Defence Against Denial of Service in Self-Aware Networks

Thesis submitted for

the Degree of Doctor of Philosophy of the University of London

and

the Diploma of Imperial College

Georgios Loukas

Intelligent Systems and Networks Group

Dept. of Electrical & Electronic Engineering

Imperial College London

ABSTRACT

Denial of Service (DoS) has become a prevalent threat in today's networks. Motivated by an impressive variety of reasons and directed against an equally impressive variety of targets, DoS attacks are not as difficult to launch as one would expect. Protection against them is, however, disproportionately difficult. Despite the extensive research in recent years, DoS attacks continue to harm. In our thesis, we start with a historic timeline of DoS incidents to illustrate the variety of types, targets and motives and how DoS attacks evolved during the last 10 years. We then present an overview of the existing proposals on both detection of such attacks and defence against them. Recognising the fact that the networks of the near future will feature self-awareness and online interaction with the users, we investigate the application of existing techniques together with novel techniques that we have designed, on the DoS resilience of Self-Aware Networks (SAN). We introduce a generic framework of DoS protection based on the dropping of probable illegitimate traffic, and we present a mathematical model with which we can measure the impact that both attack and defence have on the performance of a network. The mathematical results are validated with simulation results and experimental measurements in a SAN environment. We then introduce a variation of the generic defence, by using prioritisation and rate-limiting instead of simple dropping. We describe the implementation details and present experimental results. We also present a tool based on our mathematical model, which can recommend the optimal number and distribution of tasks among the defending nodes in a network. Last, we sketch potential future

extensions to our work and DoS research in general.

ACKNOWLEDGMENTS

I am deeply grateful to Professor Erol Gelenbe for being the kind of supervisor that PhD students hope for. Always available to listen to his students' ideas, work on their problems and suggest solutions. Always providing with opportunities; to travel, to learn, to discuss with knowledgeable people. Always making sure that his students are well-funded and well-equipped. A magnificent person. A true intellectual. My mentor.

My sincere thanks to all members of the ISN group at Imperial College for the arrestingly friendly environment that I had the pleasure to work in. Special thanks to Michael Gellman for the endless times he uncomplainingly set aside his problems to deal with mine, and to Greg Zachariadis for his invaluable support at the beginning, when everything seemed so difficult.

My deepest thanks to Georgia for her patience and understanding. Thank you for making my life beautiful and selflessly encouraging all my dreams.

I am eternally indebted to my family, who have always put my welfare before theirs. My mother's wise advice and wit, my father's work ethic and confidence in me, and my sister's incredible cheerfulness under the most difficult circumstances, have guided me through many challenges.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Denial of Service attack is any intended attempt to prevent legitimate users from reaching a specific network resource.

Denial of Service attacks are known to the network research community since the early 1980s. In a 1983 paper, V.D. Gilgor provides one of the first descriptions of denial of service in operating systems [2], and in his 1985 paper on TCP/IP weaknesses [3], R.T. Morris writes that *"the weakness in this scheme (the IP protocol) is that the source host itself fills in the IP source host id, and there is no provision to discover the true origin of the packet"*. A decade later it became clear that the attackers would routinely exploit this weakness of the IP by faking their source address, a practice which is called "IP spoofing". A typical generic DoS attack is practically always distributed (DDoS): the attacker takes control of a large number of lightly protected computers, such as those without firewall and up-to-date antivirus software, and orders them to send simultaneously volumes of meaningless traffic to a specific target (Fig. 1.1). As a result, some routers and links in the vicinity of the target are overwhelmed, and a number of legitimate clients cannot connect to it anymore. Typical targets are the servers of e-commerce websites, which can suffer significant financial losses. Other targets may be news websites, corporate networks, banks, the national infrastructure, market competitors, etc.

Recognising the fact that the networks of the near future will feature self-awareness and online interaction with the users and allow them to choose their desired level of quality of service, we investigate the application of existing techniques from the litera-

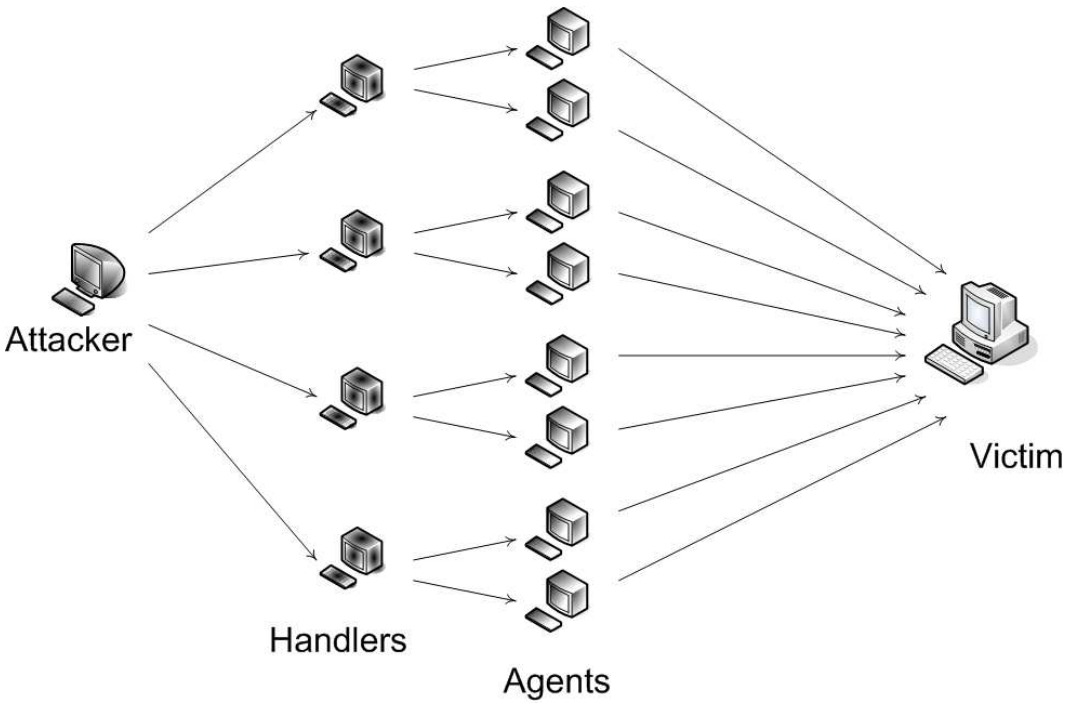

Fig. 1.1: In a Distributed DoS attack (DDoS) the attacker compromises a few vulnerable computers (handlers) and through them orders several other computers (agents) to simultaneously attack a specific target

ture together with novel techniques that we have designed, on the DoS resilience of such Self-Aware Networks (SAN). A SAN should provide a clear view of its current condition and be able to quickly recognise areas of congestion. It should feature dynamic routing of the traffic according to the requests of its users and the presence of congested areas. It should also be constantly monitoring the behaviour of all traffic, keep a history of it, and act upon anomalies. In other words, a SAN should be always fully aware of the real-time parameters that determine its performance as a service-providing medium. Naturally, a SAN will be expected to feature autonomic security, especially against Denial of Service attacks.

Denial of service attacks harm Self-Aware Networks in the same way they harm conventional networks, only to a different degree for the various network resources,

with the factor of differentiation being the dynamic routing. In a conventional network, the attack paths can be several but remain constant, which results into complete overwhelming of specific nodes or specific links on those static attack paths. A legitimate flow which does not use any of these paths, would be relatively unaffected by the attack, while the rest would suffer complete outages. In a SAN, the routing protocol attempts to accommodate all traffic by dynamically changing the paths. As a result, the attack is distributed in the whole network and affects the quality of service of the legitimate flows in a more balanced manner, in which all flows are affected, but fewer suffer complete outage.

We start in Chapter 1 with a historic timeline of DoS incidents to illustrate the variety of types, targets and motives and how they evolved during the last 10 years. In Chapter 2 we present an overview of the existing proposals on both detection of such attacks and defence against them. In Chapter 3 we introduce a generic framework of DoS protection based on the dropping of probably illegitimate traffic, and we present a mathematical model with which we can measure the impact that both attack and defence have on the performance of a network. The mathematical results are validated with simulation results and experimental measurements in a SAN environment, which uses the QoS-driven Cognitive Packet Network (CPN) packet switching architecture, currently under development in the Self-Aware Networks research group in Imperial College. In Chapter 4 we introduce a variation of this generic defence, by using prioritisation and rate-limiting instead of simple dropping. We describe the implementation details and present experimental results. In Chapter 5 we present a tool based on our mathematical model, which can recommend the optimal number and distribution of tasks among the defending nodes in a network. In Chapter 6, we identify new problems that have not been suggested before, and find solutions based on the work presented in this thesis. We conclude with a summary of our contributions and the open issues in our work, in

Chapter 7.

1.1 DoS incidents, types and motives

1.1.1 Early years

Although Denial of Service attacks existed during the 1980s and early 1990s, they were not involved in high-profile security incidents. Things started to change as the Internet was becoming a mainstream medium. DoS attacks rised from obscurity during the late 1990s, with targets being usually ISPs and governmental websites in the USA, which were considered as prestigious trophies for the hackers of the era, and this is when the most important techniques of service disruption were developed:

*The "**ping attack**"* is one of the first and simplest DoS attacks, in which the victim is flooded with more ICMP (ping) packets than it can handle. There is also the "**ping of death**" attack, in which certain systems could crash when they receive oversized ICMP packets [101].

*In "**SYN Flood**" attacks* , the attacking system sends SYN messages to the victim server system, which appear to be legitimate but in fact reference a client system that is unable to respond to the SYN/ACK messages. This means that the final ACK message will never be sent to the victim server system. The half-open connections data structure on the victim server system will eventually fill, rendering it unable to accept any new incoming connections [103].

*In an "**IRC-based DDoS attack**"* , an IRC [1] communication channel is used to connect

[1] Internet Relay Chat (IRC) is a form of instant communication over the Internet. It is mainly designed for group (many-to-many) communication in discussion forums called channels, but also allows one-to-one communication. An IRC server can connect to other IRC servers to form an IRC network [108]. IRC

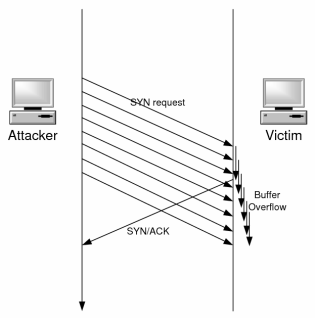

Fig. 1.2: SYN Flood attack: The victim receives an overwhelming number of illegitimate connection requests

the client to the agents. By making use of an IRC channel, attackers gain additional advantages. They can use legitimate IRC ports for sending commands to the agents, which makes tracking of the DDoS command packets more difficult. IRC servers tend to have large volumes of traffic making it easier for the attacker to hide his presence from a network administrator. The attacker no longer needs to maintain a list of agents, since he can simply log on to the IRC server and see a list of all available agents. The agent software installed in the IRC network usually communicates with the IRC channel and notifies the attacker when the agent is up and running. IRC networks also provide for easy file sharing, which is one of the passive methods of agent code distribution. This makes it easier for attackers to secure secondary victims to participate in their attacks [47].

*In the "**smurf**" attack* , the attacker is using ICMP echo request ("ping") packets directed to IP broadcast addresses from remote locations to generate DoS attacks. There are three parties in these attacks: the attacker, the intermediary, and the victim. The intermediary receives an ICMP echo request packet directed to the

networks allow their users to create public, private and secret channels, with the latter being frequently used in denial of service attacks.

IP broadcast address of their network. If the intermediary does not filter ICMP traffic directed to IP broadcast addresses, many of the machines on the network will receive this ICMP echo request packet and send an ICMP echo reply packet back. When all the machines on a network respond to this ICMP echo request, the result can be severe network congestion or outages. When the attackers create these packets, they do not use the IP address of their own machine as the source address. Instead, they create forged packets that contain the spoofed source address of the attacker's intended victim. The result is that when all the machines at the intermediary's site respond to the ICMP echo requests, they send replies to the victim's machine. The victim is subjected to network congestion that could potentially make the network unusable [102]. Similar is the "**fraggle**" attack, which uses UDP packets instead of ICMP echo packets.

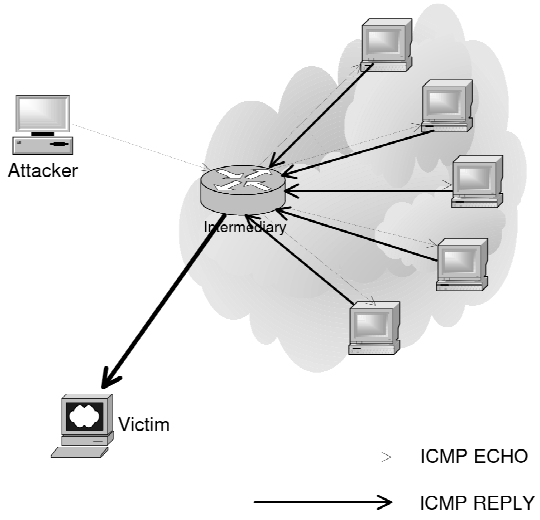

Fig. 1.3: Smurf Attack: The attacker sends a large amount of ICMP traffic to a broadcast address and uses a victim's IP address as the source IP so that the replies from all the devices that respond to the broadcast address will flood the victim

*"**Teardrop**" attacks* exploit the fact that the Internet Protocol (IP) requires fragmenta-

tion for a packet that is too large for the next router to handle. The fragmented packet identifies an offset to the beginning of the first packet that enables the entire packet to be reassembled by the receiving system. In the teardrop attack, the attacker puts a confusing value in the second or later fragment, and if the receiving operating system cannot cope with such fragmentation then it will probably crash [106]. Ironically, a month after the release of the relative security update by Microsoft, a new variety of Teardrop emerged, **"Bonk"**, which worked specifically on a loophole in this update.

1.1.2 The period of highest frequency of DoS incidents

In February 2000, the websites of Yahoo, eBay, Amazon, Datek, Buy, CNN, ETrade, ZDNet and Dell were among the targets of a 15-year old Canadian nicknamed "Mafiaboy" [2]. The attacks, which reached the rate of 1GB/s wreaked havoc in the victims' economies and changed the way people think about DoS. After that incident, DoS started moving from nuisance attacks for hacking prestige to economic crime, cyber-warfare and politically motivated attacks. Some notable recent targets were the IT systems of the port of Houston, stock markets around the world, and the DNS root servers of the Internet [3]. This is when it became clear that depending on the determination of the attacker, practically every computer network service can be disrupted with some sort of DoS attack, and often for long periods of time. The following are the most important of

[2] "Mafiaboy" was sentenced in September 2001 for the total damage of $1.7 billion that he caused. *(Source: BBC - "Mafiaboy hacker jailed", http://news.bbc.co.uk/1/hi/sci/tech/1541252.stm*).

[3] Named by the letters A through M, they are operated by U.S. government agencies, universities, nonprofit organisations and companies such as VeriSign. Of the original 13 servers, 10 are located in the U.S., one in Asia and two in Europe. A year after the incident most of the 13 servers had applied a new routing technique known as Anycast, with which their operators are replicating these servers around the world. *(Source: Network World Fusion - "Net security gets root-level boost", http://www.nwfusion.com/news/2003/1027ddos.html*).

the new types of attacks that appeared during this period:

*The "**Distributed Reflector DoS attack**"* is very similar in concept to the "Smurf" attack. The attacker orders his army of compromised computers to send connection requests to several perfectly legitimate computers, but using the victim's IP as the source in their packets. So, when the legitimate computers reply to these requests, the receiver of all is the victim. The resulting DoS attack is more distributed, more intense and significantly more difficult to trace.

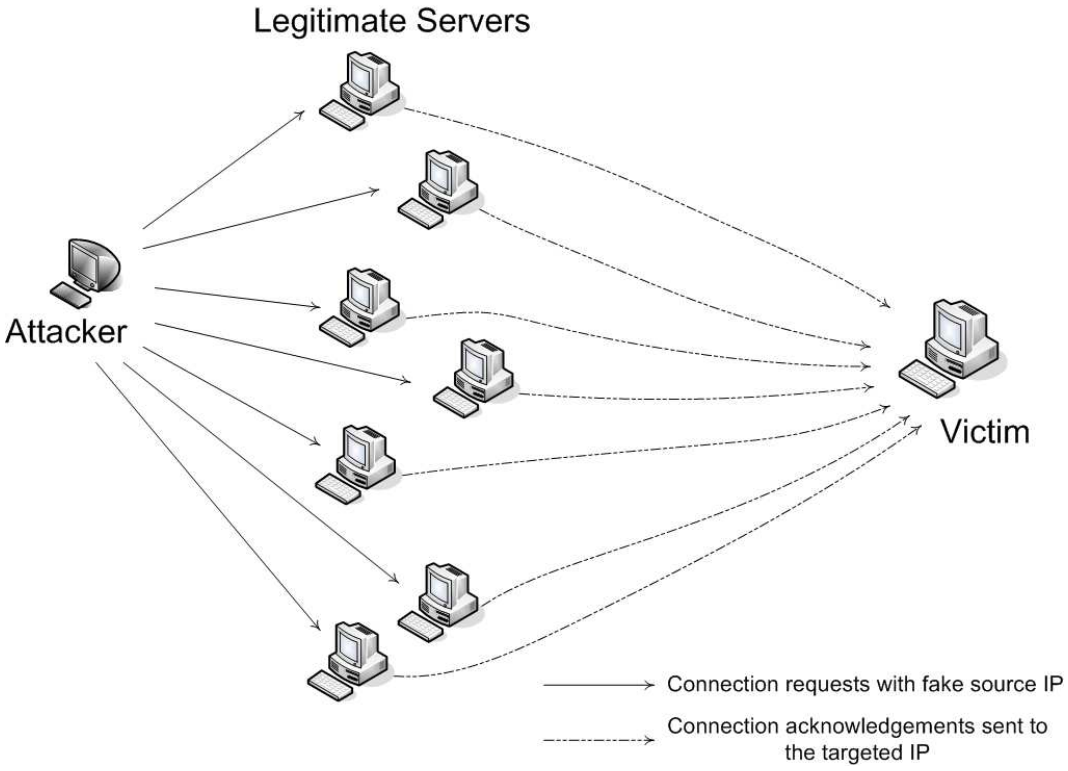

Fig. 1.4: In a Distributed Reflector DoS attack (DRDoS) the attacker uses a fake source IP (the target's) and sends connection requests to several legitimate servers. When these servers respond they send their acknowledgement packets to the attacker's target

*In an "**HTTP Flood**"* the attacker uses a large number of compromised computers, simultaneously and continuously requesting content, such as images, from a victim website. In the "**recursive HTTP Flood**" or "**HTTP Spidering**" the attack starts

from a given HTTP link and then follows all links on the website in a recursive way [107], inspired by the way search engines gather their data.

1.1.3 Current situation

Since before 2004, the DoS incidents have been deliberately not widely publicised, as the scene has shifted for good to the sensitive field of economic crime and DoS incidents harm the victims' reputation in the eyes of the increasingly security-aware public. Two major trends have emerged:

Cyber-extortion. In a famous incident from 2000, a Russian teenager stole 300,000 credit card numbers from an online music retailer and demanded $100,000 to destroy the data. When the company refused to pay, he released the numbers on the Internet, which destroyed the reputation of the company. Since then cyber-extortion has escalated and has become one of the most alarming types of cyber-crime. A common type of it is DoS extortion, in which the victims are threatened to be knocked offline until they pay, with typical targets being the betting websites and credit card processing companies. In these cases the general consensus is that the victims should avoid paying at all costs, since otherwise they appear as "soft targets" to attackers, information very easily spread among cyber-criminals. In reality, Internet downtime is so damaging for the finances and the reputation of online businesses that most victims choose to pay and simply inform the authorities (see Fig. 1.5 and 1.6).

Bot Armies. Several groups of cyber-criminals specialise in compromising large numbers of computers vulnerable to Denial of Service attacks. They build these "armies of bots", of a few thousand to allegedly up to 1.5 million computers, and rent them to potential DoS attackers.

A detailed historic timeline of landmark DoS incidents can be found in Appendix A.

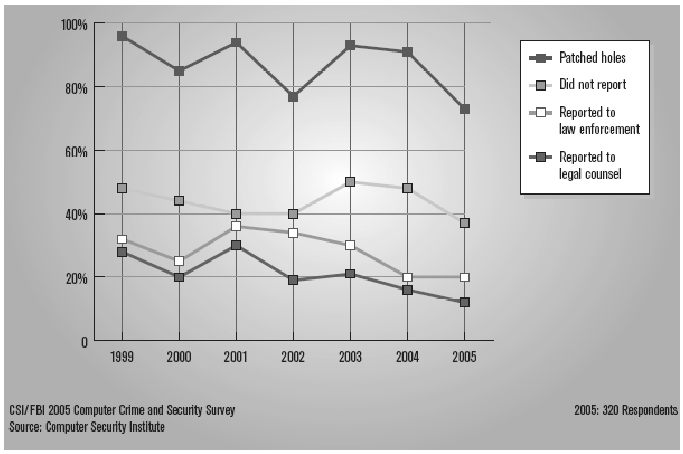

Fig. 1.5: Actions taken after computer intrusion(s) in 2005

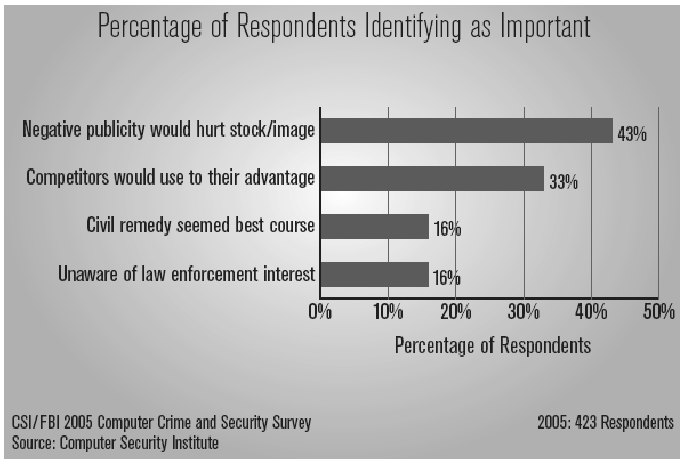

Fig. 1.6: Reason for which the organisation did not report intrusion(s) to law enforcement (by percentage of respondents identifying as important)

1.2 Statistics

Because of the distributed nature of Denial of Service and the difficulties that victims have not only in assessing the real damage but even in detecting a DoS attack altogether, there is little reliable DoS statistical data available. Still, the existing data can be indicative of the extend of the problem.

Tab. 1.1: Percentage of respondents who observed DDoS attack(s) during 1999-2005 [Source: CSI/FBI Computer Crime and Security Survey].

Year	% of respondents
1999	32
2000	27
2001	36
2002	40
2003	44
2004	38
2005	28

1.2.1 CSI/FBI Computer Crime and Security Survey

The "Computer Crime and Security Survey" is conducted annually since 1995 by the Computer Security Institute (CSI) with the participation of the San Francisco Federal Bureau of Investigation (FBI) [115]. The 700 participants in the survey are corporations, government agencies, financial institutions, medical institutions and universities, all located in the USA. The participating institutions were generally technically adept and aware of the various security threats. It is important to note that there is no evidence on whether the respondents are representative of typical or significant DoS targets and more so on whether they are representative of institutions outside the USA.

According to the findings of this survey, Denial of Service is currently the third most significant contributor to computer crime losses, behind Virus attacks and unauthorised access to information, with an approximate total loss of over 7 million dollars for the 639 respondents that were willing and able to estimate losses in 2005. The survey also shows that a significant percentage of the respondents (32%) suffered at least one DDoS attack during 2005 (see Table 1.2.1 and Fig. 1.7 and 1.8).

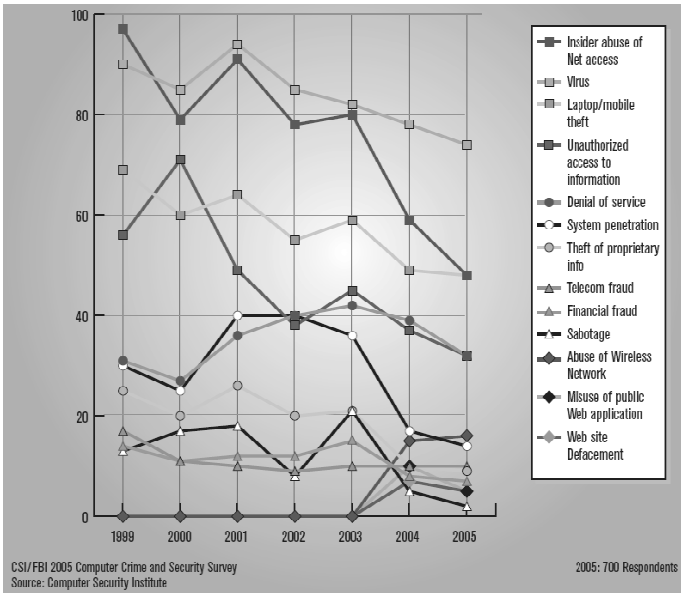

Fig. 1.7: Types of Attacks or Misuse detected in 2005 (by percentage of respondents)

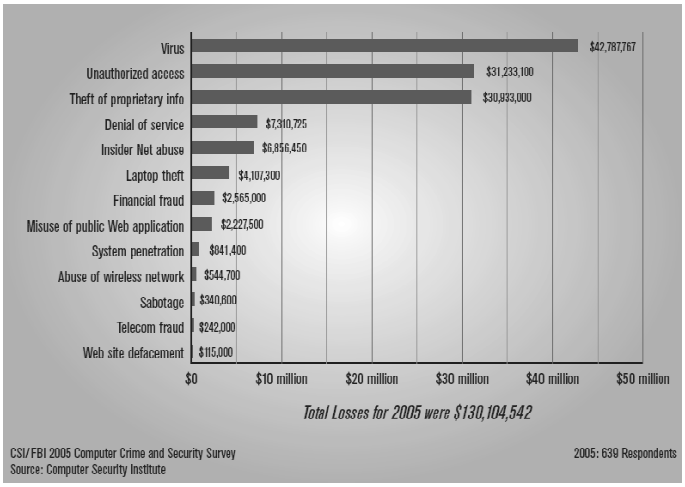

Fig. 1.8: Total dollar amount losses by type for 639 respondents

1.2.2 Backscatter analysis

In 2001, D. Moore, M. Voelker, and S. Savage proposed the “Backscatter analysis”, a clever traffic monitoring technique with which the total worldwide Denial of Service activity could be inferred [24]. The main hypothesis of backscatter is that unsolicited

packets represent responses to spoofed attack traffic. Although this hypothesis can not be validated beyond any doubt, they were able to duplicate small portions of their analysis with independent data taken from University-related networks in California and data by Asta Networks on DoS attacks which were directly detected at monitors covering a large backbone network.

Assuming that the attacking machines' sources are selected at random, the victim's responses would be equi-probably distributed across the entire Internet address range, an in-advertent effect called "backscatter" (see Fig. 1.9). Also, assuming reliable delivery and one response generated for every packet in an attack, the probability of a given host on the Internet to receive at least one unsolicited response from the victim is $\frac{m}{2^{32}}$ for a total number of m DoS packets. If one monitors n distinct IP addresses, then the expectation of observing an attack is:

$$E(X) = \frac{nm}{2^{32}}$$

The authors argue that by observing a large enough address range they can effectively "sample" all Denial of Service activity for which the assumptions hold. In fact, given the same assumptions, the average arrival rate of unsolicited responses directed at the monitored addresses can provide an estimation of the actual attack rate directed at the victim, as follows:

$$R \geq R' \frac{2^{32}}{n}$$

where R' is the measured average inter-arrival rate of backscatter from the victim and R the extrapolated attack rate in packets/s.

For their experiments the authors monitored the sole ingress link into a lightly utilised network comprising the $1/256$ of the total Internet address space). Using the

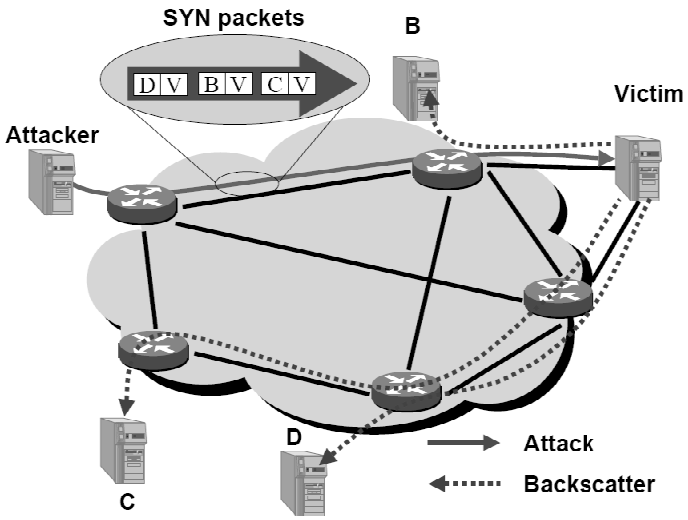

Fig. 1.9: An illustration of backscatter in action. The attacker sends a series of SYN packets towards the victim V, using a series of random spoofed source addresses: named C, B, and D. Upon receiving these packets the victim responds by sending SYN/ACKs to each of spoofed hosts.

"Backscatter analysis" technique they observed 12,805 attacks and almost 200 million backscatter packets over the course of one week. These values multiplied by 256 should provide a good estimation of the total number of DoS attacks and DoS packets in the Internet during that week of 2001. See Fig. 1.10 for the estimated number of attacks per hour as a function of time, and Fig. 1.11 for the probability density of the attack durations they observed.

1.3 Taxonomies of Distributed DoS attacks

The following taxonomies are based on [57] by Mirkovic and Reiher, and [58] by Specht and Lee, along with our personal observations.

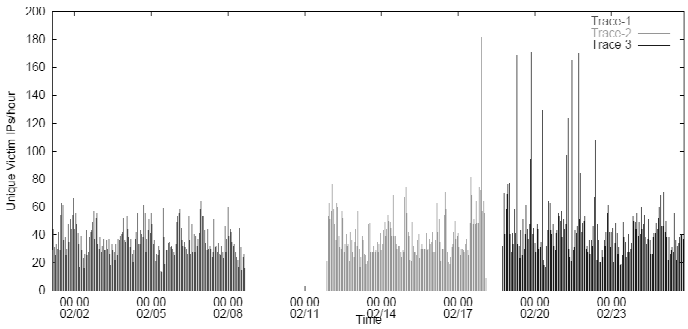

Fig. 1.10: Estimated number of DoS attacks per hour as a function of time (UTC) for three weeks of measurements.

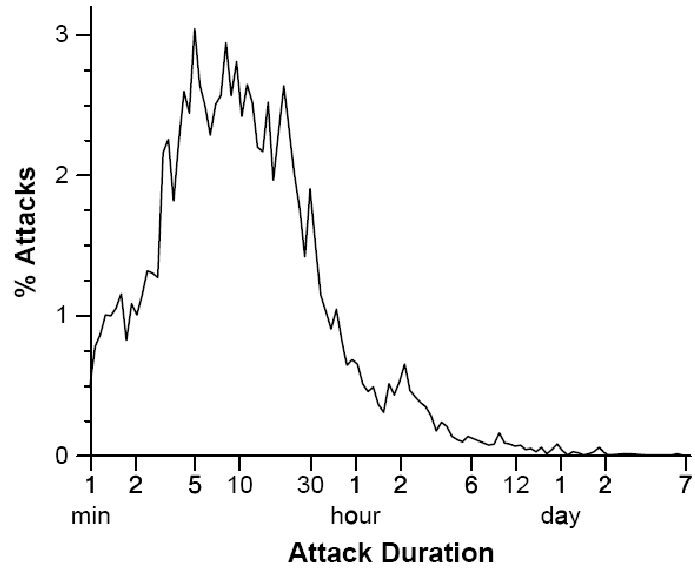

Fig. 1.11: Probability density of attack durations.

