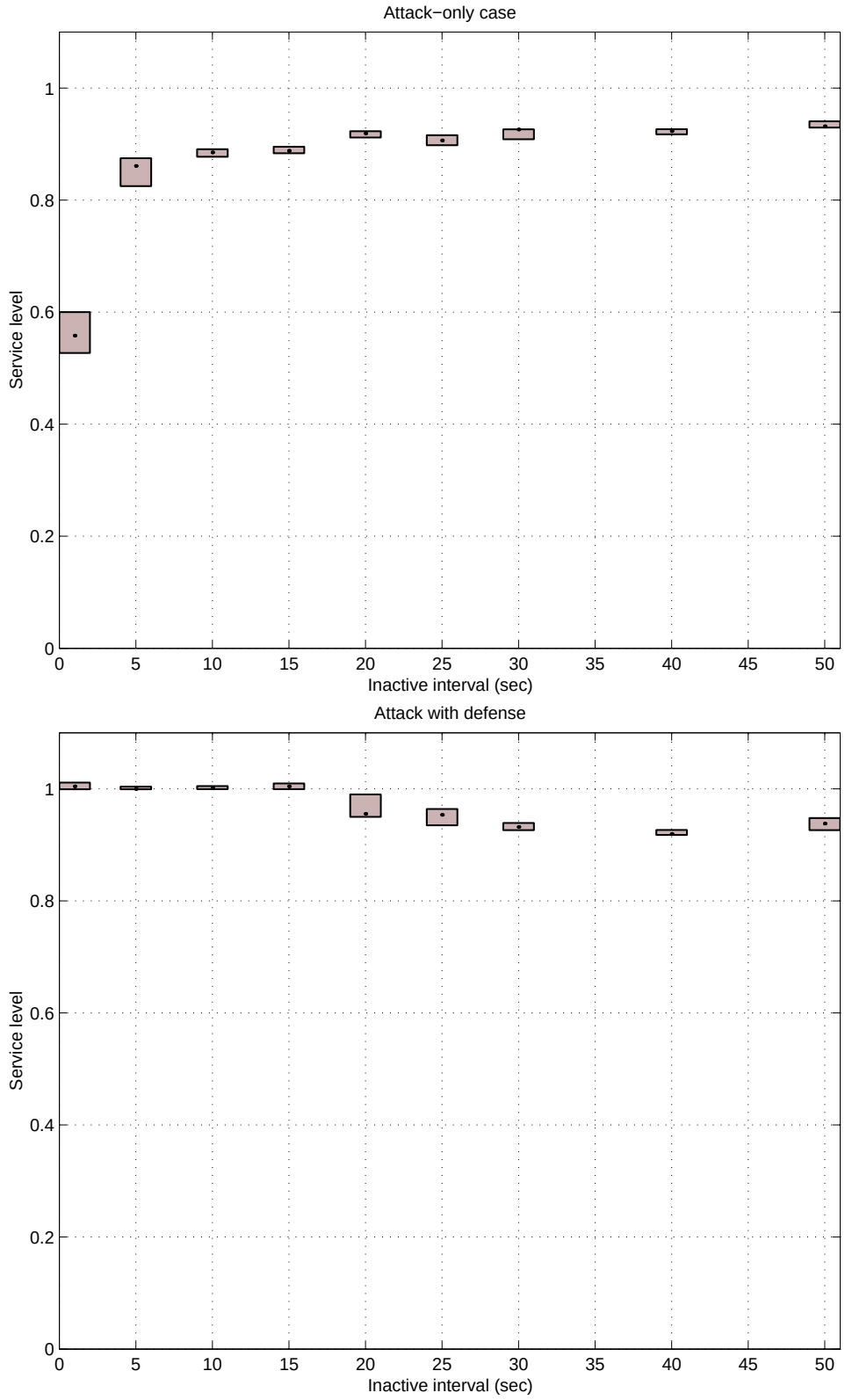

Figure 9.59: Service level in the TCP SYN pulsing attack case

The service level in the attack-only case increases as the inactive period reaches 15 seconds. This is because of the attack distribution design. Since we only have ten source networks to distribute the attack, at inactive period durations smaller than $9 * T_{on} = 27$ seconds, the attack traffic will overlap — causing a stronger denial-of-service effect. At longer durations there will be short periods of inactivity when the victim network does not receive attack traffic, thus allowing legitimate connections to recover.

Figures 9.60, 9.61 and 9.62 depict total delay for all connections and maximum and median per-connection delays. We observe that in all cases the defense performance declines only after the inactive period increases more than *Compliance_Period*.

Lastly, Figure 9.63 depicts the total number of failed connections. Just like the other metrics, the defense performance declines after the inactive period increases to more than *Compliance_Period*. It is also interesting to note that at smaller inactive periods, the number of failed connections in the attack-only case is extremely high — 160 connections out of 433 fail to complete until the end of the test run if the attack is inactive for only 1 second.

9.8 Streaming Media Models

To test D-WARD's streaming media models we use Topology 3 (shown in Figure 8.6). To create legitimate traffic we stream one Real Audio file from the server to the client. This transfer lasts 190 seconds. Around 40 seconds from the test start, we generate an 8 Mbps UDP flooding attack from the attacker to the client. Since the client has a 56 Kbps link to the router, this attack severely overloads this resource.

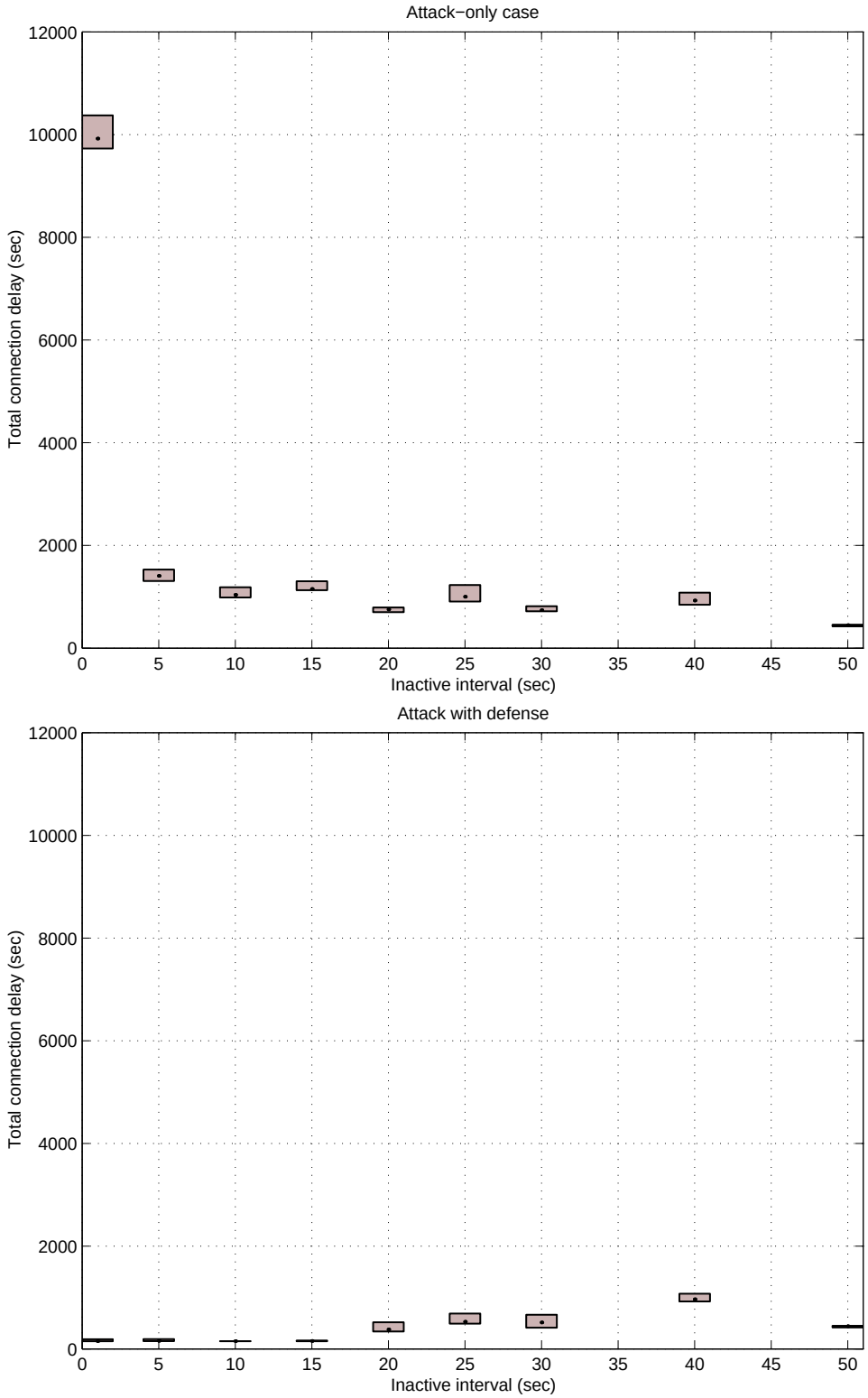

Figure 9.60: Total connection delay in the case of TCP SYN pulsing attack

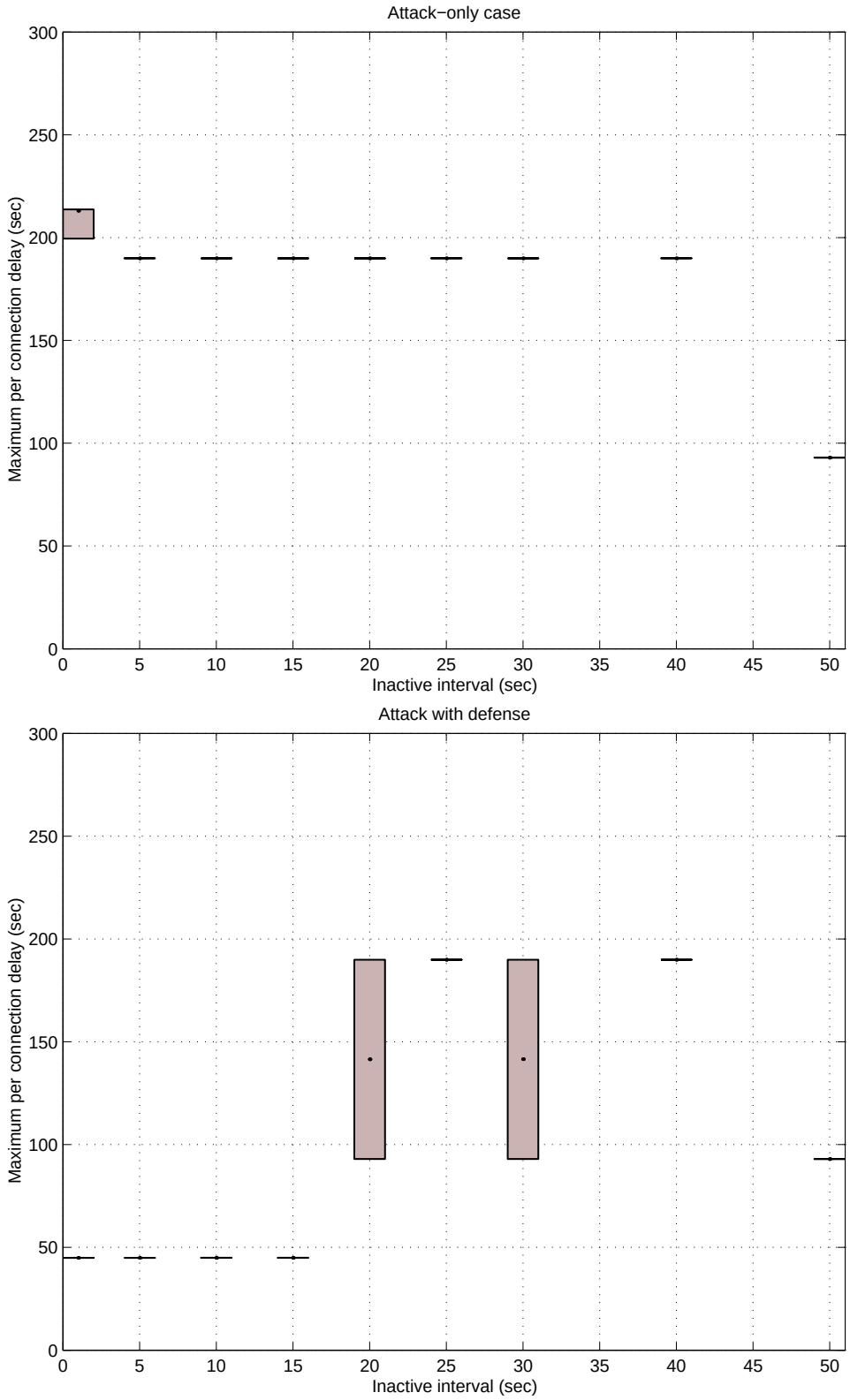

Figure 9.61: Maximum per-connection delay in the case of TCP SYN pulsing attack

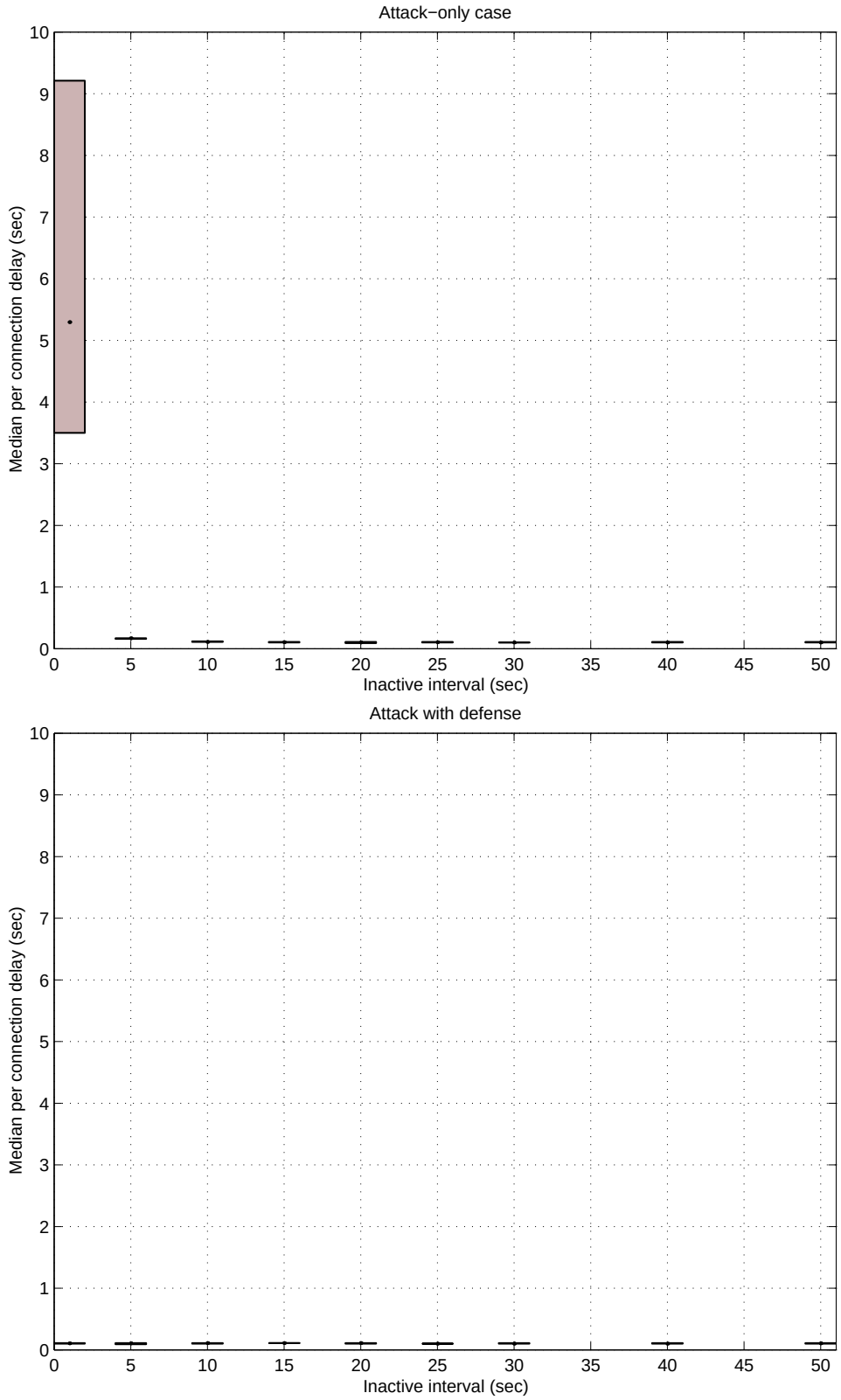

Figure 9.62: Median per-connection delay in the case of TCP SYN pulsing attack

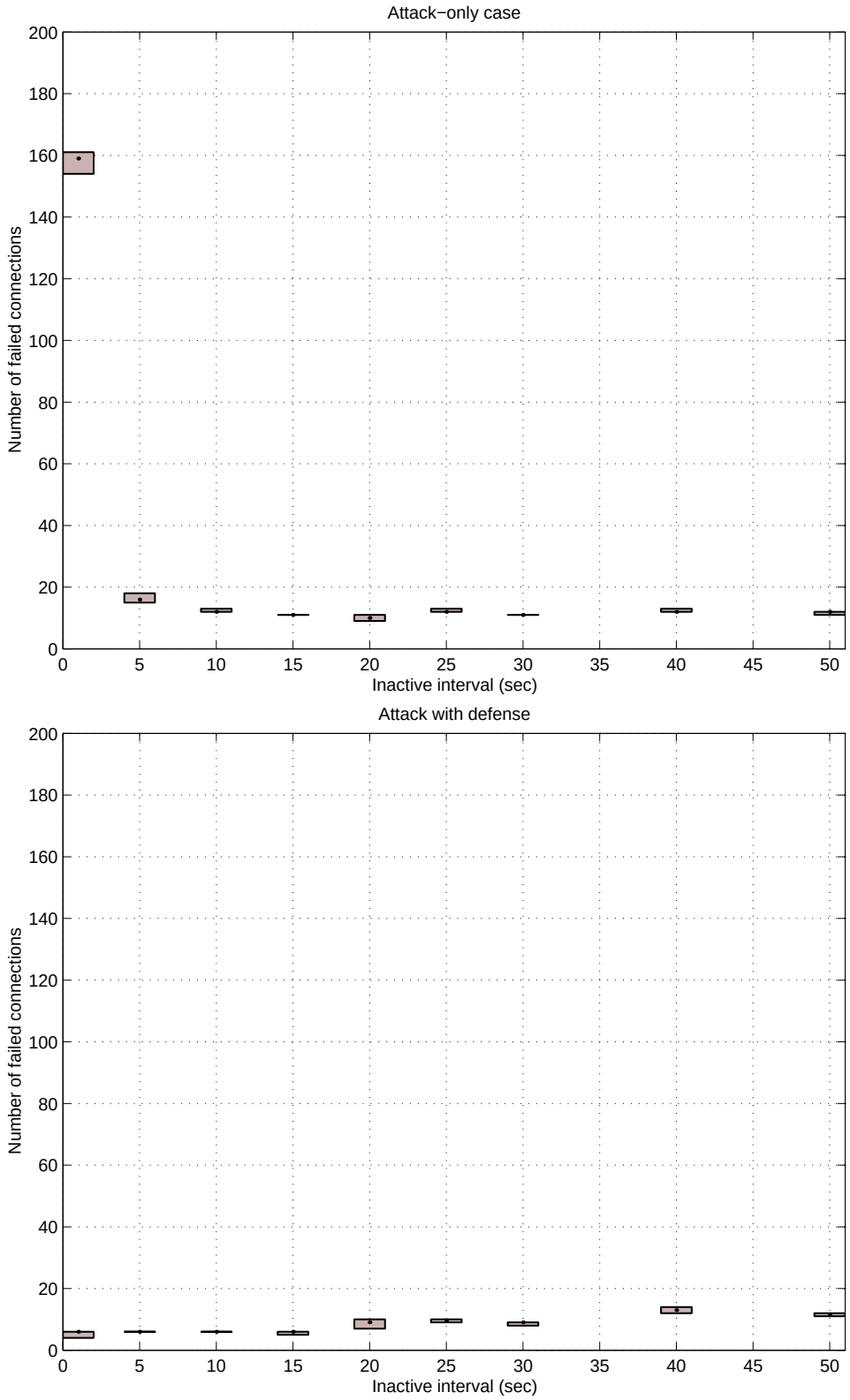

Figure 9.63: Number of failed connections in the case of TCP SYN pulsing attack

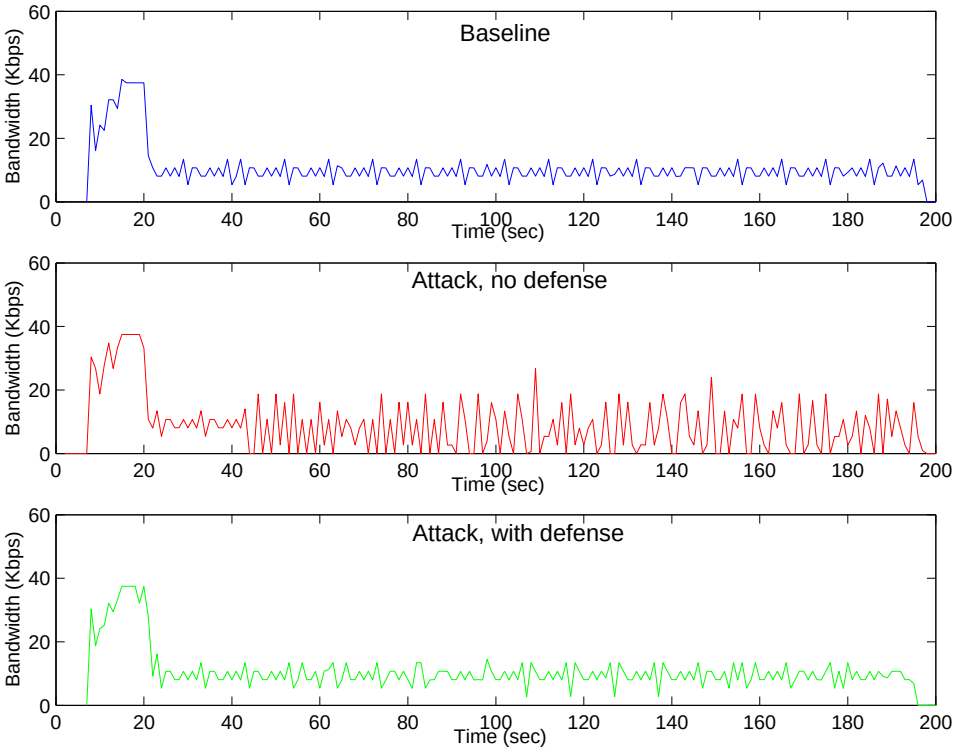

Figure 9.64: Steaming media traffic, shown in the three cases: (1) baseline, (2) attack without defense, and (3) attack with defense

Figure 9.64 shows the legitimate traffic throughput in three cases:

1. Baseline case when no attack is present and no defense is engaged
2. Attack case when the defense is not present
3. Attack case when the defense is present

In the attack-only case, the legitimate traffic exhibits a lot of jitter due to the packet loss on the congested link. When the defense is present, the legitimate traffic receives the same service as in the baseline case. Thus D-WARD successfully detects and protects streaming media traffic.

Figure 9.65 shows the attack traffic throughput in two cases:

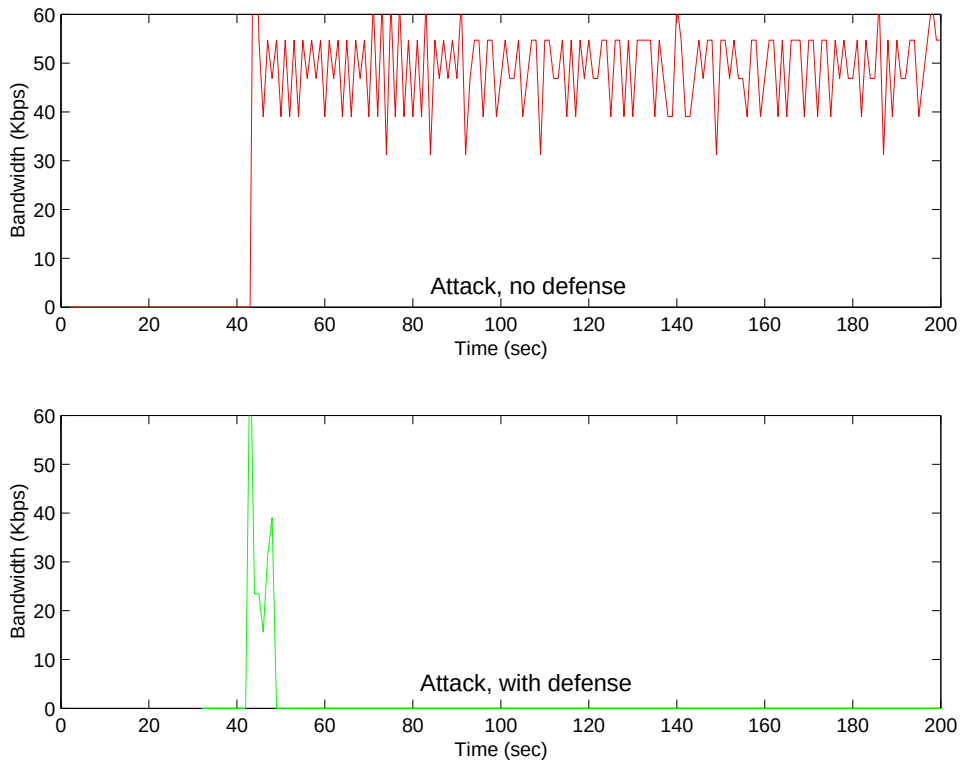

Figure 9.65: Attack traffic, shown in the two cases (1) attack without defense, and (2) attack with defense

1. Attack case when the defense is not present

2. Attack case when the defense is present

D-WARD detects and suppresses the attack traffic within 7 seconds from the attack start, thus releasing network resources from the overload.

9.9 False Alarm

We test D-WARD behavior in the case of flash crowd events on Topology 2. In addition to legitimate traffic generated in previous tests, we also generate simultaneous FTP requests for a same file, residing on *server1Netv*, from all ten

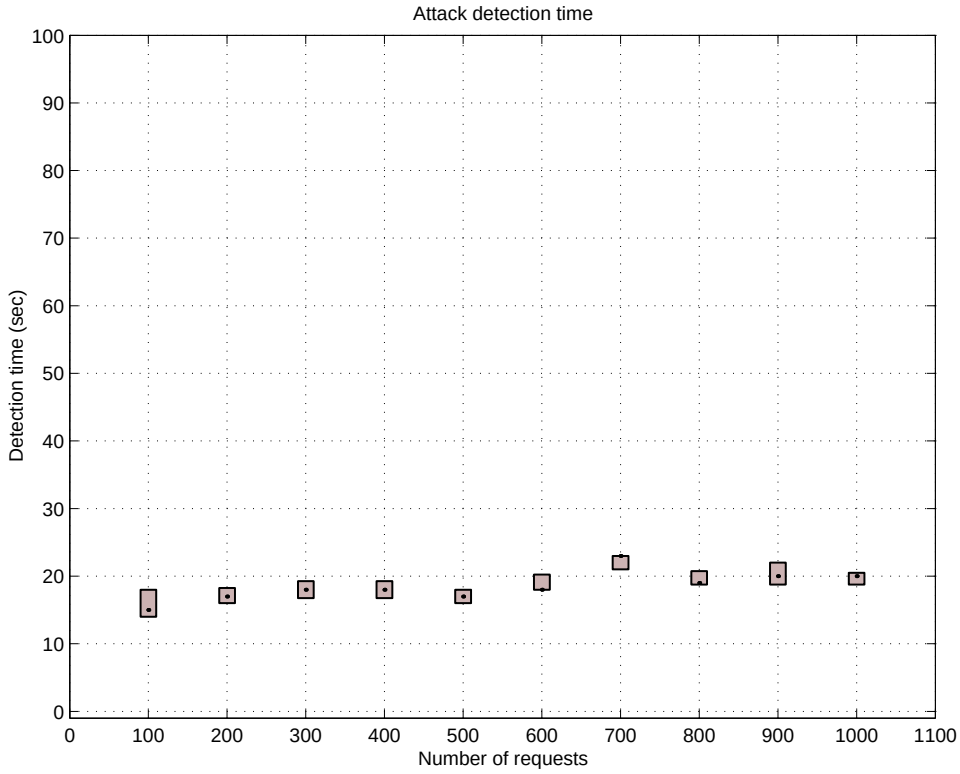

Figure 9.66: False detection time in the flash crowd case

legitimate clients. In each run a client generates N requests for a given file, spaced 1 second apart. The file is 1 MB large thus requests and replies easily overwhelm the bottleneck link. We vary N from 1 to 100, thus overwhelming the target machine by 10 to 1000 simultaneous requests.

Figures 9.66 and 9.67 show the time when D-WARD falsely detects the attack and the amount of legitimate traffic dropped due to this detection. The detection occurs between 13 and 26 seconds from the start of the FTP request series. The amount of legitimate traffic drops is now given in bytes, instead of percentage. We can observe that the highest drop is 305 bytes which is negligible compared to the 112 MB to 1.1 GB of legitimate traffic being transferred during test runs.