1.3.1 Attack architecture

- **Agent-handler model**. The attacker uses software packages (handlers) located throughout the Internet to communicate with compromised systems (agents) that will eventually carry out the attack. Usually attackers try to place the handler software on compromised routers or network servers able to handle large volumes of traffic.

- **IRC-based model**. Similar to the Agent-handler model, but the attacker communicates with the agents through an IRC channel.

1.3.2 Target resource

- **Processing**. Protocol exploit attacks and carefully malformed packet attacks belong to this category. The goal is to tie up the processing resources of the victim systems and prevent them from responding to legitimate requests, as in the SYN flood attacks (see Section 1.1).

- **Bandwidth**. The target is the victim system's network bandwidth, with either flood attacks, such as UDP floods and ICMP floods, or amplification attacks, such as the Smurf attack and the Distributed Reflector DoS attack presented in Section 1.1.

- **Infrastructure**. The attack against the 13 Root DNS Servers mentioned in Appendix A is a good example of a well organised effort to impede the operation of a crucial distributed service of the Internet.

1.3.3 Validity of source address

- **Spoofed**. Spoofing of the IP addresses is one of the most important weapons of a DoS attacker. The fake IP addresses can be real addresses of other computers which can be accessed, or non-routable unassigned ones. Then again the selected addresses can be either random spoofed, subnet spoofed [4], en route spoofed [5], or simply fixed. If the source addresses of the attack packets were the real ones, both detection of the attack and defence would be much simpler, collateral damage would be minimal and most amplification attacks would not be possible.

- **Valid address**. It is very uncommon that a DoS attacker will not hide the real source addresses of the attacking nodes, but there are such attack types.

1.3.4 Attack rate

- **Constant**. Most known attacks operate at constant rate per attacking source, since they are simple to organise and manage.

- **Variable**. A few types of attack deploy mechanisms of increasing rate, with the purpose of delaying detection at the victim's end. Although not common, there are also attacks which fluctuate their intensity according to the behaviour of the victim, or according to a preprogrammed time schedule, such as the "pulsing" attacks [6].

[4] In subnet spoofing, the attacker spoofs a random address from the address space assigned to the compromised machine's subnet. For example a machine with IP 155.195.13.101 would spoof any address from 155.195.13.0 to 155.195.13.255.

[5] In en route spoofing, the attacker spoofs the address of a machine which lies along the path from the compromised machine to the victim. Such spoofing has not been reported yet, but there are relative papers discussing it [23, 32].

[6] In a "Pulsing" DoS attack (PDoS) the agents periodically abort and resume, which when distributed, results at continuous disruption at the victim without any prolonged anomaly from any of the attacking sources. See [94, 95] for relative research on the matter.

1.3.5 Impact on the victim

- **Disruption of service**. The goal of most attacks is to completely deny the victim's service to its legitimate clients. This situation may be recoverable with an automated procedure or may need human intervention, such as the manual rebooting of machines.

- **Degrading of service**. Full-scale attacks which attempt to completely halt the victim's service are relatively easier to detect than hypothetical stealthier ones, which would aim only at degrading service. These could go undetected for long periods of time inflicting for example significant damage in lost revenue for an online shop. To our knowledge such attacks have not been reported, but there is a number of interesting papers describing possible mechanisms of both attack and defence against them (see Section 2.4, Shrew attacks).

1.4 DDoS attack tools

The following are some of the automated tools that have been used to launch Distributed Denial of Service attacks [59]:

Trinoo (or Trin00). A tool used to coordinate distributed UDP flood attacks. As a typical example of the handler-agent model (described in Section 1.3.1 the intruder connects to a so-called trinoo master (handler) and instructs it (via TCP) to launch a denial of service attack against one or more IP addresses. Then the trinoo master communicates via UDP with the trinoo daemons (agents) and orders them to attack for a specified period of time by sending UDP packets of a given size to random ports of one or more target addresses. For more details see D. Dittrich's analysis on Trinoo [8].

Tribe Flood Network (TFN). With this tool the attacker can orchestrate UDP flood, TCP SYN flood, ICMP Echo flood, and Smurf attacks. The TFN agents are controlled remotely through simple ICMP echo reply packets.

Stacheldraht. A combination of Trinoo and TFN, with some additional features, such as encrypted communication between the attacker and the handlers, and the ability to automatically update the code in the agents.

Shaft. Similar to TFN and Stacheldraht, but the attacker can now switch between handlers and agent ports on the fly. Also, communication with the handlers is done with a simple Telnet connection, and all commands contain a password and a "ticket" to keep track of the agents. The source ports and addresses are randomised, and the attack occurs in bursts of 100 packets per agent.

Tribe Flood Network 2000 (TFN2K). Among the first in the second wave of DDoS attack tools, TFN2K is an improved version of TFN. It features special adjustments to further complicate any attempts to find the true origins of the packets, allows remote execution of commands, and may even arrange the sending of "decoy" packets. It combines attacks over multiple protocols, including TCP, UDP and ICMP. It was clearly designed to counter the defence mechanisms that started to appear during the late 1990s.

Mstream. Attacker and handlers communicate via TCP, while handlers and agents communicate via UDP. The tool generates a flood of TCP ACK packets, and the IP addresses are spoofed at random.

Trinity. A DoS tool that is controlled through IRC or ICQ [7]. It can be used to launch several types of flooding attacks, such as UDP, IP fragment, TCP SYN, TCP RST,

[7] ICQ is a popular instant messaging application

and TCP ACK floods.

Agobot. The Agobot worm connects to an IRC server and allows remote users to launch DoS attacks against other IRC users. Variants of Agobot can allow remote access to the filesystem of their victims and steal specific information.

Phatbot. A direct descendant of Agobot. It implements SYN floods, UDP floods, ICMP floods, and a few newer types of floods (“Targa”, “Wonk”, etc.). Instead of IRC, it uses an unencrypted P2P protocol to control the agents.

2. EXISTING WORK ON DEFENCE AGAINST DENIAL OF SERVICE ATTACKS

The extreme diversity of DoS attacks has produced similarly diverse protection proposals from the network security research community. In most cases a complete protection architecture should include the following elements:

- ***Detection*** of the existence of an attack. The detection can be either anomaly-based or signature-based, or a hybrid of those two. In anomaly-based detection, the system recognises a deviation from the standard behaviour of its clients, while in signature-based it tries to identify the characteristics of known attack types.

- ***Classification*** of the incoming packets into valid (normal packets) and invalid (DoS packets). As in detection, one can choose between anomaly-based and signature-based classification techniques.

- ***Response***. In the most general sense, the protection system either drops the attacking packets or redirects them into a trap for further evaluation and analysis.

Detection and Classification usually overlap, since the method used to detect the existence of an attack may well provide the necessary information to start responding towards probably normal and probably DoS packets. Also, all three elements of protection may benefit by the use of an additional secondary element, which is the ***traceback*** of the real source of the traffic.

2.1 Looking for the true source

One of the first measures proposed was Ingress Filtering. It is a simple approach to thwart IP address spoofing (see Chapter 1) by configuring routers to drop arriving packets with IP addresses which are deemed to be outside a predetermined "acceptable" range [5]. In essence, packets are not allowed to leave a border network if their source address does not belong to this border network. The more widely deployed the better the protection that Ingress Filtering will offer. Of course, it requires that the receiving router has sufficient power and sufficient knowledge to examine the source address of each packet and distinguish between valid and invalid ones, while attackers can still overcome it by choosing legitimate addresses at random. Although not enough on its own, variations of Ingress Filtering can be used as a good lightweight first step to identify part of the attacking traffic.

Later, it was suggested that the real IP address could in fact be inferred. This "IP traceback" problem generated a number of important works [13, 19, 16, 17, 36, 48], where it was usually proposed either to monitor the flows traversing the network and mark packets probabilistically or to attempt to infer the attacking flows' paths based on their impact on the network. Traceback should be done with minimal cost in time and storage space and minimal number of legitimate sources erroneously detected as attack sources, while at the same time respecting the privacy of the contents of the audited packets, and not requiring interactive operational support from Internet Service Providers (ISPs). A fairly accurate and widely deployed traceback mechanism could potentially act as a deterrent, since the attackers would know that there is a high probability that their victim is protected with a system which can trace the attack flows back to the machines they used, and possibly back to them. Unfortunately, despite the elegant ideas and interesting implementations in the literature, current traceback mecha-

nisms are far from effective deterrents. In most cases, not only they cannot be deployed widely enough, but they also suffer from a common inherent weakness, which is that fake traceback messages created by attackers and included into the attack flows can help them hide their origin. However, even a rough estimate of the sources of attack packets may and usually does help the DDoS defence. In the more general sense, IP traceback can also be applied "post-mortem", after the attack is over, to help protect against future attacks. Following is a more detailed description of the main traceback proposals in the literature.

One of the first important and frequently cited works on the problem of determining the path that a packet traversed over the Internet in the context of a denial of service attack was that of Burch and Cheswick [14] in 2000. They proposed to systematically flood links which potentially belong to attack paths and observe the variations in the received packet rates. They first create a map of the routes from the victim to every network, then start with the closest router and apply a brief burst of load to each link attached to it. If the loaded link belongs to the attack path, the induced load will perturb the attacking stream. As they even themselves admit in their paper though, they assume too much. In their words, they assume that they can map the paths from the victim to all possible networks, that routes are all symmetric, can be discovered, are fairly consistent, and that the attacking packet stream arrives from a single-source network. An interesting first take on the traceback problem, but with no realistic application in today's networks; not to mention that flooding networks with traffic to protect them against flooding is at least inappropriate. The approach is usually referred to as controlled flooding, or more generally as link-testing.

Savage et al [13] were the first to propose Probabilistic Packet Marking to encode partial attack path information and include it in IP packets as they arrive in routers. They describe some basic marking algorithms, such as appending each node's address to the

end of the packet (with all the repercussions that this may have to the performance of a network), node sampling, where each router chooses to write its address in the node field with some low probability p, or edge sampling of participating routers. The victim receives all marking information and reconstructs probabilistically the attack path. The specific work by Savage et al started a wave of similar proposals in the following years, although it was not the most accurate traceback solution, and although it was practically applicable only for single-source attacks. Song and Perrig showed that the reconstruction of the attack paths based on the original Probabilistic Packet Marking techniques was computationally extremely intensive even for only a few attack sources [12].

A little earlier Bellovin had published an Internet Draft suggesting to use ICMP messages specifically generated by routers for traceback purposes. The main idea was that for one every 20,000 packets or so, the router will copy the contents of the packet and information about adgacent routers into those customised ICMP messages. Again the victim should potentially be able to reconstruct the approximate attack path [11]. Tupakula and Varadharajan [98] present a DDoS technique that could render such ICMP traceback useless, and in fact harmful for the networks it would be deployed in.

Some of the limitations of the original Probabilistic Packet Marking were addressed by Song and Perrig [19], who proposed two marking schemes for IP traceback, which they called advanced and authenticated. Since almost all Internet routes have less than 32 hops, an 11-bit hash of the IP address and a 5-bit hop count could be encoded in the 16 bits of the fragmentation field. They described two different hashing functions with which the order of the routers could be inferred when the attack path is reconstructed at the victim node.

Dean et al [17] followed a slightly different approach in the same sense of probabilistic packet marking. They reframe IP traceback as a polynomial reconstruction problem and use algebraic techniques originally developed for coding theory and machine learn-

ing. They present a series of schemes, all based on the principle of reconstructing a polynomial in a prime field, ultimately to manage to encode the probabilistic path information inside the IP header. For that they have to use its very few bits that could be considered available, or more accurately usually non-crucial. A strength of their approach is that it should potentially benefit automatically by any improvements in the underlying mathematical techniques, which are steadily being developed for the problems they were originally designed for.

The majority of the traceback solutions are targeted at high-speed attacks, where there is a large number of packets being sent during the detection phase, which in fact is what they are mainly based on. Sanchez et al [16, 36], on the other hand, propose a system to trace even single-packet attacks, such as those which exploit vulnerabilities in the packet processing of TCP/IP stack implementations ("Teardrop" was one of them, as presented in Section 1.1). This system, which they named SPIE ("Source Path Isolation Engine"), supports tracing by storing a few bits of unique information, essentially packet digests, for a period of time as the packets traverse the Internet. In order to minimise the storage needs for the digest tables, they used Bloom filters, which are space-efficient data structures with independent uniform hash functions. The software implementations of SPIE performed adequately for slow to medium speed routers. To achieve better results for faster routers, in [20] they describe in detail a hardware implementation of their system as a processing unit inserted into the router or stand-alone connected to the router through an external interface. In our opinion, either in software or hardware, the implementation cost and the computation overhead are too high for a technique oriented against an uncommon family of DoS attacks.

2.2 Investigating the source's validity

This can be done through either passive or active validity tests as described next.

Passive tests

Some passive tests which can be used for all types of traffic include the following:

- *Is the source of the packet a known "loyal client"?* In our approach, we take for granted that DoS attackers will spoof their addresses. However, that does not mean that the source IP address of a packet is not useful information. For example, if User-X, who has a static IP address, visits his favorite news website every day at 9am, stays for a while and consumes a reasonable amount of bandwidth, there is no reason for this website's detection mechanism to suspect that his packets are potentially harmful, except if his behaviour is somehow dramatically different from the usual one. Thus, in times of congestion, even if there is no official service differentiation, User-X should not be blocked in favor of a User-Y, who has not been recognised as a regular client and consumes a lot of bandwidth.

- *Did the source of the packet first appear before or after the detection of the attack?*

  Since DoS attacks are almost exclusively distributed, the distributed aspect can make it easier to detect their existence. Attack flows do not all traverse the Internet through the same paths; thus they reach the victim destination at different times, resulting in a gradual increase of the incoming traffic. This ramp-up behaviour was initially proposed as a method to tell whether an attack is distributed or single-source [45], but can additionally be a good indication that a DDoS attack has been initiated. A longer ramp-up time will also be associated with a greater number of spoofed source IP addresses. Consequently, it also means that the IP addresses which arrive after the DDoS and until it reaches its peak are more likely to be

illegitimate. Again, early detection is obviously a great advantage to identifying the spoofed set of IP addresses.

- *Is the client honouring his/her QoS agreements?*

  We are particularly interested in Self-Aware QoS-driven network environments in which clients may specify that they are a specific type of customer, or request a certain level of QoS. In case the QoS request is accepted, then the degree with which the agreement will be honoured by the client is a strong indication of his/her validity. We will describe this approach in Chapter 3.5. DoS attackers and misbehaving clients should both fail this test.

- *Does the time-to-live (TTL) field in an IP packet agree with the value that can be inferred from the packet's apparent source address?*

  This test refers to Hop Count Filtering (HCF), the ingenious idea described in [50] which exploits the fact that although the attacker can forge any field in the IP header, he/she cannot falsify the number of hops a packet needs to reach its destination starting from its source address. Their very simple algorithm infers the number of hops traversed until then (from the TTL field) and compares them to the value that can be inferred for this source, which is stored in a relevant table. If those two values are significantly different, that is a clear indication of IP spoofing, and is a good reason to treat this source's packets as probably illegitimate.

The above general passive tests have the advantage of being relatively light-weight, but are by themselves not sufficient to achieve accurate classification. More accuracy inevitably requires more specialisation.

In Mirkovic's et al leading work in [85] a set of anomaly-based classification criteria for flows and connections are proposed. For instance, a TCP flow may be classified as an attack flow if its packet ratio (TCP_{rto}) is above a set threshold. Similarly, for the ICMP

protocol they propose to use $ICMP_{rto}$ as a detector. For the UDP protocol, a "normal flow model" is proposed to be a set of thresholds based on the upper bound of allowed connections per destination, the lower bound of allowed packets per connection, and the maximum allowed sending rate per connection. The classification of connections is also done based on limits of the connections' allowed packet ratios and sending rates.

An important recent work [96] uses the bayesian concept of the conditional legitimate probability (CLP) as the basis of a packet filtering scheme. Traffic characteristics during an attack are compared with previously measured, legitimate traffic characteristics, and the computed CLP provides an indication of the legitimacy of suspicious packets. An extension of this is presented in [97], which aims to reduce the implementation complexity and enhance the overall performance. However, as with all profile-based, and particularly bayesian profile-based DoS approaches, the greatest challenge is not as much the finetuning of the defence mechanism as is to acquire dependable traffic profiles.

One more option for validity tests is to use criteria based on the type of service. This approach is closer to the QoS-driven approach of our ongoing work on CPN. For example, a network which offers Voice-over-IP (VoIP) services should include specific classification criteria based additionally on the intricacies of VoIP traffic behaviour. Then, it becomes a matter of who has the most advanced knowledge in this case on the VoIP traffic behaviour, the attacker who tries to emulate it or the defence system of the victim which examines it.

The limits and thresholds used by all these types of tests can be set by using the network administrator's experience, or with an automatic learning process using data collected from ongoing observation in all or some of the nodes.

Active tests

The first question that one has to answer when being under attack is whether it is a case of real attack or simply a case of unusually high legitimate traffic. That is because this ramp-up behaviour of traffic occurs also during "flash crowds"[1], when there is a sharp increase in the number of legitimate visitors to a website due to some significant event. Moreover, DoS attackers have recently started exploiting this fact by abandoning full strength bandwidth floods in favour of attacks which escape detection by imitating the signature characteristics of flash crowds. In response to this, detection mechanisms should try to distinguish between flash crowds and attacks after a ramp-up has been detected. Fortunately, there is a fundamental difference between the two. Flash crowd flows are generated by human users, while DDoS flows are generated by compromised computers. Thus, one can potentially use Reverse Turing tests to tell the difference. The last three years, Graphical Turing tests or visual CAPTCHAs, or simply CAPTCHAs (Completely Automated Public Turing Test to Tell Computers and Humans Apart), are commonly used to block automated requests to websites, such as the automated email account registration which has plagued Hotmail and Yahoo in the past. A human can easily tell the sequence of letters that appear in a CAPTCHA's image, while computers usually cannot (Fig. 2.1), so that CAPTCHAs have been suggested to counter DDoS attacks against webservers [49].

However, issuing a Graphical Turing test and expecting an answer requires that a connection be established between the attacker and the web-server, thus rendering the authentication mechanism a potential DoS target. The authors of [84] address this is-

[1] "Flash Crowd" is a science-fiction story written in 1973 by Larry Niven [1], about the consequences of a system of instantaneous, practically free teleportation booths that could take one anywhere on Earth in milliseconds. One consequence, not predicted by the builders of the system, was that with the almost instantaneous reporting of newsworthy events, tens of thousands of people worldwide would flock to the scene of anything interesting - along with criminals, hoping to exploit the instant disorder and confusion so created.

sue and suggest minor modifications to the TCP protocol to overcome it. They also argue that it is not the actual answer to the test but the behaviour of the client that matters. A human would solve the puzzle either immediately or after reloading the page a few times. A computer would probably continue requesting the web page. We agree with these ideas but believe that despite their value, a DoS detection mechanism cannot depend solely on CAPTCHAs. In all arms races it is only a matter of time for a countermeasure to arrive on the scene. Dedicated applications built by Computer Vision researchers achieve up to 92% success in solving commonly used types of CAPTCHAs [42]. Advances in Artificial Intelligence along with simple craftiness limit the value of CAPTCHAs, while visual CAPTCHAs are also a major impediment to computer users whose vision is impaired. Still, although CAPTCHAs cannot be the sole method of classification, we do agree with the idea that DoS attacks can be more readily detected if we can distinguish between humans and computer generated traffic, along with other techniques that recognise legitimate and DoS flows.

Fig. 2.1: This captcha of "smwm" obscures its message from computer interpretation by twisting the letters [taken from http://en.wikipedia.org/wiki/Captcha].

Several methods have been proposed for actively challenging the clients' legitimacy, with Netbouncer being the most representative [39]. It keeps a list of authorised users (beyong suspicion), while the rest undergo a series of tests, divided into packet-based (for example a simple ICMP echo request), flow-based (originally with stateless TCP/SYN cookies) and application and session-oriented (such as CAPTCHAs). Various quality of service techniques are utilised to assure fair sharing of the resources by the traffic of the legitimate clients, while this legitimacy expires after a certain interval and needs to be challenged again. Although successful in most common cases, sheer

volumes of traffic and possibly the misuse of the identities of legitimate clients can overcome this type of defence.

A very similar concept is explored in the defence solutions which are based on puzzles. In a connection-oriented DoS attack the clients are asked to solve a little cryptographic puzzle before their connection request is authorised. The puzzle may take a little while to solve, while for the defending server it is much faster to verify the result. That does slow down the attacker, but does not guarantee that it will be enough, since overwhelming the puzzle-generation process will still be possible if the attacker's rate is sufficiently high. Examples of such approaches can be found in [7], [10] and [41].

Some of the above active test solutions may achieve impressive levels of detection accuracy, but all suffer from the common weakness of being exploitable as DoS vessels themselves, in the same way as the simplest form of active validation, the ACKs, are used as DoS vessels in the reflector DoS attack [21].

2.3 Detecting the existence of a DoS attack

In the beginning of Chapter 2, we included *detection* as the first necessary element of a complete DoS protection system. This is not entirely accurate. Detection would not be necessary in the ideal case of a defence architecture with proactive qualities that would render impossible any DoS attack. In fact, interesting proactive architectures have been proposed, a few of which describe later in this chapter (2.4), but they do have some common disadvantages. Apart from the fact that to date no system is perfect, DoS attacks against one's network do not happen very often and at least resource-wise a proactive protection system is usually too expensive to operate in the absence of an attack. This can be avoided if the protection system starts operating right after a potential attack is detected, which renders the detection step necessary in most realistic cases.

In the following, we present a short summary of the existing literature on detection methods.

2.3.1 Learning techniques

Learning paradigms, such as neural networks, radial basis functions or genetic algorithms, with the intelligent and automatic classification that they can offer, are increasingly used in DoS detection. A vital problem in using this class of methods is the selection of the set of input features that will provide useful and significant information about the incoming traffic. Additionally, these systems must be designed to be fast enough to cope with the requirement of real-time decision.

Jalili et al [81] have proposed the use of an unsupervised neural network for the detection of DoS attacks. They designed a Statistical Pre-Processor and Unsupervised Neural Net based Intrusion Detector (SPUNNID), in which a statistical pre-processor is used to extract features from packets, and the feature vector is changed to numerical form to be fed to an unsupervised Adaptive Resonance Theory net (ART). In ART nets, learning is accomplished by updating the cluster weights according to the relative similarity between the weights and the input. The ART is first trained with normal and attack types of input vectors; in the testing phase it is expected to classify the packets using the adjusted cluster weights. The features they use include the percentage of ICMP packets, UDP packets, TCP packets, SYN packets in TCP packets, SYN+ACK packets in TCP packets, ACK packets in TCP packets, and the average packet header and packet data sizes.

Gavrilis and Dermatas [80] use radial basis function neural networks (RBFNN) and statistical features to detect DoS. Their scheme comprises a data collector which collects the appropriate data fields from the incoming packets, a feature estimator that evaluates the frequencies for the encoded data, and a RBF-NN detector for classification as either

normal traffic or DoS attack. For TCP traffic, the performance is evaluated with two different feature sets. The first set to contains the source port, SEQ number of the client, window size, and the SYN, ACK, FIN, PSH, URG, and RST flags, while the second is a smaller set consisting only of the source port, SEQ number and SYN flag. The source IP address is not used to decrease computational complexity. Their experiments with data obtained from a simulated public network and a real network, show that the TTL, window size and some of the flags do not provide enough information about the occurance of a DoS attack. For detecting DoS in case of UDP traffic, the source port and the TTL have been used. Gaussian functions are utilised as non-linear elements in RBFs, successful detection was possible even with a limited number of hidden neurons. Correct classification rates varied depending on the choice of time frame, number of input features, number of hidden neurons and training data. The results obtained by the three-feature set were similar to the rates obtained with the nine feature set. Experimental results reported for UDP were also successful, but with lower correct classification rate and smaller efficacy.

Gavrilis et al [76] have studied the same optimum feature selection problem using genetic algorithms and proposed a scheme for robust detection of DoS attacks. Using the same detection method as in [80], they present a method to determine the importance of each input feature in DoS detection. They select an initial set of 44 statistical features (Table 2.1) and from this set, they choose the most relevant features with a genetic algorithm. The total scheme is composed of a data collector, a features estimator and a two-layer feed-forward neural network detector. The authors observe that selected features vary with the number of hidden neurons, and with the selection and mutation probabilities. They found that generally the SYN and URG flags, and some ranges of ports were the most useful for the identification of a DoS attack.

Noh et al [55] propose a traffic rate analysis (TRA) mechanism to measure the TCP

Tab. 2.1: Statistical features for detection used in [80]

1-5	The prob. of the SYN, ACK, FIN, URG, RST flag to be raised.
6	The distinct SEQ values
7-8	The distinct values of the source and destination port.
9	The prob. of the source port to be within the first 1024 values.
10-25	The 16 possible ranges for the source port values in 1024-65535 divided in groups of 4032 ports.
26	The prob. of the dest. port to be within the first 1024 values.
27-42	The 16 prob. of the destination port value in 1024-65535 divided in groups of 4032 ports.
43	The distinct values of the window size.
44	The distinct TTL values.

flag rate and the protocol rate and utilised three machine learning algorithms, namely C4.5, CN2 and a Bayesian classifier, for detecting DoS. They provide experimental results in a simulated TCP-based network. In TRA, packet collecting agents are used for classifying IP packets into TCP, UDP or ICMP packets and for further summing up the SYN, FIN, RST, ACK, PSH and URG flags for TCP packets. For a specific sampling period the rate of a certain type of flag is determined by dividing the total number of this flag's occurences to the total number of TCP packets observed while the ratio of the number of TCP, UDP or ICMP packets to the total number of IP packets gives the protocol rate. Their experiments showed that SYN and ACK flag rates for inbound traffic provided significant information for detection of SYN flooding attacks and the best performance was obtained by the Bayesian classifier. False alarms did occur but no missed alarms were observed.

Kim et al [60] have proposed a data mining approach based on an automatic feature selection mechanism with a neural network classifier for DoS attack detection. They use a decision tree, together with entropy and chi-square concepts, to select the best out of a set of candidate features. These features are used in the neural network classifier. In their experiments, the candidate attributes were the octet count per flow, the packet

count per flow (P/F), the TCP octet count per flow, the TCP packet count per flow, the UDP octet count per flow, the UDP packet count per flow , the source port variance for TCP traffic (srcTport) , the source port variance for UDP traffic, the destination port variance for TCP traffic, the destination port variance for UDP traffic, the source IP address variance, the TCP traffic ratio (Tratio), and the UDP traffic ratio. The decision tree selected P/F and srcTport using both entropy and chi-square approaches. When information about traffic type (TCP, UDP or a mixture of the two) was also used in the selection mechanism, P/F, Tratio and srcTport were the automatically chosen attributes.

Siaterlis and Maglaris [67] have designed a data fusion system, with which they aggregate information about the internet traffic collected by different sensors, and used the Dempster-Shaefer (DS) Theory of Evidence to combine the data. In the DS framework one can state hypotheses, define membership, belief, plausibility and doubt functions regarding the hypotheses and eventually combine all evidence using a rule, to obtain a single conclusion. The proposed system is tested on a university network where information was collected with a Snort plugin and MIB entries.

He et al [87] used an Adaptive Neuro-Fuzzy Inference System (ANFIS) together with the Fuzzy C-Means Clustering Algorithm (FCM) to detect DoS attacks and tested their method by performing experiments on a DARPA/KDD99 dataset. Lee et al[88], proposed a scheme where first the content of the incoming packets are analyzed, then a probe detection system using fuzzy cognitive maps (PDSuF) is utilised to detect DoS attacks and a black list of IP addresses is constructed in accordance with the collected information. Mukkamala and Sung [62] applied three computational intelligent techniques; support vector machines (SVMs), multivariate adaptive regression splines (MARS) and linear genetic programs (LGP), in a multi-agent setting, to the problem of DoS detection and compared the experimental results obtained by all three methods. For these intelligent methods, Sung and Mukkamala [63] also addressed the feature se-

lection problem and gave feature ranking algorithms for each of them. Cheong et al [64] performed a causality analysis by combining DoS attack types and network protocol measures in a cause-effect framework. Giacinto et al [53], and Mukkamala et al [89] have used multiple classifier systems for intrusion detection and reported results for DoS detection. Chan et al [90] proposed an entropy based feature grouping method for multiple classifer systems and used RBFNNs as base classifiers. Shin et al [68] classified and minimised false alarms in DoS detection by using a decision tree approach.

2.3.2 Statistical signal analysis

Internet traffic has some statistical properties which can be used in the detection of DoS attacks. If the autocorrelation function r_{xx} of a time series x is not summable; that is if $\int_0^\infty r_{xx}(\tau)d\tau = \infty$, the series is said to be long-range dependent (LRD). Otherwise, it is a short-range dependent (SRD) series. Normal Internet traffic is known to be LRD and self-similar. Self-similarity of a time series can best be established by evaluating its Hurst parameter. If, as $\tau \rightarrow \infty$ the autocorrelation function satisfies $r_{xx} \sim c\tau^{-D}$, with $D = 2 - 2H$, where c is a positive constant, H is called as the Hurst parameter. A time series with an H value close to 1 is highly self-similar, while $0 < H \leq 0.5$ implies lack of self-similarity. Another statistical property of Internet traffic is that it can be represented as Fractional Gaussian Noise (FGN). Li et al [69] report experiments showing that Fractional Gaussian Noise (FGN) can be used to approximate autocorrelation functions of different types of traffic (TCP, UDP, IP and OTHER) with the same order of modeling accuracy.

Li [70] has also proposed a DoS detection scheme based on the autocorrelation functions of the incoming traffic. The abnormal traffic $y(t)$ is expressed as the sum of $x(t)$

the number of bytes in packets in normal traffic at time t and $n(t)$, the attack traffic:

$$y(t) = x(t) + n(t)$$

The norm of the differences of the autocorrelation functions is then used as a measure for DoS detection:

$$\begin{aligned} \|r_{xx} - r_{yy}\| = \xi > V,\ Identification \\ \|r_{xx} - r_{xl}\| = \zeta > V,\ Falsealarm \\ \xi < V,\ Miss \end{aligned}$$

where ξ and ζ are Gaussian random variables, the real number $V > 0$ is a threshold and r_{xl} is the autocorrelation function when there is no attack. The threshold V is selected with respect to predetermined numerical values of detection, false alarm and miss probabilities. Simulation results for a case study are presented and it is verified that real-time detection is possible.

Xiang et al [71] use the self-similarity property of Internet traffic to identify DoS attacks. They use the packet number or packet size as the input feature and they evaluate the Hurst parameter H by statistical techniques. The variance of H in consecutive time intervals is calculated and if there is a doubling in the variance, it is decided that there is a DoS attack:

$$DoSattack = \begin{cases} 1 & \text{if } Var(H_1, ...H_n)/Var(H_1, ...H_n) \geq 2 \\ 0 & \text{if } Var(H_1, ...H_n)/Var(H_1, ...H_n) < 2 \end{cases}$$

Gu et al [92] proposed an early-bird system of traffic anomaly detection scheme (ES-TABD) which consists of a data center, a traffic anomaly detection module and an event

correlation module. The raw variables collected at the data engines, namely the packet count variable of each protocol, the packet count variables of SYN and FIN flags, ICMP unreachable messages and the packet length count variable of packet payload, are aggregated at the data center. The traffic anomaly detection module detects anomalies from the data provided by the data center, using Statistical Prediction Theory. The traffic is observed for a long period of time and is classified into periodic and non-periodic parts. To predict anomalies in the non-periodic traffic, the "Single Exponentially Smoothed Prediction Method" is used and "Winters Level Seasonal Exponential Smoothed Prediction Method" is utilised for periodic traffic. Event correlation module evaluates these predictions with predefined pattern profiles. The use of dynamic thresholds is another novelty introduced in this study.

Kulkarni et al [26] suggest using Kolmogorov complexity metrics for DoS detection. Their work is based on the observation that, in DoS attacks, highly correlated packets are used, while legitimate traffic is in some sense random. The Kolmogorov metric, $K(x)$, is a measure of correlation in the strings forming the packets, i.e., if the complexity of a concatenated string XY, is smaller than the sum of the complexities of each string X and Y, $K(XY) \leq K(X) + K(Y)$, this implies a high correlation. Thus, they use complexity differentials:

$$[K(x_1) + K(x_2) + ... + K(x_n)] - K(x_1x_2.....x_n)$$

which yield a value close to zero in case the correlation among the concatenated strings is close to the sums of the separate correlations, indicating legitimate traffic. Since the exact values of the Kolmogorov complexity metrics cannot be calculated, estimates have been utilised [26]:

$$\hat{K}(x) \approx l(x)H(\frac{x\#1}{x\#1 + x\#0}) + log_2(l(x)) \tag{2.1}$$

where $l(x)$ denotes length of the string, $x\#0$ and $x\#1$ are the numbers of zeros and ones in the string and H is entropy.

The work of Kulkarni [26] is improved by Furuya et al in [91], where they propose a different Kolmogorov metric estimate, which measures the correlation between the first and second halves of the strings. They compare their experimental results with those obtained by using the method suggested in [26]. They also design a DoS detection algorithm which evaluates iteratively the fluctuations of the Kolmogorov complexity differentials. A significant change in the values of these differentials will indicate a change in the correlation of packets and thereby signal a DoS attack.

Feinstein et al [52], analyse the packet headers and evaluate the entropy, which is a measure of randomness, and the chi-square statistic, which can be used as a measure of the statistical significance and rough estimate of confidence in DoS detection. Hussain et al [73] also utilise signal processing techniques and classify single and multi-source DoS attacks using information of packet header content, ramp-up behaviour and spectral content. A method using collaborative detection and filtering specifically for shrew attacks is described in [99]. The autocorrelation functions and power spectrum densities of the traffic are obtained and used in a hypothesis testing scheme.

2.3.3 Wavelet transform analysis

Methods using wavelet transform analysis have also been applied in the detection of DoS attacks. Several experiments have shown that the energy distribution of normal traffic is stationary. During a DoS attack however, the traffic behaviour changes significantly, as well as the energy distribution variance. Thus, Li and Lee [93] use wavelet analysis to extract information about the energy content of the packets. For a time series $x(t)$, projected onto a wavelet subspace, the wavelet coefficients $d_x(j,i)$ are evaluated and $|d_x(j,i)|^2$ gives numerical information about the energy distribution. If $x(t)$ and $x(t+\tau)$

are the original and shifted time series, the following energy distributions are defined:

$$Eg_j^t = \frac{1}{n_j}\sum_k |d_x^t(j,k)|^2 \tag{2.2}$$

$$Eg_j^{t+\tau} = \frac{1}{n_j}\sum_k |d_x^{t+\tau}(j,k)|^2 \tag{2.3}$$

$$\tag{2.4}$$

where n_j denotes the number of wavelet coefficients. The difference in the energy distributions Eg is then computed by:

$$\Delta Eg_j = logEg_j^t - logEg_j^{t+\tau} \tag{2.5}$$

$$= log\frac{Eg_j^t}{Eg_j^{t+\tau}}. \tag{2.6}$$

$$\tag{2.7}$$

Traffic is considered to be "normal" if the variation in the energy distribution is smaller than a predetermined threshold δ:

$$x(t) \in \{x(\tau)|var(\Delta Eg_j) \ < \ \delta, \ for\ \tau > T\}, \tag{2.8}$$

where T is a time step increment related to the specific sliding window used in the computations. An important weakness of this work is that crucial issue of determining the numerical values of both δ and T are not updated adaptively.

Another wavelet transform based method is BDA-CWT, a network traffic burst detecting algorithm based on the continuous wavelet transform, proposed by Yang et al [66], where bursts in network traffic are considered to belong to three classes (long-bursts, short-bursts and one-point bursts). A continuous wavelet transform algorithm is used for the real-time identification of flat bursts in network traffic. The feature used in

the algorithm is the number of packets per second, and experiments are performed with SYNFlood software.

Wavelet based methods have advantages as well as drawbacks. A significant advantage of the wavelet method is early detection of DoS attacks, but there are important computational limitations to consider. Such methods use sliding sampling windows, the size of which, together with the choice of time step increments, determine the algorithm's performance. A small window may be inadequate to capture enough samples to calculate the energy distribution variation, while a large window results in computational overhead. Other problems include the possibility of boundary effects and the fact that wavelet based methods can mistakenly detect other (non-attack) variations in Internet traffic as DoS.

2.3.4 Multiple agents

Peng et al [54] have proposed the Source IP address monitoring (SIM) scheme, a multi-agent approach to detect DoS attacks, in which new legitimate source IP addresses appearing in the traffic are collected in the IP Address Database (IAD) in the off-line training phase. This database is utilised together with data from the incoming traffic to decide about possible DoS attacks in the detection and learning phase. The CUSUM method is used to extract information about the change in the number of new IP addresses, yielding a variable y_l to represent the change and a local threshold N_l for the $l-th$ agent. Each agent in the network applies this scheme and in case any one of them suspects an attack it will share its suspicion with the other agents. The *l*-th agent will broadcast a warning to the other agents if $N_l - y_l > T$. Since the selection of the broadcast threshold T is especially important, a gradient-based learning technique is used to select an optimum value of T that minimises both the communication overhead and the confirmation delay. This multi-agent approach will be more successful in detecting a

highly distributed DoS attack, but also slower compared to a centralised system because of the confirmation delays.

Cetnarowicz and Rojek [72] also apply a multi-agent approach to the computer security problem with an emphasis on DoS attacks, by which they differentiate between good and bad behaviour in the network using concepts from biological immune systems. Seo and Cho [29] utilise an agent based scheme with a Black Board Architecture (BBA) and firewall to detect DoS attacks and to form a black list of IP addresses.

2.3.5 Conclusions on Detection

We have seen that a wide variety of DoS attack detection methods have been suggested in the literature. Many of these are based on symbolic analysis of the traffic packets and in particular of IP addresses and other significant content of the packets. Other approaches are based on the timing characteristics of the packets streams. All of them require or assume some representation of what is a "normal" traffic stream as opposed to a DoS related stream. Many of the techniques require an on-line tuning or learning phase that is used to create patterns, data or statistics that can be used to compare with presumed attacks. Although each of these approaches offers some very interesting ideas and insights into DoS attacks, little if any work has been carried out so far to compare these different approaches against each other under similar conditions. In addition, although some of these approaches address similar types of attacks, there is also a significant difference in the actual types of attacks that these different detection schemes are trying to address.

We feel that an evaluation and accuracy comparison of the numerous contributions included in this survey is likely to be impossible, because the results reported by different authors originate from very different experimental setups or simulation data sets. Furthermore, some of the studies we have surveyed concentrate solely on detection of

DoS, while there has been a significant amount of work on extending intrusion detection techniques to DoS and developing general detection systems to work for any kind of attack including DoS. It may be misleading to consider DoS as just another type of attack and to build a detection mechanism based on this assumption. DoS attacks, with their peculiar characteristics and complexities, pose a specific challenge as a detection problem and can only be solved with approaches that concentrate on these properties.

However this survey indicates that an overall DoS detection mechanism will have to pragmatically combine different detectors in order to provide robust and effective detection, especially as attack methods and schemes evolve and are improved. Initial steps in this direction can be found in the work that we have surveyed. In [89], the outputs of intelligent decision units are combined to detect a DoS attack, while in [53], information about the data content of packets, and the past and present connection statistics are fused. In [67], information measured from different sensors is combined using Dempster-Shaefer Theory of evidence as we discussed previously. Thus, an efficient fusion on packet traffic in a learning framework may be the most appropriate approach to pursue.

2.4 Responding to an attack

Mahajan et al [18] provide a router architecture called Aggregate-based Congestion Control (ACC) agent to learn a congestion signature to identify a small number of aggregates responsible for congestion. During times of sustained high congestion, the ACC agent tries to find the congestion signature using the packet drop history (or the random samples) of the last K seconds. They propose to use destination-based identification algorithm. It first draws out a list of high-bandwidth addresses (32-bit) based on the drop history (or random sample). It then clusters these addresses into 24-bit pre-

fixes. For each of the clusters, it tries to find a longer prefix that still contains all the dropped packets because the longer prefix describes congestion signature better and will not punish a large category of traffic. Although this algorithm is simple to implement, it results in unfairness for the legitimate traffic to the congested destination. A more accurate and flexible identification algorithm is needed to maintain the fairness between friendly aggregates and misbehaving aggregates. Once the router knows the congestion signature, it then filters the bad traffic according to this signature. Furthermore, a *push-back* scheme is given to let the router ask its adjacent routers to filter the bad traffic at an earlier stage. *Pushback* [27] is a mechanism which adds functionality to the routers to detect and preferentially drop packets that probably belong to an attack. Upstream routers are also notified to drop such packets so that their resources are used to route only legitimate traffic. This is an effective scheme for various types of DDoS attacks, but its success heavily depends on the congestion signature, which is generally difficult to be sufficiently accurate.

One of the leading approaches in the proactive defence against DoS is Secure Overlay Services (SOS) [33], which is geared toward supporting Emergency Services or similar types of communication. The architecture is constructed using a combination of secure overlay tunnelling, routing via consistent hashing, and filtering. It reduces the probability of successful attacks by (i) performing intensive filtering near protected network edges, pushing the attack point perimeter into the core of the network, where high-speed routers can handle the volume of attack traffic, and (ii) introducing randomness and anonymity into the architecture, making it difficult for an attacker to target nodes along the path to a specific SOS-protected destination. The goal of SOS is to route only the "confirmed" users' traffic to the server and drop everything else. The clients are authenticated at the overlay entrance and they use the overlay network to reach the server. Only a small set of source addresses are approved to reach the server,

while all other traffic is heavily filtered out. The main advantage of SOS is that it can be applied over the existing IP infrastructure and can guarantee to some extent that in times of crisis a "confirmed" user will have access to the victim server. However, SOS does not work for public service, since the clients must be aware of the overlay network and use it to access the victim, and does not offer protection for the incoming links of the filtering router in front of the client, which can be quite easily overwhelmed by sheer volumes of DoS traffic.

The Mayday approach [61] generalises the earlier work on Secure Overlay Services [33]. Mayday combines overlay networks and lightweight packet filtering. The overlay nodes perform client authentication and protocol verification, and then relay the requests to a protected server. The server is protected from the outside by simple packet filtering rules.

Pricing techniques have also been suggested for protection against DoS attacks [25]. Dynamic Resource Pricing is a distributed gateway architecture and a payment protocol that imposes dynamically changing prices on both network, server, and information resources in order to push the cost of initiating service requests back to the requesting clients. This approach provides *service differentiation* and can therefore discriminate to some extent against adversarial behaviour.

Garg and Reddy [30] present a defence method based on regulation of the resource consumption. They built a Linux based prototype to show that traditional QoS rate-based regulation combined with their proposed window-based regulation of resources at the aggregate level at the network layer can help mitigate the impact of DOS attacks on end servers. Traffic regulation policies are enforced across traffic classes according to the resource usage of packets or flows and is done at an aggregate level and not individual flow level.

[31] presents VIPnet, a value-added network service to protect e-commerce and

other transaction-based sites from DDoS attacks. In this system, the ISPs carry the packets of the victim's "VIP" clients in a privileged class of service, protected from congestion, whether malicious or not. All non-VIP traffic is considered as low-priority and could be dropped in the case of an attack.

Quite similar is proactive server roaming [43], a novel idea that proposes that an active server changes its location within a pool of physical servers to defend itself against unpredictable or untraceable attacks. Again, only known legitimate users are explicitly informed by the roamer of its IP address and are able to follow the active server as it roams. The same authors later incorporated this idea of roaming to the concept of honeypots. Honeypots are nodes which attempt to appear as attractive targets for attackers, but provide no service to legitimate users; they only exist to capture and analyse attack traffic, and they do not receive any legitimate traffic. Since honeypots can be avoided by sophisticated attacks, in [75] they propose roaming honeypots, a mechanism that allows the locations of honeypots to be unpredictable, continuously changing, and disguised within a server pool. A (continuously changing) subset of the servers is active and providing service, while the rest of the server pool is idle and acting as honeypots.

Another well-known effort is DEFCOM [44], which suggests that the current paradigm of designing defence systems which operate in isolation should be abandoned. DEFCOM is a distributed framework that enables the exchange of information and services between existing defence nodes. Since, for example, attack detection is best done near the victim, while response is most effective and collateral damage is minimal at the source of the attack, each node should be specialised in a different aspect of the defence. Defence nodes must be able to communicate and must support at least the following messages: *Attack alerts* (generated from the Alert Generators to the rest of the network), *Rate-Limit requests* (the rate-limit requests should be sent upstream), *Resource requests* (each node should be able to issue a resource request to its downstream

neighbours), and *Traffic Classification* (classifier nodes must communicate with their downstream neighbours to ensure that the bulk of legitimate traffic will not be dropped). DEFCOM is the scientific child of some of the most important active DoS researchers, but is still in an early stage. For the time being, an obvious limitation seems to be the fact that an overlay node could be compromised and damage the operation of a large part of the system.

The COSSACK project [40] presents an architecture to explore the coordination of effective countermeasures using multicast, annotated topology information, and blind detection techniques. The cossack watchdogs are located at edge networks and are organised into a multicast tree. The client watchdog detects the attack and notifies all involved sources via multicast tree. The victim-end detection is very accurate and the source-end defence seems to effectively stop the attacks from protected networks. However, it is highly possible to inflict collateral damage if the attack traffic is similar to the legitimate traffic, and can not stop attacks from unprotected networks.

An interesting branch of DoS research is that of the low-rate attacks, nicknamed Shrew Attacks, in which a well orchestrated periodic burst of traffic can achieve disruption for which the attacker would normally need volumes of traffic. The shrew attacks, first presented by Kuzmanovic and Knightly [46], exploit the fixed minimum TCP RTO property. A few relevant papers have appeared since then, including [79] for detection of this type of attack at the edge routers and [82], which studies the effect that the buffer sizes may have.

2.5 Conclusions

The wide variety of defence approaches that were presented in this survey exhibit several similarities. *Detection* is performed by measuring features of the incoming traffic

and comparing them either to a normal profile in anomaly-based or to a DoS profile in signature-based. These features may be actual statistical features measured in real-time or simple observations acquired by actively testing the users of the network or asking them to prove their legitimacy. The same stands for the element of *classification*. Anomaly-based methods are less accurate, but apply to a broader range of DoS attacks, while signature-based methods are more dependable, but only for the attacks that they have been designed to detect. Since the two complement each other, in the work that we will present in the following chapters, we will use a combination of them.

In terms of *response*, the existing literature uses some sort of redirecting or dropping of the packets that the *detection* or *classification* methods found to be illegitimate. Redirecting the offending packets to a controlled part of the network, not only decreases the congestion in the victim network, but also provides with the opportunity to analyse the attack. However, such a safe and controlled environment usually does not exist and is difficult to build in a network. The family of response methods based on dropping packets is much more common. Such methods are simpler to implement and simpler to deploy, even in legacy systems, but they do provide opportunities for analysis and depending on the accuracy of the *detection* and *classification* methods that they are based on, may cause significant collateral damage. In our work we address both issues. We build a mathematical model to facilitate analysis of DoS and decrease the collateral damage by creating a method which drops only as many offending packets as necessary for the network to continue to operate.

Last, the evaluation of the impact of an attack and consequently of the success of a defence mechanism, is done by measuring either the average *delay* and *jitter* that the users experience, their *probability of successful connection*, or more often *goodput*, which is the rate of normal (non-DoS) packets that either have reached their destination or are ready to be forwarded to the next node in the route. The time-related QoS metrics

of delay and jitter can provide good indication of the impact of an attack, but are of secondary importance and can be misleading at times of extreme congestion, when many of the packets do not reach their destination. Also, although it is important for the users to know their probability to be connected and receive service, this metric does not provide any specific information on the impact of attack and defence on the nodes and links of the network. For this reason, in our work we will use the more common metric of goodput, with which all other metrics can be inferred, and which additionally offers a detailed view of the condition of the various components of the network.

3. SELF-AWARE NETWORKS AND DENIAL OF SERVICE

As has been discussed in [125], novel user-oriented networked systems will need to simultaneously exploit a variety of wired and wireless communication modalities to offer different levels of quality of service (QoS), including reliability and security to users; security could include protection against DoS attacks in particular. Within a single such user oriented network, different connections themselves may differ from each other with respect to QoS needs. Similarly, the communication infrastructure used by such a network will, in general, be shared among many different networks and users so that the resources available will fluctuate over time, both on the long and the short term. Such a user oriented network will not usually have precise information about the infrastructure it is using at any given instant of time, so that its knowledge should be acquired from on-line observations. Such user oriented networks should exploit self-adaptiveness to try to obtain the best possible QoS for all their connections. The QoS-driven Cognitive Packet Network (CPN) packet switching architecture [119], which is currently under development in the Intelligent Systems and Networks group in Imperial College, has been shown to meet the requirements of such self-awareness and is the system in which we develop and test our DoS protection proposals. Appendix B contains a detailed description of the CPN protocol and its underlying mathematical principles, and extensive performance evaluations can be found in [120]-[124]. The aspects of CPN that we have implemented or were already included, which qualify it as a Self-Aware Network and are directly involved in our DoS defence research are the following:

- ***Real-time QoS measurements*** in all nodes provide a clear view of the condition of the network, in terms of a variety of QoS metrics, such as incoming and outgoing traffic bitrates, average delays, packet losses, etc. As a result, areas of congestion are quickly identified.

- ***Dynamic routing***. A Self-Aware Network should always try to provide the best available routes to its users and change them dynamically when conditions change. This is achieved in CPN with the use of an additional type of packet, as explained in Section 3.1.

- ***Automatic traceback***. The path that a packet has followed within a CPN network is stored in its header and can be used to decide where to set up defences.

3.1 Networking with Cognitive Packets

CPN uses three types of packet: smart packets (SP) generated by the user for control and discovery of available paths, source routed dumb packets (DP) to carry the actual data, and acknowledgements (ACK) to bring back the discovered information. SPs associated with each flow constantly explore the network, and obtain routing decisions from network routers based on observed relevant QoS information. SPs store the identities of the nodes they visit, and collect local measurements such as delays and loss rates. At each CPN node, every new SP uses a local reinforcement learning algorithm based on measurements collected by previous SPs and ACKs, to elicit a decision from the node as to the next hop to travel to. When a SP reaches the destination node of the flow, an ACK packet is generated and returned to the source according to the opposite (destination to source) path traversed by the SP. When the ACK reaches the source, the forward route, which is the reverse of the route that it used, is stored for subsequent payload or dumb packets (DPs) which will be source-routed to the destination. Conventional IP packets

tunnel through CPN to seamlessly operate mixed IP and CPN networks.

In CPN the users are allowed to declare their QoS goals, such as *"I want to have the minimum possible delay for my VoIP conversation"*, or *"I need my packets to use the path with the minimum probability of packet loss"*, or *"I will start a multiplayer online gaming session only if there are available paths with packet loss less than 0.5% and delay less than 150ms"*. In a similar way a Self-Aware Network, such as one based on CPN, should provide the following example security option for its users: *"The resource I need is often a medium-rate DoS attack target and I want to use the path that will be less affected in case of such an attack''*, or *"In case of a low-rate DoS attack I want my delay to remain the same and a packet loss increase of less than 5%"*. A more detailed description of the Cognitive Packet Network can be found in Appendix B.

3.2 Generic DoS protection in a Self-Aware Network

To visualise the problem of Denial of Service see Fig. 3.1. The victim computer V receives packets from both attackers, such as A, and normal users, such as N. Due to the large volume of attack packets and the limited resources of the network, some normal packets may be lost and not arrive at the destination V. In the example of Fig. 3.1, some of the normal packets will be lost if node i can not process and forward quickly enough the total of normal and attack packets that go through it.

Our ultimate goal is to design a mechanism which will ensure that normal packets will arrive at their destination even when the resources are limited. In this chapter we analyse a generic method of response against a DoS attack, given a degree of differentiation between normal and attack packets.

We will consider a generic DDoS defence scheme that is based on the following principles and assumptions:

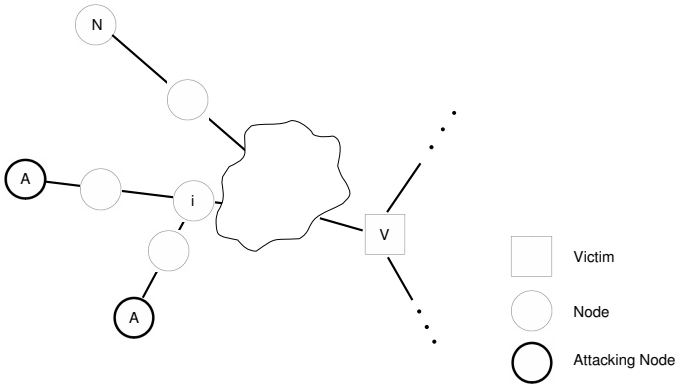

Fig. 3.1: Both normal and attack packets are sent towards the destination V. Due to the volume of the attack and the resulting congestion, some of the normal ones may be lost before they reach the destination.

- The node which is targeted by a DDoS attack (the victim node) has the ability to detect or to be informed about the attack, based on either a local or a distributed detection scheme. All nodes upstream from the victim up to the source(s) of the attack, will also be informed of the ongoing threat.

- The victim node and the informed nodes will react by *dropping packets* which are probabilistically considered to be part of the attack.

- The attack itself can produce buffer overflows and saturation of network resources such as CPU capacity, due to the inability of the nodes or routers to handle the resulting heavy packet traffic.

- The detection scheme is always imperfect, so that both false alarms and detection failures are possible. Imperfections are possible both with regard to the detection

of the attack as a whole, and the identification of the packets that belong to this attack. Thus, for any packet that flows in the network we need to consider a probability of correct identification as being an attacking packet, and a probability of false alarm, which means that some attacking packets will be missed and some non-attacking packets may be incorrectly dropped.

As explained in Section 2.5, we will evaluate the impact of the attack and the effectiveness of the defence on a given network by measuring the *goodput*, which is the rate of normal packets that either have reached their destination, or are ready to be forwarded to the next node in their route. In the same sense we will refer to *badput* as the rate of DoS packets that arrive at their destination (the victim node), or are ready to be forwarded to the next node in their route.

3.3 A mathematical model of Denial of Service protection

We have developed the following mathematical model to analyse the impact of DDoS protection on overall network performance based on the probabilities of detection and of false alarm. We assume an abstract network model in which DoS packets are identified with a certain probability and dropped, while some other valid (i.e. non-DoS) packets are mistakenly identified as DoS flows and also dropped.

The packet network consists of N nodes $1,\ ...i,\ ..., N$. At any node i, the arriving traffic is the aggregate of several normal (valid or non-DoS) flows, and possibly of several invalid (DoS) flows, where $\mathbf{n} = (n_1, n_2,\ \ldots\ , n_j,\ \ldots\ , n_{L(\mathbf{n})})$ and $\mathbf{d} = (d_1, d_2,\ \ldots\ , d_j,\ \ldots\ , d_{L(\mathbf{d})})$ are the paths followed by a normal and a DoS flow respectively. $L(\mathbf{n})$ is the path length of flow $\mathbf{n}$, and j is used to denote the position of a generic node inside the path. The total traffic rate λ_i arriving externally to node i is composed

of two parts:

$$\lambda_i = \sum_{\mathbf{n}} \lambda^n_{i,\mathbf{n}} + \sum_{\mathbf{d}} \lambda^d_{i,\mathbf{d}}, \tag{3.1}$$

where $\lambda^n_{i,\mathbf{n}}$ is the "normal" or benign incoming traffic rate which belongs to normal flow n, and $\lambda^d_{i,\mathbf{d}}$ is the arrival rate of DoS packets belonging to flow $\mathbf{d}$.

Any traffic that node i takes to be DoS traffic is dropped at the entrance of the node. Thus, a fraction $f_{i,\mathbf{n}}$ of normal traffic (the probability of false alarms) and a fraction of DoS traffic $d_{i,\mathbf{d}}$ (the probability of correct detection) will be dropped as it arrives to the node. If the node's DoS detection mechanism were perfect we would have $f_{i,\mathbf{n}} = 0$ and $d_{i,\mathbf{d}} = 1$. Once a packet is admitted into a node, it is queued and then forwarded based on its destination address. We model each node by a single server queue with service time s_i representing both the time it takes to process the packet in the node and the actual transmission time. The traffic intensity parameter ρ_i is then:

$$\rho_i = s_i(\sum_{\mathbf{n}} I^n_{i,\mathbf{n}}(1 - f_{i,\mathbf{n}}) + \sum_{\mathbf{d}} I^d_{i,\mathbf{d}}(1 - d_{i,\mathbf{d}})), \tag{3.2}$$

where for node i, $I^n_{i,\mathbf{n}}$ is the arriving traffic rate of the normal flow $\mathbf{n}$, and $I^d_{i,\mathbf{d}}$ is the arriving traffic rate of a DoS flow $\mathbf{d}$.

Since DoS attacks will tend to overwhelm the node's packet processing and transmission capability, packets will be lost by the node with probability L_i. We could use different formulas to relate traffic intensity to this probability based on modelling congestion of various type that may occur at the node. We assume Poisson arrivals for the incoming traffic and that the loss probability is due to buffer overflow. Having modelled each node as a M/M/1 queue with finite buffer capacity, we essentially adopt the relevant assumptions of Poisson arrivals, one exponential (Poisson) server, and FIFO queue of limited capacity. Thus, we use the following loss probability expression [118]:

$$L_i = \rho_i^{B_i} \frac{1-\rho_i}{1-\rho_i^{B_i+1}}, \tag{3.3}$$

where B_i is the size of the buffer at node i.

Since any traffic that is correctly or mistakenly thought to be DoS traffic is dropped at the input of the node, and since the traffic which effectively enters a node has been filtered in this manner and all droppings are independent, the traffic equations for the system become:

$$\begin{aligned} I^n_{n_j,\mathbf{n}} &= \lambda^n_{n_1,\mathbf{n}} \prod_{l=0}^{j-1} ((1-L_{n_l})(1-f_{n_l,\mathbf{n}})) \\ I^d_{d_j,\mathbf{d}} &= \lambda^d_{d_1,\mathbf{d}} \prod_{l=0}^{j-1} ((1-L_{d_l})(1-d_{d_l,\mathbf{d}})), \end{aligned} \tag{3.4}$$

where we set $L_{n_0} = L_{d_0} = f_{n_0,\mathbf{n}} = d_{d_0,\mathbf{d}} = 0$. These equations express the fact that, at any node, an incoming packet may be dropped due to correct or mistaken identification as a DoS packet, or due to buffer overflow because the node is overloaded, while all packets which enter the buffer queue and are not dropped are eventually routed to the next node on their path or absorbed at the current node if it is itself the destination node. Equations (3.4) relate input rates to the nodes to the buffer overflow or loss probabilities, while ρ_i and consequently the buffer overflow probabilities L_i in turn depend on the traffic rates.

The solution of (3.4) is obtained numerically via a non-linear iteration:

- Step 0

$$I_{i,\mathbf{n}}^{n,(k=0)} = \lambda_{i,\mathbf{n}}^{n} \tag{3.5}$$

$$I_{i,\mathbf{d}}^{d,(k=0)} = \lambda_{i,\mathbf{d}}^{d} \tag{3.6}$$

$$I_{i}^{(k=0)} = \sum_{\mathbf{n}} I_{i,\mathbf{n}}^{n,(k=0)} + \sum_{\mathbf{d}} I_{i,\mathbf{d}}^{d,(k=0)} \tag{3.7}$$

- Step $k > 0$

$$\rho_i^{(k)} = s_i I_i^{(k-1)} \tag{3.8}$$

$$L_i^{(k)} = \rho_i^{B_i,(k)} \frac{1-\rho_i^{(k)}}{1-\rho_i^{B_i+1,(k)}} \tag{3.9}$$

$$I_{n_j,\mathbf{n}}^{n,(k)} = \lambda_{n_1,\mathbf{n}}^{n} \prod_{l=0}^{j-1} ((1-L_{n_l}^{(k)})(1-f_{n_l,\mathbf{n}})) \tag{3.10}$$

$$I_{d_j,\mathbf{d}}^{d,(k)} = \lambda_{d_1,\mathbf{d}}^{d} \prod_{l=0}^{j-1} ((1-L_{d_l}^{(k)})(1-d_{d_l,\mathbf{d}})) \tag{3.11}$$

$$I_{i}^{(k)} = \sum_{\mathbf{n}} I_{i,\mathbf{n}}^{n,(k)} + \sum_{\mathbf{d}} I_{i,\mathbf{d}}^{d,(k)} \tag{3.12}$$

After the algorithm converges we obtain the goodput G_i at node i using:

$$G_i = \sum_{\mathbf{n}} I_{i,\mathbf{n}}^{n} (1-L_i)(1-f_{i,\mathbf{n}}) \tag{3.13}$$

Fig. 3.2 is a graphical representation of the model for a single flow. As explained in this section, the goodput after the last node of the specific normal flow's path will be the initial rate $\lambda_{0,\mathbf{n}}^{n}$ multiplied by the probability that no packet was lost due to buffer overflow and no packet was lost due to incorrect identification at each node of the path. The model is non-linear because the loss probabilities due to buffer overflows depend

also on the other flows of the network.

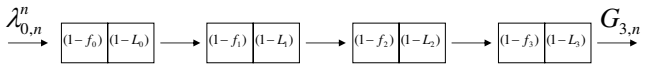

Fig. 3.2: Graphical representation of the mathematical model for a single flow.

3.3.1 A numerical example for a small network

To illustrate the use of our mathematical model we evaluate the impact of a DoS attack on the network topology shown in Fig. 3.3. In order to emphasise the impact of the attack, we have chosen a topology which provides a limited number of available paths to the target server, and consequently brings the normal traffic at an important disadvantage when competing for the available resources. We have added a second server, to evaluate the impact of attack and defence on the operation of the rest of the network.

In the example of Fig. 3.3, server (0) is being attacked by three DoS flows of 2500 packets/s each, entering the network through nodes 3, 4, and 5, to represent a distributed attack. From now on we will refer to server (0) as the victim server or simply the victim and to server (13) as the alternate server. Both servers receive normal packets by all valid clients at a rate of 100-500 packets/s per client (100 corresponds to very low and 500 to very high load level). We evaluate the impact of both the attack and the defence mechanism by considering the goodput or rate of "valid" packets which make it safely to their destination nodes. In a practical application of the mathematical model, where we would be interested in the exact numerical values during a specific attack and for a specific distribution of normal traffic, the detection probabilities would be estimated through repeated applications of the same identification mechanism. Our

goal is, however, to acquire some initial insights in the dynamics of such a simple DDoS attack, and so we vary the load levels and the probabilities of both correct detection and false alarm. We choose an average service time per packet of $s_i = 0.4ms$, which is a reasonable average value measured with experiments in our testbed. For the buffer size we choose the low value of $B_i = 50$ packets to achieve fast computations. We have repeated the numerical computations with various values for the buffer size, ranging from 20 to $10,000$ packets at each node, and found no significant differences in the numerical results.