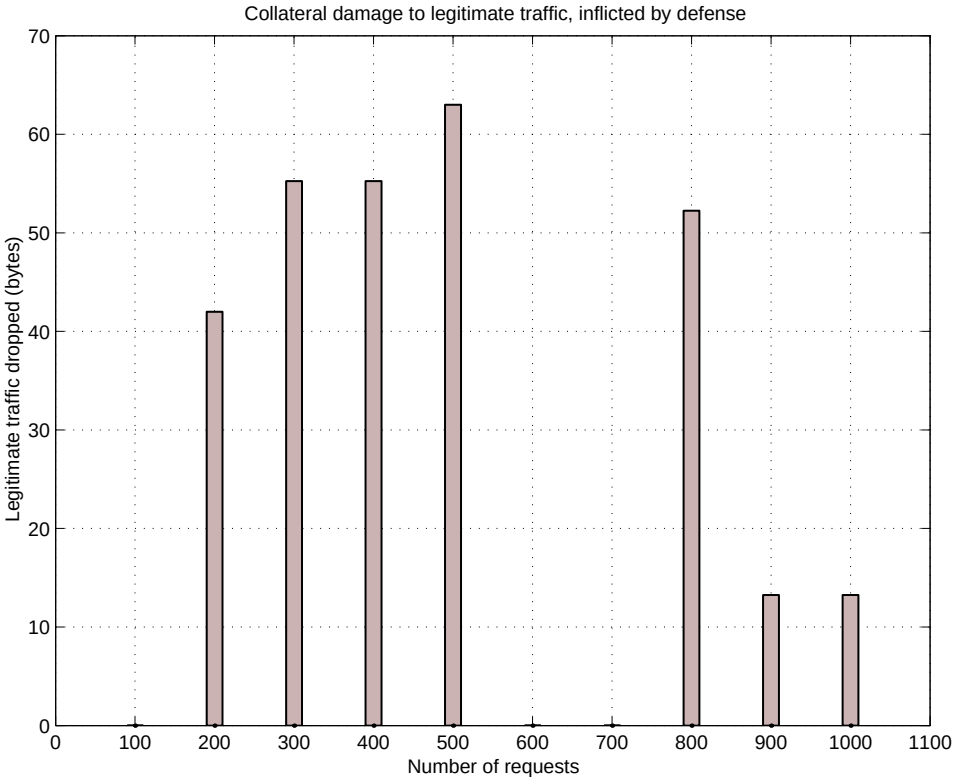

Figure 9.67: Legitimate traffic drops in the flash crowd case

9.10 False Positives

In order to test D-WARD's performance with realistic traffic, we modified the system to read packet header data from a tcpdump-generated trace file instead of sniffing it from the network. We used packet traces gathered from the UCLA Computer Science network during August 2001. The network has approximately 800 machines and experiences an average of 5.5 Mbps (peak 20 Mbps) of outgoing traffic and 5.8 Mbps (peak 23 Mbps) of incoming traffic. We assume that no attack has occurred during the trace-gathering process.

We define the following metrics for measuring the level of false positives:

1. The number of unique flows that were misclassified as attack or suspicious.[4] We report this number relative to the total number of unique flows in the trace.

2. The number of unique connections that were misclassified. We report this number relative to the total number of unique connections in the trace.

3. The number of times any flow is misclassified as attack or suspicious. We report this number relative to the total number of flow classifications performed during the trace.

4. The number of times when any connection was misclassified as bad. We report this measure relative to the total number of connection classifications performed during the trace.

Metrics 1 and 2 show what portion of the traffic will be affected by a bad decision. Metrics 3 and 4 show how good the detection process is. These are two

[4]Note that whenever the flow is misclassified, the rate limit will be imposed by D-WARD. We cannot estimate the damage that will consequently be inflicted on the flow, since any packet drops will affect the behavior of TCP connections in the flow and alter the trace.

Trace Number	Unique Misclassified Flows	Unique Misclassified Connections	Flow Misclassifications	Connection Misclassifications
1	0.28%	0.77%	0.43%	0.085%
2	0.33%	0.41%	0.06%	0.003%
3	0.23%	0.60%	0.11%	0.016%
4	1.00%	0.50%	0.10%	0.011%
5	0.63%	0.61%	0.36%	0.010%
6	1.36%	0.42%	0.14%	0.004%
7	1.29%	0.46%	0.11%	0.003%
8	1.10%	0.64%	0.18%	0.013%
9	0.97%	0.58%	0.13%	0.010%

Table 9.3: Percentage of false positives.

different things. For example, consider a case in which D-WARD sees 100 unique flows over 10 seconds, and each second it performs 10 classifications. Metric 1 reports the number of flows misclassified at any point in their lifetime. If only one flow was continually misclassified in every classification period, this metric would be 1%. Metric 3, in contrast, would be 10%, since our classification process reached a wrong decision in 10% of all classification attempts. 10% might seem rather high, but metric 1 shows that very few flows are actually affected. Table 9.3 presents the results for several traces.

As we can observe, all false positive measures are very low. A false positive measure in the flow classification case indicates that D-WARD is very unlikely to perform false detection and inflict damage on legitiamte traffic. Furthermore, even if false detection occurs, D-WARD will preserve correctly classified legitimate connections, thus further minimizing the damage.

9.11 Deployment Cost

We measure deployment cost as memory cost (for D-WARD statistics storage) and the packet handling overhead. Hash table sizes are configured by D-WARD configuration parameters. The size of the record in the flow hash table is 332 bytes. As the flow hash table size in our experiment runs was set to 1003 records, the total memory cost for the flow hash table was 333 KB. The size of the record in the connection hash table is 128 bytes. The connection hash table size in our experiment runs was set to 50003 records, so the total memory cost for the connection hash table was 6.4 MB. The size of the record in the rate limit hash table is 32 bytes. Since the rate limit hash table size in our experiment runs was set to 1003 records, the total memory cost for the rate limit hash table was 32 KB. The user-level memory cost is thus 6.7 MB.

The size of the record in the limited flow hash table is 132 bytes. The limited flow hash table size in our experiment runs was set to 103 records, so the memory cost for the limited flow hash table was 13.6 KB. The size of the record in the good connection hash table is 84 bytes. The good connection hash table size in our experiment runs was set to 2003 records, so the total memory cost for the good connection hash table was 168 KB. The kernel-level memory cost is thus around 182 KB.

Packet handling overhead under normal operation is measured using the `ping` utility in Topology 2 (shown in Figure 8.5). A total of 1000 *ICMP_ECHO* packets is sent from *clientNet1* to *server1Netv*. We compare the case when D-WARD is engaged at *routerNet1* to the case when D-WARD is not engaged. We measure the average round-trip delay of 12μsec and infer from that the average one-way delay of 6μsec.

The kernel-level incurs additional overhead for spoofed packet handling. This overhead becomes critical at rates higher than $12,000$ packets per second.

CHAPTER 10

Cooperation Efforts

D-WARD has been integrated into two distributed DDoS defense systems: COSSACK and DefCOM. It has also been independently evaluated in the DDoS-DATA project. Each of these cooperation efforts is described in the following sections.

10.1 COSSACK

COSSACK [Infa] is a distributed DDoS defense system developed under DARPA's Fault Tolerant Networking (FTN) program at Information Sciences Institute (ISI). COSSACK was integrated with D-WARD 3.0 (D-WARD without sequence number prediction and without legitimate UDP models) with the goal of enhancing performance of both systems by complementing their respective strengths. The integrated system was evaluated independently within a DARPA Red Team Experiment. This section provides details about the integrated system, the achieved synergy and an overview of preliminary experiment results.

10.1.1 COSSACK Overview

COSSACK is a distributed approach to DDoS detection and response. COSSACK components, located at the edge networks of the Internet, coordinate with

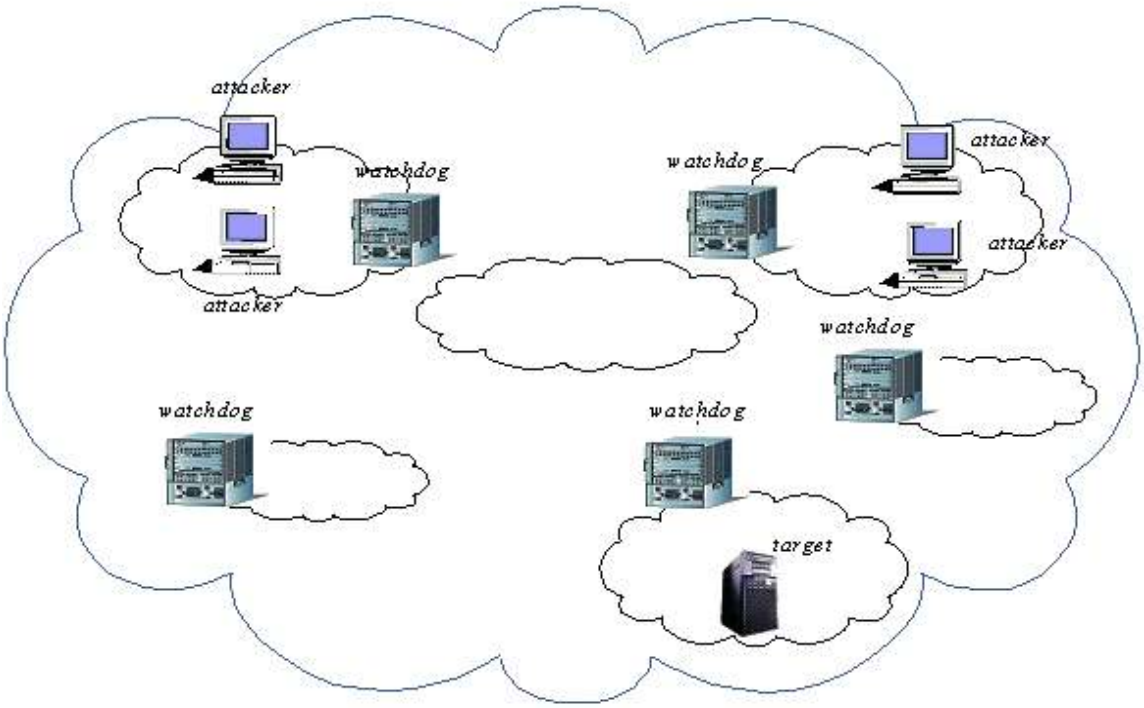

Figure 10.1: COSSACK architecture

other edge networks to collectively combat DDoS attacks.

In the COSSACK architecture, as shown in Figure 10.1, the principal element is a watchdog, a software subsystem that resides at an edge network egress point. Watchdogs communicate over multicast channels using YOID [Infc], an application-level multicast protocol. Each watchdog monitors its own network and shares information with other watchdogs. Localized information can be gleaned using a variety of collection tools, such as SNMP statistics, Cisco NetFlow, and IDS tools such as Snort. The current implementation of COSSACK uses Snort with a customized plug-in. This plug-in was developed specifically for DDoS detection and provides dynamically selectable functionality that can be controlled by the watchdog.

The watchdog performs DDoS detection, coordination with other networks originating an attack, and response. The following example outlines the operation of COSSACK in response to a typical DDoS flooding attack. Figure 10.2 shows the onset of an attack and Figure 10.3 suppression by COSSACK watchdogs. The attack is launched from a large number of machines located throughout the Internet, and their combined output overwhelms the capacity of the ingress link of the victim's network. The following operations take place:

1. The watchdog at the victim edge network detects the flooding attack at the ingress point.

2. The watchdog instructs the IDS (Snort) to compile source address information and attack signature data (rates, type of attack, etc.).

3. The watchdog multicasts an attack notification to other watchdogs in the network indicating the attacking source networks. This communication is facilitated by YOID [Infc], and application-level multicast mechanisms developed at Information Sciences Institute. It also advertises an attack-specific multicast group that will be used for subsequent coordination.

4. Watchdogs representing the implicated source networks join the coordination multicast group.

5. After receiving attack information hints, each source network watchdog performs an in-depth analysis of particular outgoing flows to determine if zombies exist within its infrastructure.

6. Source networks that identify zombies deploy countermeasures to prevent a continuance of the attack. Local responses will be dictated by a combination of local response policy and the policy information received from the victim-side watchdog

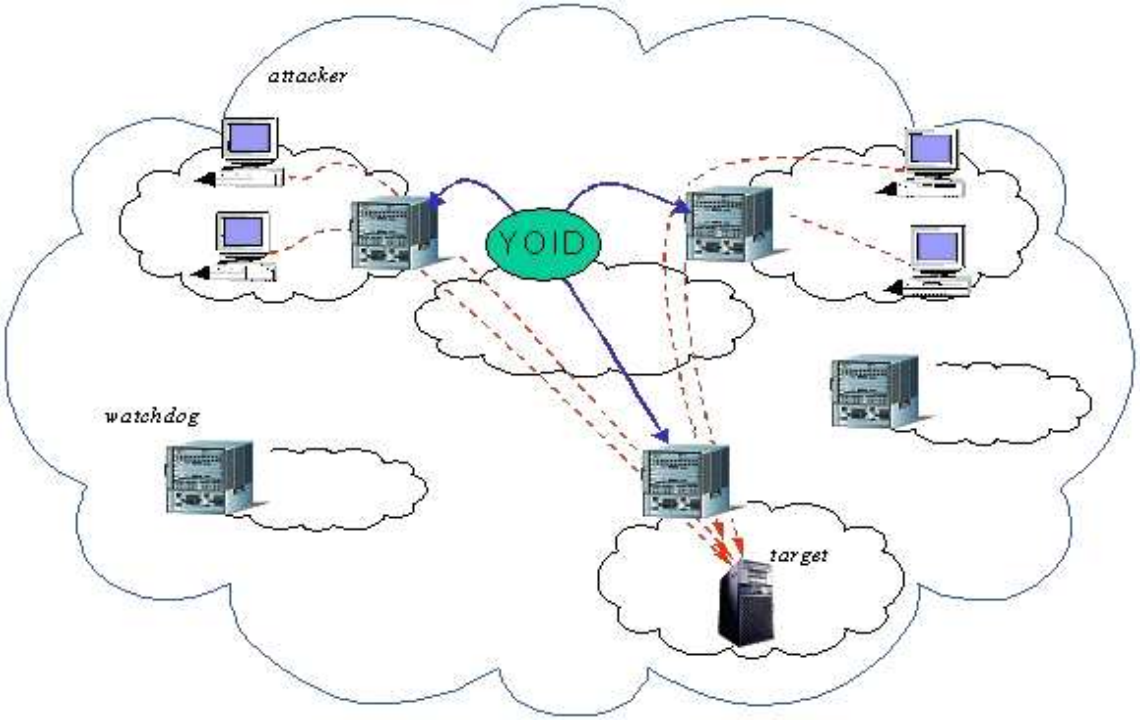

Figure 10.2: Onset of attack in watchdog protected network

Currently, it is assumed that watchdogs can figure out how to contact the source networks that are hosting the attack. At this stage of COSSACK research, random source address spoofing is not addressed. Therefore, the victim watchdog locates source watchdogs from the addresses in attack packets that are assumed to be correct.

10.1.1.1 Watchdog Overview

The watchdog is the main analysis, decision, and coordination element in the COSSACK architecture. It accepts input from one or more data sensors, analyzes the data, shares information with other watchdogs and makes decisions on how to respond to attacks. The components of the watchdog are shown in Figure

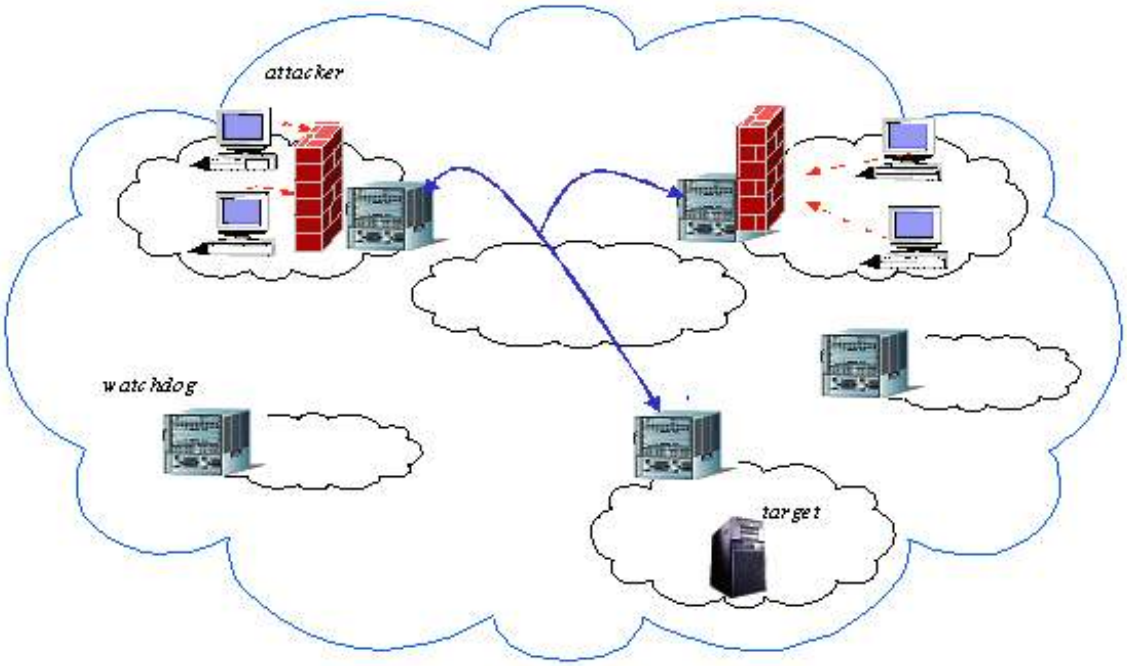

Figure 10.3: Suppression of attack by COSSACK watchdogs

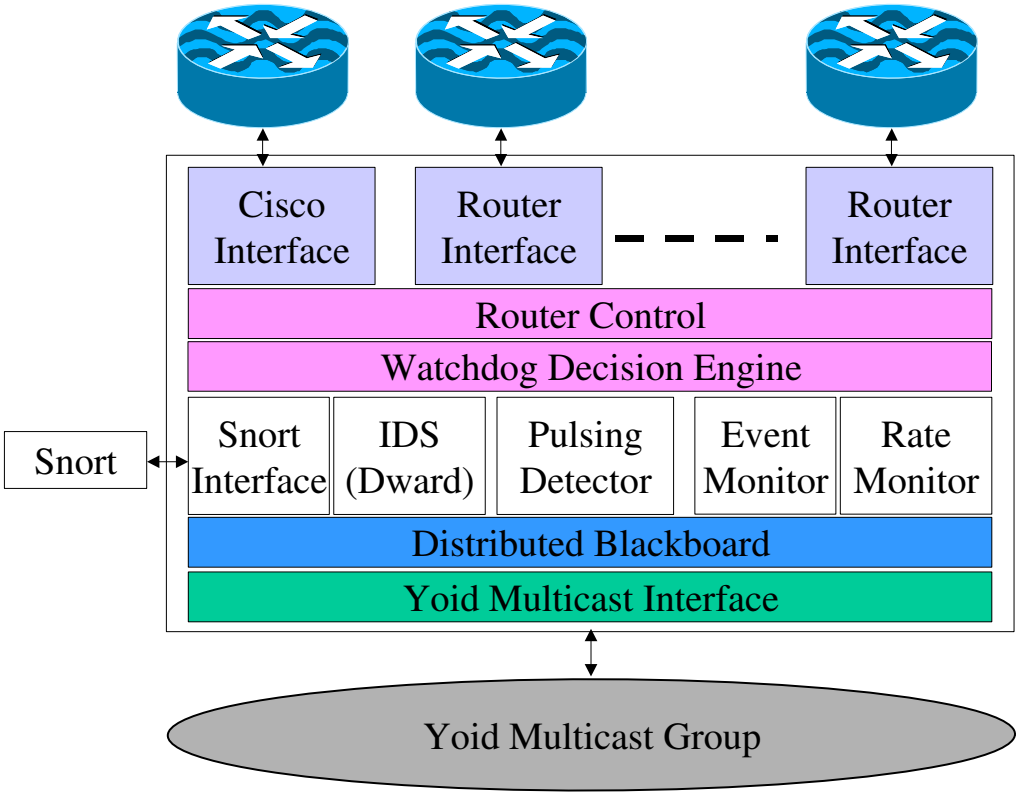

Figure 10.4: Components of COSSACK watchdog

10.4.

The core of the watchdog is implemented in the Java programming language, which allows for portability to many operating systems. No performance problems have been experienced with the watchdogs, as they deal mostly with high-level events rather than per-packet events which are handled by the sensors. In the current implementation, watchdogs accept input from one or more Snort plug-ins (see Section 10.1.1.2). Upon startup, each plug-in creates a TCP connection with its associated watchdog and begins sending flow statistics. The watchdog may control the flow of this information by supplying filtering rules to the Snort

plug-in. The watchdogs currently have an interface to control Cisco and Linux routers. Through this interface, the watchdogs may set filtering or blocking rules in response to attacks.

10.1.1.2 Snort Plug-in Overview

As mentioned earlier, COSSACK watchdogs rely on an existing IDS to detect attacks. Snort [Sou] was selected for use in COSSACK experiments. Snort has a number of desirable characteristics. It is open source, it has an established user community, and it is actively supported. In addition, there has been significant effort to optimize Snort to support traffic capture and analysis at fairly high data rates.

The internal architecture of Snort is very modular, and easily accommodates the COSSACK extensions. Packets captured by Snort are guided through a series of processing steps, one of which is filtering against a rule database containing known attack patterns that are matched against the header and payload information. Currently, the rule base is static during a Snort execution. COSSACK uses the rule database in two ways. First, it is used as a fast pre-filter, to pass only packets belonging to flows of interest to COSSACK. Here Snort is used to break up packets into different protocol groupings to keep statistics on each individual grouping. This information helps the watchdog diagnose an attack, but also allows the watchdog to define a more specific filter to install in a router to stop the attack.

The second use of the Snort database is to define specific patterns that identify individual, possibly malformed packets that attempt to exploit known flaws in software implementations. These packets are directly reported to the watchdog without being aggregated.

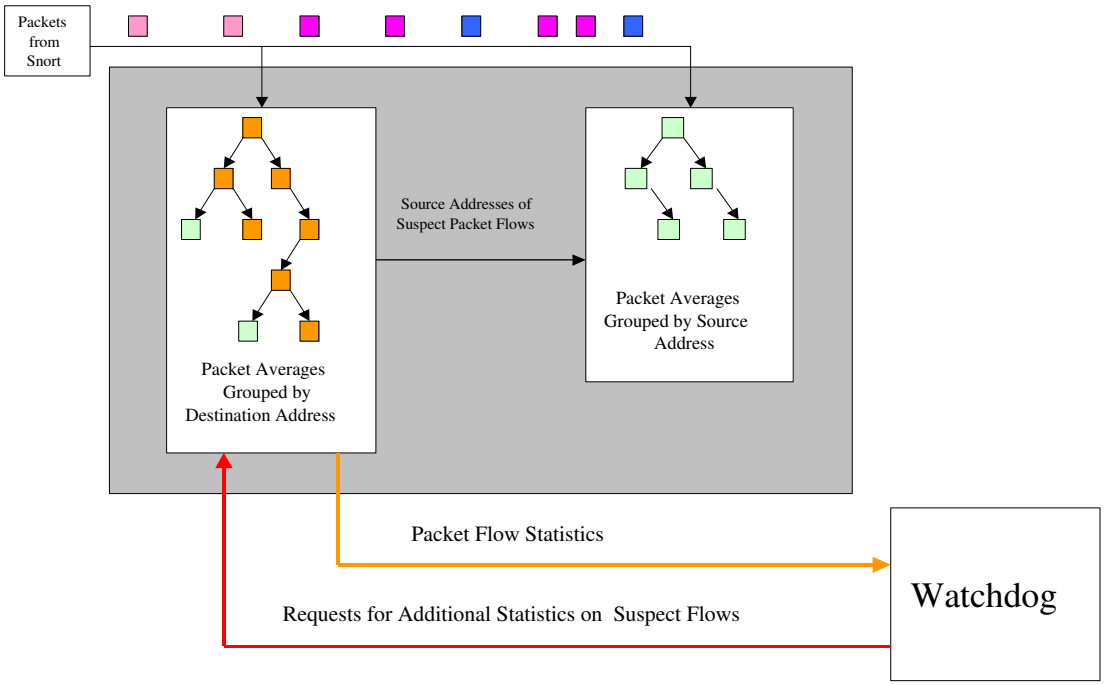

Figure 10.5: Current implementation of Snort plug-in

The current implementation of the Snort plug-in is shown in Figure 10.5. During normal operation, the plug-in keeps packet rate statistics for different flows grouped by address prefix. The plug-in constructs a prefix tree data structure, which allows for quick aggregation of prefix information. To keep the tree from growing unbounded, the plug-in performs periodic garbage collection after state expires or the tree grows beyond a certain size. The plug-in supports hundreds of simultaneous packet flows by dynamically building an aggregation tree based on packet rate.

In addition to the destination network prefix tree, the plug-in is capable of maintaining a source prefix tree. The source tree is constructed on demand, when instructions are received from the watchdog. The watchdog issues such commands if, after monitoring packet rate reports from Snort, it determines that a host may be under attack. The watchdog then asks the plug-in to construct the source prefix tree for that destination. The tree is finally reported back to the watchdog, which then contacts the watchdogs monitoring the source network(s).

10.1.1.3 Watchdog Algorithm Description

The following actions take place during COSSACK attack detection and suppression:

1. The COSSACK watchdog loads a list of detection-level thresholds from a configuration file. These rates are determined based on background traffic levels, traffic mix, types of servers being used, and connection bandwidth.

2. The COSSACK watchdog receive periodic reports from Snort on packet rates and bandwidth for destination address aggregates. Currently, these rates are reported every 30 seconds. Rates are aggregate rates of traffic at

varying prefix levels. Traffic information is sorted into different categories using Snort rules.

3. At the victim watchdog, if a bit or packet rate threshold is exceeded in the detector, which was configured by the rules in (1), the victim watchdog will ask Snort to provide source address aggregate rates for specific destinations that violated the threshold.

4. The victim's watchdog distills the aggregate information to determine the most specific address/netmask ranges, ports, and protocol numbers that are being used in an attack. The convergence tolerance is specified in the configuration file.

5. After receiving the source address aggregate rates from Snort, the victim watchdog will announce the attack to the global multicast group and create an attack-specific multicast group. Both of these actions are facilitated by YOID application-level multicast. The announcement contains the name of the attack-specific group.

6. Source network watchdogs will check the attack announcement messages from the global group to see if their network is participating in the attack. If a watchdog sees its network in the messages, it will join the attack-specific group. The list of source networks for which a watchdog is responsible is specified in a configuration file.

7. If a source network watchdog joins a specific attack group it will announce its outgoing packet rate statistics information to other group members. These announcements are stored in a blackboard data structure at each watchdog.

8. The source network watchdogs request source aggregate information with respect to the victim address.

9. The source network watchdogs also distill the aggregate information and determine the most specific address/address ranges, ports, and protocol numbers that are being used in an attack.

10. Each source watchdog will sum up other source watchdogs' traffic rates toward the victim and compare the sum to the total packet rate that the victim is reporting. This check it used to verify that the victim really has a problem and is not sending out false information.

11. If the packet rates agree within a margin of rounding error, the watchdog will install rules to block traffic from the attacking machines to the victim. The margin of error is configured in a configuration file and currently set to 80%. The source watchdog will try to be as specific as possible with its traffic blocking rules. These include source address, destination address, source port, destination port and protocol.

12. Source watchdogs will continue monitoring the traffic to the victim. When the attack from this source has stopped, the watchdog will remove the blocking rules.

13. In the case of a pulsing attack detection, the local watchdog decides when to stop blocking. In a flooding attack, each source watchdog must compare its traffic rates to the whole. The rates from other source networks are shared using the attack-specific group.

10.1.2 Overview of an Integrated COSSACK and D-WARD System

D-WARD is integrated with the COSSACK system as a source-end defense system that interfaces with local watchdogs. In the integrated system, COSSACK victim-end detection complements D-WARD source-end detection. This is beneficial for those types of attacks that D-WARD is unlikely to detect, such as non-spoofing UDP attacks. COSSACK performance is also improved by use of D-WARD's selective and dynamic response instead of COSSACK's signature packet filtering.

COSSACK and D-WARD both perform detection of distributed denial-of-service attacks in the integrated system, but only D-WARD performs responses to the attacks. The COSSACK and D-WARD detection mechanisms remain the same as described in the previous sections on the individual systems, but COSSACK never invokes a response itself, unlike its normal stand-alone operation.

When COSSACK detects an attack at the victim end, it will send a signal to other COSSACK nodes close to the sources of the attack, using its multicast mechanism. Local COSSACK watchdogs will then determine if they believe that the local network is contributing to the attack and, if so, will signal the local D-WARD software. D-WARD then performs a secondary check to verify that the COSSACK alert pertains to current source network communications. Upon successful verification, D-WARD modifies its rate-limiting. If D-WARD does not receive a COSSACK message for *COSSACK_Timeout* seconds, it falls back to autonomous operation as if COSSACK were not present. Any subsequent message received from COSSACK by D-WARD reestablishes cooperation.

Figure 10.6 shows the format of the message that the local COSSACK watchdog sends to D-WARD. The message fields are the following:

Type	Target IP	Suspect Source Networks	Rate Limit	Flag

Figure 10.6: COSSACK alert message format

- **Type** indicates the type of attack — TCP, UDP, ICMP, flooding, etc.
- **Target IP** contains the address of the machine under attack.
- **Suspect Source Networks** field indicates the ranges of networks that COSSACK believes are sources of the attack.
- **Rate Limit** is a suggested rate limit for D-WARD to impose.
- **Flag** indicates whether this is a message alerting D-WARD of a new or ongoing attack, or whether COSSACK has determined that the attack has ended. This flag value allows COSSACK to signal to D-WARD that rate limits put in place earlier may now be relaxed.

When a D-WARD node receives such a message with the flag indicating a new or ongoing attack, it treats it as advisory, not as a command. How the message is handled depends on the type of attack and any existing response to the signaled attack.

D-WARD detects few UDP attacks, so it will regard a COSSACK signal of a UDP attack as authoritative, and will impose the requested rate limit on the attack. This rate limit will not be changed by D-WARD's throttling mechanism. It will be changed only upon receipt of another message from COSSACK that modifies the limit, or after the timeout for the COSSACK message expires and D-WARD falls back to autonomous operation. Similar behavior is triggered for attacks using other traffic types (not belonging to TCP, UDP or ICMP).