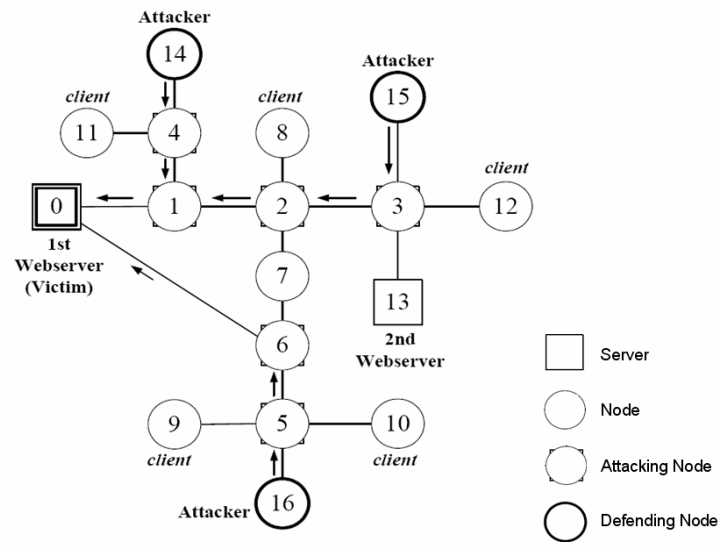

Fig. 3.3: The attacker has compromised nodes 14 - 16 and orders them to send attack flows towards the victim web-server (0).

In accordance to the assumptions of Section 3.2, the packets that are considered for dropping by the generic DoS protection mechanism are only those that are directed to the victim server (0). So, all flows directed to the alternate server (13) are subject to packet losses only due to buffer overflows and not due to the protection mechanism. Also, again

as explained in the assumptions, node (7) should not participate in the defence, since it does not belong to the attack paths and would not be informed of the attack.

To avoid confusion we refer to the sets of probabilities as simply (Pf, Pd) instead of $(f_{i,\mathbf{n}}, d_{i,\mathbf{d}})$ implying that although the mathematical model allows for more flexibility, in this scenario the two are the same for all participating nodes and flows in each case. The results presented in Fig. 3.4 show that moderate attack can cause an undefended network's performance to degrade dramatically. This is because of the concentrated nature of DoS traffic. Even when the total bitrate of DoS packets is not extremely higher than the total bitrate of normal packets, DoS traffic creates significant congestion at specific points which act as bottlenecks in the network. For example, at high load level, when under attack, the victim server operates at less than 22% of its ideal capacity. Applying a simplistic defence in which there is no mechanism to distinguish between DoS and normal traffic, and we simply drop half of the packets which are destined to the victim (Fig. 3.4 and 3.5, naïve defence) not only does not help, but causes enough collateral damage to decrease the goodput at the victim even further.

The results do improve though from the alternate server's perspective (Fig. 3.6), since the normal packets towards the alternate server do not undergo any identification process and can not be dropped due to the defence, which is the main disadvantage of the naïve defence. So, if the victim server were not a crucial part of the infrastructure, but only a decoy or a "honeypot" whose role is to attract the attacking traffic, then that very lightweight naïve defence of dropping half the packets towards the victim, would prove useful. The other two arbitrarily chosen sets of detection probabilities represent more sophisticated defence approaches. In Fig. 3.4 and 3.6 the two hypothetical sophisticated approaches show significant improvement in the goodput at both servers for all load levels.

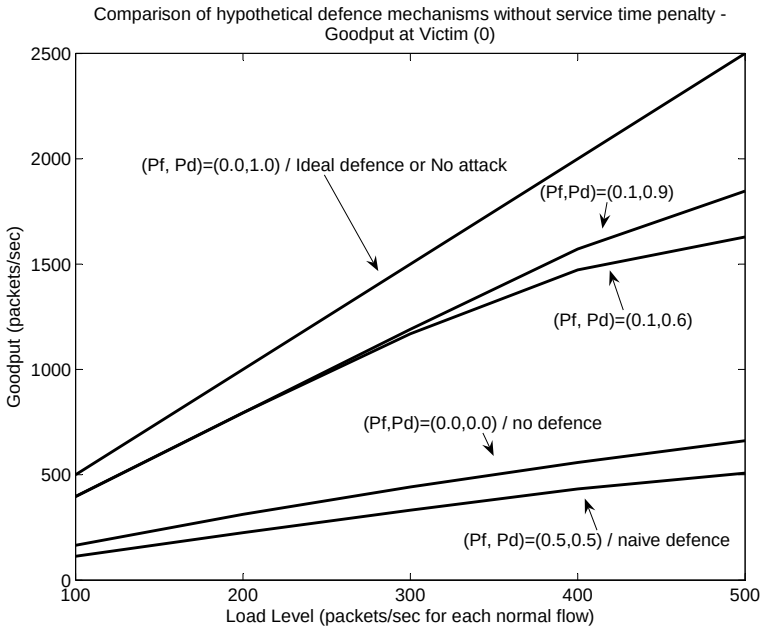

Fig. 3.4: Comparison of hypothetical defence mechanisms - Goodput at the Victim (0) *Vs.* Load Level

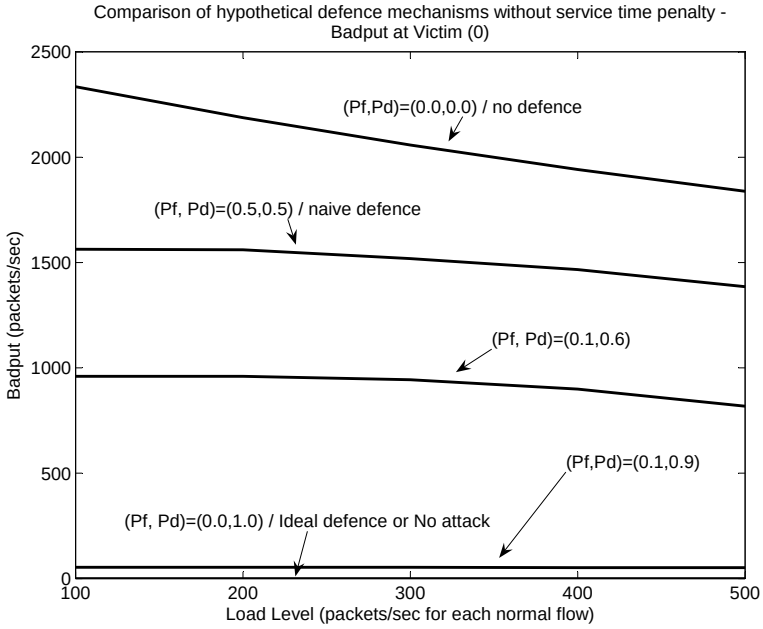

Fig. 3.5: Comparison of hypothetical defence mechanisms - Badput at the Victim (0) *Vs.* Load Level

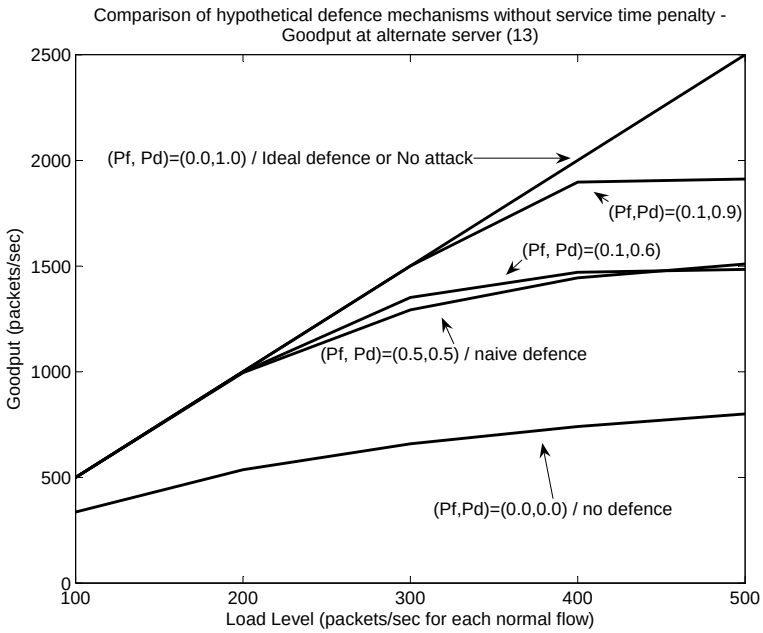

Fig. 3.6: Comparison of hypothetical defence mechanisms - Goodput at the alternate server (13) *Vs.* Load Level

Until now we have used the same average service time s_i for all nodes and all defence mechanisms. In reality we would expect the more sophisticated and more accurate defence mechanisms to introduce some penalty in the performance of the network, as it is unrealistic to assume that the additional task of defending assigned to specific nodes will not affect their efficiency at processing and forwarding packets. The service time penalty is a positive value which represents the increase of the average service time per packet for a specific defence task. To illustrate how the introduction of this parameter will affect the mathematical results, we choose for it the following values for each hypothetical defence mechanism: 5% for the naïve one, 35% for the $(P_f, P_d) = (0.1, 0.6)$ one, and 75% for the $(P_f, P_d) = (0.1, 0.9)$, which would be expected to be the most demanding. For instance, with the specific values, the two non-trivial defence mechanisms had similar performance, while the naïve one was still clearly inadequate to protect server (0) (Fig. 3.7 and 3.8). However, again in the case of server (0) being just a decoy, the naïve defence is clearly the best choice for the protection of the alternate server (13) (Fig. 3.9).

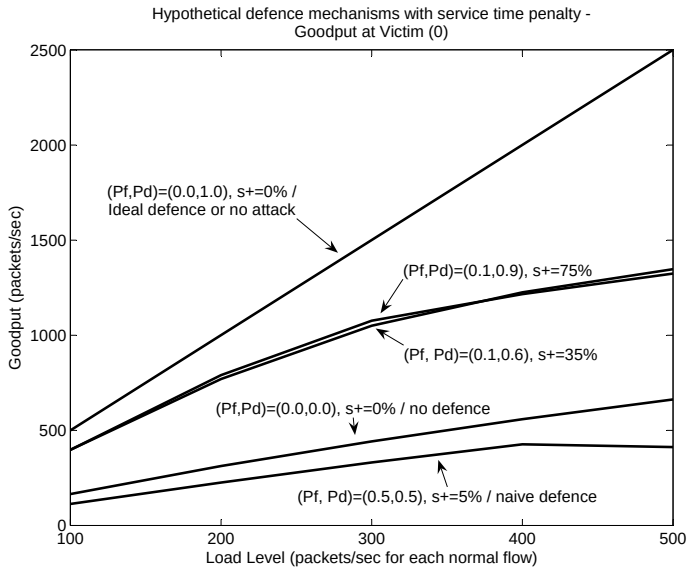

Fig. 3.7: Comparison of hypothetical defence mechanisms with service time penalty - Goodput at the Victim (0) *Vs.* Load Level

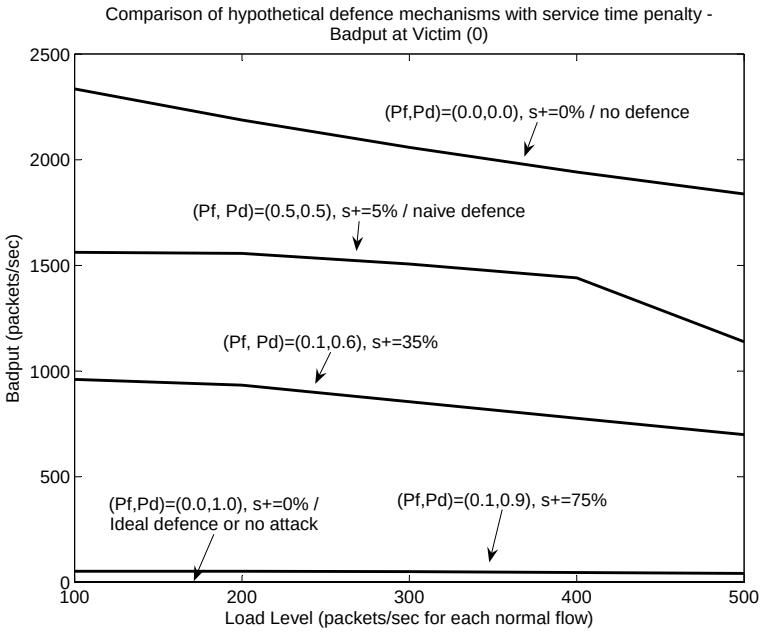

Fig. 3.8: Comparison of hypothetical defence mechanisms with service time penalty - Badput at the Victim (0) *Vs.* Load Level

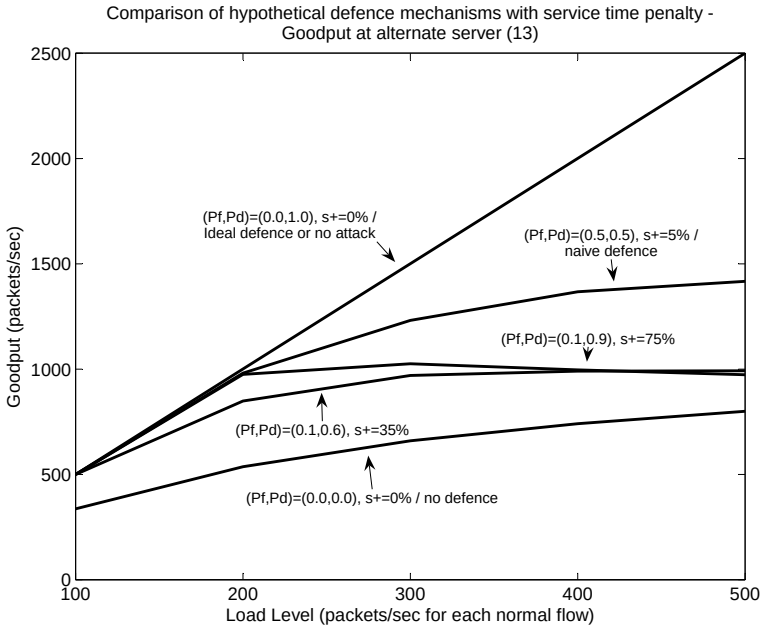

Fig. 3.9: Comparison of hypothetical defence mechanisms with service time penalty - Goodput at the alternate server (13) *Vs.* Load Level

Similar conclusions can be acquired by looking at the numerical results when instead of the load level (rate of each normal flow) we vary the rate of each attack flow (Fig. 3.10 - 3.15). In these examples, the normal rate is kept constant at 400 packets/s, while the attack rate ranges from 1000 to 5000 packets/s. As observed in these graphs, and as expected intuitively, the higher the attack rate the more useful the most accurate defence mechanisms.

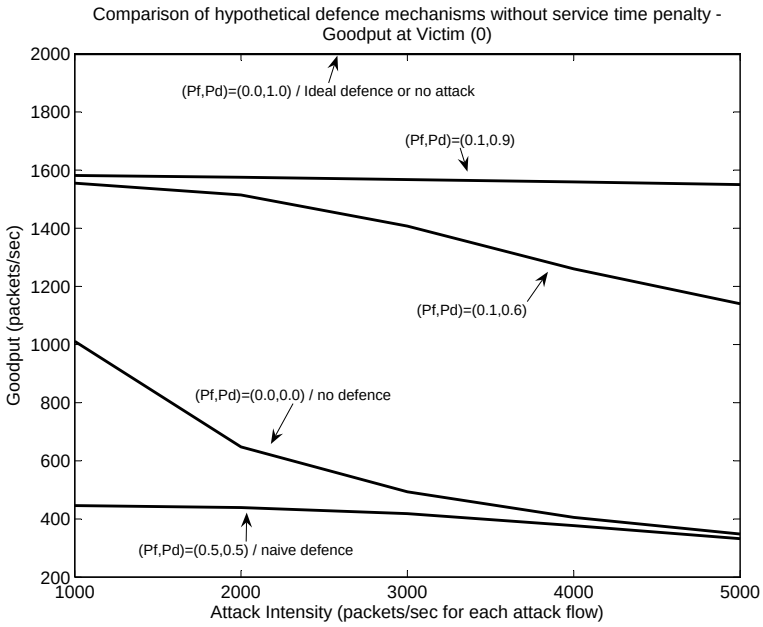

Fig. 3.10: Comparison of hypothetical defence mechanisms - Goodput at the Victim (0) *Vs.* Attack Intensity

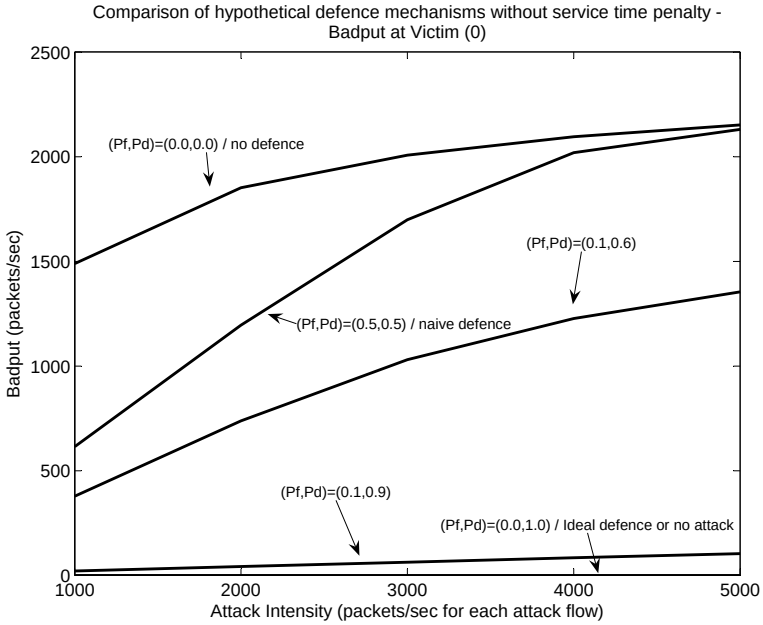

Fig. 3.11: Comparison of hypothetical defence mechanisms - Badput at the Victim (0) *Vs.* Attack Intensity

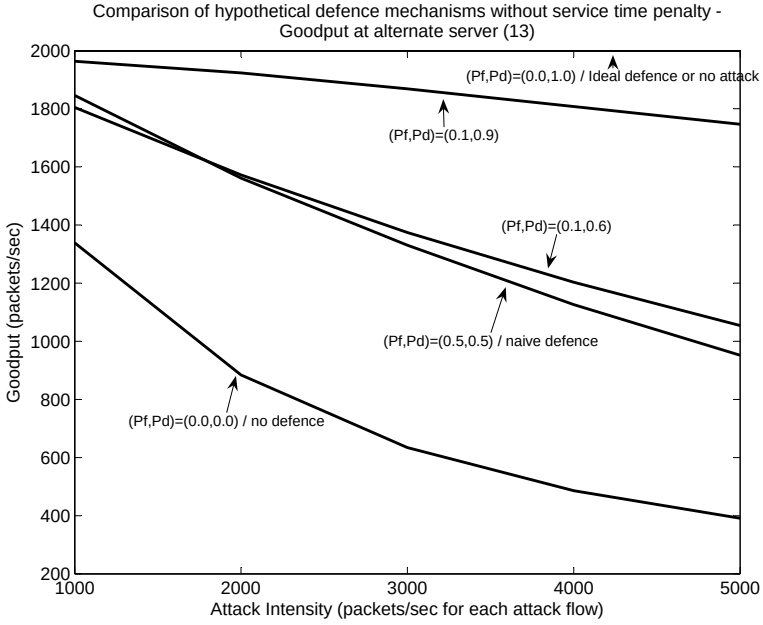

Fig. 3.12: Comparison of hypothetical defence mechanisms - Goodput at the alternate server (13) *Vs.* Attack Intensity

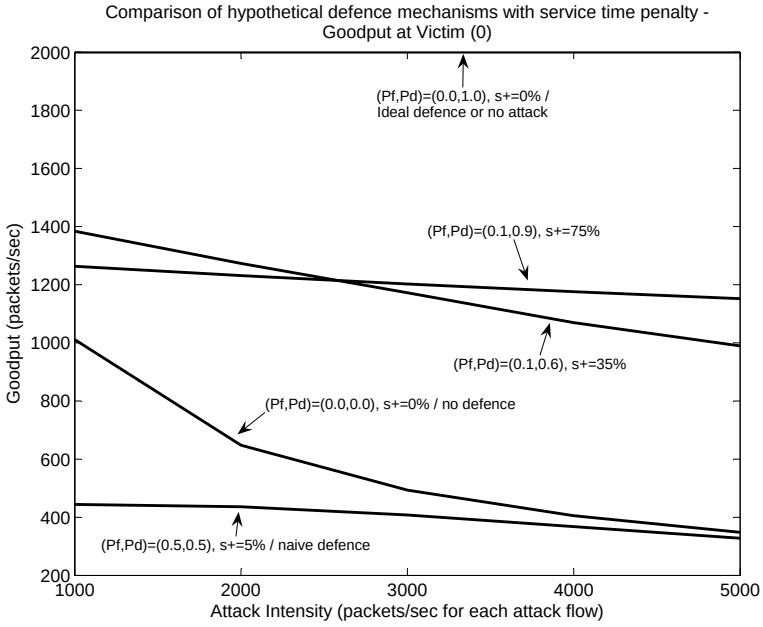

Fig. 3.13: Comparison of hypothetical defence mechanisms with service time penalty - Goodput at the Victim (0) *Vs.* Attack Intensity

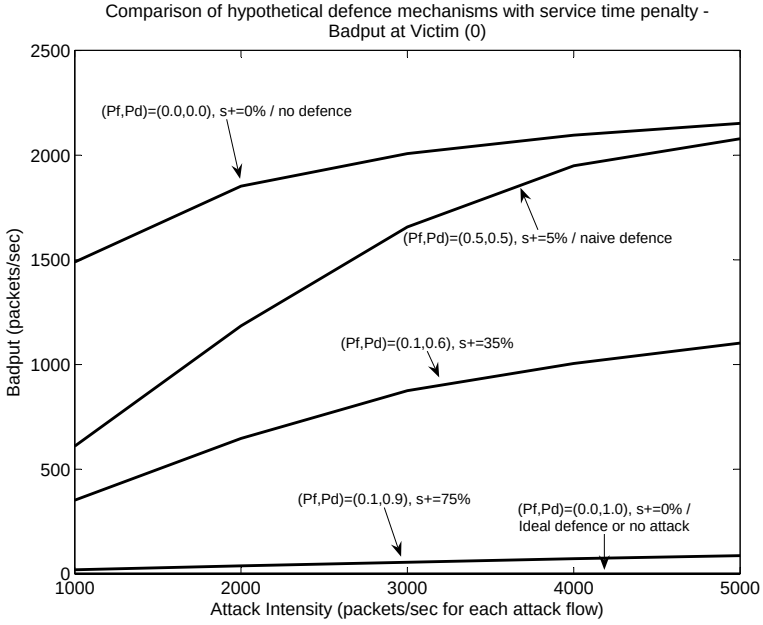

Fig. 3.14: Comparison of hypothetical defence mechanisms with service time penalty - Badput at the Victim (0) *Vs.* Attack Intensity

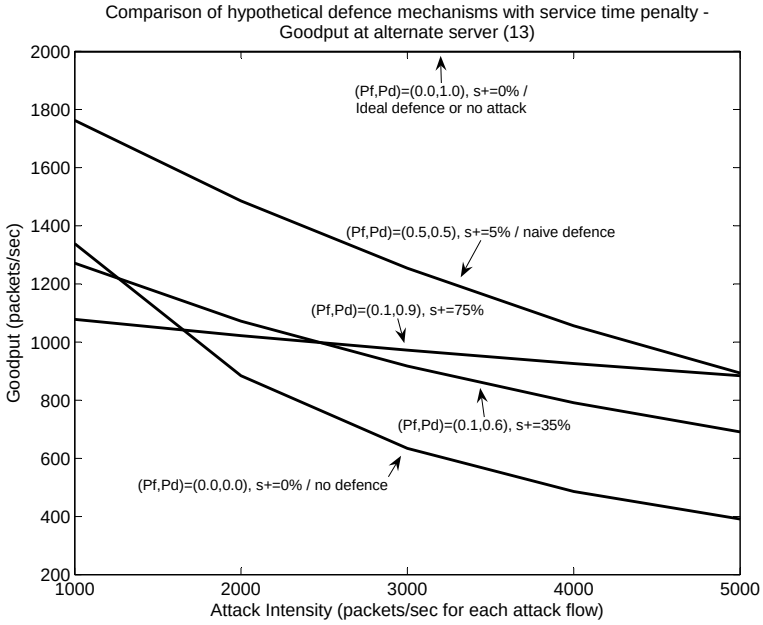

Fig. 3.15: Comparison of hypothetical defence mechanisms with service time penalty - Goodput at the alternate server (13) *Vs.* Attack Intensity

Fig. 3.16 - 3.19 compare the impact of the attack and the defence mechanisms on the performance of some of the other nodes of the network, which might not necessarily be of interest for the operator of the network or the victim server, but would be important for the operators of the specific nodes. An example observation for the specific scenario is that node 7 is largely affected by the attack, even though it neither receives attack traffic nor participates in the defence. This is wholly due to the overall congestion created by the attack flows and not due to the introduction of the defence mechanism, since packets going through 7 are not going towards the victim and therefore are not subject to defence.

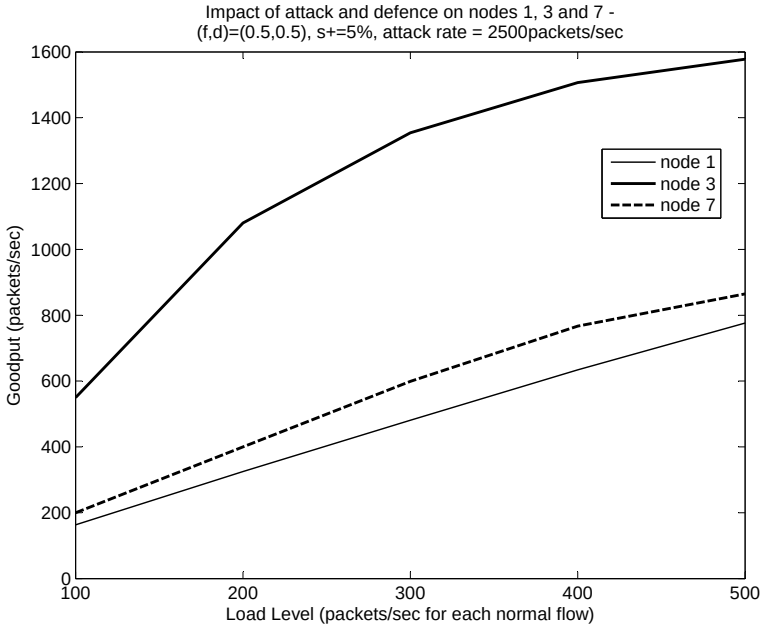

Fig. 3.16: Impact of attack and defence on intermediary nodes for naïve defence - Goodput *Vs.* Load Level

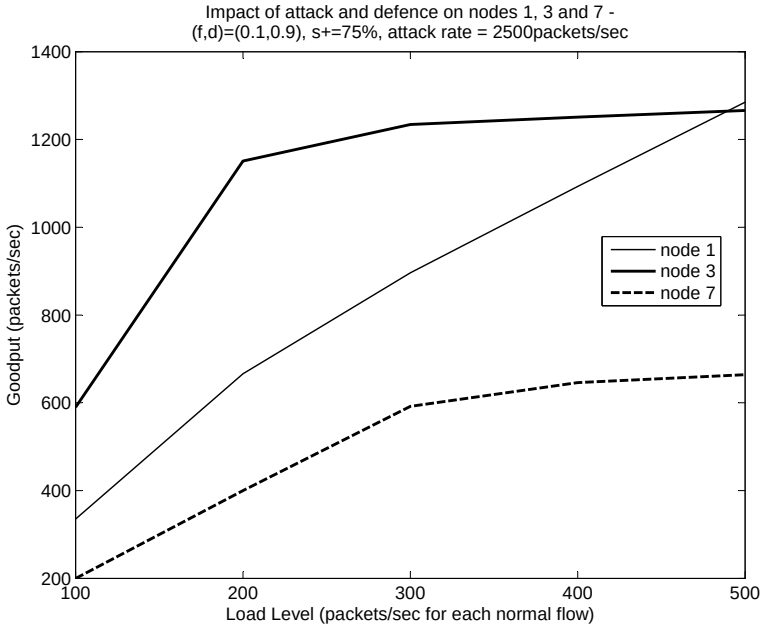

Fig. 3.17: Impact of attack and defence on intermediary nodes for sophisticated defence - Goodput *Vs.* Load Level

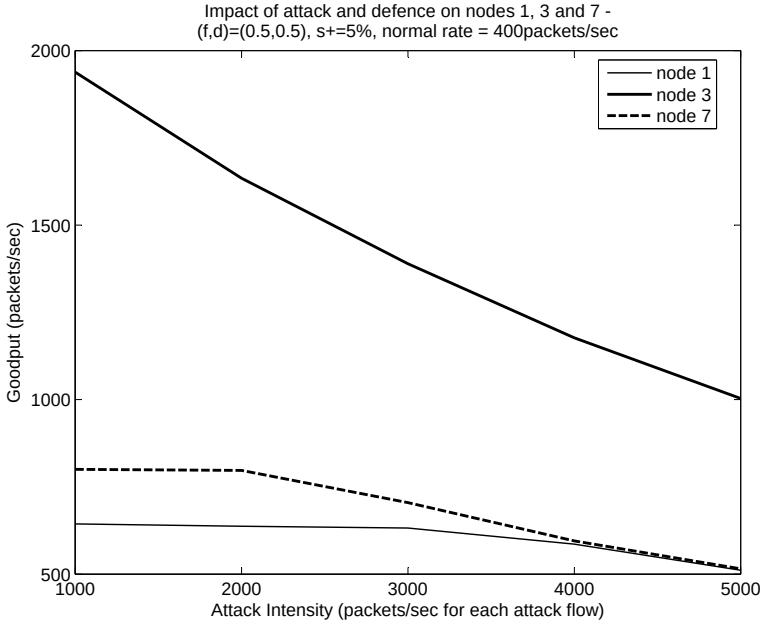

Fig. 3.18: Impact of attack and defence on intermediary nodes for naïve defence - Goodput *Vs.* Attack Intensity

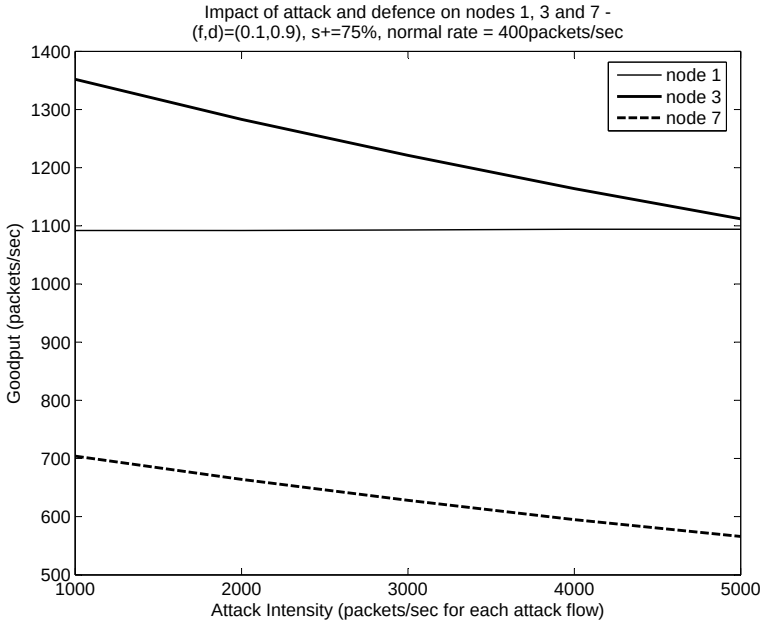

Fig. 3.19: Impact of attack and defence on intermediary nodes for sophisticated defence - Goodput *Vs.* Attack Intensity

3.4 Simulating generic DoS defence

In this section, we pursue the preceding discussion using simulations in a small network as presented in Section 3.3.1. We carried out the simulations using the NS-2 network simulator [109]. For the sake of comparison we first left the network undefended and then applied a simple defence mechanism with packet drops based on fixed probabilities along the lines of our earlier discussion. The links are full duplex and have a bandwidth of 100 Mbits/s. For all nodes the service time is $0.4ms$ and the buffer size again 50 packets. The queues are simulated with the DropTail mechanism, which implements simple FIFO scheduling and drop-on-overflow buffer management. Again, we vary the constant normal traffic rate from 100 to 500 packets/s, while the attack rate is 2500 packets/s for each of the three attack flows. We chose an average size of 500 bytes for both normal and DoS packets (more details about average packet sizes can be found in Appendix E). Fig. 3.20 is a sequence of screenshots from NS-2, with the first one showing the network before we introduce the traffic, the second one after normal traffic is added, and the third one after the attack has begun. For a choice of detection probabilities $(P_f, P_d) = (0.1, 0.6)$, Fig. 3.21 shows the impact of the attack and defence on nodes 0 and 13 for the duration of the simulation. The transitions in time shown in Fig. 3.21 are representative of the whole set of experiments and will be omitted for the rest of these simulation results. Instead, we will focus on the steady state condition, which the network reaches quickly since we use UDP traffic of constant bitrate. The results are very close to those obtained with the mathematical model for the same scenario. Fig. 3.22 and 3.23 show the goodput measurements, and Fig. 3.24 and 3.25 show the equivalent packet losses for each case.

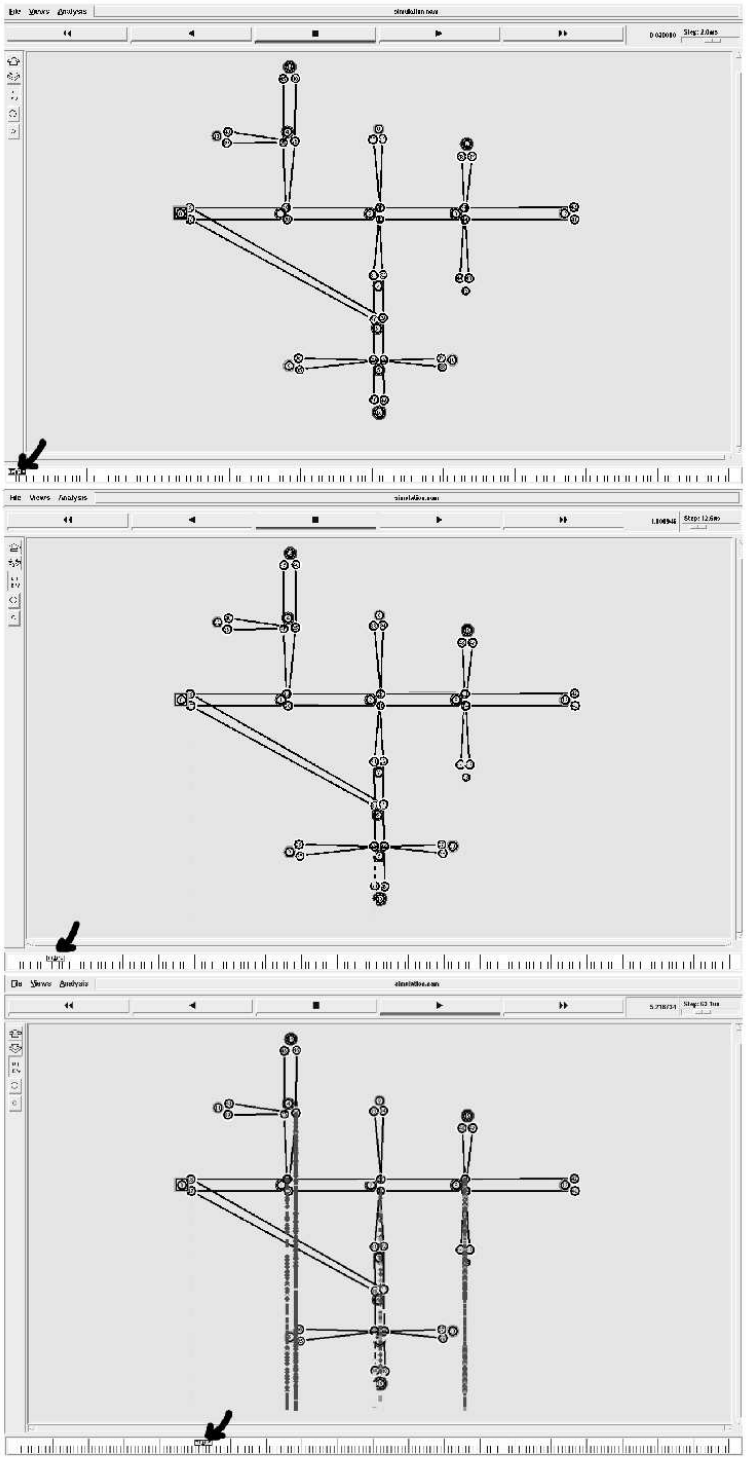

Fig. 3.20: Sequence of screenshots from NS-2 simulations. a) The network before we introduce the traffic. b) After normal traffic is added. c) After the attack has begun. Notice in the third screenshot the buffer overflows that the attack causes at most of the nodes. The arrow shows the time point and the long trails of packets are those dropped due to buffer overflows.

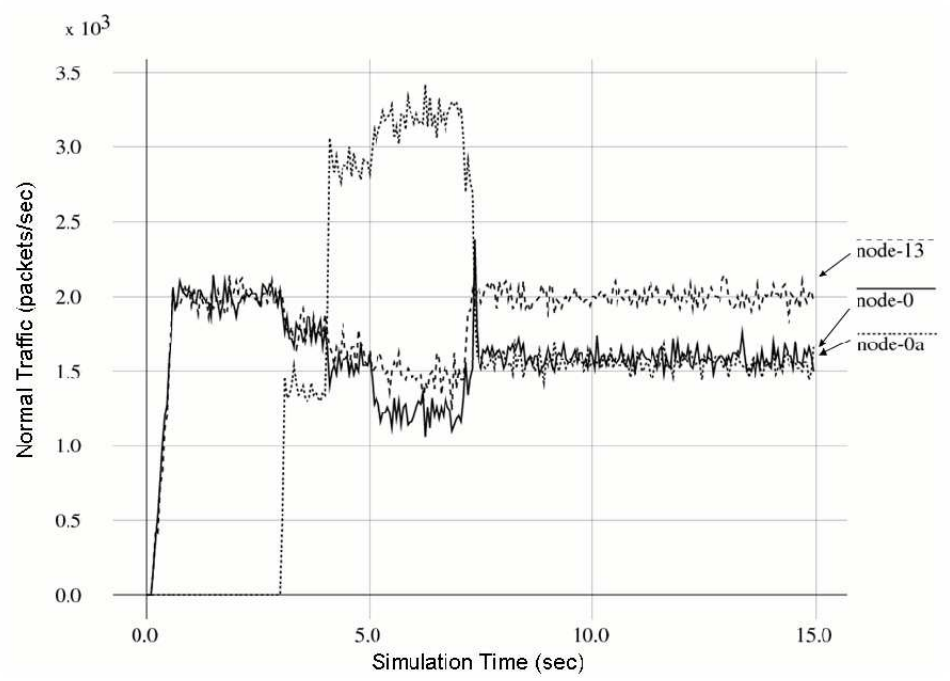

Fig. 3.21: The normal traffic (packets/s) which makes it safely to the victim (node-0) and the alternate server (node-13), and the DoS traffic that the victim receives (node-0a), *Vs.* the simulation time (sec). We start the attack at 3s and increase its intensity up to the 6th second. At 7s we start applying a $(P_f, P_d) = (0.1, 0.6)$ defence. Steady state is reached very quickly.

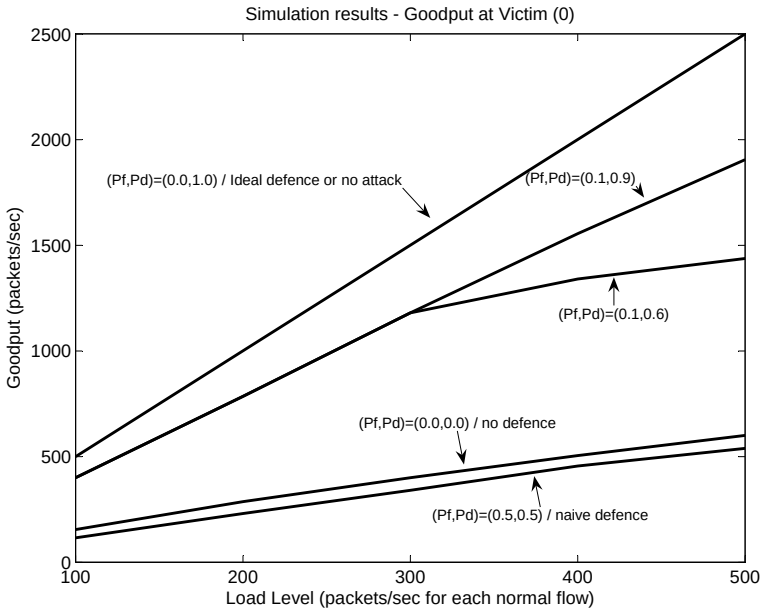

Fig. 3.22: NS2 Simulation results - Goodput at Victim (0)

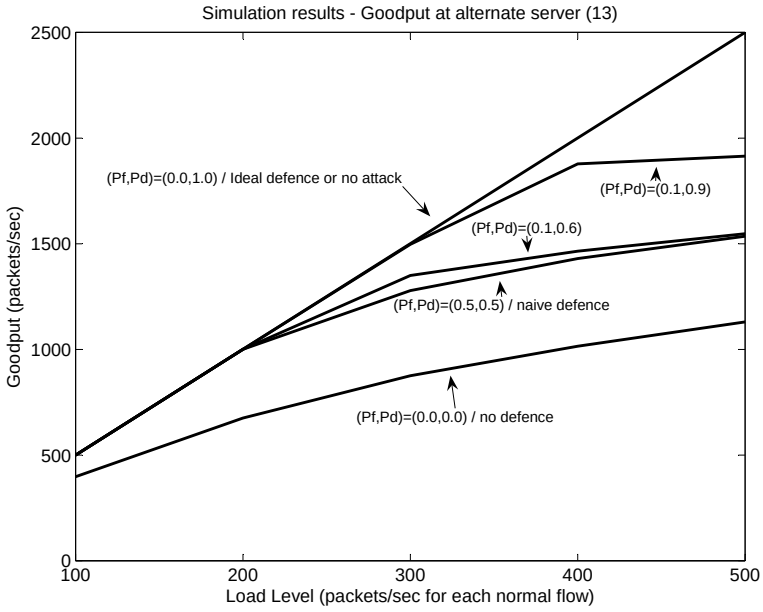

Fig. 3.23: NS2 Simulation results - Goodput at alternate server (13)

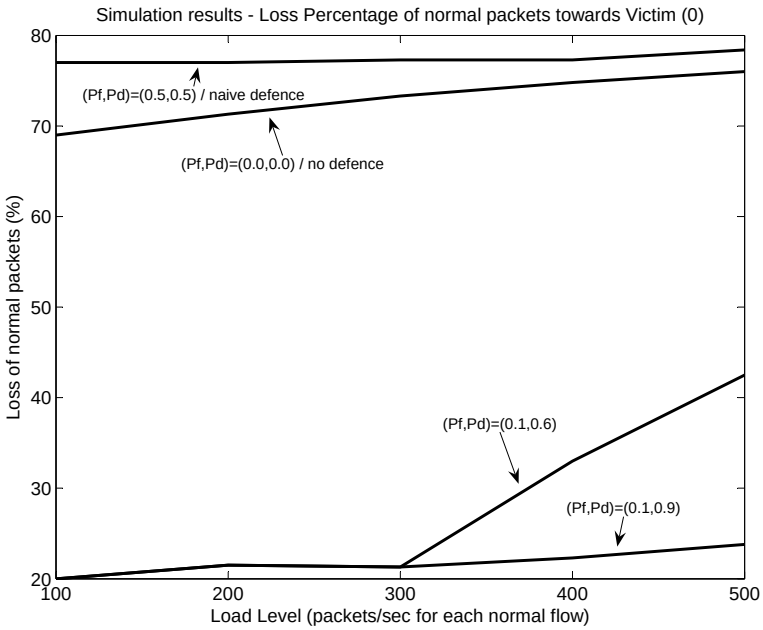

Fig. 3.24: NS2 Simulation results - Loss Percentage of normal packets towards Victim (0)

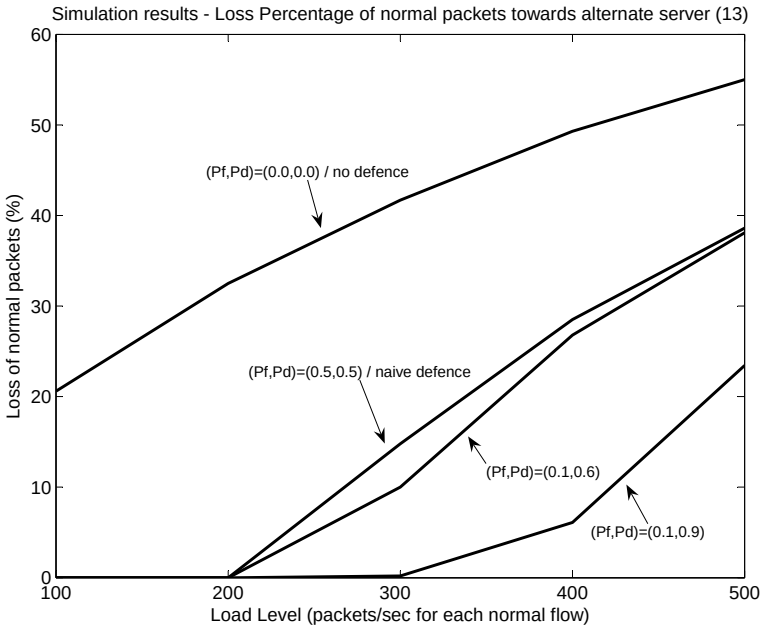

Fig. 3.25: NS2 Simulation results - Loss Percentage of normal packets towards alternate server (13)

3.5 Experimental Approach using CPN for Defence against DDoS Attacks

In this section we present an implementation in a SAN environment of the generic defence defence scheme as described in section 3.2 and evaluated in sections 3.3.1 and 3.4. The CPN-based DDoS defence technique presented here exploits the ability of CPN to trace traffic going both down- and up-stream thanks to SPs and ACK packets, so as to facilitate the stifling of malicious traffic. When a CPN node detects a DoS attack, it will use the ACKs to ask all intermediate nodes upstream to start dropping packets of the incoming flow. In a complete defence architecture these nodes would monitor all of the packets that traverse it and drop those that exhibit unwanted characteristics and are identified as parts of a DoS attack. The classification mechanism is omitted in this implementation, as our first goal is to evaluate the response mechanism given a (P_f, P_d) set of detection probabilities.

We implemented this approach on a CPN testbed consisting of 2.4GHz Pentium-4 PCs, which, for the sake of comparison with the mathematical and simulation analyses, was configured as shown in Fig. 3.3. The routes in the network were manually configured to remain static during the entire experiment, and Smart packets (SPs) were disabled from CPN to prevent the discovery and use of alternative routes. The various defence mechanisms were emulated through the use of configurable drop probabilities. We used a delay-based queueing mechanism on each outbound interface to achieve the desired value of $0.4ms$ for the average service time, so that the results will be comparable with the mathematical and simulation analyses of the previous sections. Again, as in Sections 3.4 and 3.3.1, the scenario consists of 3 attack flows of 2500 packets/s each towards the victim server (0), 5 normal flows towards the victim server (0), and 5 normal flows towards the alternate server (13), 100-500 packets/s each.

The measurements from the experiments are very similar to those obtained for the same scenario with our NS-2 simulations of Section 3.4 and the predictions of the mathematical model of Section 3.3.1. Representative comparative results showing the Loss percentages in the two servers for various load levels are summarised in Fig. 3.26 - 3.29.

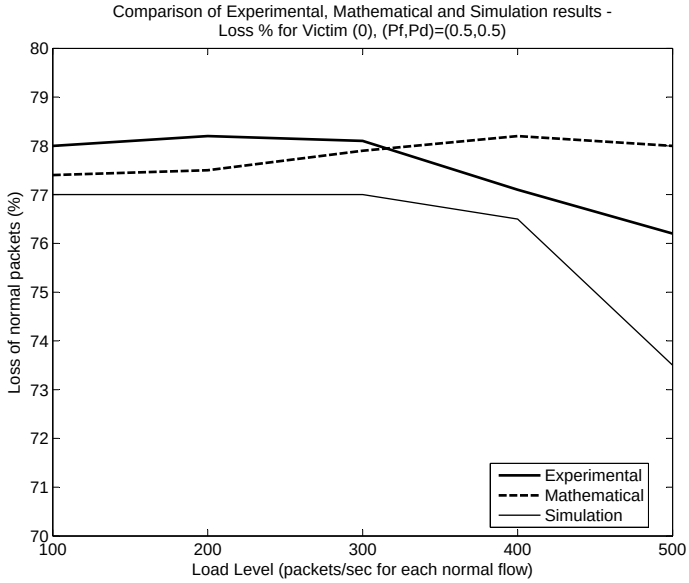

Fig. 3.26: Loss percentage comparison of the three methods for ($P_f = 0.5, P_d = 0.5$) at the victim (0)

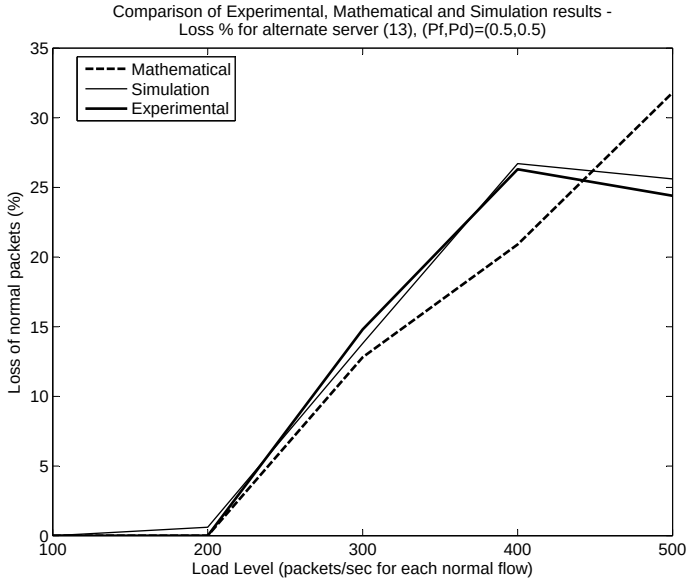

Fig. 3.27: Loss percentage comparison of the three methods for ($P_f = 0.5, P_d = 0.5$) at the alternate server (13)

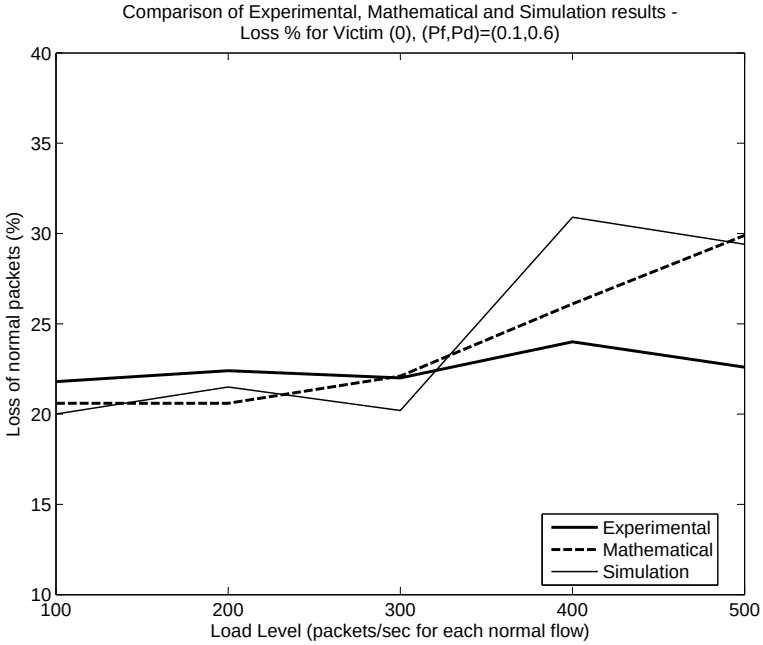

Fig. 3.28: Loss percentage comparison of the three methods for ($P_f = 0.1, P_d = 0.6$) at the victim (0)

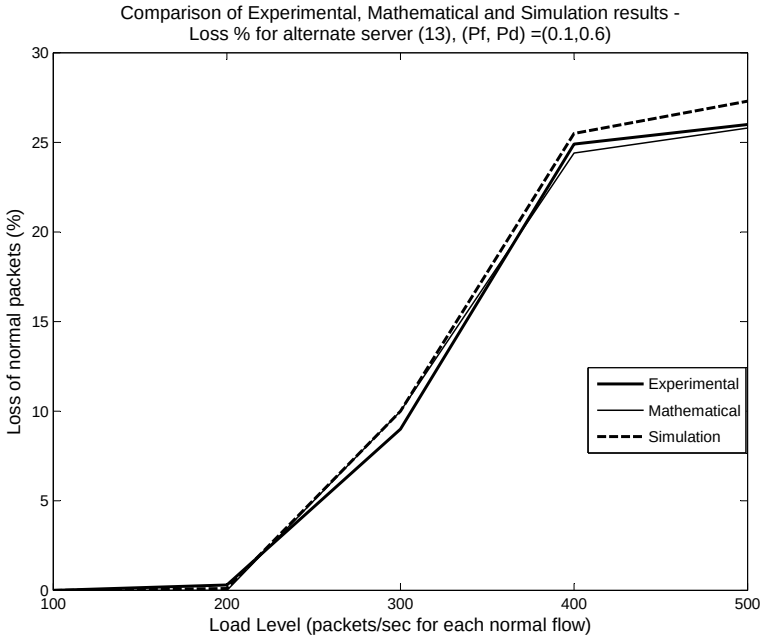

Fig. 3.29: Loss percentage comparison of the three methods for ($P_f = 0.1, P_d = 0.6$) at the alternate server (13)

3.6 Defence in the changing routes of a Self-Aware Network

A significant advantage of self-aware networks is the ability to dynamically change the routes of the flows to find the best possible according to criteria set by the users. This ability, however, makes it extremely difficult to defend against a Denial of Service attack, since the nodes that the attack flows go through are constantly changing. One solution is to install a defence mechanism in all nodes of the network. However, this would mean that a large number of defenders would not be used by the attack traffic long enough to justify the false alarms and the significant processing overhead due to the operation of the defence mechanism.

A better solution is to use as defenders only the nodes that are used by the attack traffic with certainty or with high probability. These include the nodes at the perimeter through which these flows arrive and the direct neighbours of the victim. We tested this approach in a large topology, which recreates the SwitchLAN 46-nodes backbone network [1] (Fig. 3.31).

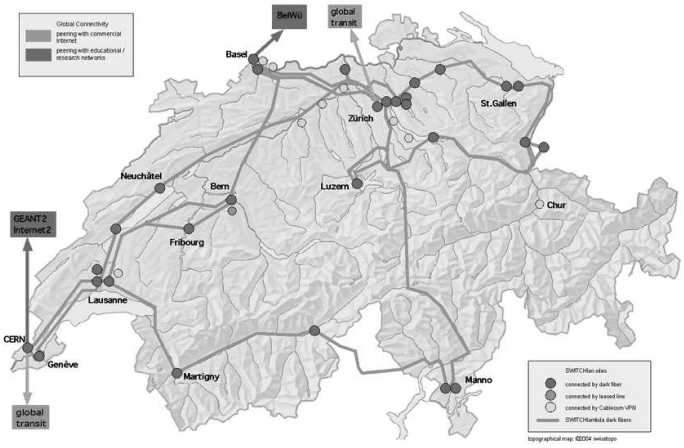

Fig. 3.30: Geographical topology of the SWITCHlan network

[1] The SwitchLAN network provides service in Switzerland to all universities, two federal institutes of technology and the major research institutes.

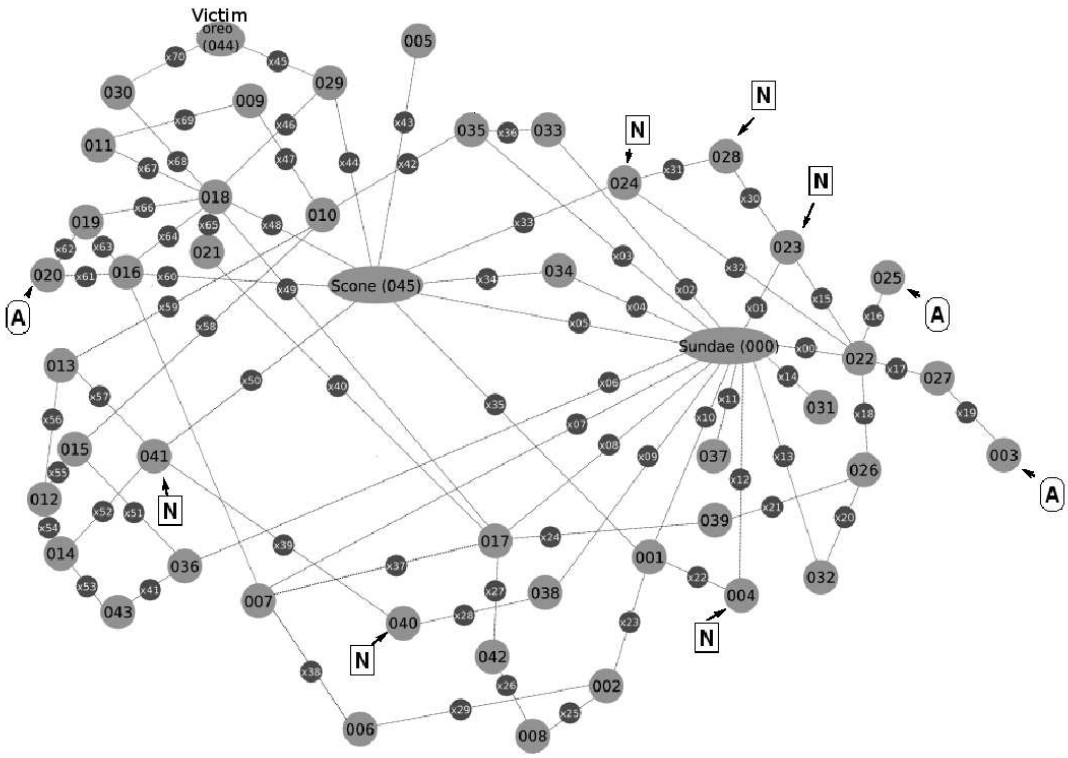

Fig. 3.31: Simple DoS scenario on the SWITCHlan topology, with 5 normal clients and 3 attackers

In the following set of experiments, 3 attack flows and 5 normal flows are sent from border nodes to node Oreo(044). The routing protocol is the CPN featuring changing routes and using delay as the QoS criterion. We measure the average goodput for the last 20s of the experiments in order to ensure that it is in steady state. Fig. 3.32 is an indicative example of the change of goodput against the time in the experiment.

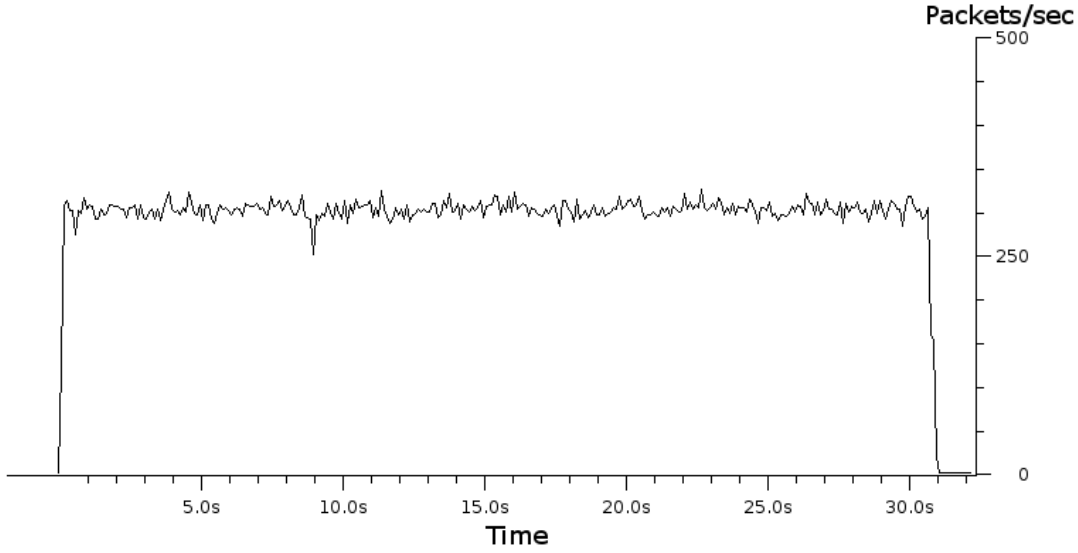

Fig. 3.32: The network reaches its steady state practically immediately. This indicative example is from one of the experiment runs for (f=0.5,d=0.5) and load level 200 packets/s per normal flow, and shows the goodput at the Victim.

Leaving the network undefended results in a 70-75% decrease of the goodput at the victim node, which improves according to the detection probabilities (P_f, P_d) of the defence mechanism. For the sake of comparison we perform the mathematical analysis considering as defenders only those we used in the experiment, the neighbours of the victim and the entrance nodes of the attack, and assuming that no packets are lost in the intermediary nodes. The comparisons of Fig. 3.33 show that the mathematical model overestimates the performance of the defence architecture. This is expected, since despite the dynamic QoS-oriented routing of CPN, it is not guaranteed that congestion will be avoided at all times and all parts of the network.