In the case of a TCP-based attack, D-WARD might already have detected the attack, in which case the message is ignored and D-WARD continues with its normal activities to throttle the attack. If D-WARD has not already detected the TCP-based attack, the incoming message is regarded as a signal of an attack similar to D-WARD's own detection. D-WARD ignores the rate limit specified in the message and invokes its own normal throttling on traffic being sent to the target IP address.

ICMP traffic will be treated the same as TCP traffic. Traffic that is not UDP, TCP, or ICMP will be treated in the same manner as UDP traffic, assuming that it is IP traffic. Non-IP traffic is out of bounds for the current integration effort.

When D-WARD receives a message with the flag indicating that the attack has stopped, it will verify that it has indeed rate-limited the associated flow. If D-WARD has deployed a rate limit for the attack, but has not yet started its slow recovery, slow recovery is started, permitting more traffic to flow toward the previous target. If slow recovery is already ongoing, the message is ignored. Exceptionally, for UDP attacks and attacks through other traffic types (non-TCP, non-UDP and non-ICMP) D-WARD lifts the rate limit immediately upon receipt of a message with a flag indicating that the attack has ended.

Because individual COSSACK messages might be lost in the network, COSSACK will resend its control messages to D-WARD periodically. In particular, the message indicating the end of an attack will be sent several times, with a delay of two to three seconds between each sending.

The COSSACK/D-WARD integration and defense strategy described in this section has been uniquely tuned/optimized for the DARPA Red Team Experiment in which: (1) the edge-networks have a single point of ingress/egress, (2) the border router at that ingress/egress point is defended, (3) the expected at-

tacks are limited to DDoS attacks, and (4) attacks on an edge network can only be mounted from outside that edge network.

10.1.3 DARPA Red Team Experiment

The COSSACK and D-WARD technology has been developed and tested in the laboratory and successfully demonstrated under controlled circumstances. In order to better understand the strengths and weaknesses of the technology, and to support the FTN program goal of technology transition to an operational environment, more extensive assessment and stress testing is required. This assessment and testing needs to be conducted outside of the development laboratory, by an independent group, and should focus on the central assumptions and claims of the technology. This is accomplished in the DARPA Red Team Experiment effort using a paradigm of experimentation, which includes well-posed hypotheses, well-defined metrics, and carefully controlled environments.

The goal of the Red Team Experiment is to investigate and quantify the ability of an integrated defensive system, comprised of COSSACK and D-WARD together, to provide effective mitigation of DDoS attacks in a representative, realistic, dynamic, environment.

10.1.3.1 Experiment Methodology

The experiment involves a network of networks, consisting of nine disjoint edge networks containing routers and clients which are interconnected by a core network of routers without clients (which is simulated by a multi-ported Cisco router). The architecture of this network is depicted in Figure 10.7

Both COSSACK watchdogs and D-WARD systems protect each edge net-

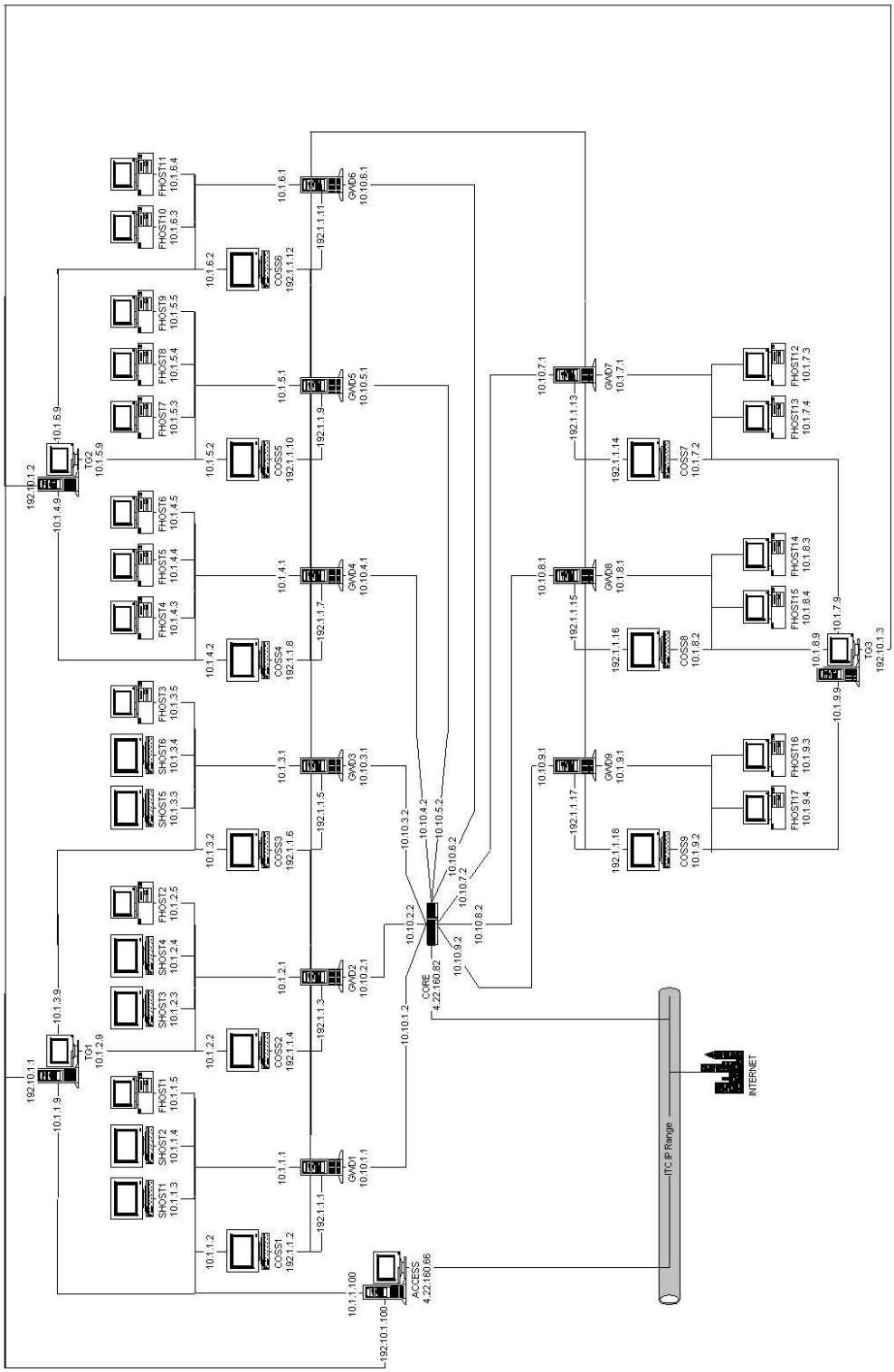

Figure 10.7: Network topology for Red Team Experiment

work's border router, connecting with the core network. COSSACK watchdogs reside at machines $COSS1$ to $COSS9$, while D-WARD systems reside at gateway routers $GWD1$ to $GWD9$. COSSACK performs detection on incoming traffic to each edge network (e.g., detecting whether it is a victim of an attack), and performs coordination and communication between all distributed components of both DDoS defensive systems. D-WARD performs detection on outgoing traffic from each edge network (e.g., detecting whether it is a source of an attack), and provides all responses by selectively throttling outgoing traffic from its own edge network. In cases of conflicting input, each D-WARD system acts in response to its own detections, rather than the broadcast warnings from COSSACK.

Based on the particulars of the scenario, representative traffic is generated to traverse the core network between the multiple edge sub-networks. This traffic, between the various clients on the edge networks, consists of both legitimate messages and DDoS attacks. The legitimate traffic is provided by three traffic generators (TG) residing on machines $TG1$ to $TG3$. There are three mixtures of legitimate traffic in the experiment: **high**, **low** and **mixed**. They are described in more detail in Section 10.1.3.2. TG outputs are dynamically varied to provide a realistic background in which the attacks can hide, and are employed during the baseline runs for determining both the false positive and false negative rates of the defensive system's detectors. In addition to traffic generators, some legitimate traffic is created live using Web server and streaming media applications deployed at machines $SHOST1$ to $SHOST6$. Those applications are invoked by clients at machines $FHOST1$ to $FHOST20$.

The attacks are developed and scripted by the Red Team in advance, and the individual attack runs are executed by the experimenters. To ensure a realistic experiment, the Red Team has the complete knowledge of the strategies and

source code of the defensive systems, but the Blue Team only has the knowledge of the Red Team's general strategy, not of the actual attacks and their order of presentation. In addition, the particular release of the defensive system's software code that is actually employed in the experiment is frozen during the period of experiment execution to avoid the problem of any changes nullifying previous results.

Because some security aspects of the existing COSSACK and D-WARD software, which are not relevant to their ability to detect and respond to DDoS attacks, have not yet been implemented (e.g., such as protecting themselves from attack), those capabilities are assumed for the duration of the experiment. The objective of the experiment is not to assess the robustness of these defensive systems in the wild, but to better understand (and communicate in the final report) the intricacies of DDoS defense.

10.1.3.2 Baseline Runs

As discussed in the above section, there are three baseline legitimate traffic mixes, referred to as **high**, **low** and **mixed**. Figures 10.8, 10.9 and 10.10 depict traffic levels at the victim for these three baseline traffic mixes. Triangles denote UDP traffic levels, squares denote TCP traffic levels and diamonds denote total traffic levels. Just as in the real Internet, TCP traffic is the largest portion of the total traffic.

Figures 10.11, 10.12 and 10.13 depict the false positives generated by the integrated system, when only baseline traffic is run. For each false-positive instance, D-WARD's observation component will detect the attack. In the subsequent intervals, the rate-limiting component will adjust rate limit rules on the misclassified flow. The allowed rate will first be exponentially decreased, then linearly

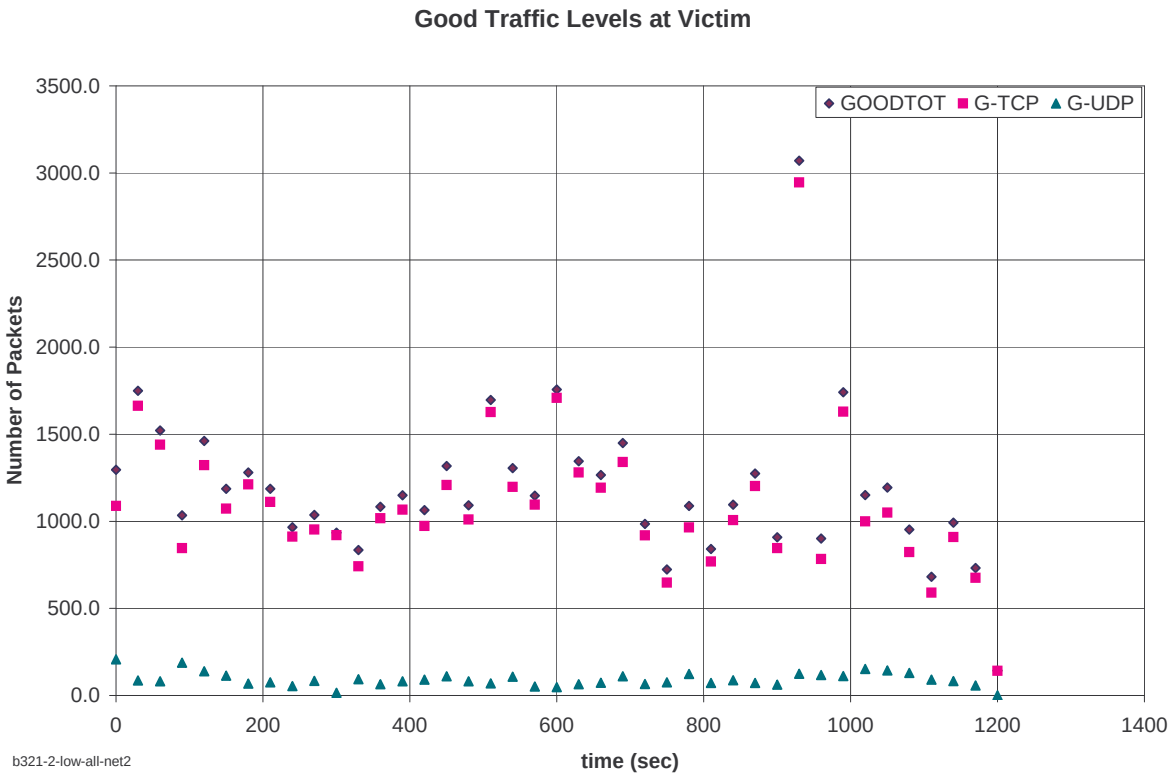

Figure 10.8: Baseline traffic — low level

Figure 10.9: Baseline traffic — high level

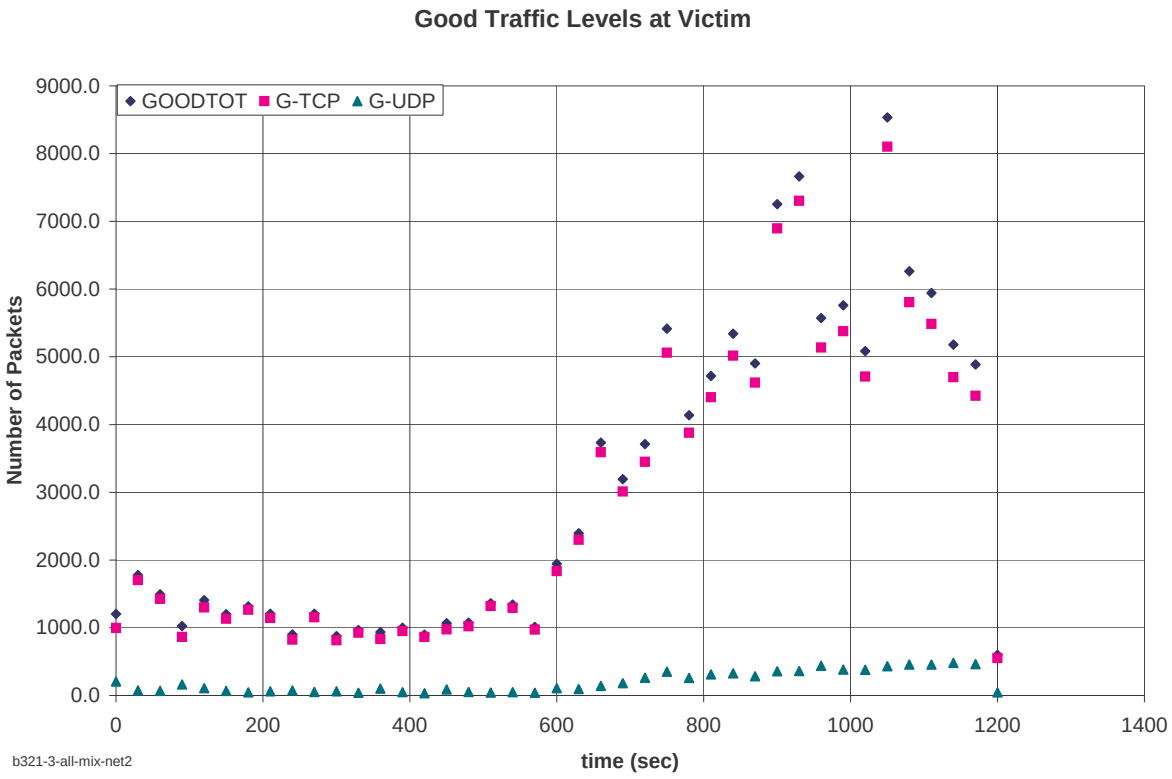

Figure 10.10: Baseline traffic — mixed level

increased once the detection becomes negative. After the *Compliance_Period* the rate will be exponentially increased until it reaches *Maximum_Rate* when it will be removed. Figures 10.11, 10.12 and 10.13 show each of these events during a test run. Diamonds denote attack detection instances, squares denote moments when the exponential decrease of the rate limit is started, stars denote moments when the linear increase of the rate limit is started, x-signs denote moments when the exponential increase is started, triangles denote time when the maximum rate limit *MaxRate* is reached and crosses denote moments when the rate limit is removed. The values on the y-axis are omitted as they only denote incidence of the above events. Test run time is shown on the x-axis.

The level of false positives is low for low levels of baseline traffic, and it is slightly higher for high levels of baseline traffic. The highest incidence of false positives is for mixed levels of baseline traffic.

The appearance of false positives during baseline runs indicates that some collateral damage may be inflicted, on legitimate traffic by the defensive system. To examine this, the percentage of legitimate traffic (sent to the victim) that arrives at the victim is measured for two source networks during these baseline runs, and depicted in Figures 10.14, 10.15 and 10.16.

Triangles denote UDP traffic levels, squares denote TCP traffic levels and diamonds denote total traffic levels. We observe that the percentage of legitimate traffic received at the victim is close to 100%, this indicates that no significant collateral damage is inflicted due to false positives.[1]

[1]The percentage dropoff at the end is the product of the measuring methodology and does not indicate real traffic loss.

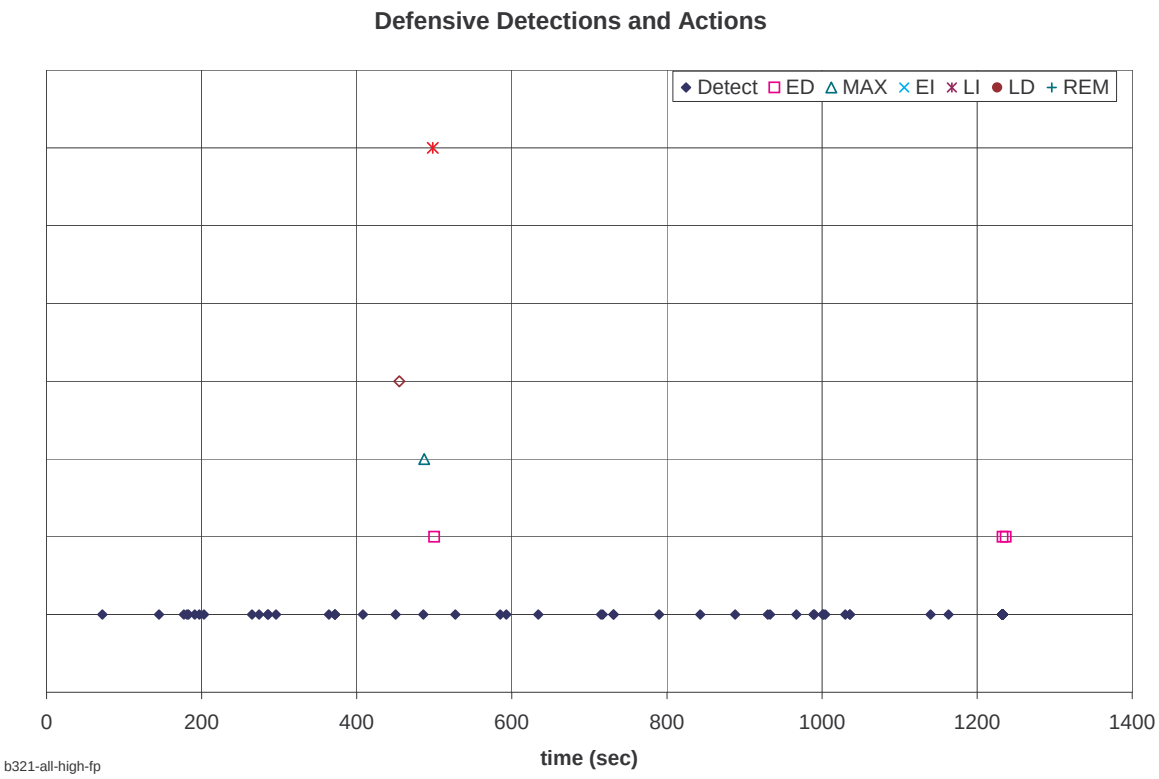

Figure 10.11: False positives — low level

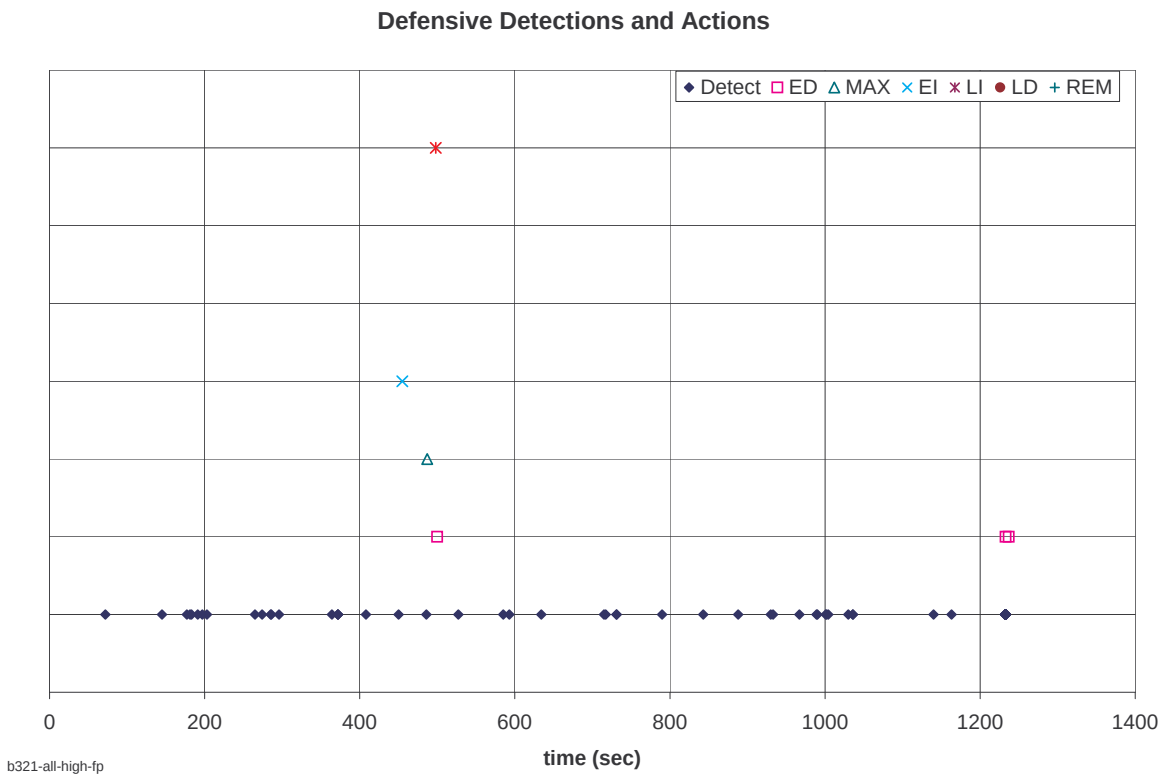

Figure 10.12: False positives — high level

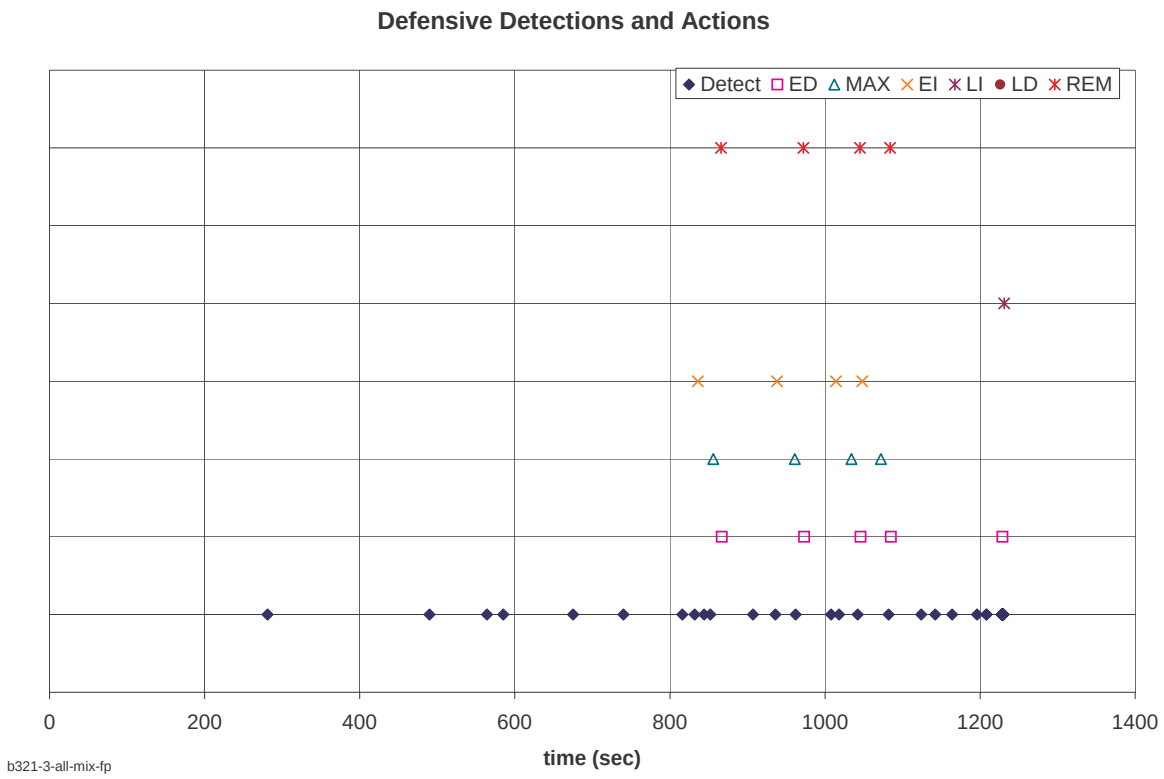

Figure 10.13: False positives — mixed level

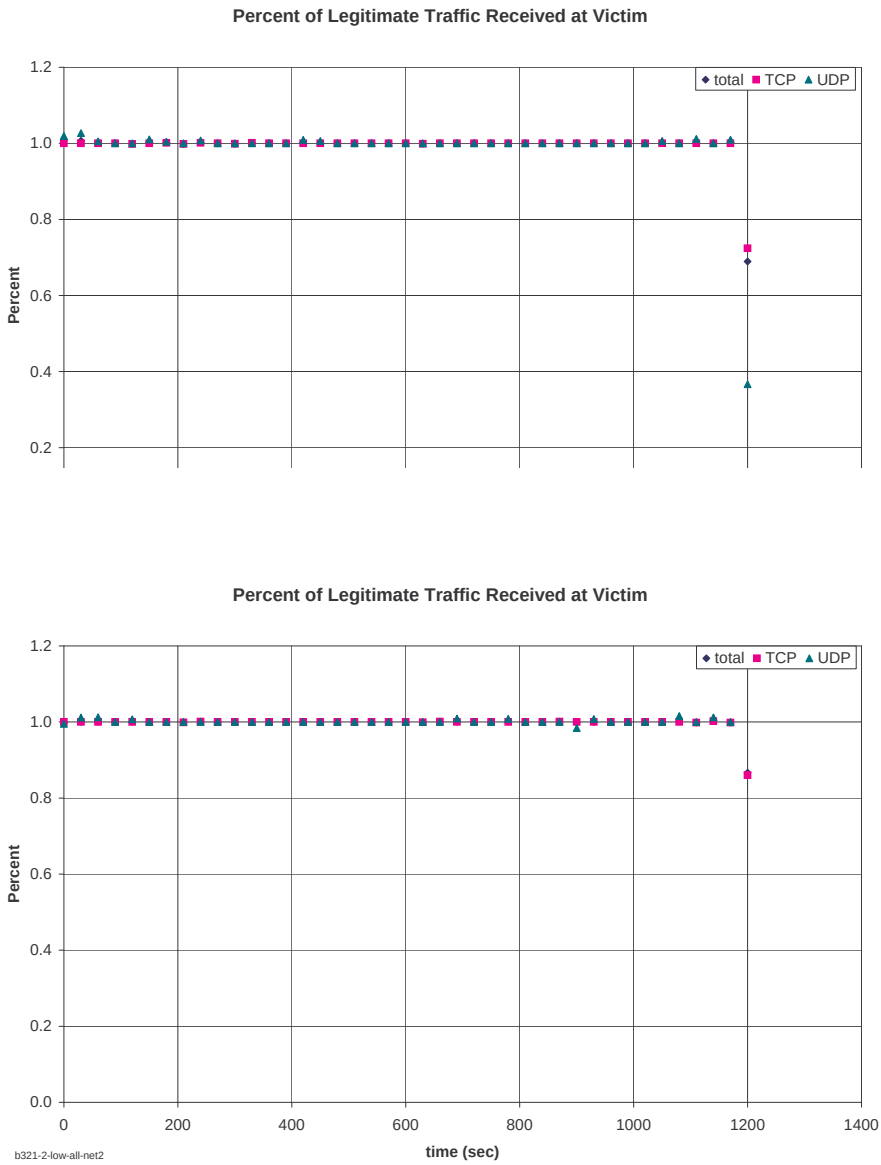

Figure 10.14: Percentage of legitimate traffic received at the victim, depicted for two source networks — low level

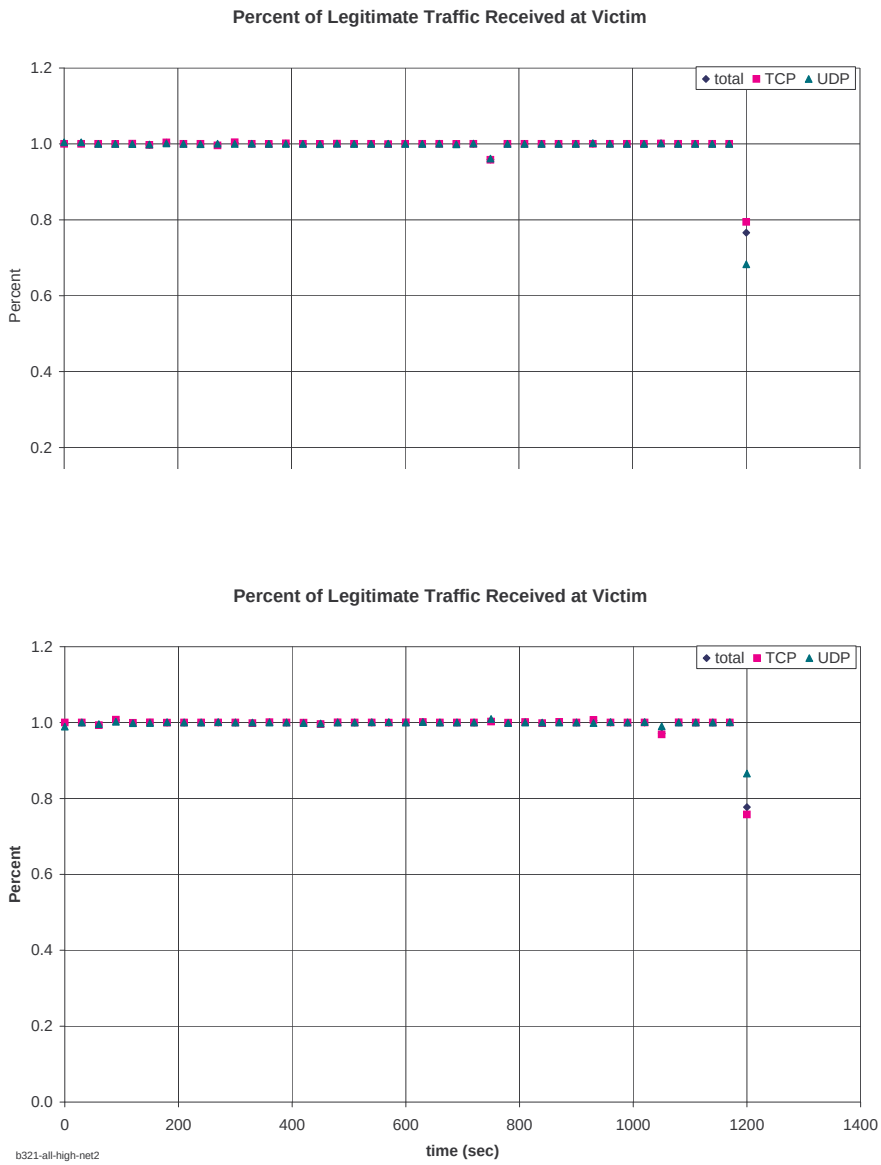

Figure 10.15: Percentage of legitimate traffic received at the victim, depicted for two source networks — high level

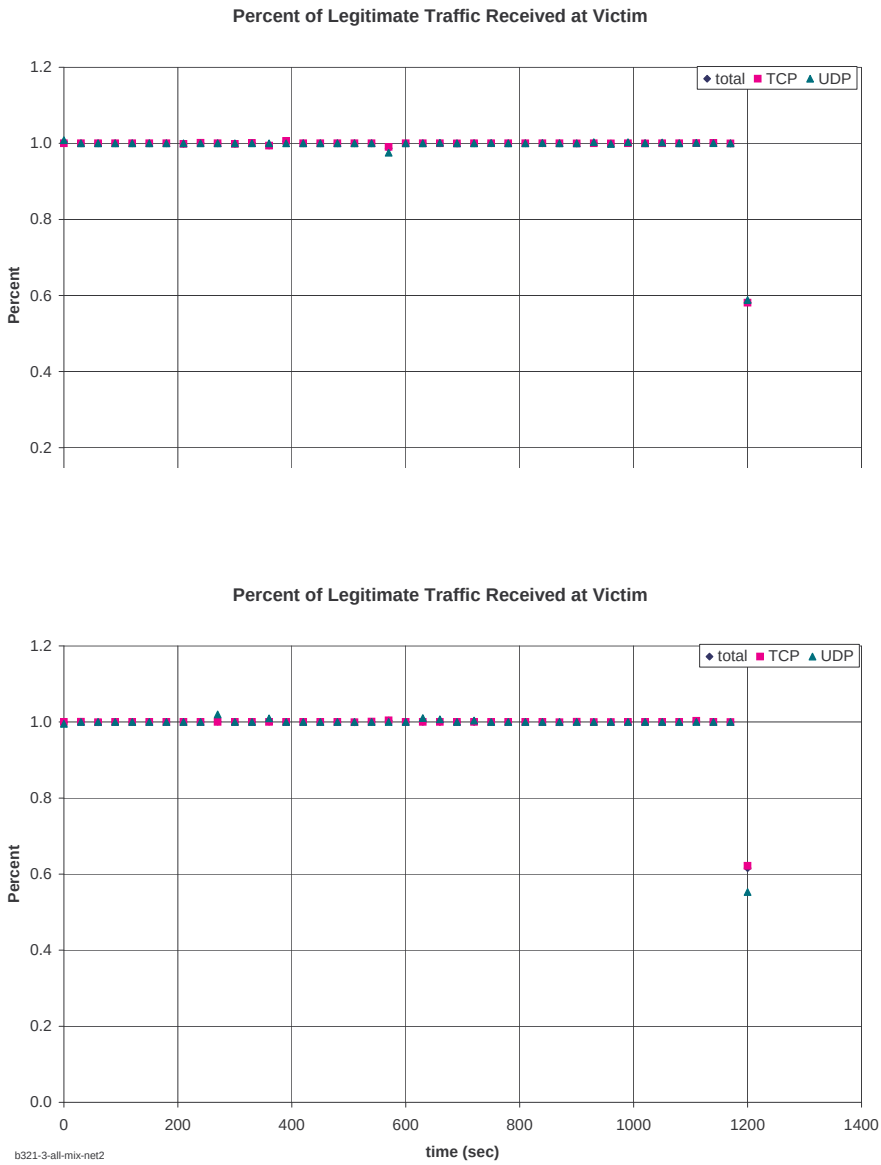

Figure 10.16: Percentage of legitimate traffic received at the victim, depicted for two source networks — mixed level

10.1.3.3 Spoofing UDP Attack

The first type of attack tested is the spoofing UDP attack. Attackers from $FHOST4$, $FHOST7$, $FHOST10$ and $FHOST12$ send UDP packets using subnet spoofing to the victim — $SHOST1$. The attack is run along with the high baseline traffic. Four metrics are defined to measure the level of service that legitimate clients perceive during the attack:

1. Ratio of sent to received legitimate traffic at the victim — measures the legitimate traffic loss due to queuing and defense system action. Ideally it should be 1.

2. Ratio of good to total traffic over time — measures the bandwidth utilization by legitimate traffic. Ideally it should be 1.

3. Traffic levels at the victim — measures the legitimate and attack traffic levels at the victim over time. Ideally legitimate traffic levels should be similar to those in baseline runs, while attack traffic levels should be close to zero.

4. Good traffic levels at the victim — measures the TCP and UDP traffic levels at the victim. Ideally these levels should be similar to those in baseline runs.

Figure 10.17 depicts these metrics when defenses are engaged during the attack. Performance results are contrasted with the case in which no defenses are present, depicted in Figure 10.18.

When defenses are present the ratio of sent to received legitimate traffic is close to 1. This is contrasted to the 0.4 ratio measured during the attack when no defenses are present. This metrics indicates that the integrated system successfully relieves the victim from the denial-of-service effect and provides good

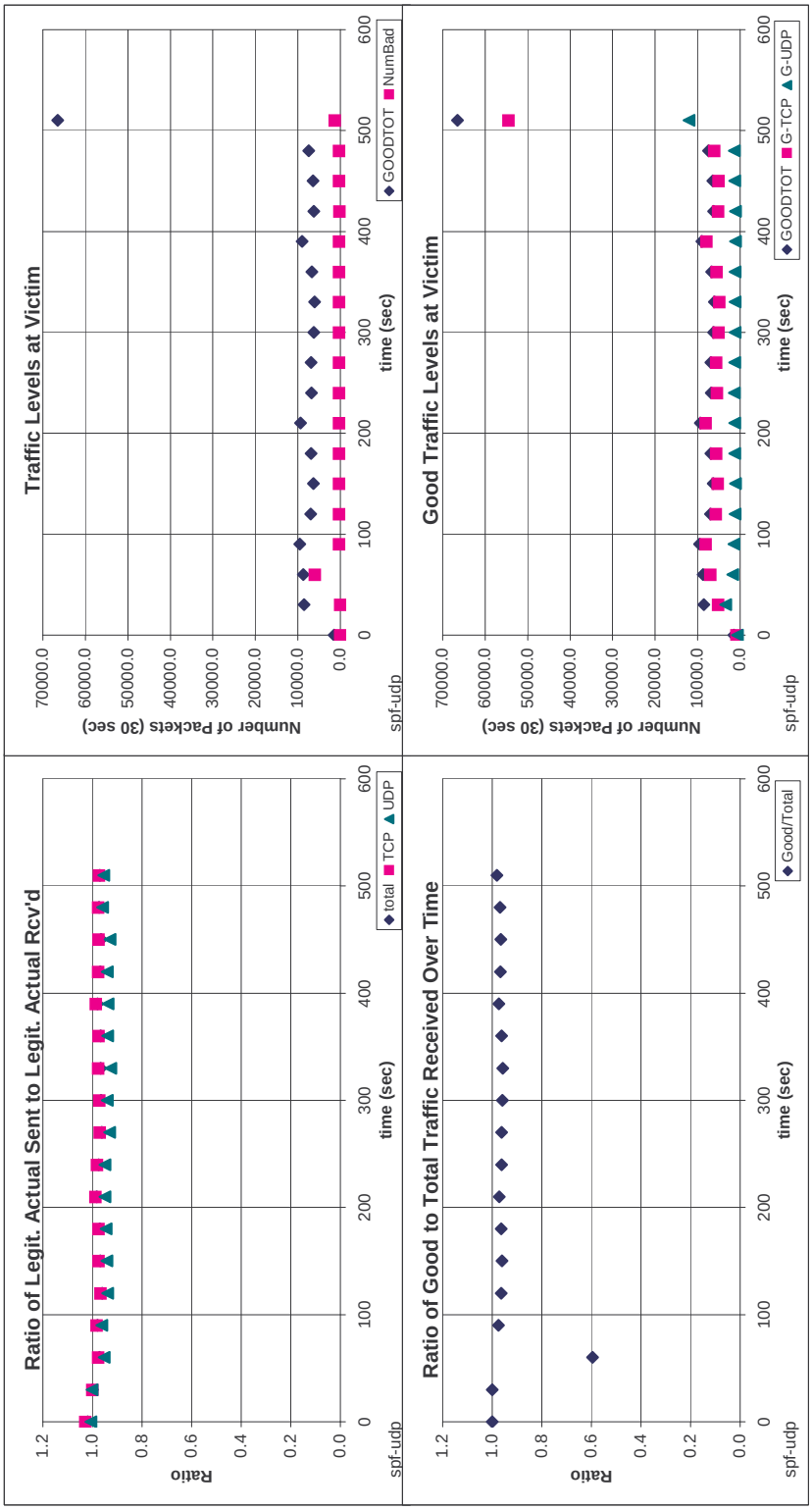

Figure 10.17: Performance metrics — spoofing UDP attack with defense

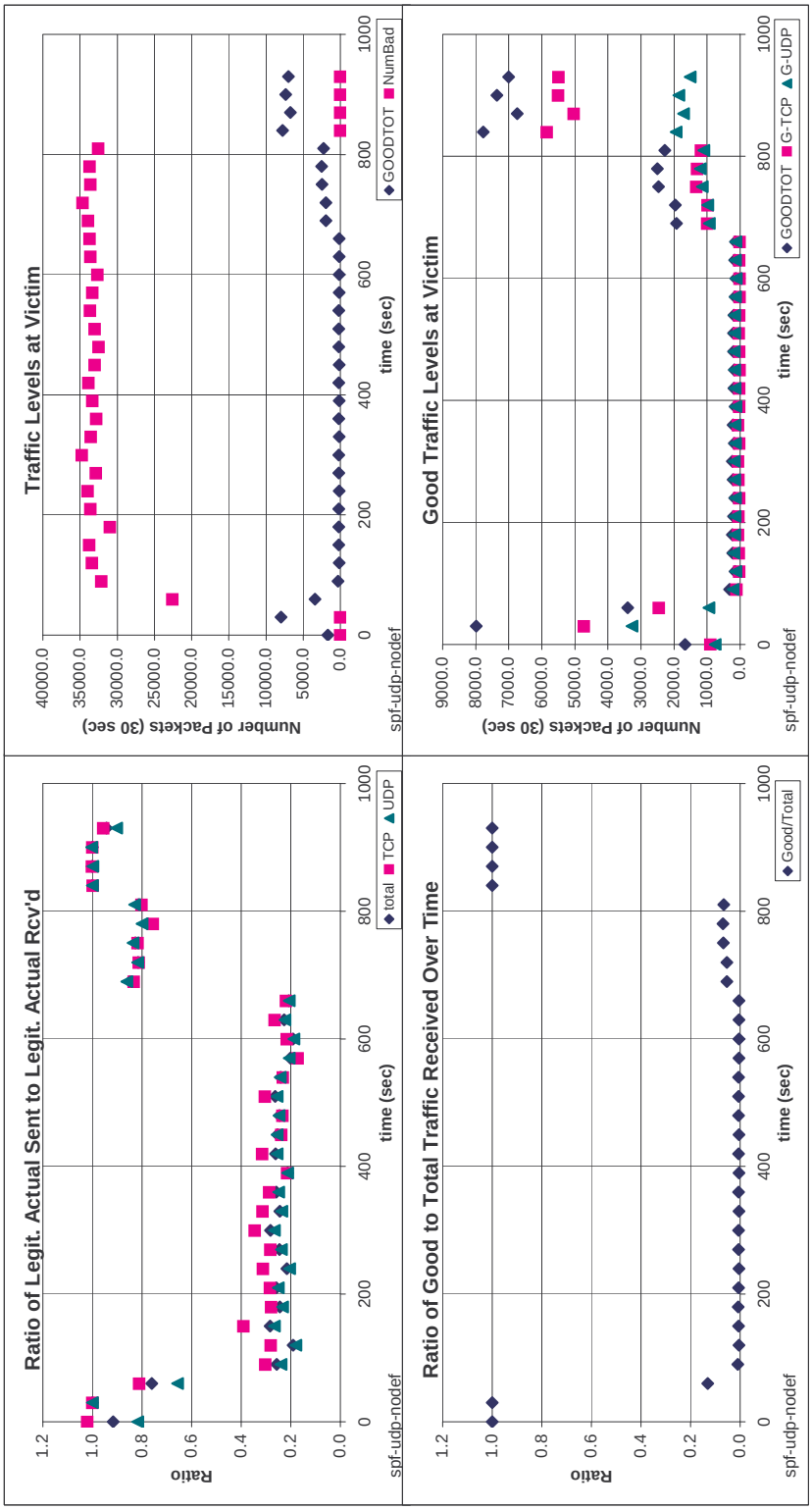

Figure 10.18: Performance metrics — spoofing UDP attack without defense

service to legitimate traffic during the attack. Similarly, the ratio of good to total traffic received over time is close to 1 when defenses are present and close to 0 when they are not present. This indicates that the integrated defense system relieves congestion from the network and assigns resources only to legitimate traffic. During defense engagement, legitimate traffic levels at the victim are similar to those when only baseline traffic is run (high baseline), while attack traffic levels are close to zero. This is contrasted to very low levels of legitimate traffic (close to zero) and high (35000 packets in a 30-second interval) levels of attack traffic when defenses are not engaged. The defense system successfully performs selective dropping of the attack traffic, which results in good service offered to legitimate traffic.

10.1.3.4 Spoofing TCP Attack

During a spoofing TCP attack, attackers from $FHOST4$, $FHOST7$, $FHOST10$ and $FHOST12$ send TCP SYN packets using subnet spoofing to the victim — $SHOST1$. The attack is run along with high baseline traffic. Figure 10.19 depicts the performance metrics in the case where defenses are engaged during the attack. Performance results are contrasted with the case where no defenses are present (depicted in Figure 10.20).

When defenses are present, the ratio of sent to received legitimate traffic is close to 1. This is contrasted to a 0.4 ratio measured during the attack when no defenses are present.[2] This metric indicates that the integrated system successfully relieves the victim from the denial-of-service effect and provides good service to legitimate traffic during the attack. Similarly, the ratio of good to

[2]Note that this metric increases to values above 1 at three points. This indicates the arrival of traffic sent at previous low-throughput intervals. This traffic was delayed in the network or lost and retransmitted by the source.

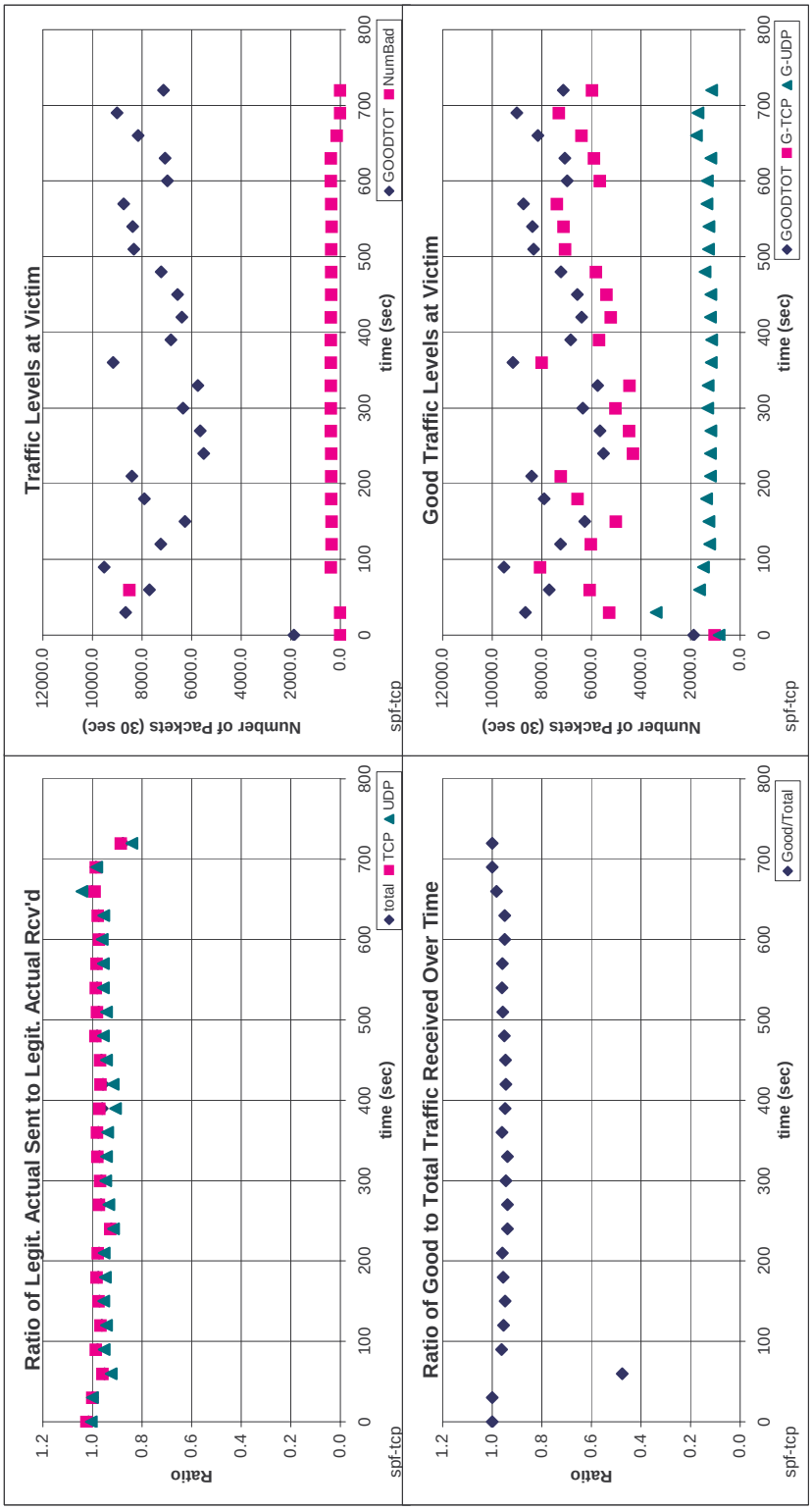

Figure 10.19: Performance metrics — spoofing TCP attack with defense

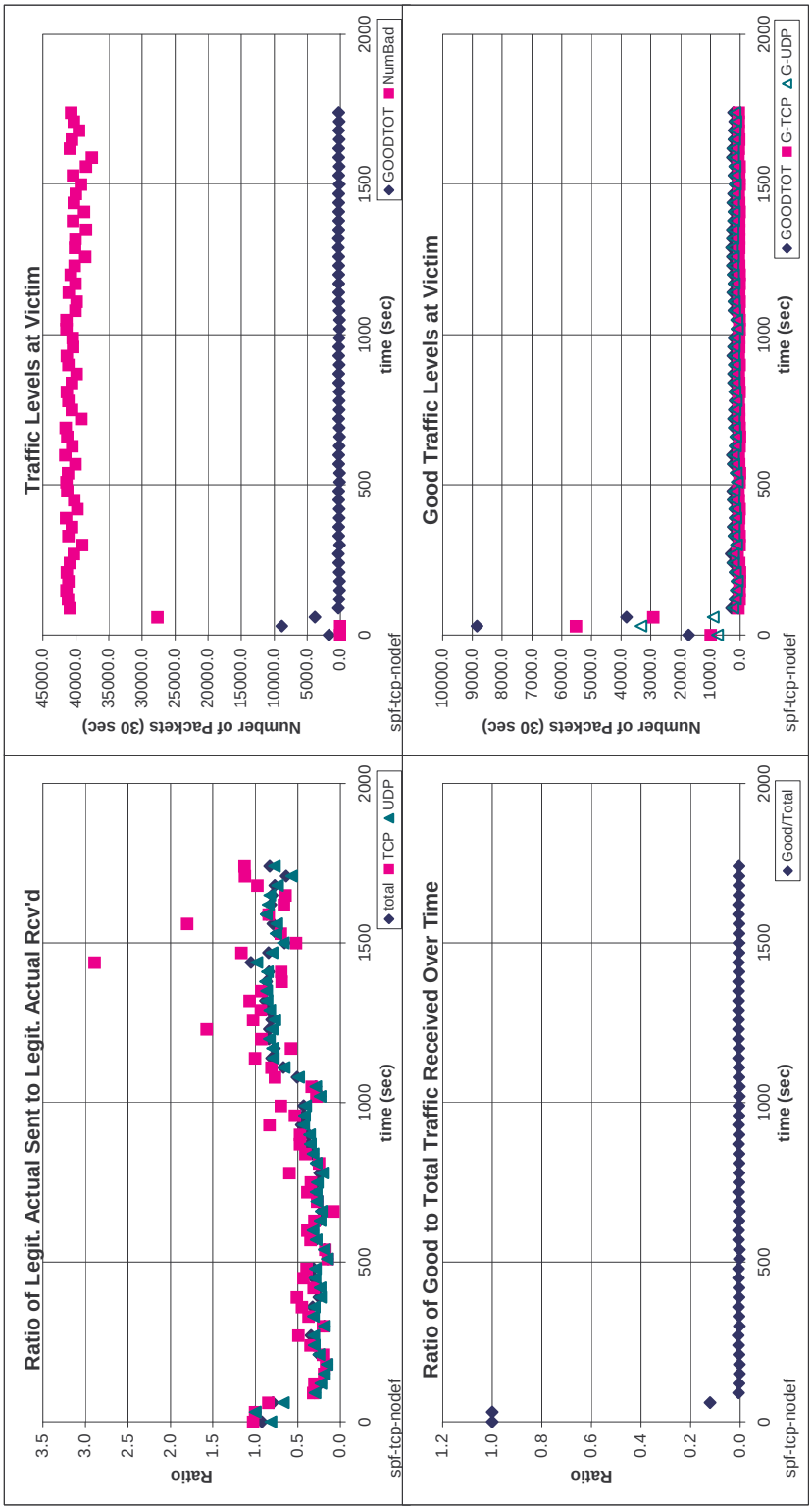

Figure 10.20: Performance metrics — spoofing TCP attack without defense

total traffic received over time is close to 1 when defenses are present and close to 0 when they are not present. This indicates that the integrated defense system relieves congestion from the network and assigns resources only to legitimate traffic. During defense engagement, legitimate traffic levels at the victim are similar to those when only baseline traffic is run (high baseline), while attack traffic levels are close to zero. This is contrasted to very low levels of legitimate traffic (close to zero) and high (40000 packets in a 30-second interval) levels of attack traffic when defenses are not engaged. The defense system successfully performs selective dropping of the attack traffic, which results in good service offered to legitimate traffic.

10.1.3.5 Spoofing ICMP Attack

During a spoofing ICMP attack, attackers from $FHOST4$, $FHOST7$, $FHOST10$ and $FHOST12$ send ICMP packets using subnet spoofing to the victim — $SHOST1$. The attack is run with high baseline traffic. Figure 10.21 shows the performance metrics when defenses are engaged during the attack. Performance results are contrasted with the case where no defenses are present (depicted in Figure 10.22).

When defenses are present, the ratio of sent to received legitimate traffic is close to 1. This is contrasted to a 0.2 ratio measured during the attack when no defenses are present. This metric indicates that the integrated system successfully relieves the victim from the denial-of-service effect and provides good service to legitimate traffic during the attack. The ratio of good to total traffic received over time is between 0.8 and 1 when defenses are present and close to 0 when they are not present. This indicates that the integrated defense system relieves congestion from the network and assigns resources more to legitimate than to

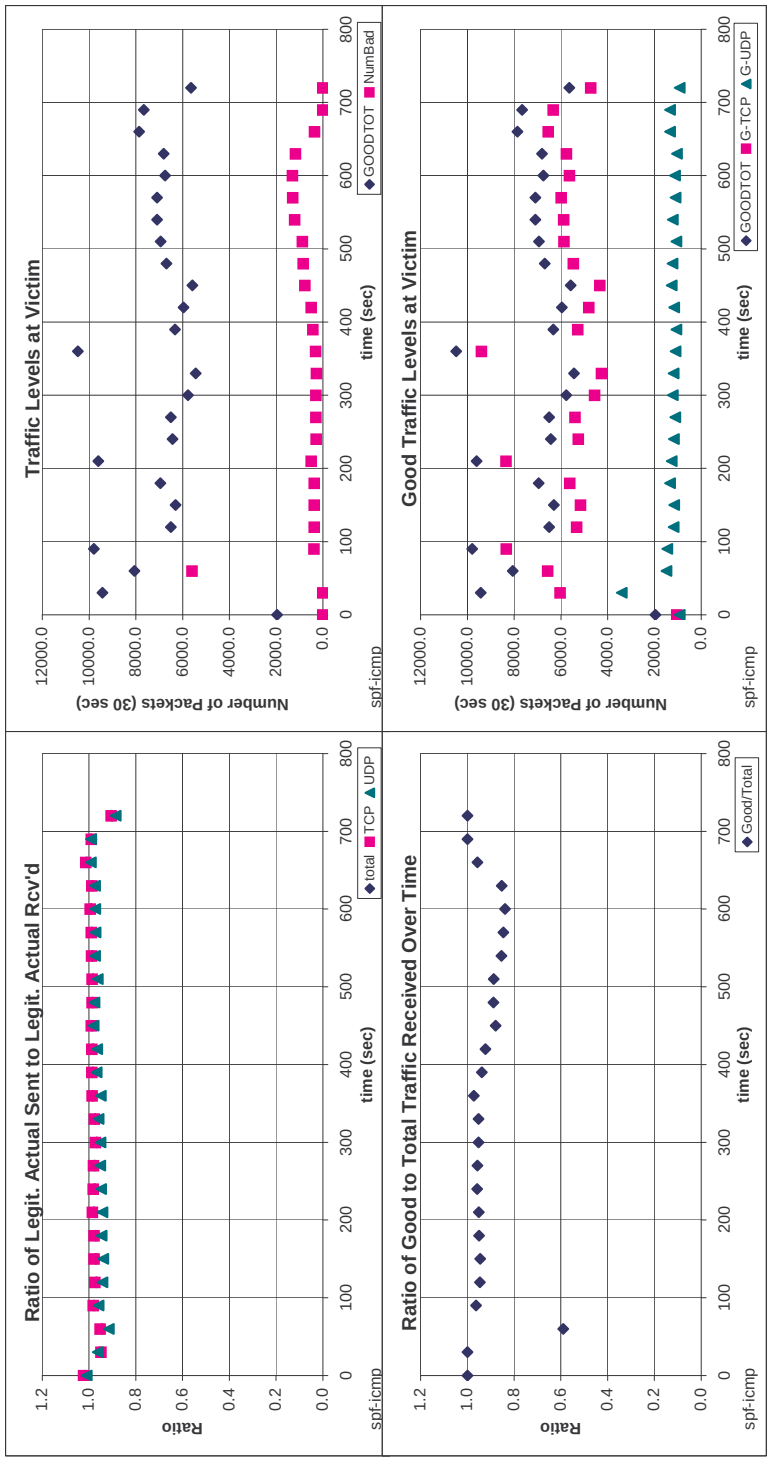

Figure 10.21: Performance metrics — spoofing ICMP attack with defense

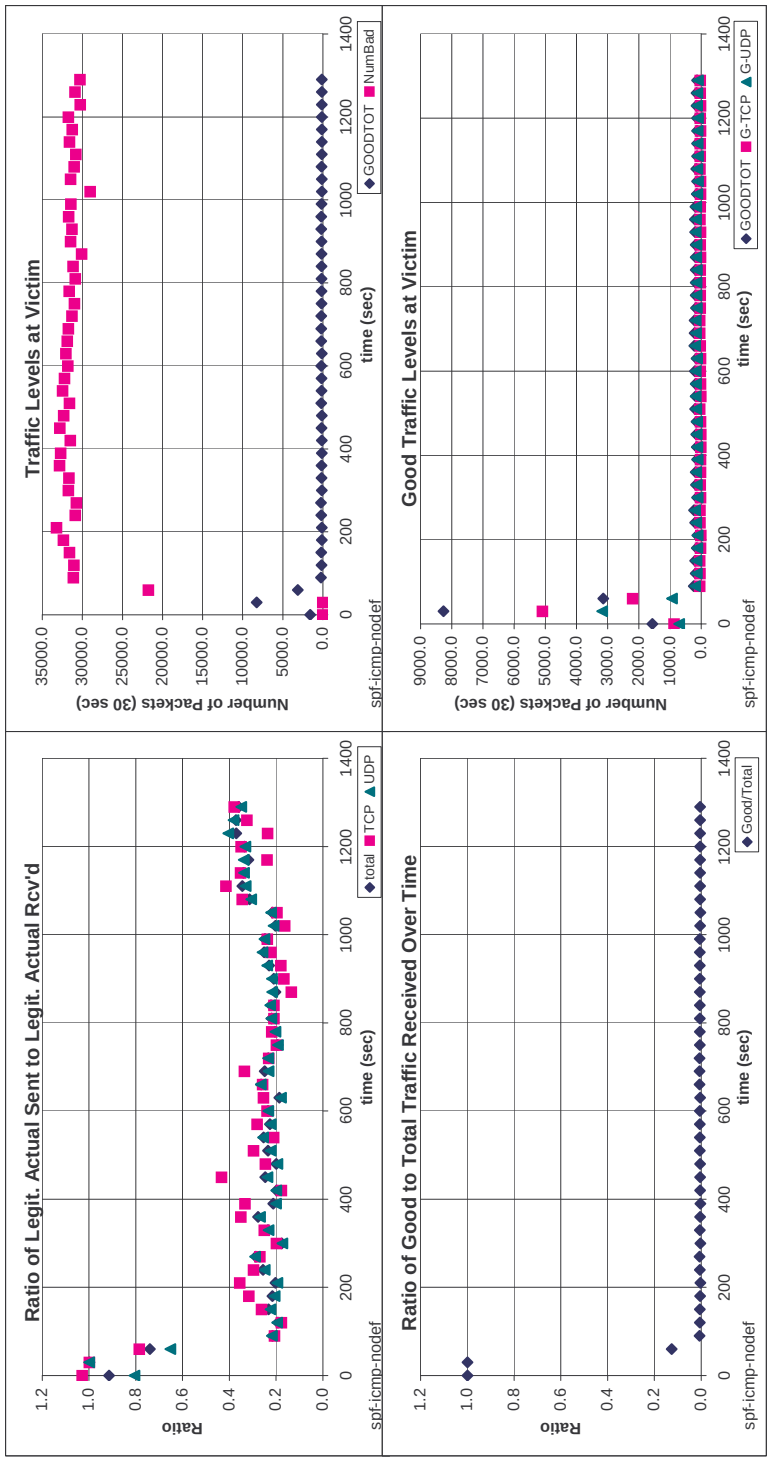

Figure 10.22: Performance metrics — spoofing ICMP attack without defense

attack traffic. Attack traffic still gets about 10% to 20% of resources. During defense engagement, legitimate traffic levels at the victim are similar to those when only baseline traffic is run (high baseline) while attack traffic levels are below 2000 packets in a 30-second interval. This is contrasted to very low levels of legitimate traffic (close to zero) and high (30000 packets in a 30-second interval) levels of attack traffic when defenses are not engaged. In this experiment defense system successfully performs selective dropping of the attack traffic, which results in good service offered to legitimate traffic.