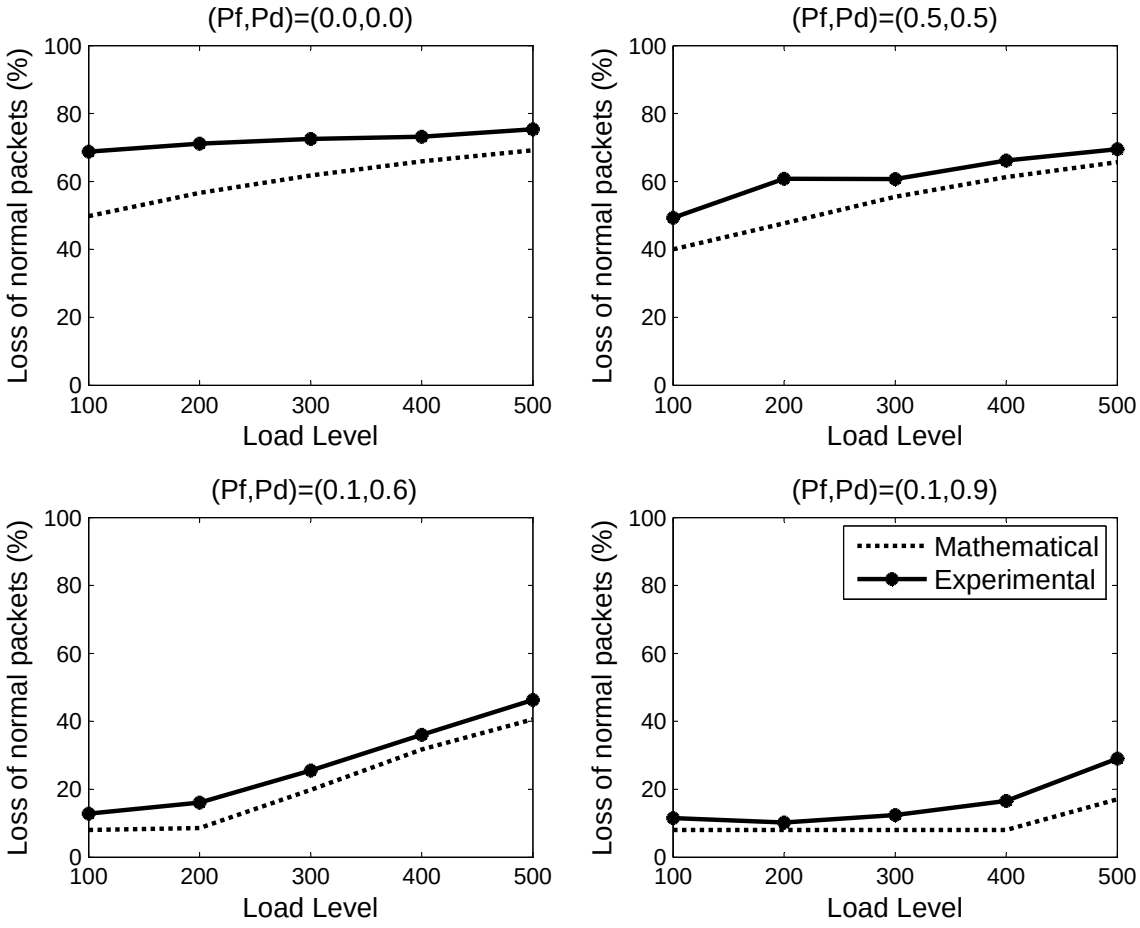

Fig. 3.33: Loss percentage comparison of the mathematical results and the experiments on a large CPN network.

Still, the defence is much more successful than with the static routes of conventional networks. This is confirmed when we repeat the same experiments using static routes according to the shortest-path rule and use as defenders all nodes upstream from the attack sources to the victim. When no defence is applied (Fig. 3.34) using static or dynamic routing does not make a noticeable difference, since normal and DoS flows compete for the limited available resources on the same terms in both cases.

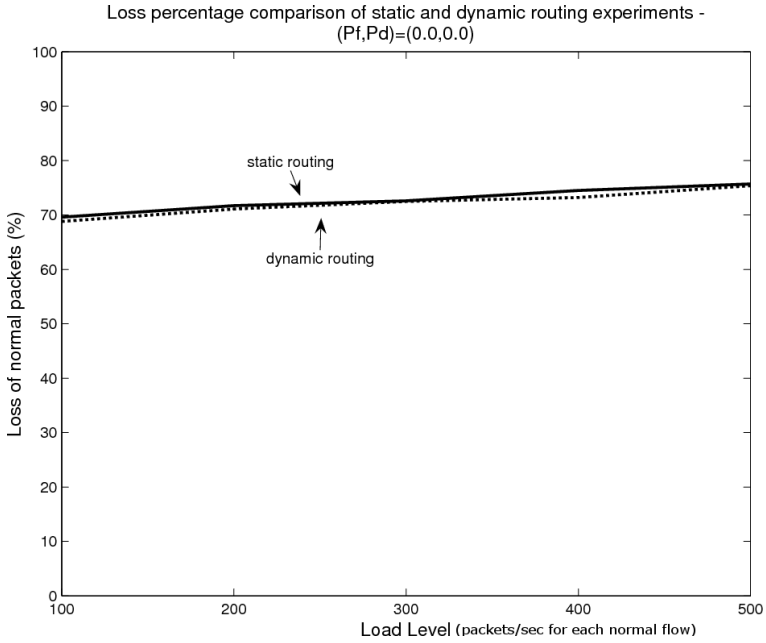

Fig. 3.34: Comparison of static and dynamic routing without defence.

When defence is applied, however, the dynamic routing significantly improves the chances of the normal traffic to find uncongested routes because of the decrease in the overall (both normal and DoS) load of the network. This results in a decrease in the loss of normal packets. As seen in Fig. 3.35, the lower the load level, the higher the decrease of the packet loss for normal traffic. When the load level is high enough to cause extreme congestion on its own (500 packets/s each), then dynamic routing has no advantage over static.

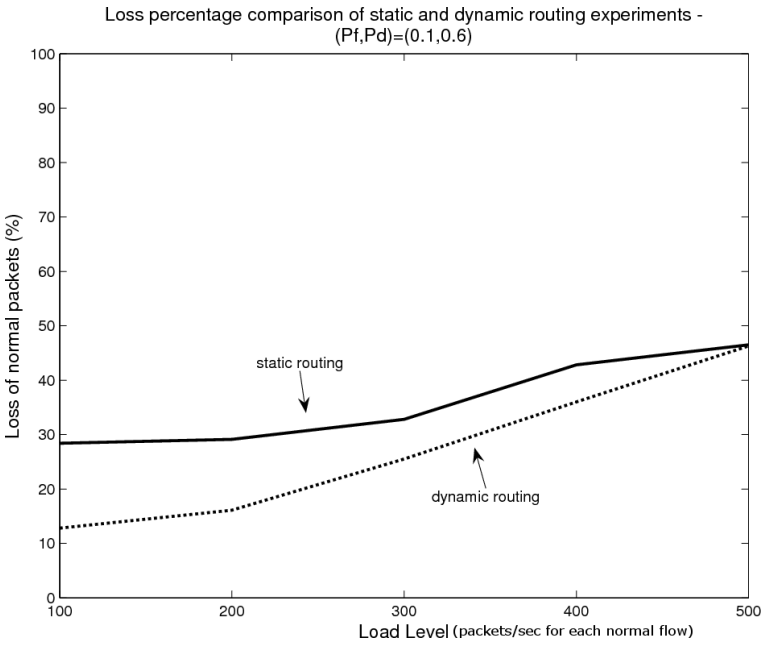

Fig. 3.35: Comparison of static and dynamic routing for $(P_f, P_d) = (0.1, 0.6)$.

3.7 Evaluating the Autonomic CPN-based Defence

It would be interesting to see how the CPN-based defence presented in Section 3.5 would perform in terms of the actual service the user of the network experiences. Thus, in a new set of experiments we tested the CPN-based defence approach for a network application (video streaming) with specific QoS requirements in terms of bandwidth.

The application which is used to illustrate the attack and defence mechanism is a UDP based MPEG1 video stream that is transferred from node 3 (S) to node 30 (D) in the CPN testbed shown in Fig. 3.36a. The video received at node 30 (D) in the un-attacked network is uncorrupted, as shown in Fig. 3.36b. Then a DDoS attack which is meant to overwhelm node 30 (D), by saturating it with incoming packets, is launched from nodes 1 (A) and 2 (A). When there is no defence, the attack corrupts the video stream, making it unintelligible as shown in Fig. 3.36c.

We then apply a simple defence algorithm, in which node 30 (D) detects a high rate of traffic on certain inbound paths and uses the CPN traceback ability to order that packets be dropped upstream on the paths, and also drops traffic coming from those paths into node 30 (D) itself. This alleviates the impact of the attack, resulting in the clear video stream shown in Fig. 3.36d. The experiment shows that if CPN or a similar network routing mechanism is used, when a node is able, even imperfectly, to recognise that it is being attacked, a sensitive real-time data stream can be protected. Of course, the protection is not instantaneous and requires some time to become effective after the attack is detected. In the specific scenario, the time for recovery was about 1 sec.

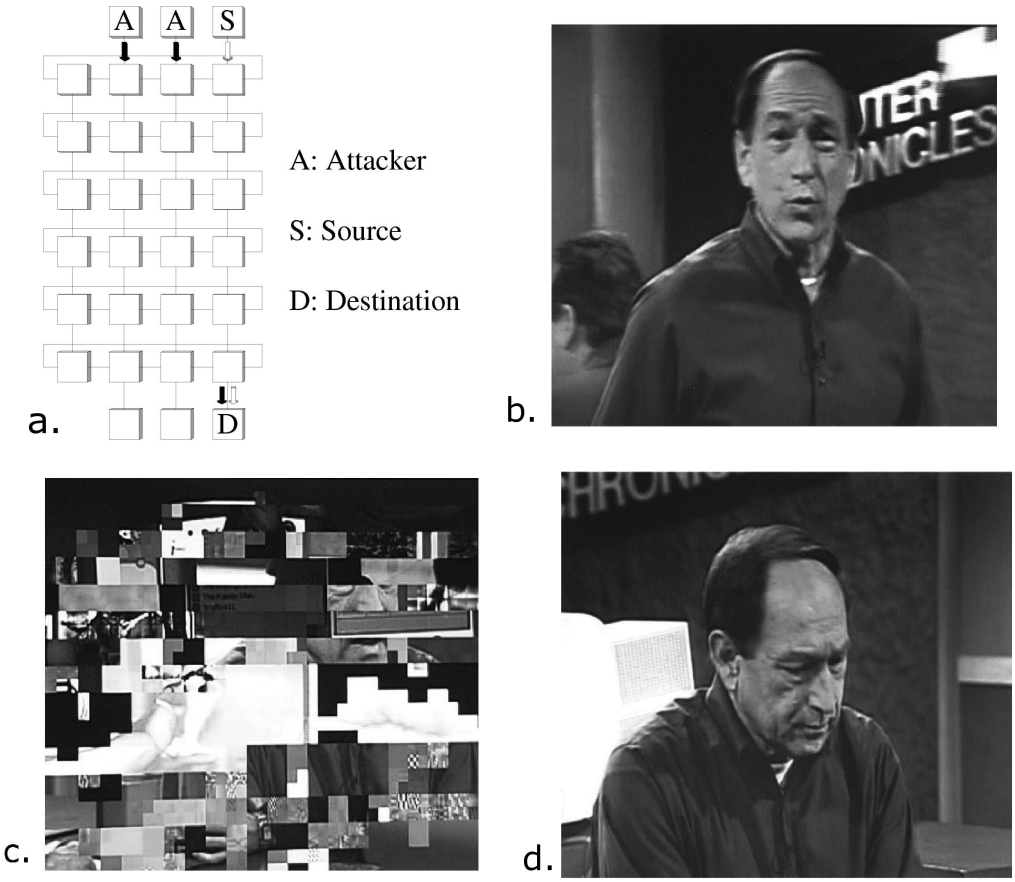

Fig. 3.36: Experimental evaluation of our defence scheme. a) The CPN testbed used to conduct the experiments. b) A frame of the video in the un-attacked network. c) The corruption in the video stream due to the attack. d) The restored video sequence after defence is enabled.

3.8 Conclusions

3.8.1 Summary of the chapter

Dropping packets which are identified as probably illegitimate is a generic type of response, used by the majority of defence mechanisms to mimimise the effects of a Denial of Service attack. In this chapter we provided a mathematical model which analyses the effectiveness of this most significant family of defence approaches and estimates the impact of the attack on the network. We validated its results by comparing them with simulations and experiments. We also implemented a defence approach which exploits the unique advantages of automatic traceback and dynamically changing routes that a Self-Aware Network ought to feature. To demonstrate the success of such a defence approach we used it within the CPN protocol to retain the image quality of a sensitive video streaming application, which when left undefended dramatically deteriorated.

3.8.2 Weaknesses and suggestions

In most of our experiments, we use UDP traffic of constant bitrate, with which the network quickly arrives at steady state. If we were using TCP traffic, then things would be different in terms of detection probabilities and loss rates. In the presence of a DoS attack, the increased packet losses would trigger a decrease in the sending rate of the normal users, and consequently a decrease in the goodput at the victim. If, as expected, the attack traffic did not decrease its sending rate, then the decrease of normal traffic would bring the latter in an even worse situation at competing for the network resources. On the other hand, the fact that attack traffic would not decrease its sending rate would make it very easy to distinguish from the normal traffic.

With the response approach described in this section, packets are still dropped if identified as illegitimate even when there are enough network resources to accommo-

date them. If these packets are correctly identified then this is not a problem. However, if normal packets are incorrectly identified as illegitimate and are dropped, although keeping them would not have harmed the network, then this is unneeded collateral damage. Also, by simply either dropping packets or letting them through, we do not take into account the level of certainty with which the classification mechanism identified them. These are some of the issues that we tackle in the next chapter, where we propose a second type of defence against Denial of Service attacks.

4. DOS DEFENCE BASED ON COMBINATION OF PRIORITISATION AND RATE-LIMITING

In Chapter 3, we saw that the performance of a defence scheme based on dropping DoS packets depends heavily on the accuracy of the detection/classification methods. Here we present an improved, more specific version of this generic scheme which attempts to decrease the impact of false alarms on the normal packet flows. In the defence approach that we propose in this chapter, instead of simply dropping traffic we suggest the use of traffic prioritisation and rate-limiting as the basis of the response mechanism.

Prioritisation. Traffic prioritisation is a queueing mechanism which serves the packets in strict order of importance (priority). Packets seen as more important (higher priority) receive service always before the ones of the next (lower) priority and so on.

Rate-limiting. Rate-limiting [1] is the process of allowing traffic only up to a maximum limit to pass. It essentially means that traffic in excess of a set limit is dropped to reduce congestion.

[1] Rate-limiting is often also referred to as throttling in the literature.

4.1 General description of the prioritisation and rate-limiting mechanism

When there is a detected attack in progress, each informed node contributes to the defence by examining every incoming packet for deviations from the normal behaviour. Packets undergo a collection of anomaly-based validity tests which may differ for each type of traffic, as seen in Section 2.2. Nodes which take part in the defence will prioritise traffic with priority levels which are related to the results of the tests. Each time a packet fails or succeeds in a validity test, its priority level will change accordingly. In our implementation, if a packet succeeds its priority level increases by one, and it decreases by one if it fails. Additionally, the upstream routers are instructed to throttle down (rate-limit) their traffic directed towards the victim node to a level which it can handle.

This two-fold protection framework ensures that packets with higher probability of being both valid and harmless, are offered preferential service. Packets which have been marginally classified as invalid may now receive service if there is available bandwidth so as to minimise the collateral damage inflicted by false detection. Packets which have been identified as being harmful are either delayed by being assigned low priority, or dropped. Various simplifications of this scheme, based for instance on grouping all traffic that has been identified as being invalid, can also be considered.

Rate-limiting as a means of controlling the attack traffic has been extensively investigated. In [28], a general framework to identify and control high-bandwidth aggregates in a network using max-min fair rate limits recursively from the victim to the upstream routers, was proposed, and in [83] the concept of router rate-limiting with a *level-k max-min fairness* algorithm has been suggested to regulate the experienced server load to below its design limit, so that it remains operational during a DDoS attack. Their

analysis with a mathematical model, simulation and experiments, proved the usefulness of smart rate-limiting during denial of service attacks. However, smartly orchestrated rate-limiting cannot be used on its own, as a stand-alone defence mechanism, since without classification, there can be extensive collateral damage, especially at periods of unusually increased legitimate traffic.

We will test this kind of protection without a sophisticated detection mechanism, but with a fairly crude mechanism to detect the existence of an attack. If a node (either a recipient or an intermediary node) receives packets towards a destination (1) at a rate higher than the current *rate threshold* ($packets/s$ or $bytes/s$) and (2) with a rate increase which is higher than the current *increase threshold* ($packets/s^2$ or $bytes/s^2$), then it announces the existence of an ongoing distributed Denial of Service attack and sends this information to all upstream nodes and to the victim. From then on, the protection mechanism is set into motion along the informed path. In case of disagreement, it is the alleged victim node's responsibility to inform those nodes that there was a false alarm and that they should return to normal operation. A detailed technical description of the mechanism can be found in Appendix D.

4.2 Experimental evaluation of prioritisation and rate-limiting

In this chapter we use our CPN testbed to evaluate a detection and defence scheme that makes use of prioritisation and rate-limiting in order to defend against a DoS attack, while reducing the impact of collateral damage to valid traffic. For rate-limiting, we will use Token Bucket Filtering (TBF) which is a simple light-weight queueing discipline that only allows packets up to a set rate to pass, with the possibility to allow short bursts in excess of that rate. For prioritisation we will use the PRIO queueing discipline, part

of the iproute2 [2] infrastructure provided with Linux 2.2 which serves packets of a certain priority level only if all higher-priority queues are empty.

The topology of the experiment, shown in Fig. 4.1, is chosen so that we have both short (1-hop) and long routes (5-hop). The hop length of the routes is important since it is effectively the maximum number of DoS defenders that can operate in each one, and the more the DoS defenders that a packet goes through, the more accurate the estimation of its validity.

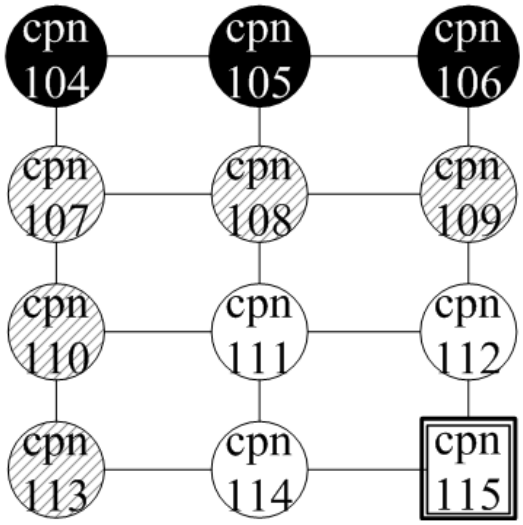

Fig. 4.1: The topology of the attack scenario. Attack nodes, cpn104-106 send up to 10 Mbits/s each, and normal nodes, cpn107-110 and cpn113, send up to 2 Mbits/s each to the target (cpn115).

The aim of the experiments is to emulate the effect of a distributed DoS attack on a Self-Aware Network. For this reason we we use the fully-featured CPN protocol, which means that both normal and attack flows change routes dynamically to find the most convenient, and with improvements in the statistics collection of the CPN code we have a clear view of the incoming and outgoing traffic at each node. The experiments last 25s

[2] The iproute2 of Linux is a system for bandwidth provisioning which supports various methods for classifying, prioritising, sharing and limiting inbound and outbound traffic.

each and are repeated five times.

At time $t = 0$ valid traffic is flowing into the network from nodes cpn107 to cpn110 and headed towards cpn115 in four constant bit rate (CBR) flows of 2 Mbits/s each. Network link capacities are 10 Mbits so that these existing flows can be carried easily by the network. At $t = 2s$ DoS traffic starts arriving through cpn104 at constantly increasing rate towards cpn115. At $t = 5s$ and $t = 8s$, cpn105 and cpn106 respectively join the attack with traffic of similarly increasing rate towards cpn115. In order to emulate the fact that some normal flows may appear after the attack starts and some others may disappear, at $t = 7.5s$ we stop the flow of normal packets from one of the nodes (cpn110) and replace it with another normal flow from cpn113 at the same rate of 2 Mbits/s. As a result of the attack most of the links get increasingly saturated and the rate of valid packets received at cpn115 drops dramatically. It reaches a minimum when the attack stabilises at its peak rate (30 Mbits/s in total) at about $t = 14s$. As seen in Fig. 4.2, the total attack rate has the familiar ramp-up shape of DDoS, as explained in the literature [45]; very fast increase at the beginning, which reaches a peak when all attack flows have arrived.

We repeat the same experiment after applying the following defence scheme. When the total incoming rate of cpn115 is over 15 Mbits/s (cpn115 has only two incoming links of 10 Mbits capacity each), this is considered an attack situation and all nodes which are at most 5 hops from the victim, essentially all in this scenario, are requested to act as defenders.

The distributed classification method that we have designed is based on the use of a marking system, from 1 to 7. If a packet which is headed towards the recognised victim (cpn115) arrives for the first time at a DoS defender node, then it receives an initial mark of 4, and undergoes three classification tests. The mark changes according to the result of each test:

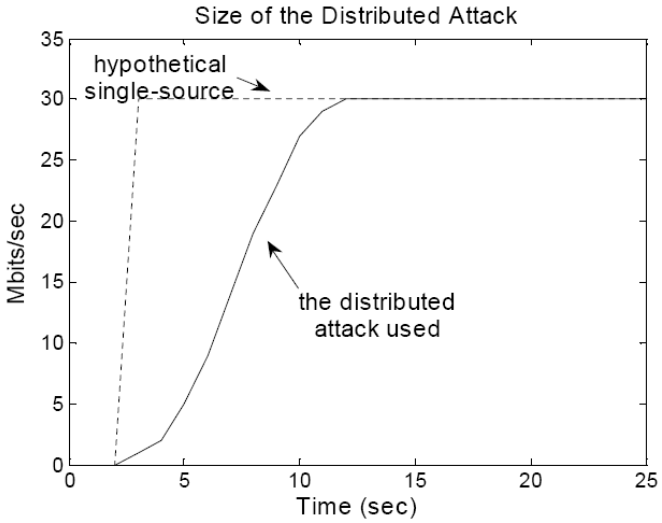

Fig. 4.2: The total attack rate; cpn104-106 send up to 10 Mbits/s each to the target node (cpn115).

a *Is the packet's source among the recognised ones?* If yes then increase the mark by 1. We have arbitrarily chosen three nodes (cpn107, cpn110, cpn113) to belong to the list of recognised customers. The remaining two nodes with valid traffic and the three nodes with DoS traffic are not subject to a change in their mark.

b *Did the packet's source appear after the attack was detected?* If yes then decrease the mark by 1. Nodes cpn106 and cpn113 appear after the attack is detected and receive that penalty (the attack is detected at about t = 7s when the total incoming rate at cpn115 is over 15 Mbits/s).

c *Is the current bitrate from the packet's source over 7 Mbits/s?* If yes then decrease the mark by 1. In this scenario cpn104, cpn105 and cpn106 all exceed that limit after

a while.

The defenders use a priority scheme for their outgoing packets, with priority bands from 1 to 7. Each band allows up to a maximum rate of 7 Mbits/s, less than the capacity of each port, so as to avoid full starvation of lower bands. The packets are allocated to them according to the mark that was assigned to them and stored in their header, so that packets with mark 1 go to the worst band and so on.

As a result of this defence scheme, the normal flows receive preferential treatment at every node throughout the experiment, while the attack flows compete with each other for the low to middle priority bands. The experimental results show that prioritisation according to these simple classification tests allows for practically all valid packets to reach their destination as shown in Fig. 4.3 - 4.7, while the DoS traffic reaching cpn115 is decreased to 50% of each maximum value as seen in Fig. 4.8 - 4.10.

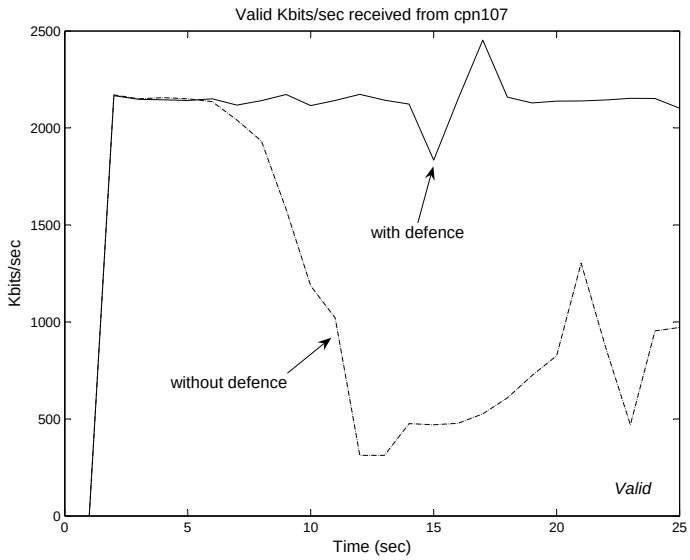

Fig. 4.3: The bitrate of normal traffic from node cpn107 that made it safely to their destination (cpn115).

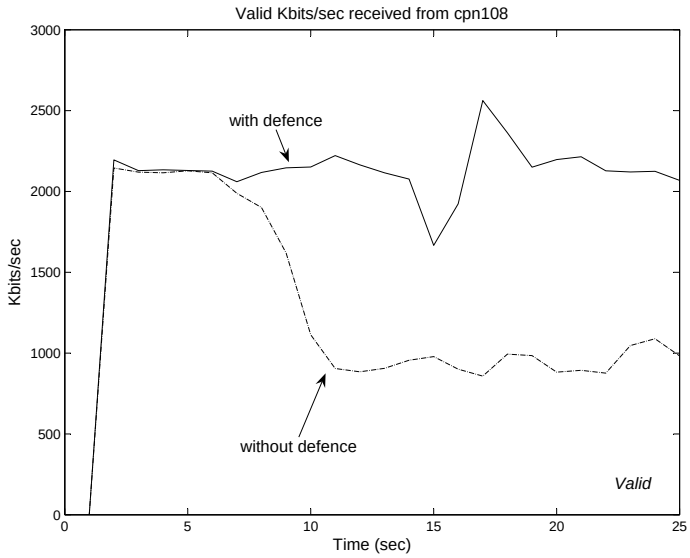

Fig. 4.4: The bitrate of normal traffic from node cpn108 that made it safely to their destination (cpn115).

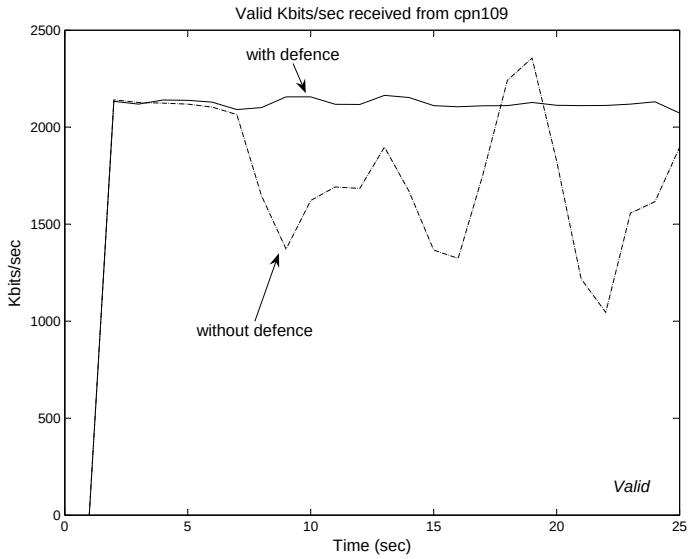

Fig. 4.5: The bitrate of normal traffic from node cpn109 that made it safely to their destination (cpn115).

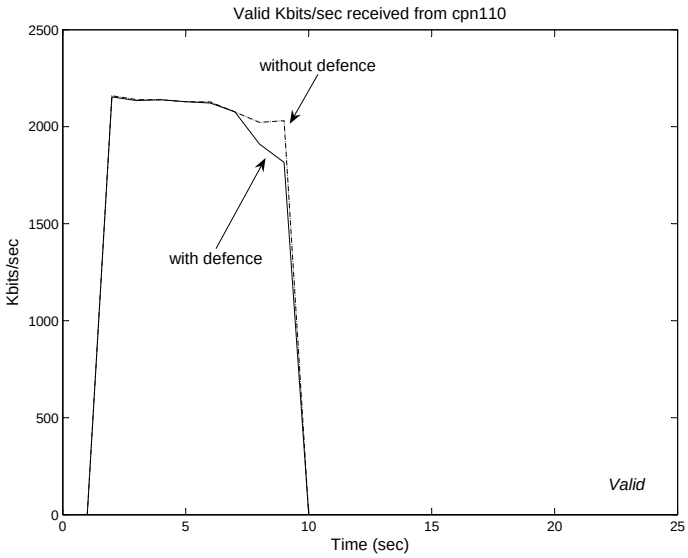

Fig. 4.6: The bitrate of normal traffic from node cpn110 that made it safely to their destination (cpn115).

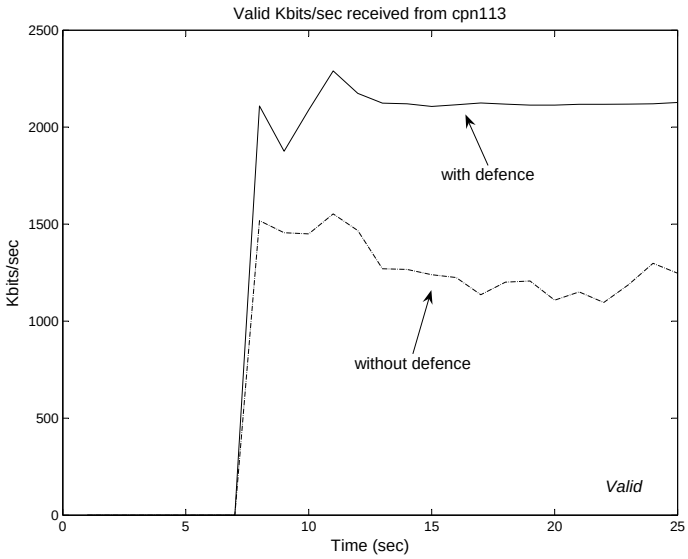

Fig. 4.7: The bitrate of normal traffic from node cpn113 that made it safely to their destination (cpn115).

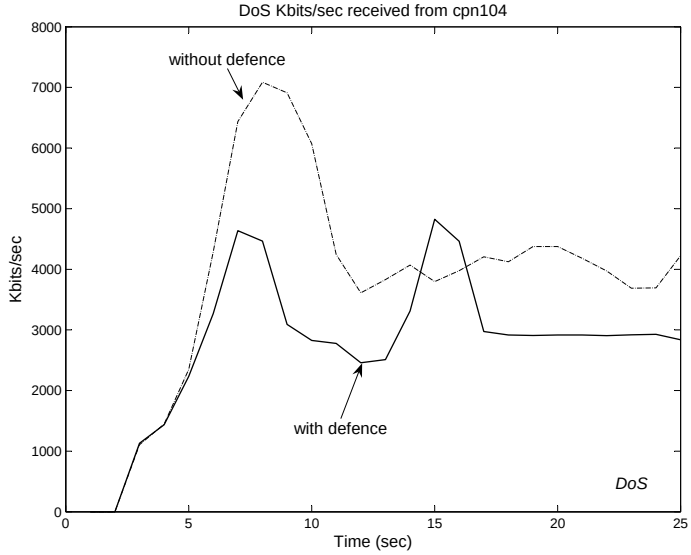

Fig. 4.8: The bitrate of DoS traffic from attacker cpn104 that reached its target (cpn115).

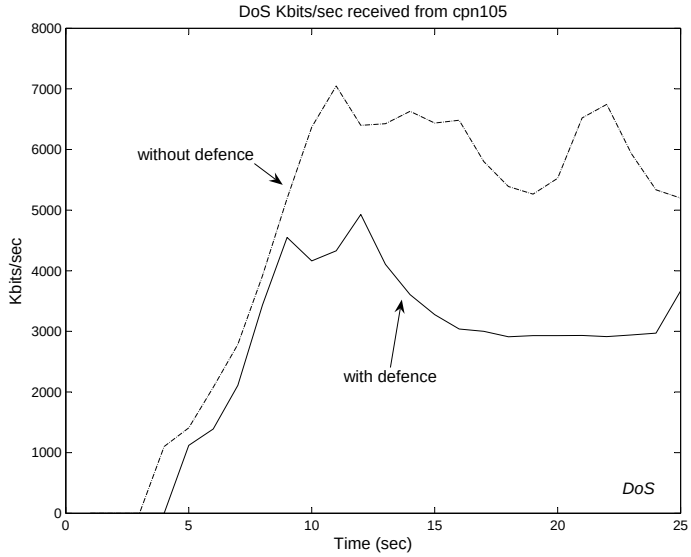

Fig. 4.9: The bitrate of DoS traffic from attacker cpn105 that reached its target (cpn115).