10.1.3.6 Spoofing-All Attack

During spoofing-all attack, attackers from $FHOST4$, $FHOST7$, $FHOST10$ and $FHOST12$ send a mix of UDP, TCP SYN and ICMP packets using subnet spoofing to the victim — $SHOST1$. The attack is run along with high baseline traffic. Figure 10.23 depicts the performance metrics when defenses are engaged during the attack. Performance results are contrasted with the case where no defenses are present (depicted in Figure 10.24).

When defenses are present, the ratio of sent to received legitimate traffic is close to 1. This is contrasted to a 0.2 ratio measured during the attack when no defenses are present. This metric indicates that the integrated system successfully relieves the victim from the denial-of-service effect and provides good service to legitimate traffic during the attack. The ratio of good to total traffic received over time is between 0.95 and 1 when defenses are present and close to 0 when they are not present. This indicates that the integrated defense system relieves congestion from the network and assigns resources more to legitimate than to attack traffic. During defense engagement, legitimate traffic levels at the victim are similar to those where only baseline traffic is run (high baseline), while attack

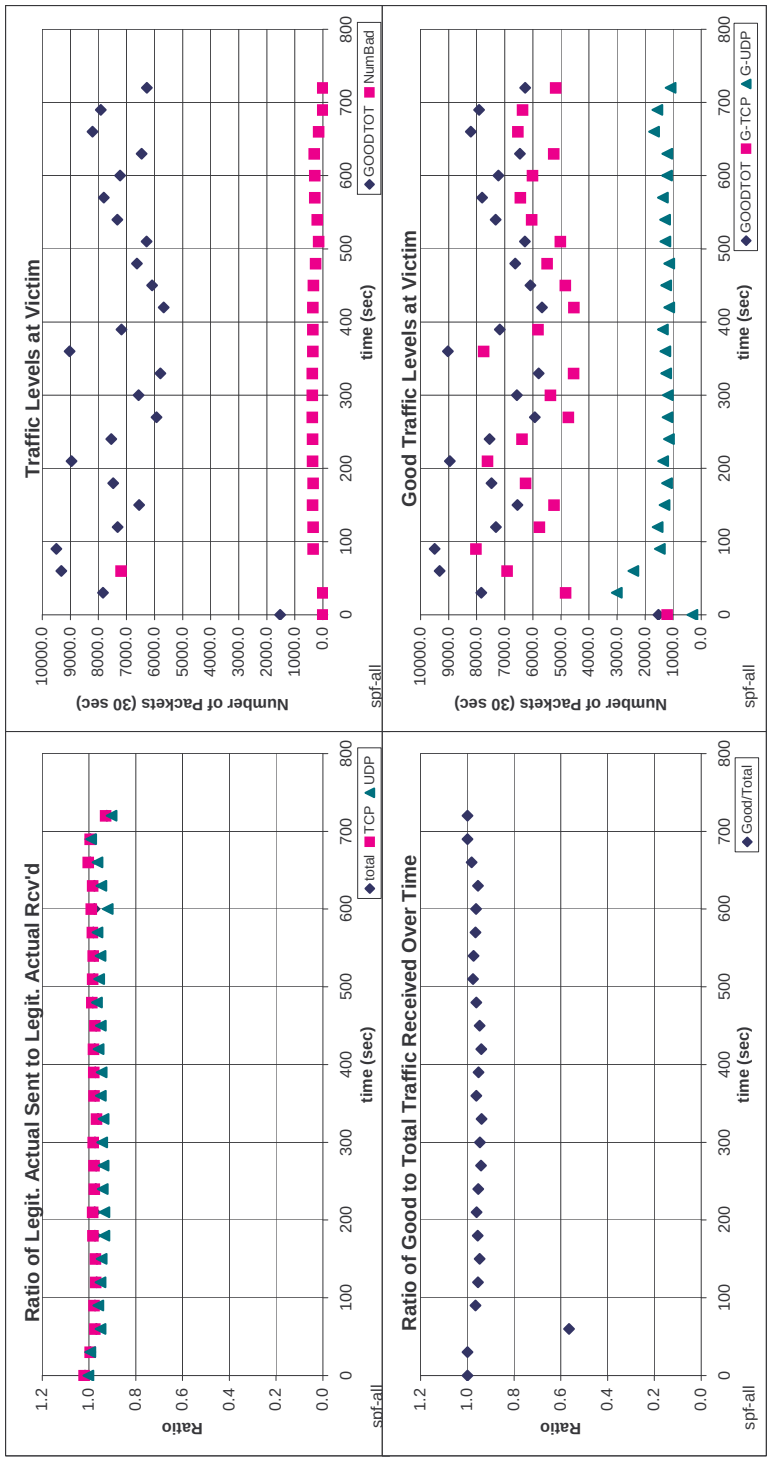

Figure 10.23: Performance metrics — spoofing-all attack with defense

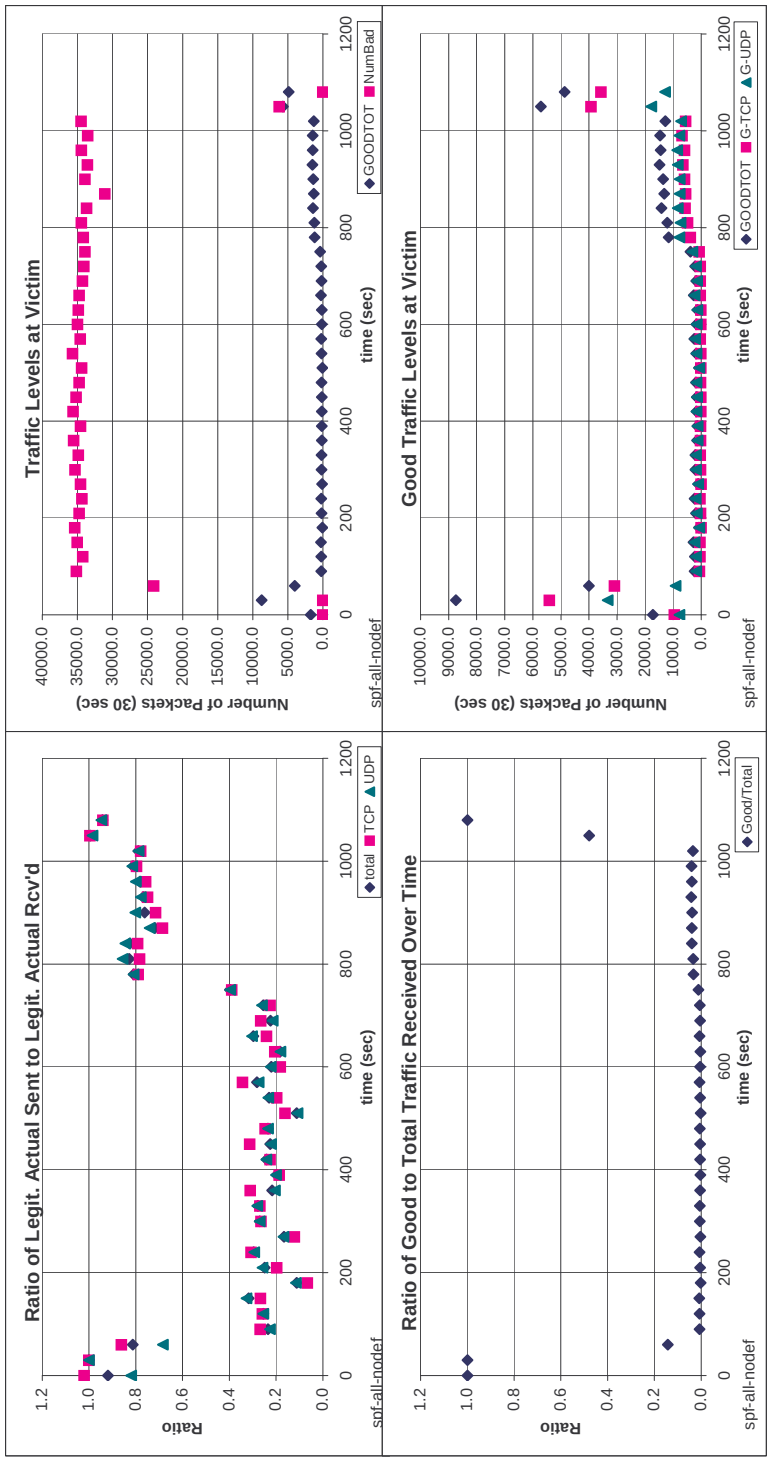

Figure 10.24: Performance metrics — spoofing-all attack without defense

traffic levels are below 1000 packets in a 30-second interval. This is contrasted to very low levels of legitimate traffic (close to zero) and high (35000 packets in a 30-second interval) levels of attack traffic when defenses are not engaged. All of the above metrics indicate that the defense system successfully performs selective dropping of the attack traffic, which results in good service offered to legitimate traffic.

10.1.3.7 Rolling Attack

During a rolling attack, attackers from $FHOST3$, $FHOST4$, $FHOST7$, $FHOST10$, $FHOST12$, $FHOST14$ and $FHOST16$ send packets using subnet spoofing to the victims — $SHOST1$ and $SHOST2$. This attack takes advantage of the timing involved in the COSSACK/D-WARD system. Red Team trials indicated that it took approximately 5 seconds to get throttled by D-WARD after continuous flooding. It was also observed that it took approximately 60 seconds for a machine to get its "bad" reputation cleared from COSSACK after launching a flooding attack. The rolling attack coordinates several flooding agents to take advantage of these timing characteristics. Each agent cycles through its targets, flooding them for 5 or 6second periods, it then lies dormant for a 60 second period. Except for timing characteristics, this scenario is almost identical to the pulsing attack scenario described in section 9.7. The rolling attack is run along with high baseline traffic. Figure 10.25 depicts the performance metrics when defenses are engaged during the attack. Performance results are contrasted with the case when no defenses are present (depicted in Figure 10.26).

When defenses are present the ratio of total sent to received legitimate traffic is close to 1, while the ratio of UDP sent to received traffic is between 0.8 and 1. This is contrasted to the ratio measured during the attack when no defenses

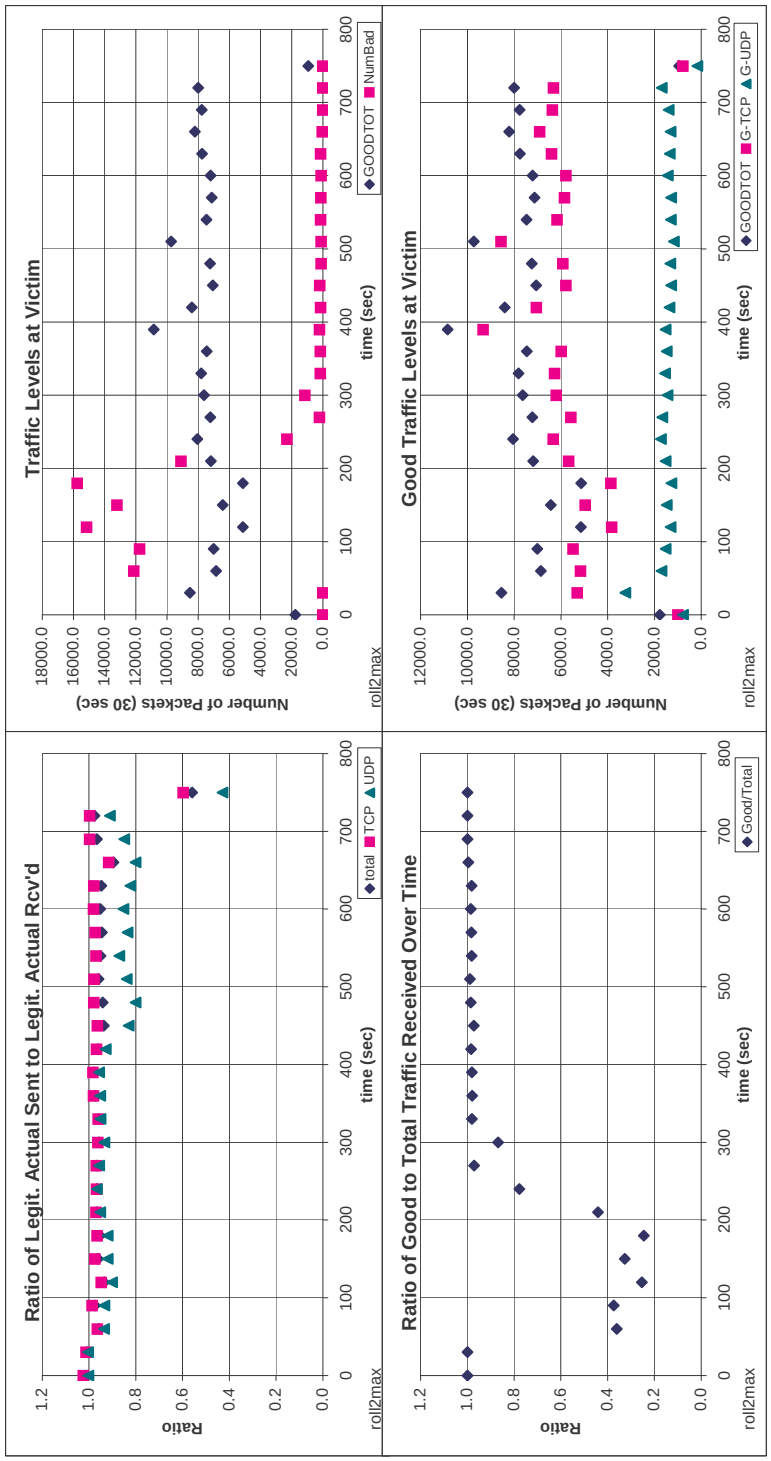

Figure 10.25: Performance metrics — rolling attack with defense

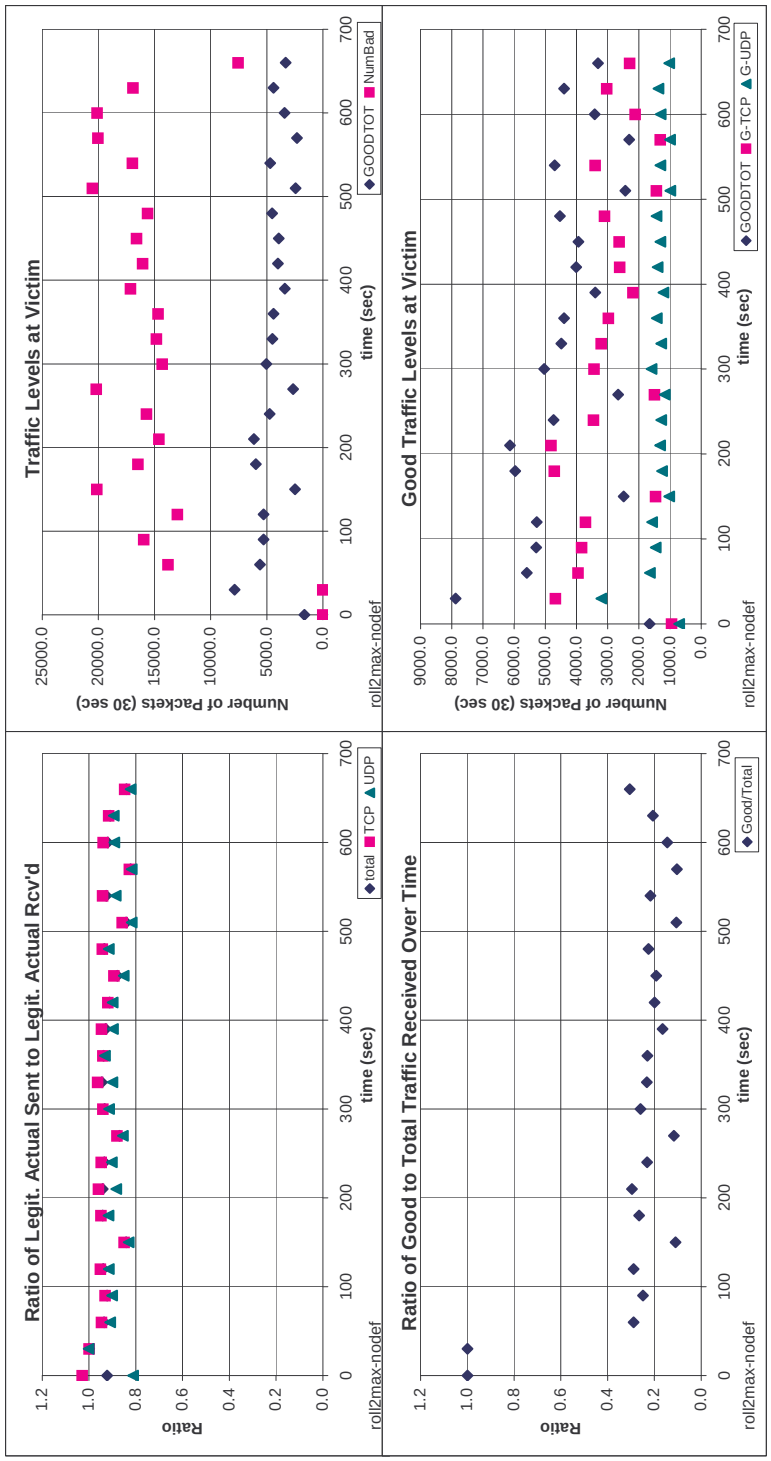

Figure 10.26: Performance metrics — rolling attack without defense

are present, which is between 0.8 and 1 both for TCP and for UDP traffic. The attack-only metric indicates that the rolling attack does not inflict a significant denial-of-service effect at the victim. This effect is successfully relieved with defense system action. The defense system brings the TCP traffic ratio close to 1. As the D-WARD 3.0 version does not deploy legitimate UDP connection models, it cannot differentiate between legitimate UDP traffic and attack traffic. This accounts for the drop in the UDP traffic ratio. This ratio is between 0.8 and 1 even in the presence of the defense system. The ratio of good to total traffic received over time declines to 0.4 when the attack starts and recovers to values close to 1 after 250 seconds in the case when defenses are present. This ratio is close to 0.2 when defenses are not present. This indicates that the integrated defense system relieves congestion from the network and assigns resources more to legitimate than to attack traffic. The low values of the ratio of good to total traffic that persist for 250 seconds indicate that the rolling attack can successfully deny service to legitimate traffic, in spite of defense presence. As experiments in section 9.7 suggest, this effect can be ameliorated by increasing the value of the *Compliance_Period* parameter in D-WARD.

During defense engagement, legitimate traffic levels at the victim are about 80% of those when only baseline traffic is run (high baseline), while attack traffic levels are close to zero. This is contrasted to 50% levels of legitimate traffic and high ($15000 - 20000$ packets in a 30-second interval) levels of attack traffic when defenses are not engaged. The defense system successfully performs selective dropping of the attack traffic which results in good service offered to legitimate traffic. Selectiveness of response is lower than in previous tests which is manifested in service level reduction to UDP traffic.

10.1.3.8 HTTP Request Flood Combined with Rolling Attack

An HTTP request flood attack launches an unending stream of redundant HTTP requests. This uses up the targets resources in generating responses. It has been noted that two attacking FHOST machines were sufficient to degrade service from a single SHOST machine. No spoofing was present. This attack is not interesting by itself under the rules of engagement for COSSACK/D-WARD since they do not defend against this type of attack. This attack is included by itself for engineering purposes only, and is intended to be run in conjunction with other packet flooding attacks to provide redeeming traffic.

In this test run, an HTTP request flood is combined with a rolling attack. Attackers from $FHOST2$ and $FHOST13$ launch an HTTP request flood, targeting the victims $SHOST1$ and $SHOST2$, while $FHOST3$, $FHOST4$, $FHOST7$, $FHOST10$, $FHOST12$, $FHOST14$ and $FHOST16$ launch rolling attack against $SHOST1$ and $SHOST2$. The attack is run along with high baseline traffic. Figure 10.27 depicts the performance metrics when defenses are engaged during the attack. Performance results are contrasted with the case where no defenses are present (depicted in Figure 10.28).

When defenses are present, the ratio of total sent to received legitimate traffic is close to 1, while the ratio of UDP sent to received traffic is between 0.8 and 1. This is contrasted to the ratio measured during the attack when no defenses are present, which is between 0.8 and 1. Both metrics are similar to those in the rolling attack case. They indicate that the HTTP/rolling attack does not inflict a significant denial-of-service effect at the victim, and that this effect is successfully relieved with defense system action. Low levels of UDP traffic ratio occur because D-WARD 3.0 does not implement legitimate UDP connection models. The ratio of good to total traffic received over time declines to 0.4 when the attack starts

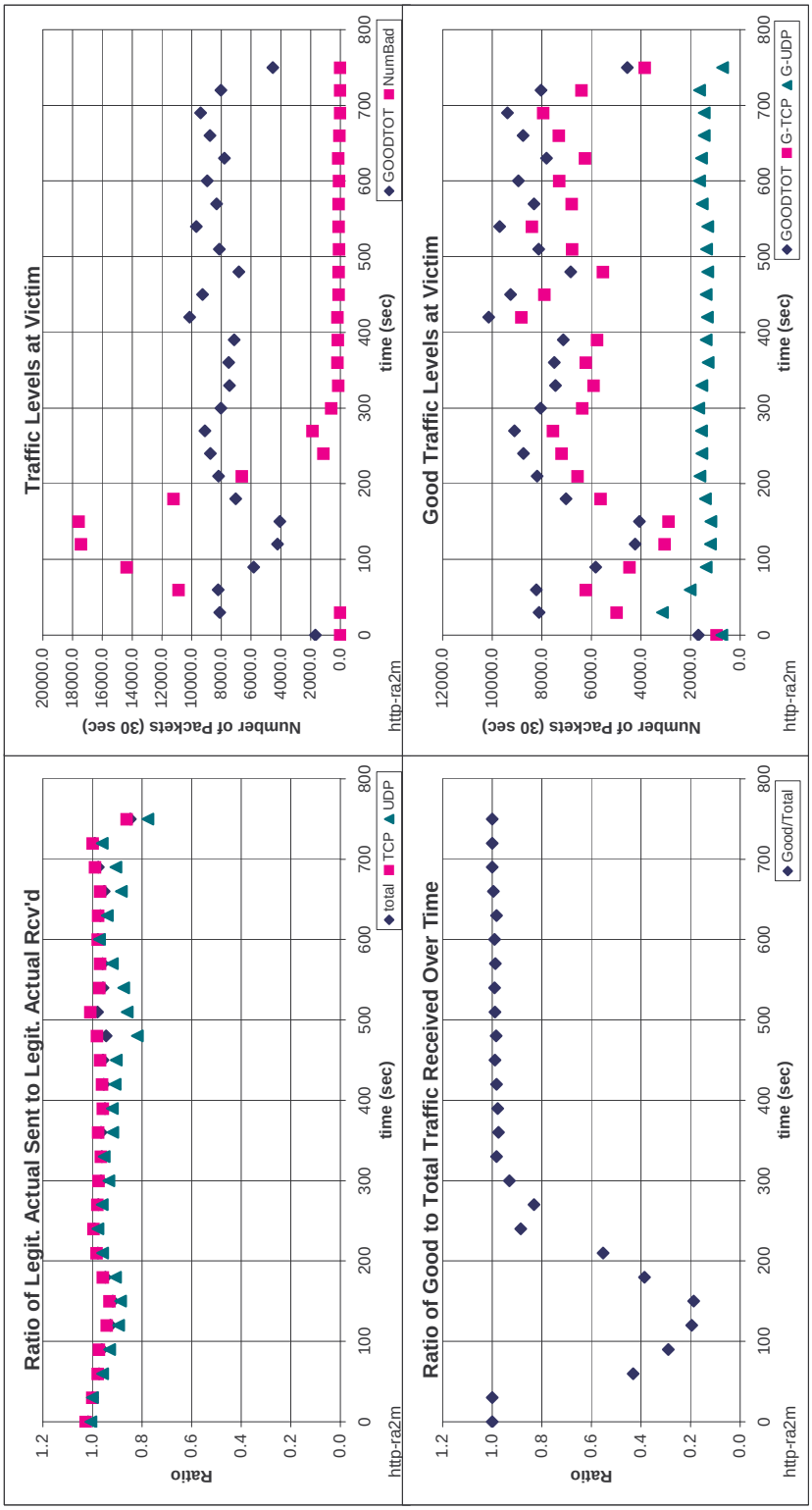

Figure 10.27: Performance metrics — HTTP/rolling attack with defense

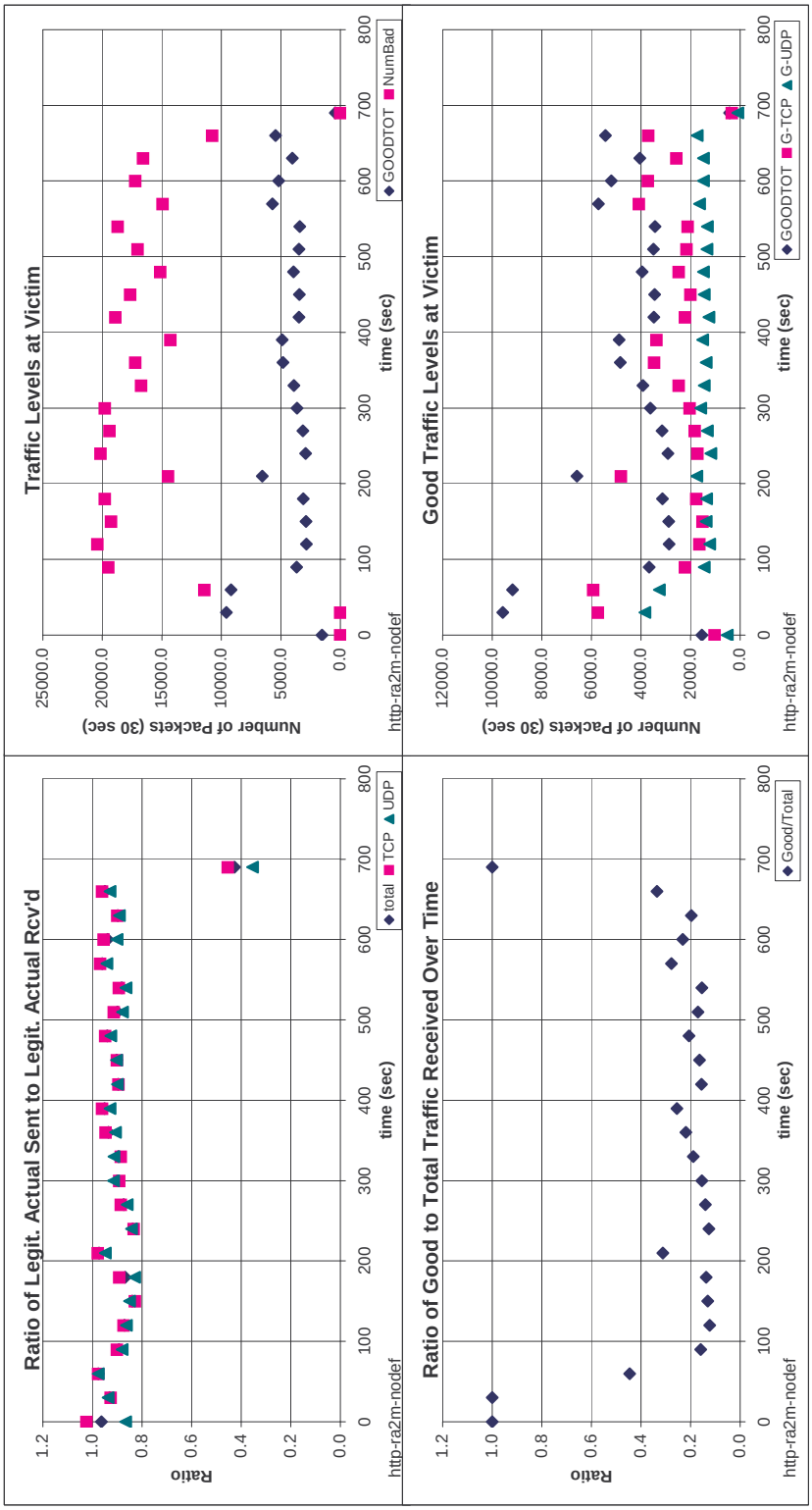

Figure 10.28: Performance metrics — HTTP/rolling attack without defense

and recovers to values close to 1 after 250 seconds in the case when defenses are present. This ratio is close to 0.2 when defenses are not present. This indicates that the integrated defense system relieves congestion from the network and assigns resources more to legitimate than to attack traffic. The explanation for the long delay in controlling the attack (250 seconds) is the same as in the case of the rolling attack given in Section 10.1.3.7. During defense engagement, legitimate traffic levels at the victim are about 80% of those when only baseline traffic is run (high baseline), while attack traffic levels are close to zero. This is contrasted to 50% levels of legitimate traffic and high (20000 packets in a 30-second interval) levels of attack traffic when defenses are not engaged. The defense system successfully performs selective dropping of the attack traffic, which results in good service offered to legitimate traffic. Selectiveness of response is lower than in previous tests, which is manifested in a service-level reduction to UDP traffic.

10.1.3.9 Connection Request Flood Combined with Rolling Attack

All network-capable operating systems have a hard limit on the number of TCP connections allowed per process. These connections are listed in a file descriptor (FD) table of some sort. On virtually all operating systems, this table has a limit of 1024 entries. It is possible to fill the FD table completely with spoofed connections to force the target machine to refuse all subsequent attempted connections. This is an effective DOS against TCP traffic only. However, this does not discount the attack's significance, as the majority of network traffic is TCP. During this attack, all ports on the target machine will be targeted by more than 1024 spoofed connections. This attack was included by the Red Team for engineering purposes only and is intended to be run in conjunction with other packet flooding attacks.

In this test run, a connection request flood is combined with a rolling attack. Attackers from $FHOST2$, $FHOST5$, $FHOST13$ and $FHOST16$ launch an connection request flood, targeting the victims $SHOST1$ and $SHOST2$, while $FHOST3$, $FHOST4$, $FHOST7$, $FHOST10$, $FHOST12$, $FHOST14$ and $FHOST16$ launch rolling attack against $SHOST1$ and $SHOST2$. The attack is run along with high baseline traffic. Figure 10.29 depicts the performance metrics in the case where defenses are engaged during the attack. Those results are contrasted with the case where no defenses are present (depicted in Figure 10.30).

When defenses are present, the ratio of total sent to received legitimate traffic oscillates between 0.4 and 0.6 for the first 100 seconds of the attack, and is close to 1 for the remainder of the attack. The ratio of UDP sent to received traffic is between 0.6 and 0.8 for the entire duration of the attack. This is contrasted to the ratio measured during the attack when no defenses are present. In this case both UDP and TCP ratios are between 0.8 and 1. The attack-only metrics indicates that the connection/rolling attack does not inflict a significant denial-of-service effect at the victim. The low TCP traffic ratio for the first 100 seconds when the defense is engaged occurs because of the dual negative effect of attack and defense on legitimate TCP traffic. The attack fills up the FD table with spoofed connections, denying the service to legitimate TCP traffic. Thus most new TCP connections appear only as SYN packet retransmissions. On the other hand, D-WARD 3.0 does not deploy initial sequence number prediction and thus cannot classify first packets on new connections as legitimate. These packets have a high probability of being dropped, increasing the denial-of-service effect. The ratio of good to total traffic received over time declines to 0.4 when attack starts and recovers to values between 0.6 and 1 after 50 seconds in the case where defenses are present. This ratio is close to 0.2 when defenses are not present. This

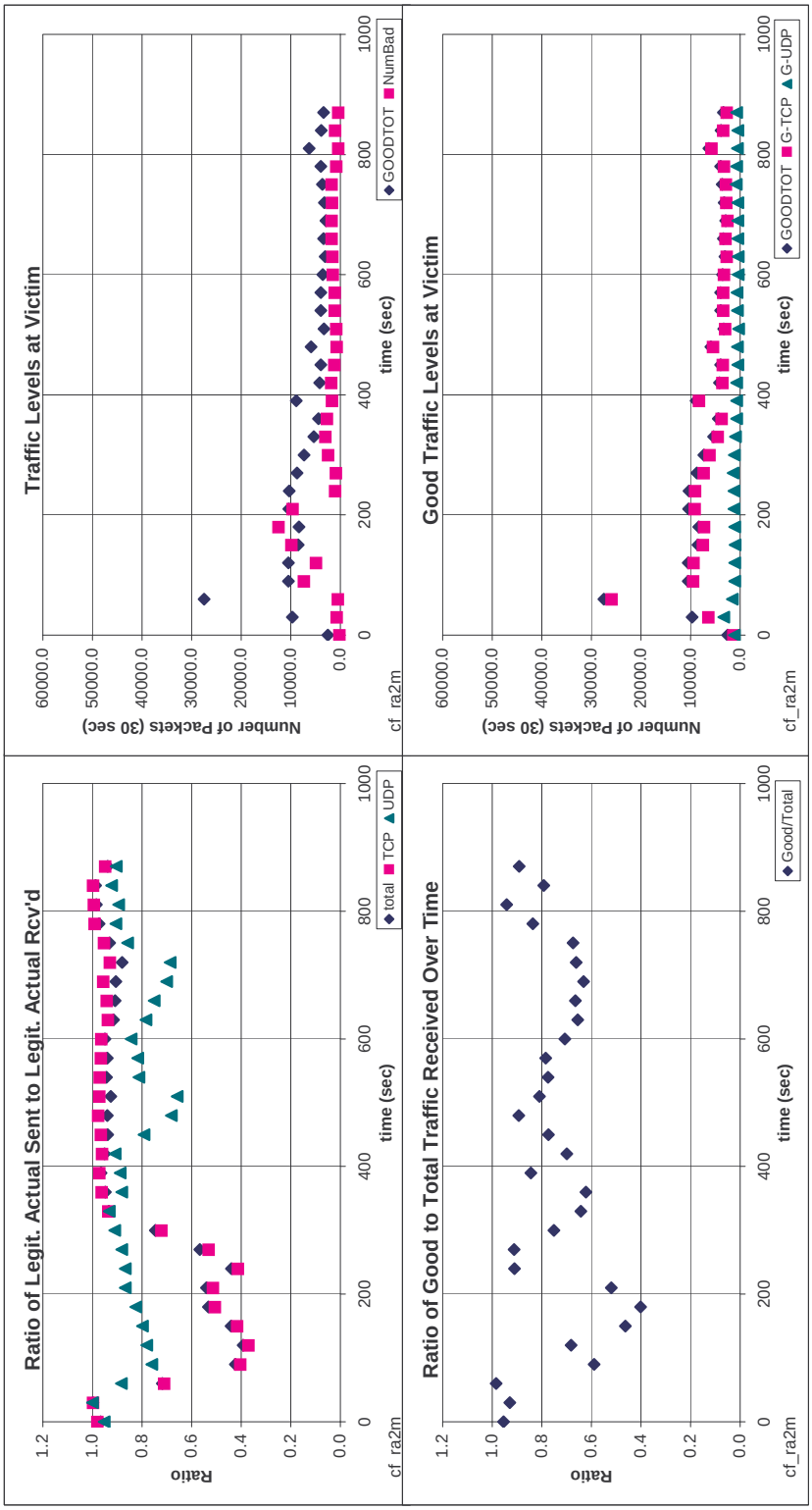

Figure 10.29: Performance metrics — connection/rolling attack with defense

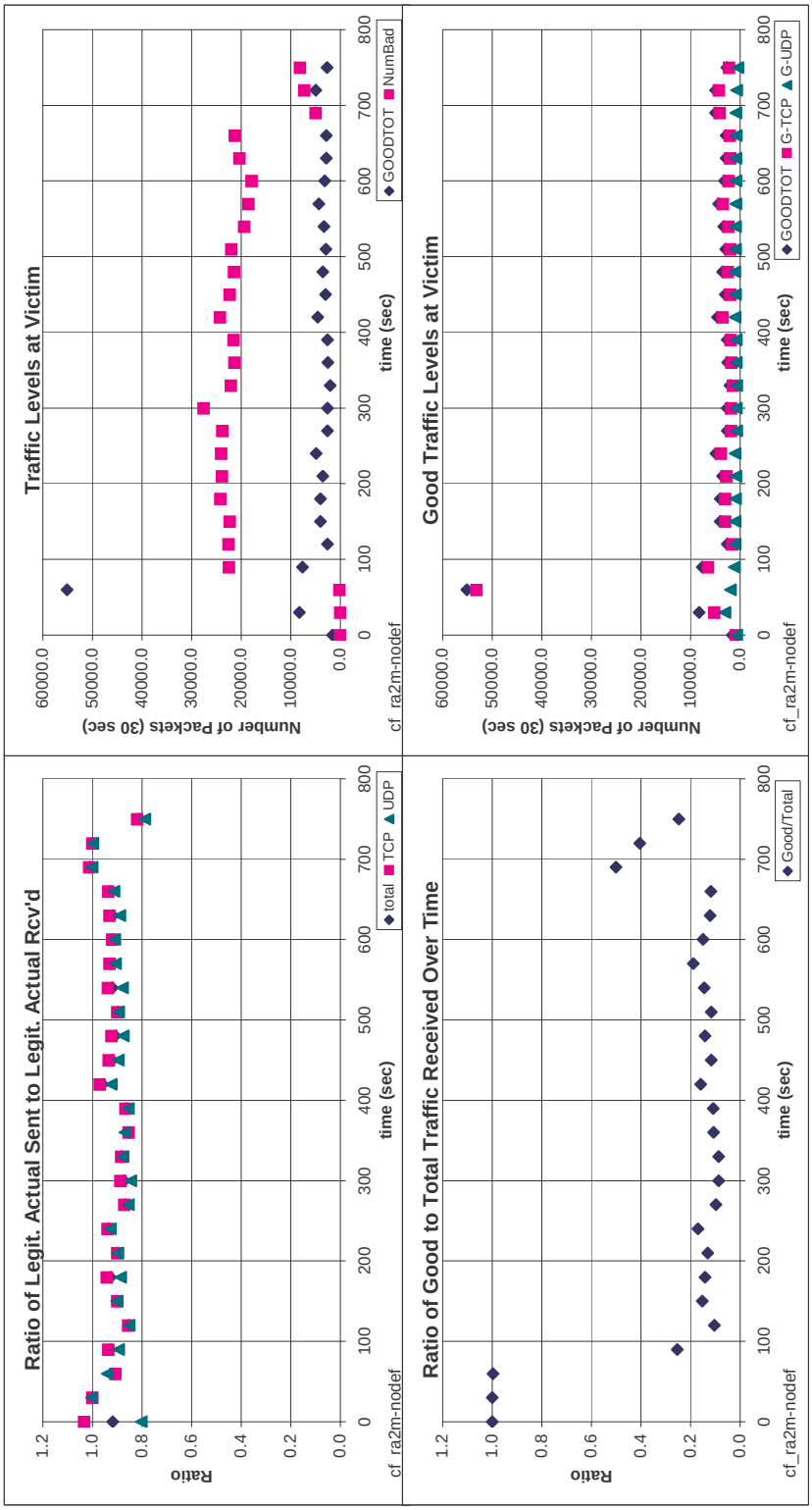

Figure 10.30: Performance metrics — connection/rolling attack without defense

indicates that the integrated defense system relieves congestion from the network and assigns resources more to legitimate than to attack traffic. During defense engagement, legitimate traffic levels at the victim are about 70% of levels when only baseline traffic is run (high baseline), while attack traffic levels are around 3000 packets in a 30-second interval. This is contrasted to 50% levels of legitimate traffic and high (20000 packets in a 30-second interval) levels of attack traffic when defenses are not engaged. It can be concluded that the connection/rolling attack defense system does relieve congestion from the network by removing a significant portion of the attack traffic. However, legitimate traffic receives slightly improved service.

10.1.3.10 ACK Proxy Attack

The ACK proxy attack investigates the idea that the attack connections can be validated by D-WARD, if for every TCP SYN packet sent to flood a target, an ACK is spoofed back from an unprotected source as if it had come from the target. This attack consists of several flooding agents. Each time an agent sends out a packet, it makes a request to a dedicated ACK server via a socket connection. The ACK server at the receiving end of this connection reads the desired packet information from the socket and creates the ACK reply as if it were coming from the target of the TCP flood.

Attackers from $FHOST7$, $FHOST10$, $FHOST13$ and $FHOST15$ launch a TCP SYN flooding attack targeting the victim $SHOST1$, while proxies from $FHOST2$, $FHOST3$, $FHOST4$ and $FHOST17$ send the acknowledgment traffic back to D-WARD, spoofing the victims address. The attack is run along with high baseline traffic. Figure 10.31 depicts the performance metrics when defenses are engaged during the attack. These results are contrasted with the case where

no defenses are present (depicted in Figure 10.32).

When defenses are present, the ratio of total sent to received legitimate traffic is close to 1. This is contrasted to the ratio measured during the attack when no defenses are present, which is close to 0. These metrics indicate that the ACK proxy attack inflicts an enormous denial-of-service effect at the victim. This effect is successfully relieved with defense system action. The ratio of good to total traffic received over time declines to 0.7 when attack starts, but quickly recovers to close to 1 in the case where defenses are present. This ratio is zero when defenses are not present. This indicates that the integrated defense system relieves congestion from the network and assigns resources more to legitimate than to attack traffic. During defense engagement, legitimate traffic levels at the victim are similar to those when only baseline traffic is run (high baseline), while attack traffic levels are close to zero. This is contrasted to zero levels of legitimate traffic and high (28000 packets in a 30-second interval) levels of attack traffic when defenses are not engaged. The defense system successfully performs selective dropping of the attack traffic which results in good service offered to legitimate traffic.

10.2 DefCOM

The Defensive Cooperative Overlay Mesh (DefCOM) is a distributed DDoS defense system being developed in our research group, the Laboratory for Advanced Systems Research (LASR) at UCLA. DefCOM consists of heterogeneous defense nodes organized into a peer-to-peer network. These nodes communicate to achieve a dynamic cooperative defense. The high-level overview of DefCOM's operation is given in Figure 10.33. It shows the presence of legacy routers (white circles), core defense nodes (black circles), alert generators (striped circles), clas-

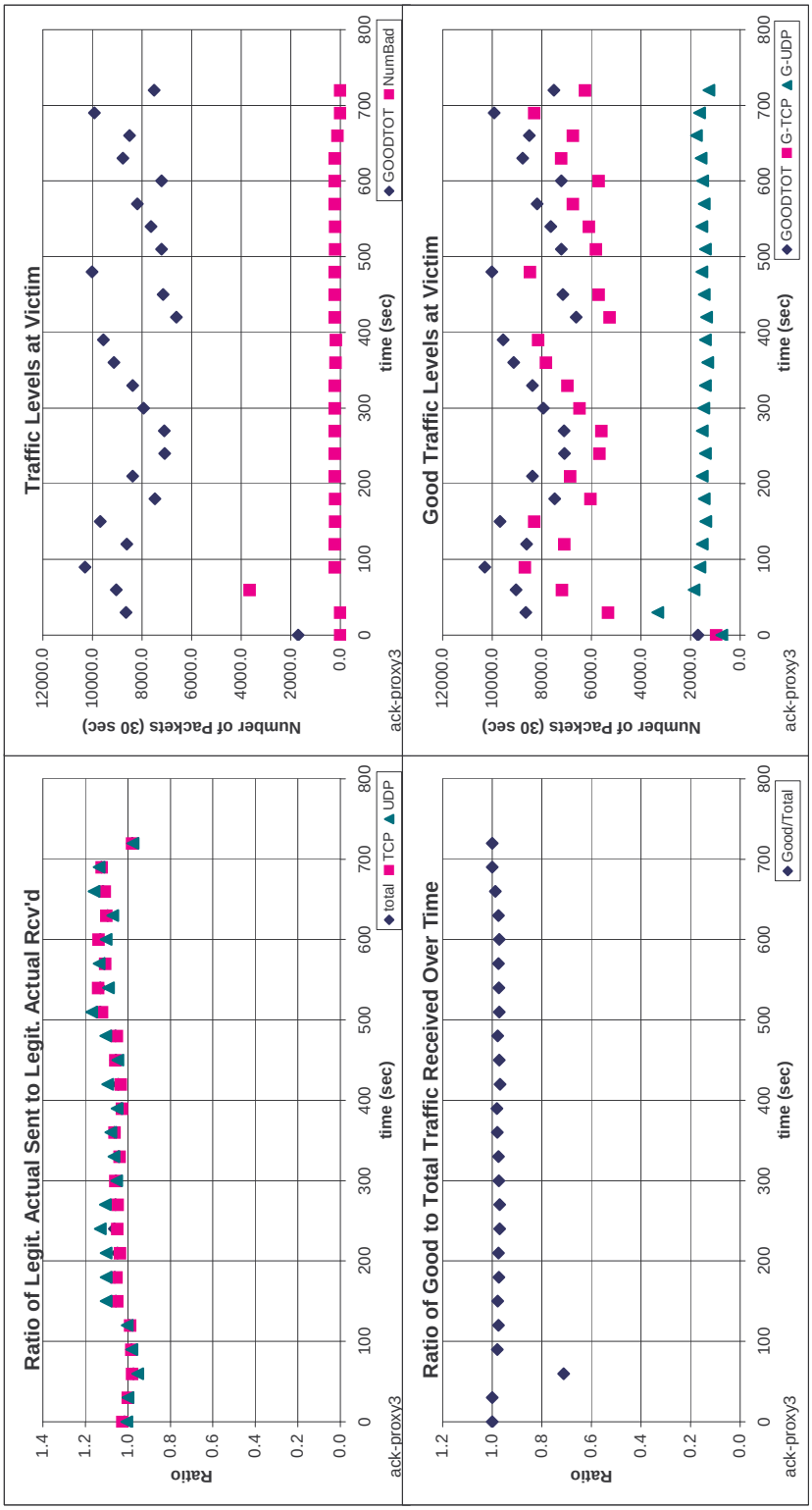

Figure 10.31: Performance metrics — ACK proxy attack with defense

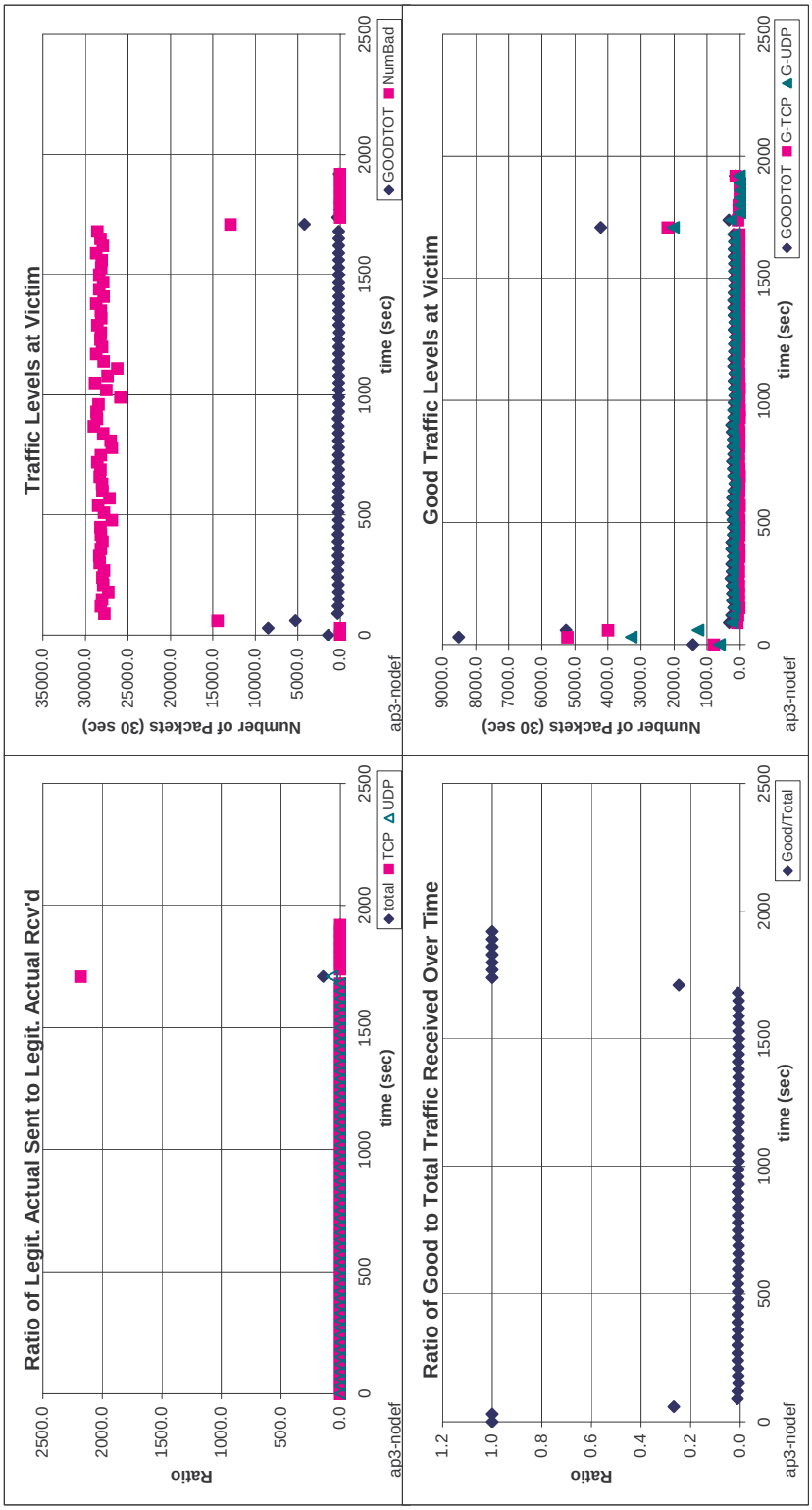

Figure 10.32: Performance metrics — ACK proxy attack without defense

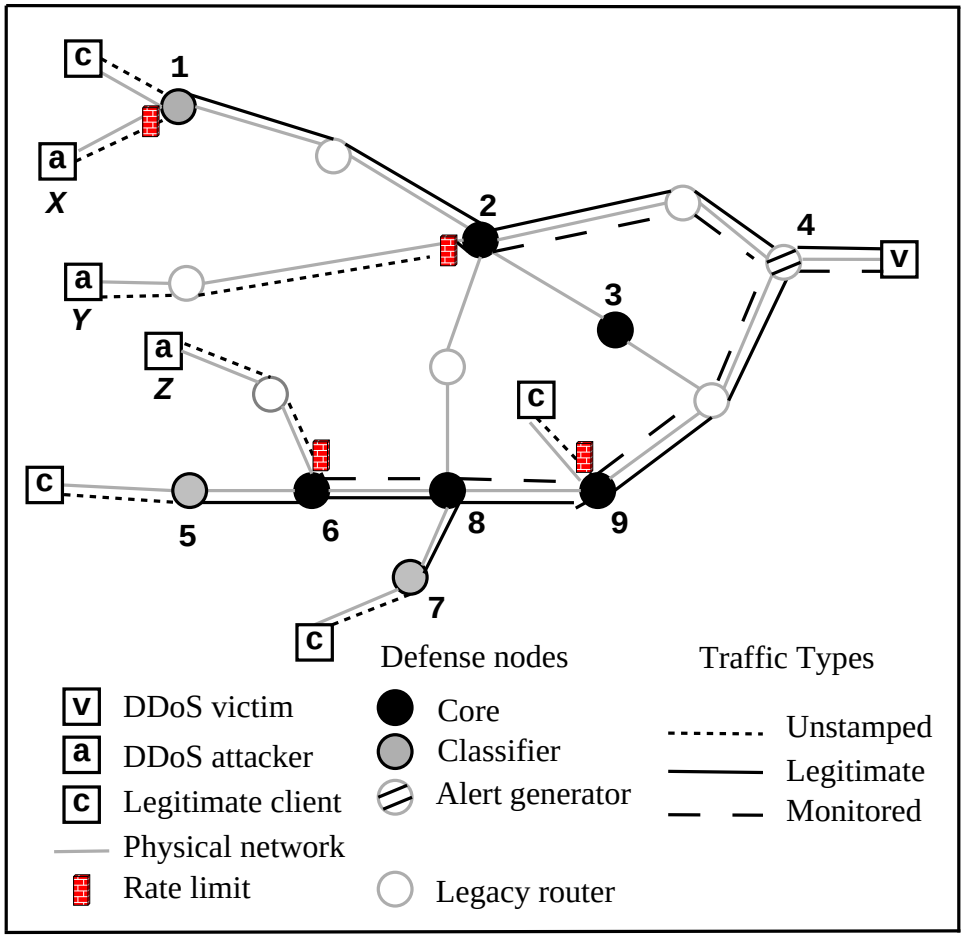

Figure 10.33: DefCOM overview.

sifier nodes (grey circles) and the two types of traffic sources — legitimate clients and attackers. Some traffic sources are behind classifier defense nodes, while others are not. The following sections provide a description of the DefCOM system and how it is integrated with D-WARD.

10.2.1 DefCOM Overview

DefCOM defense nodes are organized in a peer-to-peer network whose topology construction allows approximation of the underlying physical network.[3] During the attack they discover the *victim-rooted traffic tree*, thus identifying upstream-downstream relationships between peers. DefCOM nodes then devise the appropriate rate limits to restrain the attack traffic and place them as close to source networks as possible. At the same time, classifier nodes differentiate legitimate from attack streams, and all nodes in the framework give preferential service to legitimate traffic.

10.2.1.1 Traffic Tree Discovery

When a DDoS attack occurs, the alert generator node close to the victim detects it and propagates an *alert message* to all nodes in the peer network. In Figure 10.33, alert generator node 4 detects a DDoS attack on victim V and informs all other defense nodes through the DefCOM peer network. This process is optimized so that those nodes that do not forward any traffic to the victim do not propagate the signal further; e.g., node 3 will receive the alert but will not send it to its neighbors. Those nodes that observe traffic to the victim (1, 2, 4, 5, 6, 7, 8, and 9) are called *activated nodes*. They cooperate to trace out the topology of the victim-rooted traffic tree by deploying *secure traffic stamping*. Tracing of the tree structure enables each node to assign upstream or downstream classification to its peers, thus defining its policy and the message types to be sent to these peers. Secure packet stamping actually serves four purposes: (1) discovery of the victim-rooted traffic tree topology, (2) differentiation of traffic types, (3) protec-

[3]Nodes can either start off by carefully constructing a peer network topology to resemble the physical topology, or converge to it through reconfiguration guided by traffic observation.

tion of legitimate traffic and (4) transparent operation through legacy routers. Each activated defense node picks a *stamp* and communicates it securely to its neighbors. The node places this stamp in the header of packets that it forwards to the victim.[4] It also observes packets that it receives from its neighbors, looking for their stamps. A node becomes a parent of a neighbor whose stamped traffic it observes. A parent sends an explicit message to its children to inform them of their child status.

To protect the packet stamping mechanism from misuse, every pair of neighboring nodes uses stamps unique to them, and changes the packet stamps on a frequent basis, using the secured communication channel.

To detect and discard false alert messages, the DefCOM network only accepts alarms from registered and authenticated alert generator nodes. Alert generator nodes require the advance permission of a victim before they issue attack alerts. Thus, each alert generator node may only raise alerts for a particular set of victims with whom the alert generator node has established a trust relationship.

10.2.1.2 Distributed Rate-Limiting

Once the tree topology is determined, the nodes cooperate to deploy rate limits that limit attack traffic while protecting and granting preferred service to legitimate traffic. An optimal deployment puts the rate limits as close to the leaves of the tree as possible. This relieves congestion higher up the tree near the victim-root. In Figure 10.33, core defense nodes 1, 2, and 6 deploy rate limits to stop or reduce attack traffic from attack nodes X, Y and Z to V.

The rate limit is initially propagated from the root of the tree (the node that

[4]For instance, the stamp could be placed in the IPv4 identifier field, or in the IPv6 flow identifier field.

has no parents) downstream. Each node assigns an equal share of its rate limit to its children and communicates this through *rate limit requests*. The resource division created through this initial rate limit process can be modified through *resource requests* issued by the node to its parents.