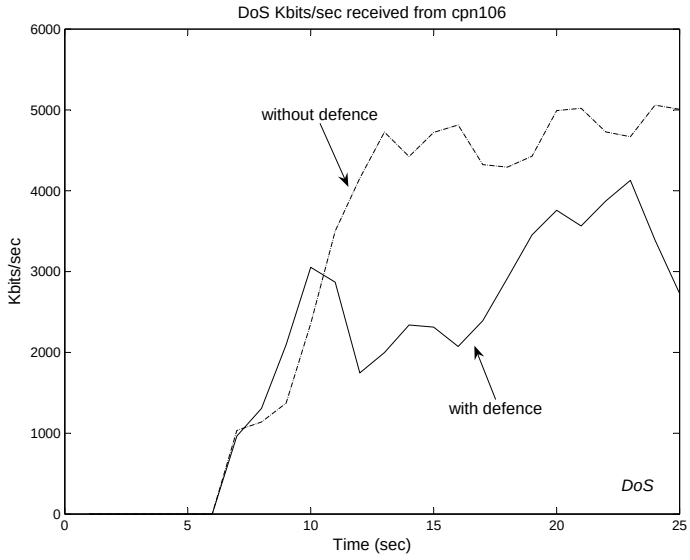

Fig. 4.10: The bitrate of DoS traffic from attacker cpn106 that reached its target (cpn115).

We repeat the same scenario with fewer nodes participating in the defence, varying the distance of defending nodes from 1 to 4 from the attacked node as shown in Fig. 4.11. The results show that a radius of defence in excess of 1 can achieve a level of success comparable to the case when all nodes are used as defenders (Fig. 4.12). Note that all numerical values shown are the averages of five independent repetitions of a given experiment.

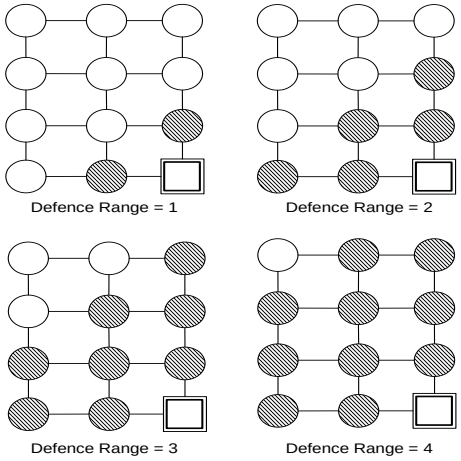

Fig. 4.11: Explanation of the defence ranges used in our experiments.

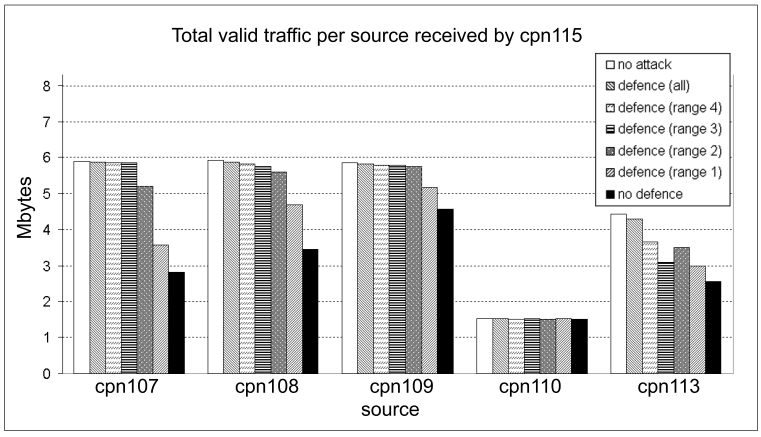

Fig. 4.12: The total normal traffic that makes it to the destination (cpn115) when there is no attack, when there is attack and defence from range 5 to range 1, and when there is attack and no defence.

4.3 Conclusions

4.3.1 Summary of the chapter

In this chapter we proposed a complete DoS defence system, in which (i) the detection is based on the incoming bitrates and their change at the victim node, (ii) the classification uses several validity tests to assess the probability of the packets being legitimate or DoS, and (iii) the response mechanism rate-limits the incoming traffic and prioritises the packets according to the result of the classification process.

With the help of the dynamic behaviour of CPN, which improves the chances of finding a good route, practically all the traffic of the legitimate users arrives at the destination. Packets with higher probability of being both valid and relatively harmless for the network are served with higher priority. Packets which would be marginally classified as invalid may receive service if there is available bandwidth, this way minimising the potential collateral damage inflicted in case of false classification. Meanwhile, invalid packets or valid packets which at the specific time would be excessively harmful for the network, are served with lower priority, which essentially means that they are either delayed or dropped due to buffer overflow. This framework is easy to expand as new DoS validity tests are proposed and can be seamlessly integrated in existing service prioritisation schemes. Also, since marking is only looking for anomalies (behavioural and statistical) and not for known attack signatures, it can still be effective against undiscovered yet types of attacks.

4.3.2 Weaknesses and suggestions

Every priority-based system faces the problem of starvation, since without some kind of precaution, low-priority flows may never receive service. Depending on the accuracy of the validity tests of the classification process, starvation is not necessarily a negative

thing, as the lower-priority flows should be the DoS or the misbehaving ones.

5. DISTRIBUTING DEFENCE TASKS

Extensive filtering and complicated defence systems will introduce significant overhead. Thus different parts of the defence mechanism may be allocated to different nodes, with some specialising in detection and others in aspects of traffic classification or active response. The corresponding design may be performed with techniques based on our mathematical model of Chapter 3.3. For example, it is often stated that defence can be carried out more effectively closer to the attackers, while detection is more accurate at the victim or at its close neighbours. We decided to test this commonly held view for the allocation of defenders by building a tool, based on the specific mathematical model. We demonstrate how such a tool can recommend the numerically optimal way to allocate defence tasks in a distributed defence system and identify inherent weaknesses of a given topology in terms of the vulnerability of specific nodes.

5.1 Distributing tasks in a small network, without bandwidth considerations

Consider the generic attack scenario of Fig. 3.3. Instead of using all upstream nodes for defence, we only use the three nodes that are closest to the attackers (3, 4, 5), and then as a contrasting alternative only the ones closer to the victim (1, 5, 6) as shown in Fig. 5.1. Call these Case 1 and 2, respectively. For a modest detection performance of $f_{i,\mathbf{n}} = 0.1$ and $d_{i,\mathbf{d}} = 0.6$, the results are contrary to our expectations (Fig. 5.2, 5.3), as Case 1 proves worse than Case 2. By this, we do not wish to imply that the commonly held

view is wrong, but simply that a mathematical model can help make the best choices as a function of a set of measurable parameters. In fact, if we were to allocate exactly 3 nodes to do the dropping (or rate-limiting) in this network, our tool went through all $\frac{7!}{3!(7-3)!} = 35$ possible combinations for a given range of potential rates of attack and rates of normal traffic, and provided us with the optimal choice (in terms of goodput to the victim), which is $\{1, 4, 6\}$ if the three attack flows have the same rate (at the range 1000-8000 packets/s that we checked) and the ten normal flows also have the same rate (100-800 packets/sec). The resulting curves for the optimal distribution are provided for comparison in the same figures. If we keep only the optimal result for each number of defenders over a range of normal traffic load level, we get the two surfaces of Fig. 5.6. From these graphs we see that in our scenario using more than 5 defenders will not increase the performance of the defence in any but the highest load level, with $\{1, 3, 4, 5, 6\}$ proving to be the best choice.

The previous results were obtained with the assumption that the additional task of defending assigned to specific nodes will not affect their efficiency at processing and forwarding packets. In the real defence implementations, however, this is not the case. To study this additional parameter we repeated the same analysis for various levels of increase of the average service time in each of the defending nodes. As expected, the higher the performance penalty for the defenders the lower the optimal number of them. Some representative results are shown in Fig. 5.7. The same concept can be extended to all kinds of DoS defence tasks if we know or measure the performance penalty (the average increase in the service time) due to each task. Then the optimal allocation of multiple tasks can be found with the use of this tool.

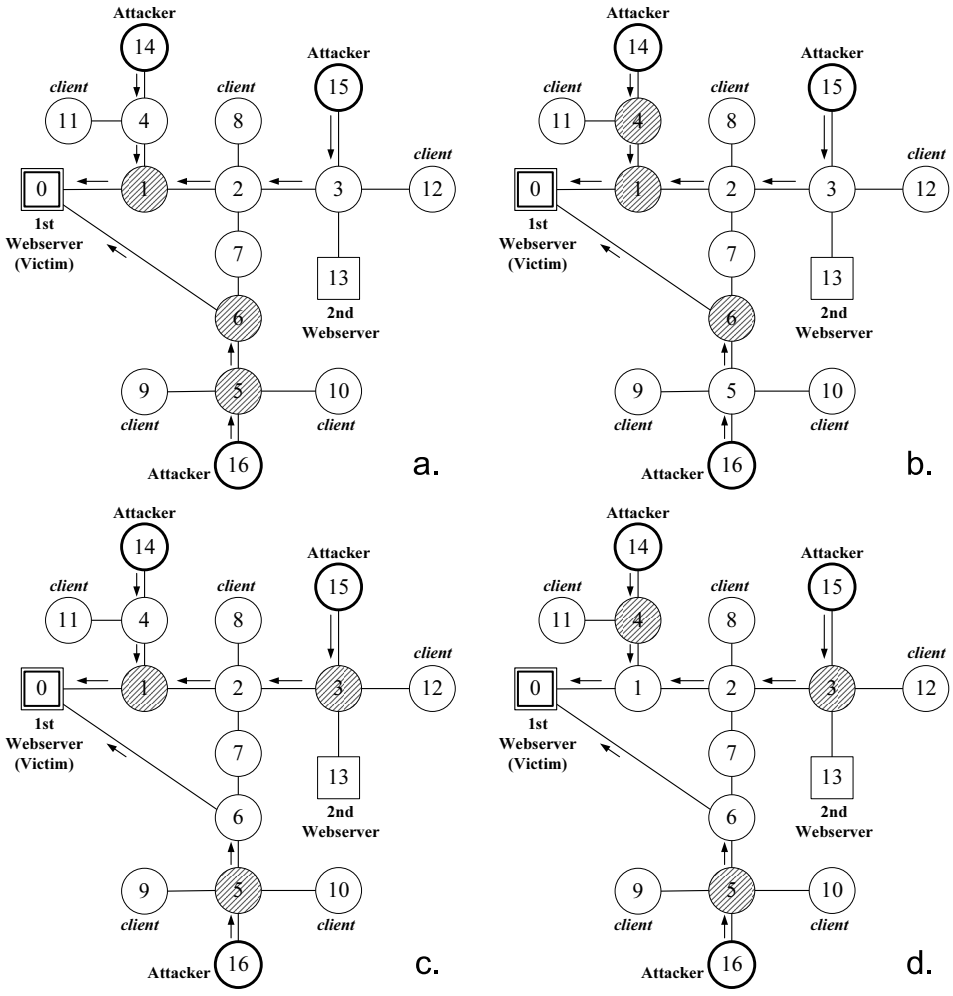

Fig. 5.1: Partial defence distribution: a) nodes 1, 5 and 6. b) nodes 1, 4 and 6. c) nodes 1, 3 and 5. d) nodes 3, 4 and 5

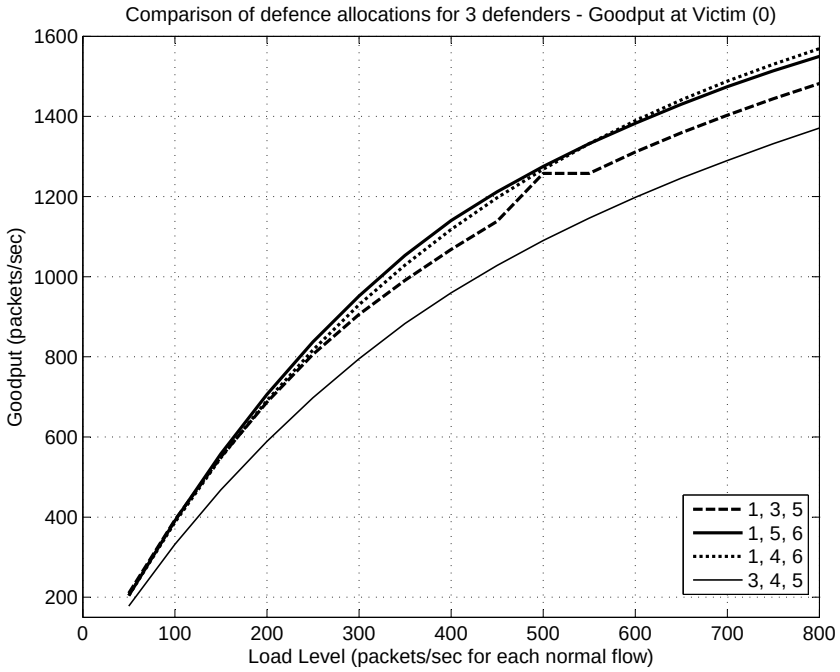

Fig. 5.2: Comparison of defence allocations for 3 defenders - Goodput at the Victim (0) *Vs.* Load Level

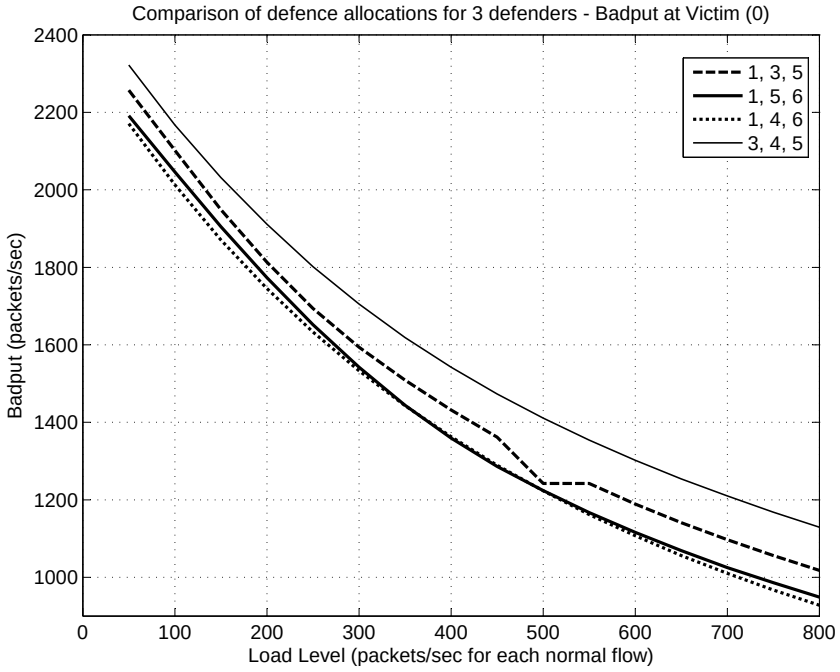

Fig. 5.3: Comparison of defence allocations for 3 defenders - Badput at the Victim (0) *Vs.* Load Level

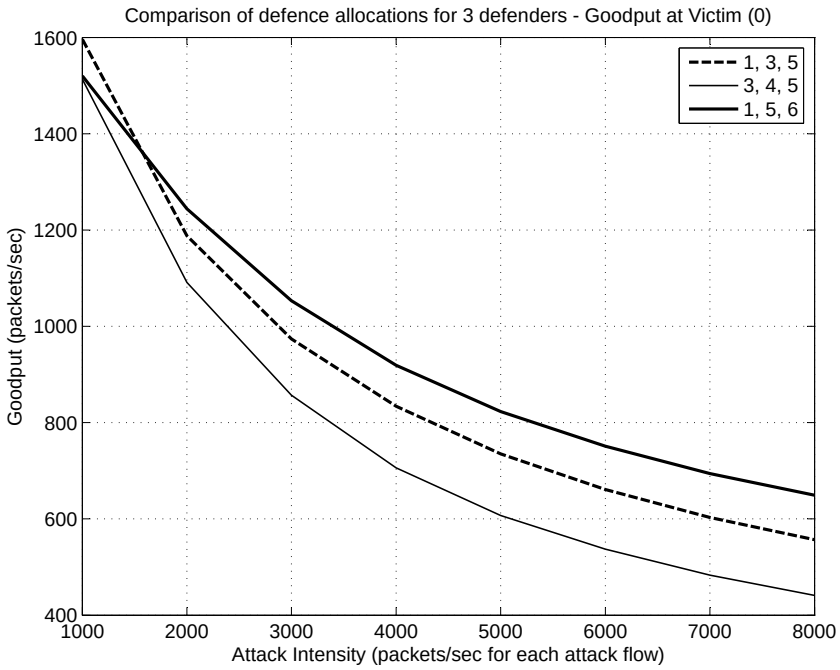

Fig. 5.4: Comparison of defence allocations for 3 defenders - Goodput at the Victim (0) *Vs.* Attack Intensity

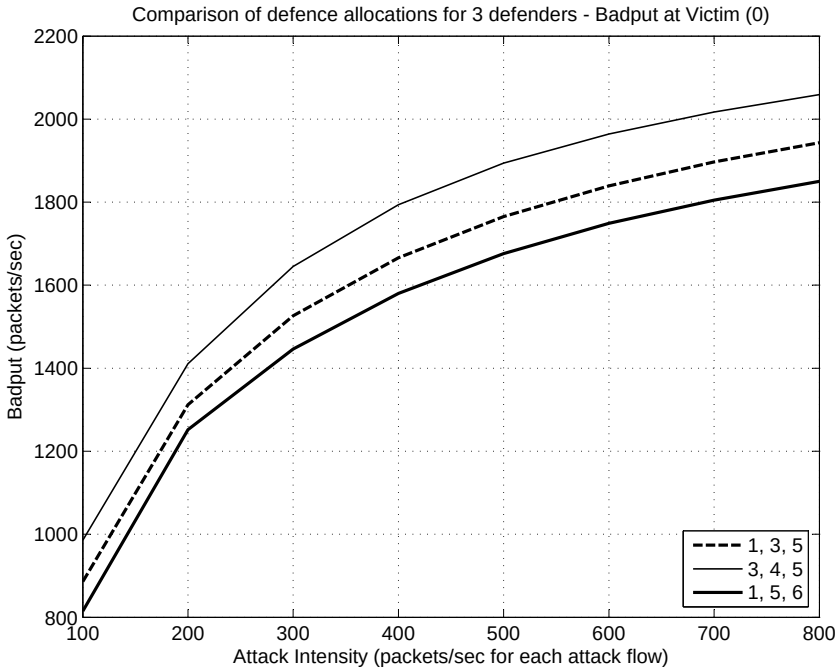

Fig. 5.5: Comparison of defence allocations for 3 defenders - Badput at the Victim (0) *Vs.* Attack Intensity

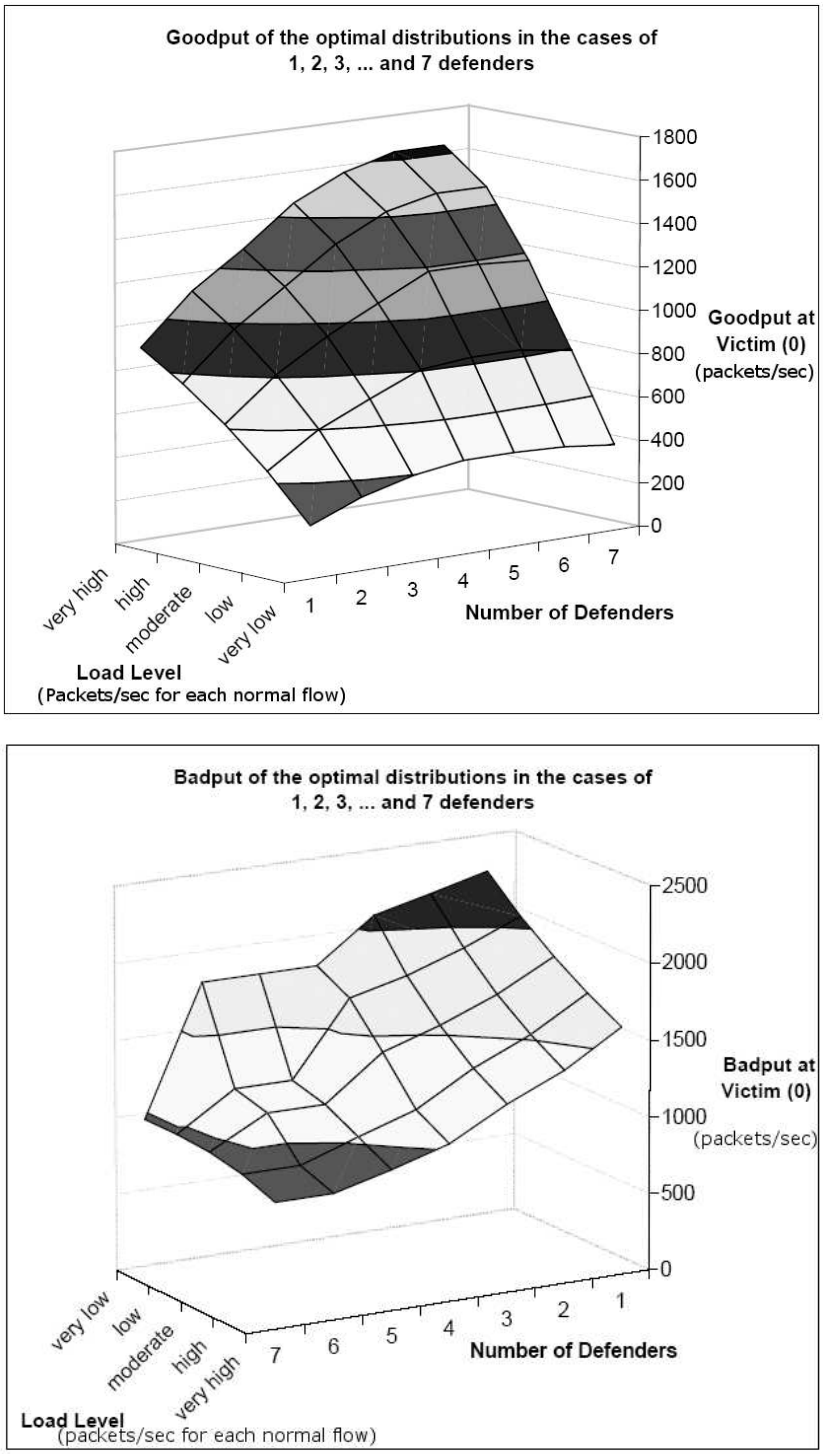

Fig. 5.6: Comparison of optimal distributions of the defenders for 1, 2, 3, ..., 7 defenders

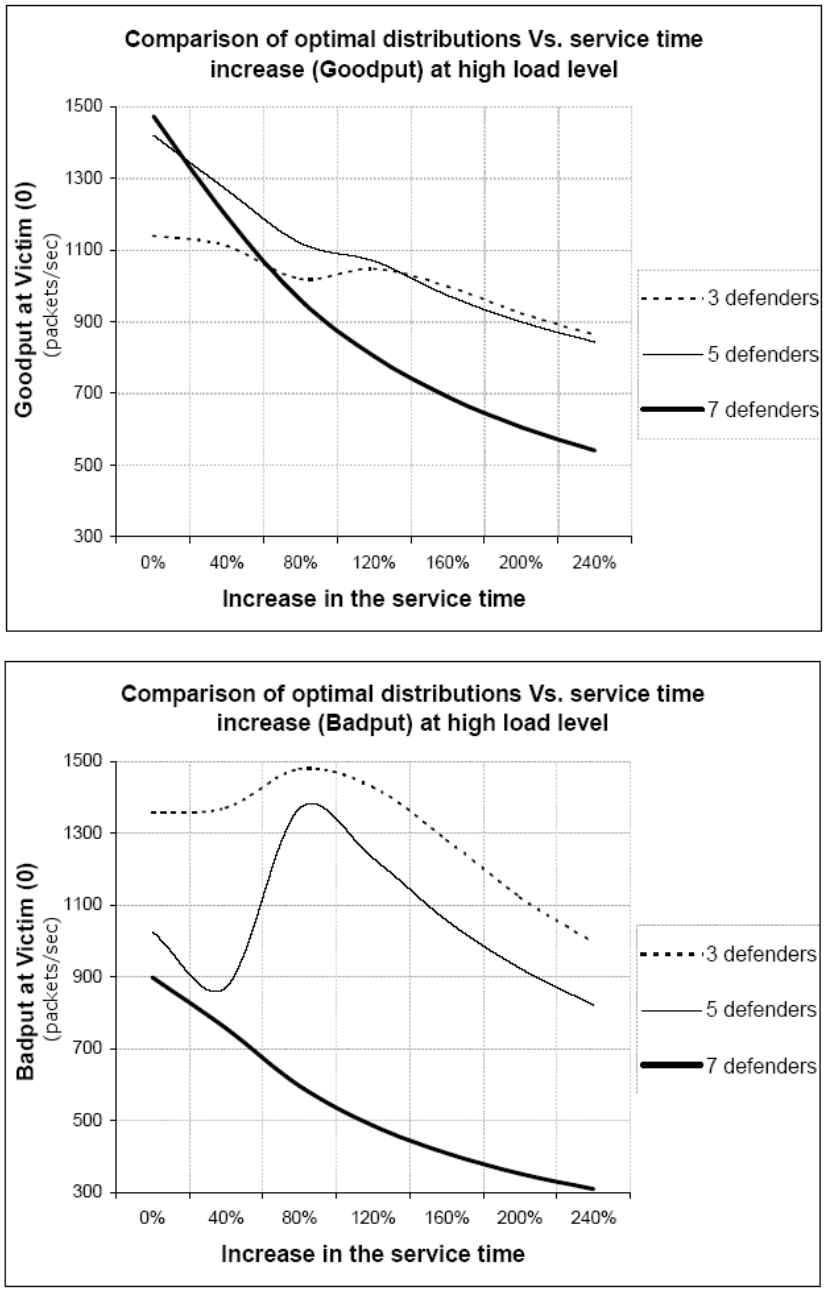

Fig. 5.7: Comparison of optimal distributions of the defenders for various service time increases

5.2 Description of the optimal allocation tool

The following is a high-level description of the way the optimal allocation tool works. As seen in Fig. 5.8, it uses as input a description of the attack scenario (topology file, normal traffic file, DoS traffic file) and a desription of the defence mechanism (VictimID, excluded nodes, $\{P_f, P_d\}$, service time penalty, number of defenders). The output is either the optimal allocation of defenders or the performance of a specified allocation, according to the goodput at a specified node (measuredNodeID - usually the same as the VictimID). A few parameters referring to the precision of the numerical computations can also be customised (number of iterations and buffer sizes).

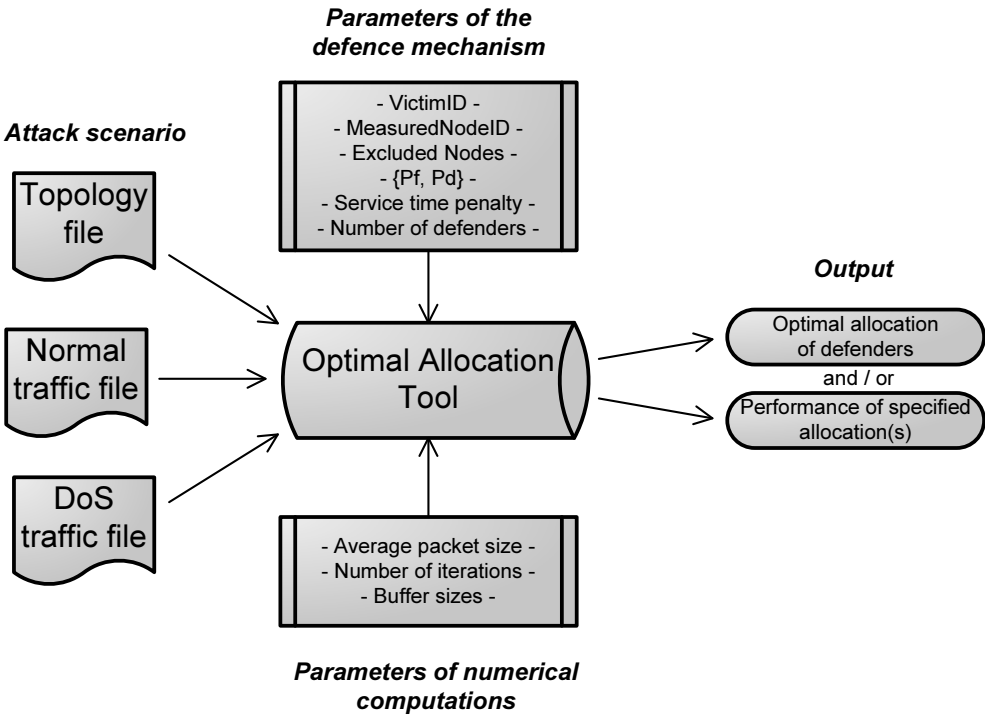

Fig. 5.8: Graphical representation of the mechanism of the tool for optimal allocation of defenders

Normal traffic file. This text file contains all information for the normal traffic in steady state. Each line in the file represents a flow of normal traffic with the format

being:

Unique ID of the flow / sequence of nodes in the flow's path starting from the source and ending to the destination / flow's rate in packets/s or Kbits/sec. For example:

0/4-1-0/300

1/4-1-2-3-8/300

2/2-1-0/300

...

The tool stores this information in separate arrays to allow for more flexibility.

DoS traffic file. Similarly, this file contains all information for the DoS traffic in steady state, with each line representing a flow of DoS traffic:

0/4-1-0/2500

1/3-2-1-0/2500

...

Topology file. The topology information can be fed to the tool either as a text file which is converted to a NxN topology array or directly as a hardcoded array (N is the number of nodes in the topology). For each pair of nodes (i,j) with $i \neq j$ the value is 1 when there is a direct link or 0 otherwise. In addition, for the (i,i) pairs the stored value is the processing rate in Kbits/s of node i. For example, the following is the topology file used in the scenarios of Chapter 3:

10000,1,0,0,0,0,1,0,0

1,10000,1,0,1,0,0,0,0

0,1,10000,1,0,0,0,1,0

0,0,1,10000,0,0,0,0,1

0,1,0,0,10000,0,0,0,0

0,0,0,0,0,10000,1,0,0

1,0,0,0,0,1,10000,1,0

0,0,1,0,0,0,1,10000,0

0,0,0,1,0,0,0,0,10000

This processing rate (10000 in the example) is used to compute the service time of each node:

$$s = \frac{8 \cdot 1000 \cdot average\_packet\_size}{processing\_rate} \tag{5.1}$$

Alternatively, the service time can be used directly in the topology file or converted from a processing rate given in packets/s instead of Kbits/sec.

VictimID. This is the node ID of the victim in the scenario or generally the node that the defence system is set to protect.

MeasuredNodeID. This is the ID of the node of which we measure the goodput. In reality the tool measures and is able to show both the normal and the DoS incoming rates for all nodes in a given topology, but by choosing a specific measuredNodeID the tool has a benchmark metric according to which it suggests the optimal allocation of defenders.

Excluded nodes. In specific scenarios we may wish to exclude some nodes from the pool of potential defenders. This option is provided in the form of the excluded nodes ArrayList. By default potential defenders are all the nodes in the given topology, including the victim.

$\{P_f, P_d\}$. This is the global set of probabilities for false alarm and correct detection. Alternatively we can set a different set of probabilities for each node if this is

needed in our scenario. For example, later, in section 6.1.2 we present an example where there are two different types of defence tasks to allocate, with different sets of $\{P_f, P_d\}$ for each node according to their task.