Resource request functionality must be provided by the system, as some children of a node may have a greater number of legitimate clients upstream than other children. On the other hand, the resource request process is particularly vulnerable to manipulation by malicious participants (who could request a large amount of resources at the expense of others) and must be regulated. Generally, the problem of having malicious participants exists in any distributed system (such as routing and the DNS infrastructure), and is as yet unsolved. However, we believe that by limiting the extent of trust between nodes we can limit the amount of unfairness that the malicious participant can introduce in the framework. Nodes should devise monitoring and policing functions to ensure that rate requests are obeyed and resource requests are granted in accordance to negotiated policy. Unreasonable requests should be dropped and trust should be reduced for the requesting node. Subsequent requests from the untrusted node will have a lower probability of being granted.

10.2.1.3 Differentiated Service for Legitimate Traffic

Defense nodes use secure packet stamping to provide different service levels, ensuring that the policed traffic has priority in resource allocation.

Each defense node maintains a *legitimate stamp* and a *monitored stamp*. Classifier defense nodes profile the traffic originating from their networks, and securely mark those packets that are deemed legitimate with a legitimate stamp. Each core node then polices the traffic in the following manner:

- The traffic bearing the legitimate stamp is forwarded to the victim (within the rate limit) and restamped with the node's legitimate stamp.
- The traffic bearing the monitored stamp is forwarded to the victim (within the leftover resources from the legitimate traffic) and restamped with the node's monitored stamp.
- The unstamped traffic is forwarded to the victim only if there are leftover resources from the legitimate and the monitored traffic. It is stamped with the node's monitored stamp.

In Figure 10.33, classifier nodes 1, 5 and 7 vouch for traffic from legitimate clients C. Node 9 also rate-limits traffic from legitimate client C, if necessary. As this client does not have a classifier node to vouch for it, it is an unknown source and thus subject to rate-limiting. The traffic that passes a rate limit in a core node will bear the monitored mark, informing downstream nodes that it has been policed.

The rate limit algorithm in core nodes should provide preferential service to marked traffic by apportioning its allowed resources first to legitimate traffic (that has passed through a classifier node), then to monitored traffic (that has passed through a core node and has been policed), and lastly to unstamped traffic (containing an unknown mix of legitimate and attack traffic).

Figure 10.34 illustrates in more detail how preferred service for legitimate traffic is achieved using the packet-stamping mechanism. This figure is, in fact, a magnified lower branch of Figure 10.33.

A legitimate traffic flow originating in the 192 subnet is shown passing through defense node 5 in Figure 10.34. Originally unstamped, it leaves defense node 5 bearing stamp 222, classified as legitimate traffic. Upon reaching defense node 6,

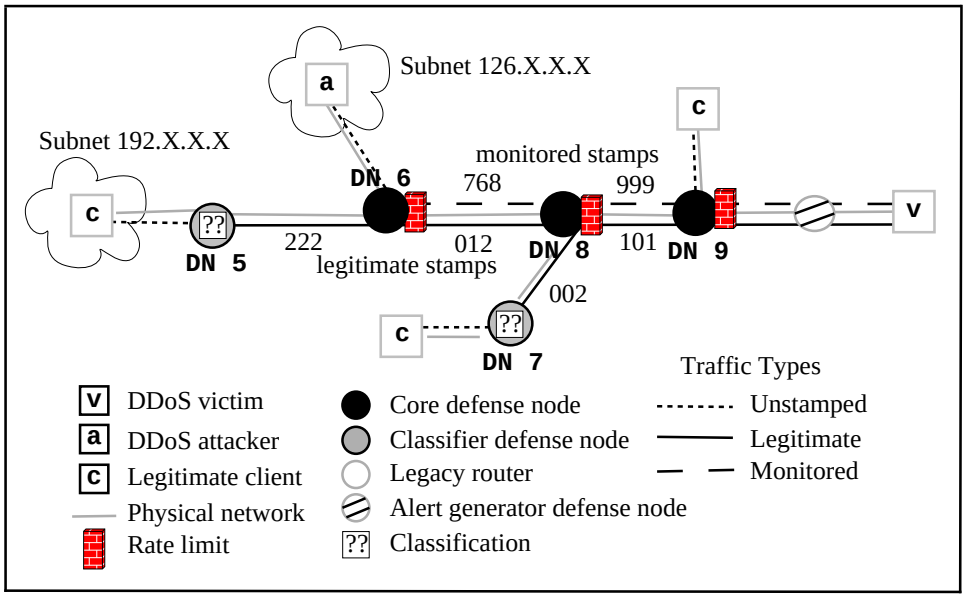

Figure 10.34: Packet stamping and flow classification.

it is restamped with stamp 012. Upon reaching defense node 8, it is aggregated with another incoming legitimate traffic flow (bearing stamp 002), and both flows leave this node 8 bearing stamp 101 and continue toward the victim. In contrast, attack traffic from the 126 subnet is rate-limited by defense node 6, and becomes a monitored flow leaving this node with stamp 768. This monitored flow is restamped at node 8 with stamp 999. Further aggregation of monitored flows is shown in defense node 9, where fresh unstamped traffic from a legitimate client enters, is rate-limited, and aggregated with the 999 flow. This example shows how two stamps must be maintained between any two defense nodes. At this snapshot in time, for defense node 8 and 9, the 999 stamp defines monitored traffic, and the 101 stamp defines legitimate traffic.

10.2.2 Overview of an Integrated DefCOM and D-WARD System

The D-WARD 3.1 system (D-WARD with sequence number prediction) is integrated with DefCOM as a classifier node. It receives attack alerts and rate limit requests from the DefCOM system, and regards them as authoritative. DefCOM attack alerts are handled in the same manner as autonomous attack detections. If a D-WARD system has information on the flow associated with the target from an attack alert message, it will install a rate limit on this flow. The imposed rate limit will be the rate limit requested in the DefCOM message. If for some reason DefCOM messages do not arrive in time, D-WARD falls back to autonomous operation.

D-WARD benefits from integration with DefCOM by improving its detection mechanism with more accurate victim-end DefCOM detection. DefCOM also benefits from integration effort, as D-WARD provides excellent classifier functionality. It successfully separates legitimate from attack traffic, thus supporting DefCOM's goal of guaranteed good service to legitimate clients.

10.2.3 Experiments

DefCOM experiments are currently ongoing. Their results will be available soon at the Web site: `http://lasr.cs.ucla.edu/defcom`. The preliminary results indicate that legitimate service level in the integrated system is much higher than the service level when D-WARD is not deployed. However, no performance graphs are currently available.

10.3 DDOS-DATA

The Johns Hopkins University Applied Physics Laboratory Distributed Denial of Service Defense/Attack Tradeoff Analysis (DDOS-DATA) project, under the sponsorship of DARPA's FTN program, is currently analyzing DDOS attacks and mitigation technologies. DDOS-DATA's goal is to use analysis to quantify how well mitigation technologies work, how attackers can adapt to defeat mitigation technologies, and how different mitigation technologies can be combined. While a variety of analysis approaches exist, DDOS-DATA is using modeling and simulation. Modeling and simulation provide an approach with several advantages over closed form and real world testbed analysis: the ability to vary key parameters that may not be easily modifiable in a testbed, the ability to easily repeat a given analysis scenario, and the use of models without debilitating simplifications.

Using modeling and simulation, the DDOS-DATA team has analyzed five mitigation technologies (D-WARD, NetBouncer, Active Monitor, Rate Limiting, and Proof-of-Work) to develop quantitative evidence regarding the mitigation technologies operation both individually and when deployed in combinations. The analysis results have provided insights into properties of mitigation technologies that make them robust in the face of an adapting attacker and conditions that must be satisfied for mitigation technologies to work together.

DDOS-DATA analysis begins with developing requirements and designs for the necessary models. The requirements reflect the analysis goals while the design is based on all available information for the technology in question. Once the models have been developed, verification and validation begins. Verification ensures that the model correctly implements the developer's intent, while validation compares the model to the real system ensuring correct model behavior.

Once verified and validated, the mitigation technologies are deployed in a 500+ node target network based on a subset of the JHU/APL Intranet. DDOS-DATA metrics examine the performance of each mitigation technology in several attack scenarios.

DDOS-DATA experiments with D-WARD are currently ongoing. Their results will be available soon at the Web site: `http://lasr.cs.ucla.edu/d-ward`.

CHAPTER 11

Overview of Related Work

There are numerous approaches to DDoS defense and network security in general. Chapter 3 gave a thorough analysis of the DDoS defense problem and an overview of the most commonly used approaches, and Section 4.2 provided a detailed taxonomy of DDoS defenses. In almost all cases, proposed DDoS defense systems are compatible with D-WARD, and often the use of D-WARD with these other systems could improve D-WARD or offer synergistic protection from DDoS attacks. In the following sections we provide detailed descriptions and comparisons with related DDoS defense approaches.

11.1 Protocol Security Mechanisms

Protocol security mechanisms address the problem of a bad protocol design by changing and securing current protocols to eliminate the possibility of vulnerability attacks.

TCP SYN cookies [SKK97] address the half-open connection problem in the TCP protocol that is misused for TCP SYN flooding attacks. They modify the TCP protocol at the protected target, replacing the server state (usually instantiated after SYN packet receipt) with cryptographic information encoded in SYN ACK packets. Once the final ACK is received and the connection has been established, the server recreates the state from the ACK packet. This delayed

commitment of resources successfully handles the TCP SYN attack and requires no changes to the client's TCP mechanism.

Delayed commitment of resources has also been explored in [LNA00, Mea99, ANL01]. Those papers propose protocol designs in which a server allocates resources for communication with the client only after sufficient authentication is done [LNA00, Mea99], or the client has paid a sufficient price [ANL01]. Designing a protocol along these guidelines will not completely remove the possibility of DDoS attacks. The attacker can still acquire a large number of agent machines and form an overwhelming number of legitimate connections, or he can generate high-volume traffic that consumes a target's network resources. However, these guidelines prevent small-rate attacks that use spoofed addresses, thus forcing the attacker to resort to cruder and easily detectable attack strategies. These approaches all have the disadvantage that they require changes in existing client software.

Protocol and application scrubbing [MWJ00] (typically applied at the entry point to a victim network) has been proposed to remove ambiguities from transport and application protocols. Scrubbing detects malformed or ambiguous packets and modifies them to ensure proper operation. It can eliminate many vulnerability attacks that misuse protocol ambiguities to bypass intrusion detection systems. A protocol scrubber could be installed at the exit point of the source network and thus prevent vulnerability-based attacks originating from this network. This would nicely complement D-WARD's defense, thus enabling it to detect and handle vulnerability-based DDoS attacks.

Several filtering mechanisms have been proposed to prevent spoofing of source addresses in IP packets ([FS00, LMW02, PL01]). While IP spoofing is not necessary for DDoS attacks, it helps the attackers hide the identity of attacking

machines so they can reuse them for future attacks. D-WARD and many other DDoS prevention mechanisms would benefit from more reliable packet source addresses. Elimination of spoofing would reduce DDoS defense to a resource-sharing problem where many known solutions can readily be deployed.

11.2 Victim-End Defense

Most systems for combating DDoS attacks work on the victim side. As discussed before, victim-end defense cannot provide complete protection from DDoS attacks because the defense system may itself be overwhelmed by the attack traffic. Another possible approach would be to deploy a victim-end defense system at the source-end, reversing its functions so that it examines and polices the outgoing traffic. While attack traffic does not create the same set of anomalies at the source-end as it does at the victim-end, it is likely that the sensitivity of most victim-end defenses will be inappropriate for source-end defense. However, some victim-end defenses can successfully be combined with D-WARD to achieve a synergistic effect.

Intrusion detection systems such as NetRanger [Cisa], NID [Com], SecureNet PRO [Mim], RealSecure [Int], and NFR-NID [NFR] use signature and anomaly detection to detect all sorts of security breaches that may indicate intrusion incidence. The EMERALD IDS [NP99] is particularly relevant, since it analyzes TCP/IP packet streams to detect intrusions through anomalous behavior. Those IDS systems could be used to detect anomalies and attack signatures in outgoing traffic. While outgoing traffic at the source-end is of small volume and may not exhibit the same level of anomalies as traffic near the victim, coupling IDS systems with D-WARD would surely improve detection accuracy. As most IDS systems do not take automated action to stop the attack, they would benefit,

through a combining with D-WARD, by receiving an automated, selective and dynamic response to the attack.

In [LXS00] Liu et al. propose to detect DDoS attacks at a router located close to the victim. The router monitors its buffer queue size and starts throttling incoming traffic once the queue size grows over a specified threshold. When the queue size is reduced, throttling is stopped. This approach is similar to the RED congestion control mechanism [FJ93]. Its application at the source network is not likely to be very effective since the attack is usually sufficiently distributed so that it does not create congestion. On the other hand, D-WARD detection occurs promptly, even for small-rate attacks that do not create source-end congestion. Further, the approach proposed in [LXS00] is not selective, since it probabilistically drops any of the packets in the queue. While the high amount of attack traffic ensures that attack packets will be dropped with a higher probability than legitimate ones, collateral damage is still possible. As D-WARD invests great effort in distinguishing between legitimate and attack traffic, its response is more selective and thus promises smaller collateral damage.

Netbouncer [OB] establishes a list of legitimate clients (source IP addresses) prior to the attack. The legitimacy of the client is established through "legitimacy tests." These tests send a request packet to the client and wait for the appropriate response to include the client into the list of legitimate clients. The sent requests can be ICMP_ECHO packets testing to see if the IP address belongs to a real machine, client puzzles or more complicated requests (such as Turing tests) that test whether a live user is generating the traffic. Under attack, Netbouncer forwards only those packets that come with source addresses from the legitimate client list, and drops everything else. This approach is likely to accurately detect legitimate clients, but it can be easily tricked into forwarding the attack packets,

if the attacker can fake addresses from the legitimate client list.

Resource accounting mechanisms police the access of each user to resources based on the privileges of the user and his behavior. The user in this case might be a process, a person, an IP address, or a set of IP addresses having something in common. Resource accounting mechanisms guarantee fair service to legitimate well-behaved users. In order to avoid user identity theft, they are usually coupled with legitimacy-based access mechanisms that verify the user's identity, such as client puzzles [JB99]. Approaches proposed in [JB99, ZL97, SP99, GR02, LRS00] illustrate resource accounting mechanisms. As DDoS is in fact a resource overloading problem, resource accounting approaches are well suited to address it. One drawback of these mechanisms is that clients must be aware of the defense and install special software, enabling them to reply to legitimacy tests. Another drawback is that resource accounting requires keeping state per user, thus storage requirements grow with the number of legitimate users. This indicates that the system must be installed at the end network and not in the core. As discussed before, victim-end defense cannot handle high-volume attacks that overwhelm the defense system. On the other hand, a source-end defense that deploys resource accounting would be beneficial to D-WARD. As D-WARD establishes connection legitimacy regardless of the spoofed addresses, no other legitimacy test would be required. A resource accounting approach could then divide resources fairly among legitimate connections, preventing subtle attacks that do not deploy spoofing and that create legitimate-like connections.

Several DDoS defense systems [Maz] and [Arb] perform anomaly detection (usually at the victim network) by observing numerous traffic parameters and defining a range of allowed values based on the analysis of packet trace data. The attack response is to impose a non-selective fixed rate limit to offending streams,

thus likely damaging legitimate traffic. Instead of fixed legitimate traffic models that D-WARD uses, [Maz] and [Arb] train their models based on the observed traffic in the networks. Once underlying traffic patterns change, models need to be retrained to avoid false positives. Like all learning approaches, these detection models can be mistrained by the attacker to regard attack traffic as legitimate. On the other hand, models tailored especially to traffic characteristics of deploying network are likely to yield better detection accuracy than those deployed by D-WARD. Thus, integration of a defense system that deploys trained models with D-WARD would likely enhance detection accuracy. D-WARD's selective response guarantees good service to legitimate traffic and is likely to inflict less collateral damage than the fixed rate-limiting deployed by [Maz] and [Arb].

Resource multiplication mechanisms provide an abundance of resources to counter DDoS threats. The straightforward example is a system that deploys a pool of servers with a load balancer and installs high bandwidth links between itself and upstream routers. The other approach is to acquire resources dynamically once the attack has been detected [YEA00]. These approaches essentially raise the bar on how many machines must participate in an attack to be effective. While not providing perfect protection, for those who can afford the costs resource multiplication has often proved sufficient. For example, Microsoft has used it to weather large DDoS attacks. Another approach is the use of Akamai services for distributed Web site hosting. User requests for a Web page hosted in such a manner are redirected to an Akamai name server, which then distributes the load among multiple, geographically distributed Web servers hosting replicas of the requested page.

11.3 Source-End Defense

We are aware of two source-end defense systems - MULTOPS [GP01] and Reverse Firewall [Cs3].

MULTOPS [GP01] is a heuristic and a data-structure that network devices can use to detect DDoS attacks. Each network device maintains a multi-level tree, monitoring certain traffic characteristics and storing data in nodes corresponding to subnet prefixes at different aggregation levels. The tree expands and contracts within a fixed memory budget. The attack is detected by abnormal packet ratio values, and offending flows are rate-limited. The system is designed so that it can operate as either a source-end or victim-end DDoS defense system. While the high-level design of this system has much in common with D-WARD, the details are different. MULTOPS uses only the aggregate packet ratio to model normal flows. Non-TCP flows in a system using MULTOPS can either be misclassified as attack flows, or recognized as special and rate-limited to a fixed value. In the first approach, harm is done to a legitimate flow, while in the second approach, a sufficiently distributed attack can still make use of the allowed rate to achieve the effect. On the other hand, D-WARD adds legitimate UDP and ICMP flow models, thus detecting a broader range of attacks. MULTOPS imposes a fixed, non-selective rate limit on outgoing or incoming traffic (based on its deployment at source- or victim-end), thus likely inflicting collateral damage on legitimate traffic. D-WARD offers much more selective response with almost no legitimate traffic drops.

Reverse Firewall [Cs3] prevents DDoS attacks by limiting the rate of "unexpected" TCP packets at a network's exit router. The specifics of expected packets are derived from the foreign-peer acknowledgements to previously sent traffic. Only the outgoing packets matching the expected range will be forwarded.

This technique is similar to sequence number prediction but it requires storing TCP data for each legitimate connection. While it is likely to be more accurate in distinguishing legitimate from attack traffic, this improvement can only be marginal compared to D-WARD's already excellent traffic separation. The improvement is gained at large storage and processing costs that may make the system unable to handle a large traffic volume. Reverse Firewall deploys a fixed rate limit on "unexpected" packets and on non-TCP traffic, and thus cannot adapt to dynamic changes in network conditions. Rate limits are engaged at all times, which leads to a difficult trade-off for setting the limit. Small values lead to poor resource utilization, as they cannot accommodate traffic bursts. Large values, on the other hand, do not control highly distributed attacks, as their traffic passes below the rate limit. D-WARD successfully addresses this problem by dynamically adjusting rate limits based on current traffic observations.

11.4 Distributed Defense

Distributed defense systems locate their nodes either at edge (victim and source) networks, or at the edge and in the core.

Distributed defense at the Internet core has a definite advantage over single-point defense. Due to the highly interconnected core topology, a few strategically deployed defense systems can monitor and control a large portion of traffic circulating in the Internet. For instance, in [PL01] Park et al. have evaluated the effectiveness of spoofed address filters deployed at the core. Their results indicate that with deployment at a chosen 18% of core administrative systems, almost all spoofed traffic can be detected and dropped. A similar claim would stand for DDoS defense — it is likely that distributed defense deployed at a small percentage of core routers would effectively stop a large portion of DDoS attacks.

Distributed defense at the edge networks has the advantage of easy attack detection. Because the victim-end defense system raises the attack alarm, the detection is very reliable. The response is then engaged at source networks that participate in the attack. A distributed DDoS defense that involves only edge networks is ineffective against attacks from legacy networks (i.e., those networks that do not deploy source-end defense). The attack traffic generated by agent machines in legacy networks reaches the victim unimpeded.

Distributed systems usually attempt to engage defense actions as close to the sources as possible. This minimizes collateral damage since a response only affects those traffic mixes that contain the attack. Combining the D-WARD system with a distributed DDoS defense can greatly enhance performance of both parties. D-WARD can improve its detection by receiving victim-end attack alert and advisory on desired actions, while distributed defense can improve its selectiveness by deploying D-WARD's source-end response. This effect was demonstrated, both in the integration of the D-WARD system with COSSACK [Infa] and with DefCOM [MRR03].

In [MBF02] Floyd et al. have proposed Aggregate Congestion Control. ACC augments routers with the ability to detect and control flows that create congestion (frequently a sign of DDoS attacks). Flows are detected by monitoring the dropped packets in the router queue and identifying high-bandwidth aggregates that are responsible for the majority of drops. The rate limit is then imposed on the aggregate. Pushback can be used to install rate limits at upstream routers if the congested router cannot control the aggregate itself. This approach faces challenges related to augmentation of large numbers of routers, handling legacy routers, and cooperation among different administrative domains. Further, since the rate limit is imposed at the aggregate traffic close to the victim, legitimate

flows suffer collateral damage [IB02]. Combining ACC with D-WARD would address this problem, as D-WARD offers a more selective response that guarantees low collateral damage.

Secure Overlay Services (SOS) [KMR02] prevent denial-of-service attacks on critical servers by routing requests from previously authenticated clients to those servers via an overlay network. All other requests are filtered by the overlay. SOS is a distributed system that offers excellent protection to the specified target at the cost of modifying the clients's machines to be aware of the overlay and use it to access the target. This prevents the use of SOS for protection of public servers. Additionally, large numbers of overlay nodes are required to make the system resilient to DoS attacks. Also, SOS offers no protection from insider attacks. Namely, an attacker could trick the authentication mechanism (or subvert a legitimate client's machine) and direct its attacks to the victim via the overlay.

DefCOM [MRR03] builds an overlay of defense nodes along the traffic path and uses it to suppress attack traffic, thus providing resources for legitimate traffic. More details about DefCOM are given in Section 10.2. D-WARD has been successfully integrated with DefCOM and has demonstrated excellent performance.

COSSACK [Infa] forms a multicast group of defense nodes which are deployed at source and victim networks. Each defense node can autonomously detect an attack and issue a signal to the group, inviting defense nodes from participating source networks to deploy defense measures. There are several deficiencies in the COSSACK system. Its use of multicast communication suggests poor scalability of the communication system and offers many points of failure. D-WARD has been successfully integrated with COSSACK, and results of this integration are presented in Section 10.1. Both systems benefit from the integration and provide

better defense than in the autonomous operation.

The Active Security System (ASSYST) [CCP01] consists of defense nodes deployed at edge networks. Once the attack is detected, the nodes discover their peers using probe messages, and organize themselves into a peer-to-peer network. The blocking requests are then propagated to upstream peers and a response is installed as close to the sources as possible. ASSYST does not provide separation of legitimate from attack flows, and is thus likely to inflict collateral damage. Integration of ASSYST with D-WARD would have the same synergistic benefits as the D-WARD/COSSACK integration. ASSYST would gain a selective and dynamic response mechanism, while D-WARD would improve its attack detection by coupling it with a victim-end attack alert.

IDIP [SDS01] is an application-level protocol that coordinates detection and responses of multiple intrusion detection systems. IDIP nodes are organized into neighborhoods and communities. Coordinated detection, attack tracing and response within a community are managed by a component called a Discovery Coordinator. IDIP assumes contiguous deployment of neighborhoods that share information, and thus its ability to suppress attacks is limited in a partial deployment scenario.

MANAnet [Cs3] forms cooperative neighborhoods of defense nodes around the victim. These nodes stamp packets that they forward, encoding the path to the victim in the packet header. The router at the victim then offers a fair share of resources to each encoded path. Source networks can further deploy the Reverse Firewall to prevent outgoing attacks. MANAnet requires contiguous deployment of defense nodes near the victim and changes to the IP protocol to facilitate packet stamping—both of these features are likely to hinder wide deployment.

Several systems combat DDoS attacks by using a traceback mechanism to

locate attacking nodes ([SWK00, BLT01, DFS01, SP01, SPS01]. These systems provide information about the identity of attacking machines, but do not themselves stop DDoS attacks. The complexity of a traceback mechanism is large if the attack is distributed, and the mechanism itself may be susceptible to attacks. Both of these issues must be resolved before traceback can be widely deployed on the Internet and used for DDoS defense. D-WARD could benefit from integration with a traceback system, as it could receive reliable information that the attack has been detected and that the source network is participating in it. Traceback systems would also benefit from this integration as it would enhance the identification process with automated and selective response.

The approach proposed in [PLR02] combines the Pushback [IB02] and traceback [SWK00] approaches to reduce collateral damage to legitimate traffic due to non-selective pushback action. All routers involved in defense perform probabilistic packet marking (PPM), as described in [SWK00]. The victim builds a "normal profile" on each of the upstream routers, using marked packets. The victim also collects current statistics on upstream routers, building a temporary profile. The attack is detected when the victim notices a large discrepancy between the normal and temporary profiles. The victim then sends Pushback messages only to those routers whose profiles disagree with the model, installing responses as close to the sources as possible. This minimizes collateral damage since those routers that relay only legitimate traffic are not likely to create anomalous profiles, and thus will not install responses and drop legitimate packets. Collateral damage will still be inflicted at those routers near the source that install the response. Coupling D-WARD with this approach would clearly address this problem and thus improve defense performance.

Dynabone [Infb] is a system for the rapid configuration, deployment, and

management of protective layered overlays that reconfigure the network on attack alert, to cut off the attack traffic. DynaBone interconnects numerous parallel networks (innerlays) into a single "outer" overlay (outerlay) that presents an interface compatible with COTS applications and operating systems. The result is a parallel set of innerlays, any subset of which can be disconnected in response to an attack while the outerlay continues to provide effective service over the remaining innerlays. Additionally, Dynabone provides a variety of network management and security algorithms in innerlays, all of which must be simultaneously attacked to deny services. Dynabone requires formation of redundant, disjoint innerlays to provide desired re-routing service. While re-routing will successfully address flooding attacks by removing high-volume traffic from a bottleneck link, it is still likely to inflict collateral damage to legitimate traffic sharing the path. Coupling Dynabone with D-WARD would provide selection function to Dynabone, thus enabling it to separate legitimate from attack traffic and apply the re-routing response only to the attack traffic.

CHAPTER 12

Beyond D-WARD

The previous chapters have described the design and implementation of D-WARD, a source-end system for DDoS defense. Numerous performance results show that D-WARD detects a wide range of attacks and successfully controls those attacks, providing good service to legitimate clients. However, distributed denial-of-service is a complex problem. While D-WARD demonstrates a successful approach to DDoS defense, it is by no means a "silver bullet" for DDoS attacks. In this chapter we reexamine the DDoS problem and propose a fundamental solution needed to handle this threat. D-WARD represents a crucial piece of this solution.

12.1 DDoS Revisited

On the surface, DDoS seems to be a very simple problem — a large number of machines are generating a lot of traffic that overwhelms some critical resource at the victim. DDoS attacks are successful because of brute force, not sophistication. They work because attackers are "stronger" (have more resources) than the victim. Two obvious solutions come to mind:

1. Equip the victim with more resources. However, the attacker can recruit hundreds and thousands of machines. If the victim purchases sufficient resources to weather the largest possible attacks, it will simply waste money

because these resources will be constantly under-utilized.

2. Perform resource management with fair-sharing among users. As the defense system on the victim's network link may easily be overloaded with the attack, the resource management unit should be placed at least one hop upstream. The goal of resource management would be to divide resources equally among users, so those users that generate a lot of requests cannot steal resources from users that generate a few. This approach fails for two reasons. First, attackers use IP spoofing, so instead of seeing many requests from a few users, the unit will see a few requests from numerous users. Second, even if IP spoofing were countered, the attacker could recruit a sufficient number of machines to obtain almost all of the critical resources for himself, even in face of resource management. For example, let us observe a Web server WS that usually receives 10 requests per second, and its resources can support up to 1000 requests per second. At the time of the attack, this server has three legitimate users that each generate a request per second. The attacker subverts 10000 machines and instructs them to generate 1 request per second directed at the Web server WS. If the server is protected by a resource manager unit, it will detect 10003 legitimate users per second. As it can only accommodate 1000, it will choose 1000 among 10003 requests and forward them to the server. The attacker obviously stands $\binom{10000}{1000}/\binom{10003}{1000} = 73\%$ chance of getting all of the server's resources.

The other straightforward solution is to prevent the attacker from gaining numerous agent machines, thus limiting his power. To achieve this, we must secure (and keep up to date) the majority of machines in the Internet. While this is a worthy goal (insecure machines are exposed to many malicious actions, such

as worm and virus poisoning, that can do significant damage) it is unattainable. There are simply too many machines owned by too many users all over the world, and no single policy can be enforced among them.

Obviously, DDoS proves to be tougher than it may seem at first glance. This is because it misuses the very principles upon which the Internet was built — best-effort service and the end-to-end paradigm, as discussed in Chapter 2. The intermediate network blindly forwards requests to their destination, regardless of the destination's ability to handle them. In the case of DDoS attack (or a flash crowd) this leads to overload. The solution to DDoS must be as fundamental as the misused vulnerabilities to successfully address the threat.

12.2 DDoS Nemesis

Obviously, many aspects of DDoS can more successfully be addressed near the source than near the victim or in the middle of the Internet:

- Any response deployed near the source has less collateral damage than if deployed further downstream. This is because pushing responses downstream limits the amount of legitimate traffic affected by the response. For example, let us compare two of the same defense systems, with different deployment scenarios. System A is deployed at the victim. System B is deployed as two instances located at the only two ingress points at the victim network. Both systems receive the same attack alert and attack characterization. Defense system A can police (and thus possibly drop) all legitimate traffic that a victim receives, and will be engaged at each attack alert. Defense system B, on the other hand, can choose to activate either one or both of its instances. It will choose to activate only one instance if some deci-

sion function indicates that the majority of attack traffic is traversing this instance; otherwise both instances will be activated. Even if the decision function is imprecise or can be tricked by the attacker, there will be cases in which only one response will be engaged. In these cases only the portion of legitimate traffic coming over the ingress point with the active response runs the risk of being dropped, while the rest of legitimate traffic will be delivered to the victim.

- Source address spoofing could be successfully countered near the source. For example, if ingress filtering were deployed everywhere, the source addresses could be trusted and resource management could be performed based on the source network prefix yielding much less collateral damage. Even various traceback mechanisms would perform better near the source, as they could take advantage of known network topology.

- Selectiveness of the response is better near the source. This point was demonstrated by the D-WARD system. The traffic volume is low at the Internet edges and high in the core. As traffic differentiation requires significant amounts of processing and storage, the only two places to perform this are near the victim or near the source. The differentiation at the victim runs the risk of being overwhelmed by the attack traffic. Thus the only available deployment point is near the source.

However, source-end defense has two major drawbacks that hinder its ability to completely address the DDoS problem: (1) source-end detection may be inaccurate, and (2) attacks from unprotected networks are still possible. The first problem can be addressed by combining victim- and source-end defense. Victim-end defense can perform detection while source-end defense can respond to the attack. The second problem can be addressed by deploying defense nodes in

the core. Those nodes cannot achieve desired selectiveness of the response, but they can successfully control traffic from unprotected networks. In the meantime, traffic from protected networks can be assumed to be legitimate and can be forwarded freely to the victim. This leads us to the approach that we believe is a fundamental answer to the DDoS threat — a distributed defense system incorporating defense nodes at the victim- and source-end and in the core. This approach is taken by DefCOM, a logical extension of our work on D-WARD.

We can also arrive at the same solution from the other direction. DDoS is, in essence, a resource overloading problem. The fundamental solution to the problem is to assign each user his fair share of the resources. In the case where there are too many users so that this strategy yields too small a share of resources, we need to make a more sophisticated choice. One approach, to choose users randomly, fails because there may be many more malicious than legitimate users and a random strategy would reduce legitimate users' chances of getting the resource. The only other option is to somehow differentiate among users and perform fair resource sharing only among those users that "pass the test." As we have discussed before, this separation of legitimate from attack traffic can be done near the source. The legitimate packets can then be grouped by the "user" (a network, IP address, application, etc.), and fair sharing can be enforced among groups. To accommodate legacy networks, we must assign some share of resources to them. The fair-sharing enforcement can be performed either in the core or close to the victim. The benefit of performing it in the core is that the policing load is shared among multiple defense points and not concentrated on one point near the victim. Finally, as defense is expensive and is not perfect (i.e., occasionally a legitimate packet may be dropped), it should be engaged only during the attack. The attack signal should come from the victim, as the victim is best informed as to when its resources are being overloaded. Thus we arrive at

DefCOM from the other direction: attack alert from the victim-end defense, non-selective response at the core, and selective response near the source. DefCOM core nodes do not currently perform fair sharing of resources among legitimate users, but this feature could be easily added.

12.3 D-WARD's Place in the Solution

The goal of DDoS defense is not only to stop the attack but to detect legitimate traffic that is mixed with the attack stream, sieve it and deliver it safely to the victim. D-WARD is crucial for attaining this goal. It provides a highly selective response that enables legitimate clients of the victim to freely communicate with it in spite of the attack. This makes the occurrence of the attack almost transparent to legitimate clients, thus cancelling the DoS effect.

Without D-WARD, the distributed defense system we have proposed as a fundamental solution would, at best, divide the collateral damage equally among the source networks involved in the attack. However, this is a poor deployment motivation. Who would deploy a defense system that will likely drop its legitimate traffic during the attack, when this traffic would stand a fair chance of reaching the victim without the defense? Even if defense nodes are deployed by the core routers they would produce numerous dissatisfied clients once the responses were activated. On the other hand, with D-WARD such a system offers a "carpool lane" to the victim. Those networks that deploy D-WARD can reach the victim undisturbed even while the attack is on-going. Furthermore, if D-WARD's response is engaged and produces packet drops, this indicates that some machines within deploying networks are hosting malicious code. This can be a worthy signal to network administrators to examine their machines and detect any that are subverted. While there are many issues still to be answered in the DefCOM

project, node and communication security being the pressing one, we believe that this approach promises excellent performance and has a good economic model. Both of these features should facilitate quick and wide deployment, necessary for a successful DDoS defense.

12.4 Open Issues

While D-WARD demonstrates excellent performance as is, there is a lot of space for improvements:

1. UDP legitimate connection models could be improved. Currently they only contain fairly simple DNS, NTP and streaming media models. Adding more UDP application models and introducing more complexity into the current ones would definitely improve D-WARD's performance with UDP traffic, and make it resilient to stealthy attacks.

2. A value prediction technique could be added for UDP connections to assure that first connection packets are not dropped during the attack, similar to sequence number prediction techniques for legitimate TCP connections. This technique would significantly improve service offered to short UDP connections such as DNS and NTP that do not benefit much from legitimate connection models. The question then remains: which parameter should be chosen for this value prediction technique?

3. A better hashing technique would reduce packet handling overhead, thus enabling D-WARD to handle higher packet rates.

4. A reduction in state storage requirements could be made to enable D-WARD to handle very high traffic volumes, consisting of numerous flows

and connections. Such a modified version could be installed further downstream, possibly at core routers or at the victim. While some selectiveness would likely have to be sacrificed for speed and throughput, this unit would significantly improve the performance of current defense systems.

5. A better UDP flow model could be designed that would facilitate detection of non-spoofed attacks. For instance, D-WARD could select a set of parameters and monitor this set under normal operation, building a legitimate flow model. D-WARD could then monitor these parameters and signal an attack once a large discrepancy is detected between the model and the observed values. This approach is successfully deployed in many defense systems. Although it has severe drawbacks (as discussed in Section 5.1), such as the mistraining of models and proper model update, if carefully designed this approach can improve system performance.

6. D-WARD could be equipped with more sophisticated flow models, enabling it to detect attacks that do not create anomalies at the transport layer, such as attacks that request expensive cryptographic operations or that overload a database engine with bogus requests.

The remaining open issues do not pertain exclusively to D-WARD but to DDoS defense in general:

1. **Defense node deployment.** DDoS is a distributed problem and must be addressed in a distributed manner. While some level of defense can be achieved by deploying the defense nodes only at a single point or at a limited set of points in the Internet, the quality and effectiveness increase dramatically as the number of deployment points increases. The question then remains: how to achieve wide deployment of any defense? One ap-

proach, taken by D-WARD and DefCOM, is to design a defense system with good performance and a good economic model, and hope that this will represent sufficient incentive for wide deployment. The other option is to hope that the legal system will take interest in network security and mandate deployment of DDoS and other security solutions at all public networks. Yet another approach is to request defense deployment in policy agreements between Internet service providers or between administrative domains. Because networks that do not deploy an adequate defense represent a hazard and inflict damage to other networks, it would not be unreasonable to require them to show evidence that they have been secured before entering into a peering agreement with them.

2. **Defense update.** DDoS attacks are adversarial. They mutate and adjust themselves to avoid current defenses. If a defense mechanism disables a particular attack, attackers will try to find other ways to make effective attacks. It is impossible to predict all attack variations and design defenses that will work at all cases. Thus we must be able to update current defenses with better traffic models and response strategies in the future. We also must be able to layer new defenses and interface them with the old ones so that they work in concert.

3. **Defense interaction.** There are currently many solutions, both in the DDoS field and in network security in general. It is difficult to predict how all these solutions may interact if widely deployed. D-WARD is mostly compatible with and complements other DDoS solutions, but it may fail to work well with encrypted traffic or with traffic that is subject to anonymization such as onion routing. Further, many approaches address only one part of the DDoS problem. It is likely that, if combined, they could fill a large

portion of the DDoS defense puzzle. The open question is: how to combine the existing defenses so to achieve this synergistic effect?

CHAPTER 13

Conclusion

Distributed denial-of service is a grave problem that requires a complex solution. This thesis has presented D-WARD, a source-end defense system that successfully handles a broad range of the attacks. D-WARD achieves excellent performance with a modest set of models and per packet overhead. It provides good service to legitimate traffic during the attack, which is the ultimate goal of DDoS defense. It also detects and controls numerous DDoS attacks within seconds. In this chapter we summarize the DDoS problem, the D-WARD system's features and performance and the lessons learned in the course of this dissertation work.

13.1 Summary of the DDoS Problem

Distributed denial-of-service attacks appeared only five years ago, and they have quickly become a major problem in the Internet. This is not likely to change any time soon. The attackers are becoming more sophisticated in their attacks, using legitimate packets that a victim cannot ignore, communicating with agents via public services (such as IRC) and deploying their agents strategically to avoid detection while still performing successful attacks. In the meantime, more and more machines are connecting to the Internet via cable modems and broadband connections. These machines are frequently not secured against intrusions, and their continuous, high-bandwidth Internet connection makes them very appealing

for subversion. Further, we increasingly rely on the Internet for various services, as it offers great convenience and speed, and abounds in information. This makes many Internet sites critical resources that should be accessible at all times.

DDoS defenses are lagging far behind the attacks. While they successfully handle specific attacks (such as TCP SYN flooding, for instance), they are frequently designed to be deployed at a single spot and work in isolation from other security systems. Thus they fight a quixotic battle — trying to beat the army of attacking machines from a single point in the network. The attacker can either bypass this point or simply overwhelm it with a strong attack in order to achieve his goal. Further, there is a disproportion in the effort invested in the attack and in the defense. A small modification in the attack traffic enables an attacker to bypass or trick current defenses. Meanwhile, defenses must be significantly changed to handle new attacks. For each step the attacker takes, defenses must take ten steps to catch up. How can we change the game?

First, DDoS defense must be **distributed**. This is a natural approach since a the large army can only be defeated with an army equal in number or in power. Since it is difficult to assure wide deployment of a single defense system, defenses must be ready and able to interface with one another and to cooperate for the common good. Ideally, a defense system should be independent so that it can work in isolation and still achieve good performance. It should also be able to combine its actions with other systems to further improve performance. D-WARD meets this goal, as demonstrated in the COSSACK and DefCOM integration efforts.

Second, DDoS defense must be **selective**. To achieve the wide deployment goal, it is necessary to offer incentives to various networks to deploy the defense. If a system is selective, this incentive is naturally present in the good service

level that it promises to legitimate traffic. Otherwise, a non-selective defense system is no better than the "drop-tail" queuing policy in routers — it controls the congestion but it offers no fairness. D-WARD meets the selectiveness goal, as demonstrated in numerous experiments.

Third, DDoS defense must be **adaptive**. As new attacks appear, the defense must be easily changed to accommodate new models and policies. If a new set of attacks is better handled with a new defense system, an old system should be able to interface with the new one, enabling the customers to purchase upgrades, not complete replacements. D-WARD's models can easily be modified using its configuration parameters. Due to modular design, new models and policies could also be easily added. Lastly, D-WARD can easily be interfaced with new defense systems.

Fourth, DDoS defense must be **customizable**. The Internet has long passed the era when "one size fits all" protocols were sufficient. It is desirable for new systems and services to offer a means for policy definitions so that users can specify custom behaviors of their liking. The system must also provide a policy checker that warns the user if the desired behavior will result in inefficient or insecure operation. D-WARD provides a set of configuration parameters that can be modified by the user and that affect the system's performance and operation cost.

13.2 D-WARD Solution

D-WARD approaches the DDoS problem from a new direction. It is a source-end solution whose goal is to autonomously detect and stop outgoing attacks from the deploying network. In its design and implementation, D-WARD adheres to

several principles that are main contributors to its good performance:

1. It makes very few assumptions. The only assumption D-WARD makes is that it is aware of its *policy address set* and that it observes symmetric traffic. D-WARD does not outlaw spoofing. Instead it assumes stealthy attackers and deploys several strategies to detect sophisticated attacks.

2. It detects legitimate, not attack behavior. There are many characteristics of DDoS attacks, but most of them are consequences of choice, not of necessity. A system that attempts to detect malicious behavior becomes useless as soon as the attack changes (e.g., this effect has been noticed in virus signature detection). D-WARD thus builds a set of legitimate models and attempts to detect legitimate connections and provide good service to their packets. The rest of the traffic is then controlled and must fight for resources. Models of legitimate traffic are designed so that the attacker must sacrifice the attack's effectiveness to match them. This makes the game counterproductive to the attacker.

3. It builds models based on protocol specification rather than on observed behavior. With the exception of legitimate UDP flow models, all other models are built upon protocol specifications. This removes the need for model update as traffic changes, and guarantees that all legitimate connections and flows will obey the model.

4. It applies dynamic response. D-WARD acknowledges the possibility that its observations and actions may occasionally be wrong. The system is also aware that network conditions change and the response must be adjusted to them. D-WARD therefore frequently reevaluates its response and adjusts it promptly to observations. This allows fast recovery from false alarms,

fast relief for the victim once the attack has been detected, and speedy restoration of throughput when the attack subsides.

5. It has modular design. This enables easy modifications and integration with other defense systems.

6. It is an autonomous system. While parts of autonomous operation may be overloaded by cooperative action, if D-WARD is integrated with other defense systems, the D-WARD system achieves excellent performance in autonomous operation. Current integration efforts were realized to complement the autonomous operation, enhancing D-WARD with cooperation messages. In the absence of messages, the system falls back to autonomous mode, thus offering robustness and resiliency to message loss.

D-WARD achieves excellent performance, as has been demonstrated in numerous experiments performed by ourselves and by independent evaluators. Its simple models account for its small storage cost. Nonetheless, as its models are fairly fundamental, D-WARD is able to detect and handle even sophisticated attacks such as an HTTP request flood (see Section 10.1.3.8). While there are certain aspects of D-WARD that could be improved, these results seem very promising.

13.3 Key Contributions

D-WARD has demonstrated the main claim of this dissertation — that source-end defense can detect and control a broad range of attacks in autonomous operation and at a low cost. We believe that D-WARD has been a missing piece from the DDoS defense field. Many networks host machines that are poorly secured and so numerous that keeping them secure and up to date is infeasible. A ready example

is provided by university networks that have numerous machines that are poorly administered by students. Even though the whole network has capable administrators that take care of a few important machines, such as mail servers and routers, they cannot possibly attend to each machine in the network. Installing a D-WARD system at the egress point of those networks would serve multiple purpose: (1) D-WARD would protect the rest of the Internet from DDoS attacks that originate from the insecure machines in the network, (2) D-WARD would ensure good service to legitimate clients during the attack, (3) D-WARD response would alert network administrators to the presence of infected machines in their network, calling for detailed investigation, and (4) D-WARD makes the deploying network highly unattractive to the attackers, thus reducing the probability of intrusions.

D-WARD has achieved an excellent selectiveness of response. This feature would be beneficial to many other DDoS defense systems. Combining D-WARD with other defense systems improves the defense performance of the given system and is likely to achieve a synergistic effect, where the integrated system can handle a broader range of attacks than any of its parts.

D-WARD provides a dynamic response that is self-adjusting. By carefully choosing the criteria for adjustment, D-WARD is able to promptly react to changes in network conditions, while being resilient to an attacker's attempts to trick the system into rate limit removal. While attackers can still design stealthy attacks that will bypass the system (such as low-rate pulsing attacks with long period), D-WARD forces them to sacrifice attack effectiveness or to recruit many more agent machines and to distribute those over many source networks. This makes D-WARD deploying networks unattractive to attackers.

More than just the design of a DDoS defense system, this thesis has been

an exploration of the DDoS problem and possible solutions. This effort resulted in the creation of attack and defense taxonomies that enhance the understanding of the problem and solution space, and facilitate design of better defenses. Parallel work has been performed on the evaluation strategies for DDoS defense systems. These strategies are lacking at the present time, which hinders testing and comparisons of DDoS defense systems. As a result of this effort, a customizable attack tool *cleo* and a trace reconstruct tool *tracegen* have been created. We have also selected a set of metrics that can be used to describe performance of DDoS defense systems.

13.4 Final Comments

Distributed denial-of-service requires a distributed solution. This thesis has presented a crucial building block of this solution — D-WARD, a source-end defense system that provides a selective and dynamic response. D-WARD can offer excellent performance, even in autonomous operation. This performance is further enhanced if D-WARD is combined with other defense systems. We believe that these features will lead to quick and wide deployment of D-WARD and integration with other systems, and thus to further improvement of Internet security.

References

[ABK01] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. "Resilient Overlay Networks." In *Proceedings of 18th ACM SOSP*, October 2001.

[ANL01] T. Aura, P. Nikander, and J. Leiwo. "DOS-Resistant Authentication with Client Puzzles." *Lecture Notes in Computer Science*, **2133**, 2001.

[Arb] Arbor Networks. *The Peakflow Platform*. http://www.arbornetworks.com.

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

[Axe00] S. Axelsson. "Intrusion detection systems: A survey and taxonomy." Technical Report 99-15, Department of Computer Engineering, Chalmers University, March 2000.

[BBNa] BBN Technologies. *Applications that participate in their own defense*. http://www.bbn.com/infosec/apod.html.

[BBNb] BBN Technologies. *Intrusion tolerance by unpredictability and adaptation*. http://www.bbn.com/infosec/itua.html.

[BCC98] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. "Recommendations on Queue Management and Congestion Avoidance in the Internet." *RFC 2309*, April 1998.

[Bel96] S. Bellovin. "Defending Against Sequence Number Attacks." *RFC 1948*, May 1996.

[BLT01] S. Bellovin, M. Leech, and T. Taylor. "ICMP Traceback Messages." *Internet draft, work in progress*, October 2001.

[CAI] CAIDA. *Characterization of Internet traffic loads, segregated by application*. http://www.caida.org/analysis/workload/byapplication/.

[CCP01] R. Canonico, D. Cotroneo, L. Peluso, S. P. Romano, and G. Ventre. "Programming routers to improve network security." In *Proceedings*

of the OPENSIG 2001 Workshop Next Generation Network Programming, September 2001.

[CERa] CERT Coordination Center. *CERT Advisory CA-2001-19 "Code Red" Worm Exploiting Buffer Overflow In IIS Indexing Service DLL.* http://www.cert.org/advisories/CA-2001-19.html.

[CERb] CERT Coordination Center. *CERT Advisory CA-2001-26 "Nimda Worm".* http://www.cert.org/advisories/CA-2001-26.html.

[CERc] CERT Coordination Center. *Code Red II.* http://www.cert.org/incident_notes/IN-2001-09.html.

[CERd] CERT Coordination Center. *Denial of Service Tools.* http://www.cert.org/advisories/CA-1999-17.html.

[CERe] CERT Coordination Center. *DoS using nameservers.* http://www.cert.org/incident_notes/IN-2000-04.html.

[CERf] CERT Coordination Center. *erkms and li0n worms.* http://www.cert.org/incident_notes/IN-2001-03.html.

[CERg] CERT Coordination Center. *IP Denial-of-Service Attacks.* http://www.cert.org/advisories/CA-1997-28.html.

[CERh] CERT Coordination Center. *Nimda worm.* http://www.cert.org/advisories/CA-2001-26.html.

[CERi] CERT Coordination Center. *Ramen worm.* http://www.cert.org/incident_notes/IN-2001-01.html.

[CERj] CERT Coordination Center. *Smurf attack.* http://www.cert.org/advisories/CA-1998-01.html.

[CERk] CERT Coordination Center. *TCP SYN flooding and IP spoofing attacks.* http://www.cert.org/advisories/CA-1996-21.html.

[CERl] CERT Coordination Center. *Vulnerability in Certain TCP/IP Implementations.* http://www.cert.org/advisories/CA-1998-13.html.

[CER01] CERT Coordination Center. *Trends in Denial of Service Attack Technology*, October 2001. http://www.cert.org/archive/pdf/DoS_trends.pdf.

[Cisa] Cisco. *NetRanger Overview.* http://www.cisco.com/univercd/cc/td/doc/product/iaabu/csids/csids1/csidsug/overview.htm.

[Cisb] Cisco. "Strategies to Protect Against Distributed Denial of Service Attacks." http://www.cisco.com/warp/public/707/newsflash.html.

[Com] Computer Incident Advisory Capability. *Network Intrusion Detector Overview*. http://ciac.llnl.gov/cstc/nid/intro.html.

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

[DFS01] D. Dean, M. Franklin, and A. Stubblefield. "An Algebraic Approach to IP Traceback." In *Proceedings of the 2001 Network and Distributed System Security Symposium*, February 2001.

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

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

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

[DO] T. Darmohray and R. Oliver. *Hot spares for DDoS attacks*. http://www.usenix.org/publications/login/2000-7/apropos.html.

[DWD] D. Dittrich, G. Weaver, S. Dietrich, and N. Long. *The mstream distributed denial of service attack tool*. http://staff.washington.edu/dittrich/misc/mstream.analysis.txt.

[FJ93] S. Floyd and V. Jacobson. "Random Early Detection Gateways for Congestion Avoidance." *IEEE/ACM Transactions on Networking*, August 1993.

[Flo00] S. Floyd. "Congestion Control Principles." *RFC 2914*, September 2000.

[FS00] P. Ferguson and D. Senie. "Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing." *RFC 2827*, May 2000.

[GP01] T. M. Gil and M. Poletto. "MULTOPS: a data-structure for bandwidth attack detection." In *Proceedings of 10th Usenix Security Symposium*, August 2001.

[GR02] A. Garg and A. L. N. Reddy. "Mitigation of DoS attacks through QoS Regulation." In *Proceedings of IWQOS workshop*, May 2002.

[HM91] K. Hafner and J. Markoff. *Cyberpunk: Outlaws and Hackers on the Computer Frontier*. Simon & Schuster, 1991.

[IB02] J. Ioannidis and S. M. Bellovin. "Pushback: Router-Based Defense Against DDoS Attacks." In *Proceedings of NDSS*, February 2002.

[Infa] Information Sciences Institute. *COSSACK Homepage*. http://www.isi.edu/cossack.

[Infb] Information Sciences Institute. *Dynabone*. http://www.isi.edu/dynabone/.

[Infc] Information Sciences Institute. *YOID Homepage*. http://www.isi.edu/div7/yoid.

[Ins81] Information Sciences Institute. "Transmission Control Protocol." *RFC 793*, September 1981.

[Int] Internet Security Systems. *Intrusion Detection Security Products*. http://www.iss.net/securing_e-business/security_products/intrusion_detection/index.php.

[Jac88] V. Jacobson. "Congestion Avoidance and Control." *ACM Computer Communication Review; Proceedings of the Sigcomm '88 Symposium in Stanford, CA, August, 1988*, **18, 4**:314–329, 1988.

[JB99] A. Juels and J. Brainard. "Client puzzles: A cryptographic countermeasure against connection depletion attacks." In *Proceedings of the 1999 Networks and distributed system security symposium*, March 1999.

[KMR02] A. D. Keromytis, V. Misra, and D. Rubenstein. "SOS: Secure Overlay Services." In *Proceedings of SIGCOMM 2002*, 2002.

[LMW02] J. Li, J. Mirkovic, M. Wang, P. Reiher, and L. Zhang. "SAVE: Source Address Validity Enforcement Protocol." In *Proceedings of INFOCOM 2002*, June 2002.

[LNA00] J. Leiwo, P. Nikander, and T. Aura. "Towards network denial of service resistant protocols." In *Proceedings of the 15th International Information Security Conference*, August 2000.

[LRS00] F. Lau, S. H. Rubin, M. H. Smith, and Lj. Trajkovic. "Distributed Denial of Service Attacks." In *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2275–2280, Nashville, TN, USA, October 2000.

[LXS00] S. Liu, Y. Xiong, and P. Sun. "On Prevention of the Denial of Service Attacks: A Control Theoretical Approach." In *IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, June 2000.

[Maz] Mazu Networks. *Mazu Technical White Papers*. http://www.mazunetworks.com/white_papers/.

[MBF02] R. Mahajan, S. Bellovin, S. Floyd, V. Paxson, and S. Shenker. "Controlling high bandwidth aggregates in the network." *ACM Computer Communications Review*, **32**(3), July 2002.

[McAa] McAfee. *Personal Firewall*. http://www.mcafee.com/myapps/firewall/ov_firewall.asp.

[McAb] McAfee. *VirusScan Online*. http://www.mcafee.com/myapps/vso/default.asp.

[Mea99] C. Meadows. "A formal framework and evaluation method for network denial of service." In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, June 1999.

[Mim] MimeStar.com. *SecureNet PRO Feature List*. http://www.mimestar.com/products/.

[Moo] D. Moore. *The spread of the code red worm (crv2)*. http://www.caida.org/analysis/security/code-red/coderedv2_analysis.xml.

[MPR02] J. Mirkovic, G. Prier, and P. Reiher. "Attacking DDoS at the Source." In *Proceedings of the ICNP 2002*, November 2002.

[MRR03] J. Mirkovic, M. Robinson, and P. Reiher. "Forming Alliance for DDoS Defenses." In *Proceedings of the New Security Paradigmes Workshop*, August 2003.

[MVS01] D. Moore, G. Voelker, and S. Savage. "Inferring Internet Denial-ofService Activity." In *Proceedings of the 2001 USENIX Security Symposium*, 2001.

[MWJ00] G. R. Malan, D. Watson, F. Jahanian, and P. Howell. "Transport and Application Protocol Scrubbing." In *Proceedings of INFOCOM 2000*, pp. 1381–1390, 2000.

[Nag84] J. Nagle. "Congestion Control in IP/TCP." *RFC 896*, January 1984.

[Nar02] R. Naraine. *Massive DDoS Attack Hit DNS Root Servers*, October 2002. http://www.esecurityplanet.com/trends/article/0,,10751_1486981,00.html.

[Nat01] National Infrastructure Protection Center. *Advisory 01-014: New Scanning Activity (with W32-Leave.worm) Exploiting SubSeven Victims*, June 2001. http://www.nipc.gov/warnings/advisories/2001/01-014.htm.

[net] netfilter.org. *netfilter/iptables Homepage*. http://www.netfilter.org.

[NFR] NFR Security. *NFR Network Intrusion Detection*. http://www.nfr.com/products/NID/.

[NP99] P. G. Neumann and P. A. Porras. "Experience with EMERALD to DATE." In *1st USENIX Workshop on Intrusion Detection and Network Monitoring*, April 1999.

[OB] E O'Brien. *NetBouncer: A practical client-legitimacy-based DDoS defense via ingress filtering*. http://www.networkassociates.com/us/_tier0/nailabs/_media/research_projects/development_solutions/netbouncer_presentation.pdf.

[Pax01] V. Paxson. "An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks." *ACM SIGCOMM Computer Communications Review*, **31**(3), July 2001.

[PL01] K. Park and H. Lee. "On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets." In *Proceedings of ACM SIGCOMM 2001*, August 2001.

[PLR02] T. Peng, C. Leckie, and K. Ramamohanarao. "Defending against distributed denial of service attack using selective pushback." In *Proceedings of the Ninth IEEE International Conference on Telecommunications (ICT 2002)*, 2002.

[Pos80] J. Postel. "User Datagram Protocol." *RFC 768*, August 1980.

[Pos81] J. Postel. "Internet Control Message Protocol." *RFC 792*, September 1981.

[Pri03] G. Prier. *"iDward: Implementing D-WARD in the IXP."*. Master's thesis, University of California Los Angeles, 2003.

[Reaa] RealNetworks. *Helix Universal Server*. http://forms.real.com/rnforms/products/servers/eval/mbps.html.

[Reab] RealNetworks. *Real One Player*. http://www.real.com/realone/index.html.

[rou97] route. "LOKI2: (the implementation)." *Phrack Magazine*, **7**(51), September 1997.

[SAN00] SANS Institute. *NAPTHA: A new type of Denial of Service Attack*, December 2000. http://rr.sans.org/threats/naptha2.php.

[SD00] N. Long S. Dietrich and D. Dittrich. "An Analysis of the "Shaft" distributed denial of service tool." In *Proceedings of LISA 2000*, 2000. http://www.adelphi.edu/∼spock/shaft-lisa2000.pdf.

[SDS01] D. Schnackenberg, K. Djahandari, and D. Sterne. "Infrastructure for Intrusion Detection and Response." *Advanced Security Research Journal*, **3**(1), 2001.

[SH02] J. Shapiro and N. Hardy. "EROS: A principle-driven operating system from the ground up." In *IEEE Software*, pp. 26–33, January/February 2002.

[Sha03] S. Shankland. "SCO Web site slammed by Net attack." *ZDNet.com*, May 2003. http://zdnet.com.com/2100-1105_2-999584.html.

[SKK97] C. Schuba, I. Krsul, M. Kuhn, G. Spafford, A. Sundaram, and D. Zamboni. "Analysis of a denial of service attack on TCP." In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, May 1997.

[Sou] Sourcefire. *Snort: The Open Source Network Intrusion Detection System*. http://www.snort.org/.

[SP99] O. Spatscheck and L. L. Petersen. "Defending Against Denial of Service Attacks in Scout." In *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation*, February 1999.

[SP01] D. X. Song and A. Perrig. "Advanced and authenticated marking schemes for IP Traceback." In *Proceedings of IEEE Infocom 2001*, 2001.

[SPS01] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer. "Hash-Based IP Traceback." In *Proceedings of ACM SIGCOMM 2001*, August 2001.

[SWK00] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. "Practical Network Support for IP Traceback." In *Proceedings of ACM SIGCOMM 2000*, August 2000.

[Sym] Symantec. *An Analysis of Slapper Worm Exploit.* http://securityresponse.symantec.com/avcenter/reference/analysis.slapper.worm.pdf.

[Tri] Tripwire. *Tripwire for servers.* http://www.tripwire.com/products/servers/.

[Wea] N. Weaver. *Warhol Worm.* http://www.cs.berkeley.edu/∼nweaver/worms.pdf.

[WLS02] 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." In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*, pp. 255–270, Boston, MA, December 2002. USENIX Association.

[YEA00] J. Yan, S. Early, and R. Anderson. "The XenoService – A Distributed Defeat for Distributed Denial of Service." In *Proceedings of ISW 2000*, October 2000.

[ZL97] Y. L. Zheng and J. Leiwo. "A Method to Implement a Denial of Service Protection Base." In *Information Security and Privacy*, volume 1270 of LNCS, pp. 90–101, 1997.