Service time penalty. As explained in Section 5.1, it is usually unrealistic to assume that the additional task of defending assigned to specific nodes will not affect their efficiency at processing and forwarding packets. The service time penalty is a positive value which represents the increase of the average service time per packet for a specific defence task. A penalty of 1.2 for example translates into a 120% increase over the node's initial average service time computed by (5.1).

Number of defenders. The tool is designed to suggest the optimal allocation by comparing the goodput of the measured node for all allocations of a specific number of defenders.

Average packet size. This is used for the necessary conversions only when the processing rates of the nodes in the topology file are written in Kbits/s instead of packets/sec. Details about average packet sizes in the Internet can be found in Appendix E.

Number of iterations. The number of iterations used for the numerical solution of the non-linear system, as explained in Section 3.3. Obviously the higher the number of iterations the better the accuracy, and the slower the computation. When the tool is used simply to compare different allocations of defenders then numerical accuracy is not that important and we can choose a low number of iterations (20 was enough in most cases). For the computations of Section 3 where we wanted to compare numerically the mathematical results with the results of the simulations and the experiments, we used several thousand iterations for each scenario.

Buffer sizes. These are the sizes of the queues of the nodes in the given topology. For the sake of simplicity we choose the same value for the buffer sizes of all nodes.

The computation time depends on the number of iterations for the numerical solution of the non-linear systems, the sizes of the buffers and the number of nodes and flows in the scenario.

5.3 A numerical example for a real large topology, with bandwidth considerations

It is important to note that we built the mathematical model considering that the most crucial and most vulnerable resource in a network during a DoS attack would be the processing capacity of the nodes. That we represented with the average service time s_i for each node i. However, although adequately fast links are becoming increasingly available and affordable at a rate much higher than the increase in processing power, it can still be expected that an attack may overwhelm the links before it overwhelms the processors. In this case, where link capacity matters, we have to slightly alter the mathematical model. We now have to refer to i as an entity of the network, which can be either a node or a link with processing capacity represented by a rate r_i of packets per time unit: For a node, $r_i = \frac{1}{s_i}$, and for a link, $r_i = \frac{\text{capacity of link}}{\text{average packet size}}$. With this simple change, (3.2) becomes:

$$\rho_i = \frac{1}{r_i}(\sum_{\mathbf{n}} I^n_{i,\mathbf{n}}(1 - f_{i,\mathbf{n}}) + \sum_{\mathbf{d}} I^d_{i,\mathbf{d}}(1 - d_{i,\mathbf{d}})), \tag{5.2}$$

and (3.8) becomes:

$$\rho_i^{(k)} = \frac{1}{r_i} I_i^{(k-1)} \tag{5.3}$$

The rest of the equations of the mathematical model (Section 3.3) remain the same.

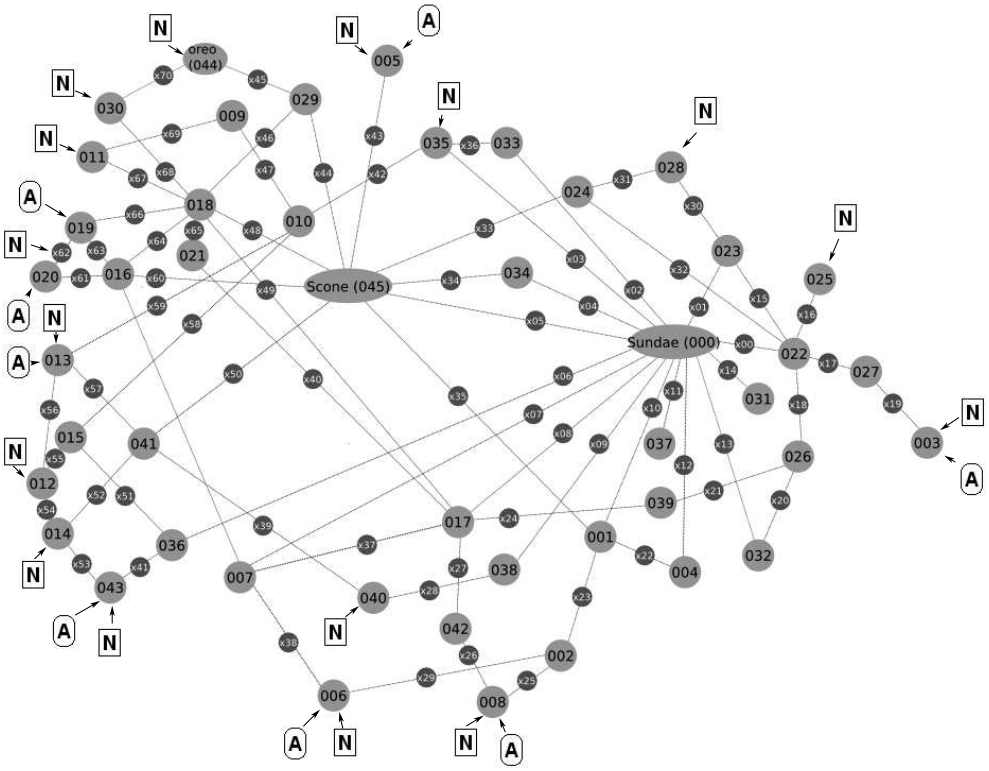

Fig. 5.9: Simple DoS scenario on the SWITCHlan topology, with 16 normal clients and 8 attackers, with all traffic directed towards node Sundae(000)

To test this additional capability of the tool, we test it on an attack scenario in the real topology of the SwitchLAN network. As seen in Fig. 5.9, 8 attacking and 16 normal flows are directed towards node Sundae(000). The majority of the links have 10 Mbits/s capacity except for a select few of 100 Mbits/sec, such as the one between nodes 045 and 000. For an average packet size of 500 Bytes, the 10 Mbits/s links have $r = 2500$ packets/s and the 100 Mbits/s have $r = 25000$ packets/sec. For all nodes we have chosen $r = 10000$ packets/sec. The numerical results shown in Fig. 5.10 and 5.11 are indicative of the impact that the attack and defence are expected to have in such an attack scenario.

It is more interesting, however, to look at the damage that each attack flow would inflict on its own. To determine the entrance node where the network is most vulnerable we repeat the scenario using only one of the 8 attack flows each time and we compute the

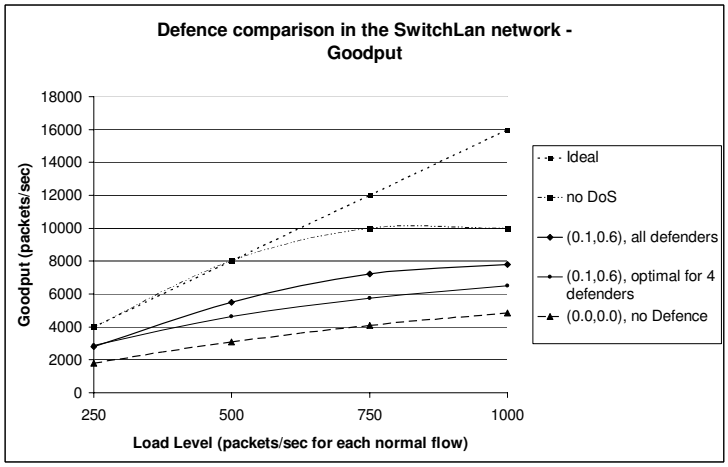

Fig. 5.10: Evaluating the DoS scenario on SwitchLAN - Goodput

goodput at the victim for the optimal distribution of (0.1,0.6) defenders each time, with load level of 750 packets/s per normal flow. Fig. 5.12 shows that the attack flow which enters through node 043 is more difficult to defend than the rest. This result, which is repeated for different load levels, attack levels and detection probabilities, exposes a vulnerability of the specific topology in terms of the protection of node Sundae(000). An obvious solution would be to change the topology by increasing the capacity or the number of intermediary links and nodes. Alternatively a more accurate defence mechanism should be installed in the nodes between 043 and Sundae(000). The above analysis took under consideration only the attack flows of the specific arbitrarily chosen scenario. For a more systematic investigation of the vulnerabilities of a given topology one should repeat the analysis for all potential entrances of attack flows and also not only for single flows, but for combinations of them.

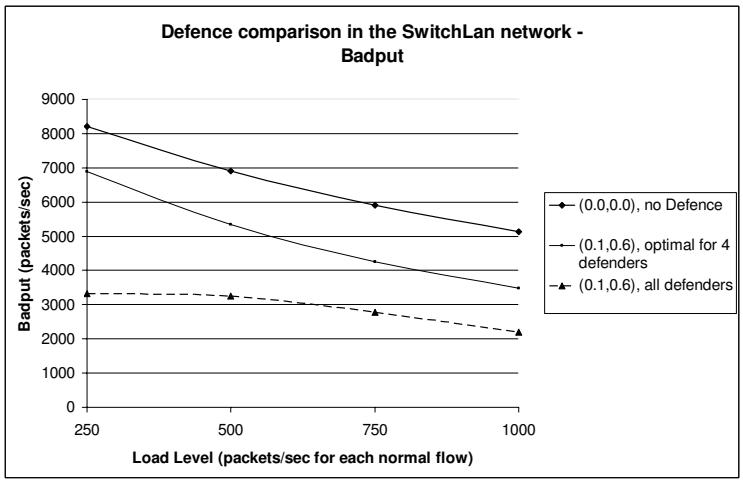

Fig. 5.11: Evaluating the DoS scenario on SwitchLAN - Badput

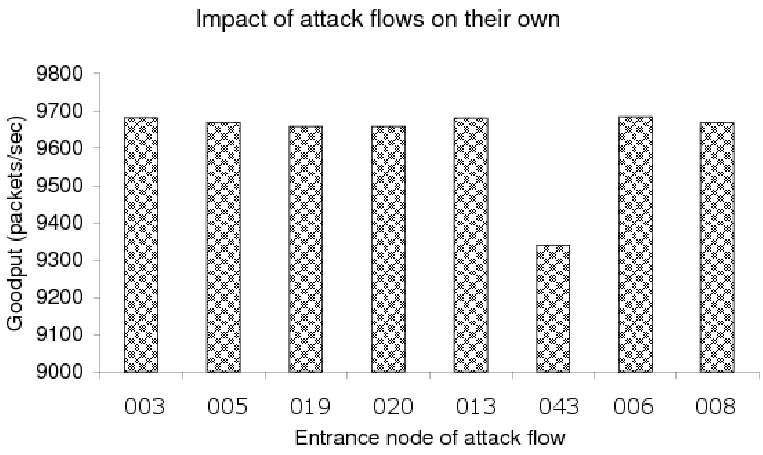

Fig. 5.12: Evaluating the DoS scenario on SwitchLAN - Impact of attack flows on their own

5.4 Conclusions

5.4.1 Summary of the chapter

Using all available nodes to defend during a DoS attack is usually undesirable as this practice may completely impair the normal operation of the network, to a large extent due to the performance penalty that defence tasks impose on the nodes. In this chapter we presented a way of numerically computing the performance of the various possible allocations of defence tasks and suggesting the optimal one in terms of the resulting goodput given a user-defined desired number of defenders.

5.4.2 Suggestions for the use of the tool and potential weaknesses

As described in this chapter, the optimal distribution tool can be particularly useful for the administration of a network at setting up a defence architecture and even the physical topology. It can be used offline to find the more vulnerable areas in a topology or in realtime after a DoS attack is detected, right before the allocation of defenders. For the real time case, it needs to receive as input an estimation of the current legitimate and attack traffic, as received by measurement of the bitrate at each node. The tool inherits the weaknesses of the mathematical model it is based on (Section 3.3), which means that its accuracy depends largely on the information that it receives as input. In both offline and realtime case, in order to have a good view of the potential damage that an attack could inflict on a network and the defence that one should deploy, a wide range of attack distributions and rates should be used, as DoS attacks differ completely from case to case. If the normal traffic used in the optimal distribution tool also varies a lot, then this would increase the complexity even further and impair the accuracy of the analysis.

6. IDENTIFICATION OF NOVEL PROBLEMS AND SOLUTIONS BASED ON OUR WORK

Our technical investigations of Chapters 3, 4 and 5 has led to the identification of new problems, never suggested before, and to potential solutions for them based on the work presented in this thesis. In Section 6.1, we identify the interdisciplinary issue of acquiring information during an attack which will be of use for law-enforcement, and we extend our tool of Chapter 5 to find the optimal distribution of this new type of task in the victim network. In Section 6.2, we introduce the concept of charging to provide better DoS protection for some of the users or nodes. We describe a simple static charging scheme and two more advanced ones based on a modification of the prioritisation mechanism of Chapter 4 and an application of the mathematical tool of Chapter 3. In Section 6.3, we propose the idea of on-demand third-party DoS protection as a solution to the inherent problem of overhead during the normal operation of a network, that all DoS defence mechanisms introduce to it, including the ones presented in this thesis. Last, to tackle the same problem of overhead during normal operation, in Section 6.4, we describe an advanced mechanism to detect the existence of a DoS attack and trigger the initiation of our response mechanisms of Chapters 3 and 4 only when they are needed.

6.1 Law-enforcement defenders

Many of the issues that perplex DoS researchers, pose as significant complexities for legislators too, and could produce an interesting topic of interdisciplinary research.

6.1.1 Legislation complexities

Intent

In Section 2.2 we talked a bit about flash-crowds, which are cases of perfectly legitimate traffic, but with repercussions equally unwanted as with DoS attacks. There is the "slashdot effect" [1], where a popular website links to a smaller one as the original source of the news it presents. That often causes the smaller website to receive more traffic than it can handle, with effects very similar to a DDoS attack. The fundamental difference of Denial of Service and these situations of increased traffic is the intent.

Virtual sit-ins

In Appendix A, we mention an incident where an angry customer used common DoS tools to launch an attack against his telecommunications provider as a form of protest for what he considered as bad service. That was just one frustrated individual, but there are also several groups of "hacktivists" [2] who organise such attacks. The "Electronic Disturbance Theater" group has targeted the Mexican government, the US department of Defense, the Frankfurt Stock Exchange, and the US Border Patrol website and is still very active. The French group "Federation of Random Action" has targeted the International Monetary Fund and the World Bank [100]. Such events of "electronic civil disobedience" [3], which are pre-announced and involve DDoS attacks launched not by one individual, but by often several thousand Internet users are called "virtual

[1] Slashdot.com is a popular website with technology news

[2] From wikipedia (http://en.wikipedia.org/wiki/Hacktivism): Hacktivism (from hack and activism) is often understood as the writing of code, or otherwise manipulating bits, to promote political ideology - promoting expressive politics, free speech, human rights, or information ethics. Acts of hacktivism are carried out in the belief that proper use of code will have leveraged effects similar to regular activism or civil disobedience.

[3] From wikipedia (http://en.wikipedia.org/wiki/Electronic_civil_disobedience): Electronic Civil Disobedience (ECD) is any type of civil disobedience in which the participants use information technology to carry out their actions.

sit-ins". Since usually they intentionally do not last too long or cause much damage, they are academically also known as "cyber-graffiti" [38]. In danger of confusing the reader with excess terminology we could say that virtual sit-ins are an electronic hybrid between the sit-ins of the civil rights movement of the 1960s and the "flash-mobs" of the 21st century [4]. As a new type of protest and political expression, hacktivism through DDoS attacks has both many ideological supporters and many enemies, but what matters for the research community is that there is no clear legislation addressing it. Since we, the researchers, are not responsible for distinguishing between what is legal and what is not, we are still not in position to address this issue technically, apart from considering it as yet another type of flash-crowd.

So, who is to blame?

Currently, no technical approach can be used to find the exact source of a DoS attack. In section 2.1 we presented the existing traceback mechanisms and explained that even if they were perfect they could only identify the source IP addresses of the compromised computers and not of the attacker. For that reason, most DoS attackers are tracked down almost exclusively by traditional "offline" police investigations rather than by locating the source of the attack [100]. Even then though, the suspect may claim that his/her computer was compromised by a hacker and was not the one who launched the attack.

Estimating losses

In most DoS cases it is very difficult to estimate the losses caused by an attack. The duration, the number of compromised computers, and the full scale of the collateral damage are all unknown. The lost revenue is also not clear unless there is a clear indi-

[4] From wikipedia (http://en.wikipedia.org/wiki/Flash_mob): A flash mob is a group of people who assemble suddenly in a public place, do something unusual for a brief period of time, and then quickly disperse. They are usually organised with the help of the Internet or mobile telephony.

cation of the level of QoS degradation (delays, packet losses, etc.) that the legitimate traffic suffered. From the point of view of a legislator, this vagueness has important implications even in terms of the jurisdictional thresholds [5]. From the point of view of a DoS researcher, these costs incurred due to the interruption of services (and indeed the intensity and seriousness of the attack) should be estimated in real time and be taken into account before deciding what types of (or even whether) defence mechanisms should be initiated [6]. We have partly addressed this issue with our mathematical model. Although not specifically designed to measure monetary costs, with simple adaptations it could prove useful, since monetary costs are directly dependent on the rate of the legitimate traffic, which is computed with our mathematical model.

6.1.2 Positioning a law-enforcement defender

Apart from the current legislative inefficiencies, an important factor that hinders prosecution of DoS attackers is the lack of evidence. Thus, it would be reasonable to envision a distributed DoS defence scheme, such as the ones considered in this thesis, but with one additional task to allocate, the task of collection of evidence. A node assigned with this task, together with (or instead of) actively participating in the defence by detecting, classifying, or responding to an attack, it would collect information which could be used as evidence in court. The location of such a node would be particularly important not only to be secret but also to avoid further deterioration of the performance of the network. To illustrate this issue of the allocation of multiple tasks, consider again the example shown in Fig. 3.3 that we have used before. Suppose a scenario with 2,500 packets/s for each DoS flow, 400 packets/s for each normal flow, and 35% increase in

[5] Depending on the cyber-law of each country, there are usually thresholds of financial damage to place cyber-crimes within the jurisdiction of a specific authority.

[6] In fact, a DoS attacker with sufficient knowledge on the kinds of defences used in a network could achieve significant damage by simply triggering the initiation of the defence mechanisms. A former employee of a company with information on its security would fit that profile.

the service time due to active defence with probabilities $(P_f, P_d) = (0.1, 0.6)$ in the nodes used by the attack flows (1, 2, 3, 4, 5 and 6). If there is no topological advantage for one node or another in terms of the efficiency of evidence collection, then the choice of location for this task would be based only on its impact to the performance of the network. In Fig. 6.1 and 6.2 we show how this choice (or generally for any task with no active participation in the ongoing defence) would be based on the performance penalty due to the task.

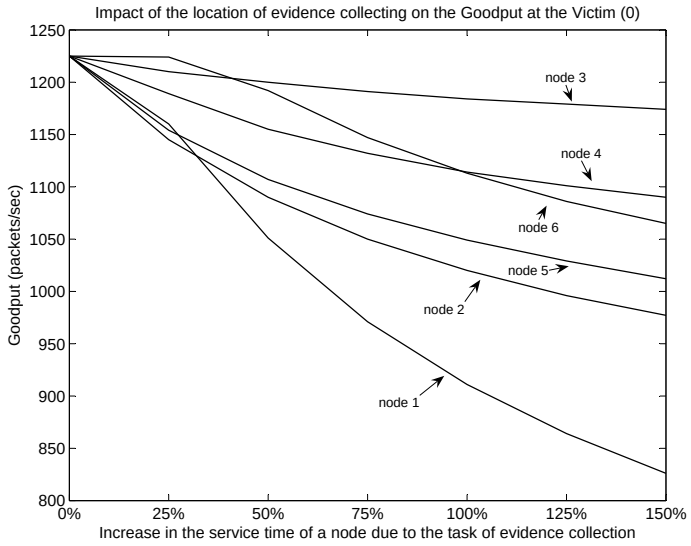

Fig. 6.1: Impact of the location of the evidence collection task on the Goodput at the Victim (0).

In fact, it would be a reasonable choice for all surviving defending nodes to turn into evidence collectors when it is realised that it is not possible to counter the attack with technical measures. To the best of our knowledge the task of evidence collection in a DoS defence mechanism has not been suggested before [7]. An interesting aspect of such research is that different legal systems deal with Denial of Service in a different way, and there is currently no standardised set of evidence that is needed. Thus, the

[7] The only resembling concept that we can think of is that of the "honeypots", nodes which attempt to appear as attractive targets for attackers, but provide no service to legitimate users; they only exist to capture and analyse attack traffic, and they do not receive any legitimate traffic.

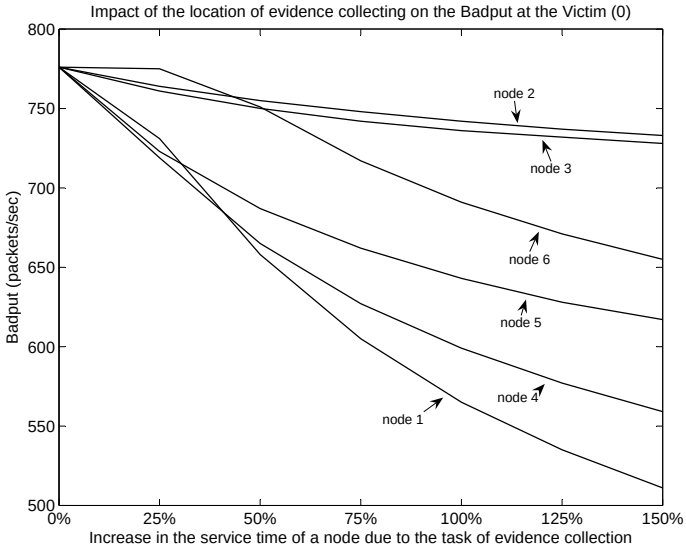

Fig. 6.2: Impact of the location of the evidence collection task on the Badput at the Victim (0).

design and implementation of a DoS evidence collector system would need flexibility and adaptability to be suitable for different countries and updates in their legal systems. References [22, 56, 35, 105, 77] would be useful to begin with.

6.2 Charging for protection against DoS

During a Denial of Service attack incident it would be reasonable to introduce a charging scheme which would attempt to improve the chances of better service to those paying for it.

6.2.1 Charging according to defence choice

A simple approach to providing such a differentiation in terms of protection during a DoS attack would be to have a collection of available defence choices and ask for a different price for each one according to its sophistication. Suppose that each node can have an additional type of defence installed in it, independently and in addition to the network's global defence mechanism. Then, the customer's choice of protection would correspond to the choice of this additional defence per node for the nodes in the paths that his traffic would use. Along the lines of our previous discussions this would mean that installing in those nodes an additional defence choice with (historically) expected probabilities of false alarm and correct detection $(P_f, P_d) = (0.1, 0.9)$ would have to cost more than a $(P_f, P_d) = (0.1, 0.6)$ one. If we wanted to take this a step further then two defence mechanisms with the same set of expected detection probabilities should not cost the same if they would introduce a different penalty in the performance of the nodes they would be installed in (Fig. 6.3).

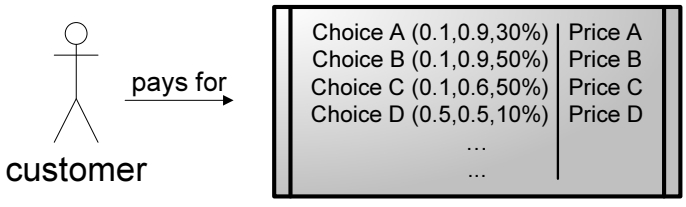

Fig. 6.3: The customer selects a defence mechanism for the nodes his traffic will use. Straightforward, but impractical in most cases.

As straight-forward as it may seem though, this paradigm of charging is based on the assumption that it is both possible and reasonable to have different defence mechanisms per node. In most cases there is no practical reason for the network's administration to offer anything less than the best available defence choice to all nodes and to all customers. It would only make sense if there were some kind of limitation in the available defence resources. This would be the case if, for example, the additional processing power used for a defence mechanism would decrease the quality of some other user's service, or if the network's administration were paying for its security needs to third parties for on-demand real-time protection (see Section 6.3).

6.2.2 Charging for higher initial mark during DoS

In Chapter 4 we presented a defence architecture in which packets undergo validity tests during a DoS attack and receive a mark according to the results of these tests. Packets with higher probability of being legitimate should receive a higher mark than those belonging to misbehaving clients or attackers, and the higher the mark the higher their priority at the defending nodes. So, if some customers wish to receive preferential treatment during a DoS attack, all the network has to do is to charge them according to the initial mark that they want their packets to enter with (Fig. 6.4). Such a charging system would have significant advantages:

- A network's customer who pays for preferential treatment, especially during an attack, is almost definitely not a DoS attacker, since in the vast majority of DoS incidents the purpose of the attack is to cause damage to the victim with minimal or no cost for the attacker. Thus, it is usually safe to consider a paying customer as a legitimate customer. Even if this does not hold, however, by choosing to increase the mark of his packets instead of simply allocating them to a high-priority queue, we make sure that these packets will also have to go through the classification

process. Then, the initial advantage that they had will not mean much if there is strong evidence that they are not legitimate, or if they are excessively harmful to the overall performance of the network.

- During a crisis, such as a DoS attack, it makes sense to try to temporarily reduce the number of customers, ideally to those few who urgently need to receive service. An example would be a website providing real-time stock exchange information. If it is under attack, then occasional visitors who do not need access to the website as much as someone who actively uses it, would not pay for preferential treatment; they would probably just leave the website, this way freeing resources for those who need them more.

Fig. 6.4: The customer pays for an increase in the DoS mark so that his packets will have higher priority during a DoS attack (see Chapter 4 for details on the DoS mark).

6.2.3 Charging according to estimated defence result

Preferential treatment in the form of personalised defence choices, higher initial marks, or higher priorities indicate but do not guarantee better performance for the customer's traffic. While providing such guarantees may be highly unlikely in today's networks, the situation could be at least improved with some sort of estimation of the results of a type of defence. In this third charging scheme we assume that there can be such an estimation, for example by applying the mathematical model we presented in Chapter 3. During a DoS attack the customer pays according to the estimation provided for the QoS criteria of his choice, such as the expected goodput (Fig. 6.5).

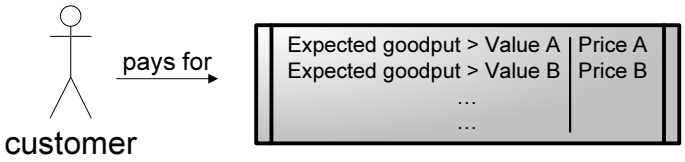

Fig. 6.5: The customer pays according to the estimated goodput during a DoS attack.

Significant weakness of the scheme, which would have to be dealt with, is the fact that not only the legitimate customers would be informed of the estimated performance of their traffic, but also the attackers. In fact, in this way the attacker would receive feedback about the impact of his traffic and could accordingly alter his attack on-the-fly to maximise its strength.

6.3 On-demand third-party DoS protection

For many organisations DoS attacks happen too rarely to justify an investment of resources on DoS defences. In Appendix C we describe how for this reason commercial DoS protection products need added value to be appealing to a large enough target group, and often have to include network monitoring, network management capabilities etc. This problem would be solved with the introduction of a complete third-party on-demand DoS solution. In accordance with the categorisation of Section 2:

Detection. The possibility of third-party detection has been mentioned in [47]. Mechanisms deploying third-party detection would rely on external messages which would signal the existence of an attack and possibly provide a characterisation of its type. These external messages would be generated by the monitoring systems of an external organisation (the third party), which would be dedicated to remote DoS intrusion detection for registered networks. These monitoring systems would also provide the processing resources for analysing the attack. In this way, not only the nodes' processors in the protected network are relieved from the detection task, but to an extend it is also ensured that the detection system is more up to date and of better quality. Such remote DoS detection could probably be achieved with techniques similar to the ones used by traceback systems (e.g. interception of a sample of packets, analysis of routers' logs, etc.).

Classification. Instead of a fully third-party classification system, which in our opinion does not seem practically viable, we could have a system in which the classification architecture exists in our network, but the actual parameters of the filters that would be used in the classification would be determined by the analysis that the third party would have carried out. For example, in the case of the classification architecture described in Chapter 4, the thresholds used in the validity tests would

be determined fully or partly by the analysis that the external detection system would carry out.

Response. Similarly to the classification, response would probably have to be only partly carried out by a third-party. The response architecture should exist in the network, but for example the limits used for the token bucket filter rate-limiting of Chapter 4 could be determined by external analysis. We could also have partial third-party response by using an external honeypot where attack traffic would be redirected for analysis or to mislead the attacker. Again, we assume that an external organisation, dedicated to the use of honeypots and analysis of DoS traffic, would be better informed and generally more suitable to carry out these tasks. However, redirecting DoS traffic to a honeypot is technically not different to attacking it, which could cause damage (overwhelming of processors and links) on the way from the network to the honeypot. This issue is not very important when the honeypot, the nodes and the links that forward the traffic, belong to the victim network, but it has implications when they do not. A solution would be to decrease the concentration of the redirected attack traffic by using more than one external honeypots, managed by the same entity, which would then be responsible for the distributed analysis that would be needed.

A third-party on-demand approach to DoS defence would have some significant advantages over today's approaches of in-house defence:

- The nodes of the network would spend much less resources for security against DoS. Detection could be fully external, and the classification and response tasks could be at least assisted by third-party analysis.

- An external organisation dedicated to a specific DoS defence task would probably be able to allocate more processing resources and would be better informed about

current attack types. This would mean more accurate analysis of an attack and consequently more accurate determination of the defence parameters.

On the other hand, some new issues would have to be dealt with:

- The fact that the defending system would be heavily dependent on the network connection between victim and third party is a serious inherent limitation of the concept. An attacker with inside knowledge of the protection system of the victim network, could severely impede the defence by first targeting the perimeter nodes through which the external messages are exchanged. A solution would be to use both external and in-house protection for all defence tasks and periodically switch between the two, so as to uncover possible efforts of an attacker to compromise one of the two systems.

- The results of the third-party analysis, in the form of external messages, would have to be delivered in a timely manner; apart from the processing delay for the analysis, the network delay between the third party and the victim network should also be taken into account.

- External messages can always be forged. Then, it is a matter of the efficiency of cryptographic techniques, authentication methods and secure network paths to decrease the probability of fake messages. Of course, the same issue exists in all distributed defence systems which use some form of control messages.

- Since for bandwidth reasons external detection would probably be based on the interception of a sample of packets per time unit, the accuracy of the external analysis would depend heavily on the suitability of the sampling mechanism.

6.4 DoS detection with Bayesian decision

The problem of distinguishing between normal and DoS traffic during an attack falls into the general category of pattern classification problems. A way of looking at this subproblem is with the help of Bayesian decision theory. The following is a brief introduction based on [15] and [4], and then a description of our initial implementation of a DoS detector and its initial results.

6.4.1 Brief introduction

The Bayesian Decision theory is a major pattern recognition technique based on a probabilistic description of the underlying features of a problem. It aims to minimise the risks encountered by the decision taking process by evaluating the various tradeoffs between decisions [15]. For a classification problem of two categories (w_1 and w_2), the use of Bayesian classifiers entails evaluating the likelihood ratio, which is the ratio of the probability density functions $\Lambda(x) = \frac{f(x|w_1)}{f(x|w_2)}$, for the measured value x of the observation variable, and comparing it with a threshold T. Then, x is assigned to category w_1 if $\Lambda(x) > T$; otherwise it is assigned to category w_2 [9].

The task of DoS detection can be considered as a two-category classification problem, where w_1 corresponds to normal network condition and w_2 to existence of DoS attack). We may use multiple Bayesian classifiers to take individual decisions for the monitored features of the traffic and combine them in an information fusion phase to detect DoS attacks in incoming traffic. In the following sections we present a first implementation of this approach, including the selection of the input features, the offline statistical information gathering and decision taking.

6.4.2 Description of detection mechanism

Fig. 6.6 shows a schematic representation of a detection mechanism based on the principles of Bayesian decision. The following is a quick summary of the step-by-step detection process:

Select the input features

- Select the decision variables which will be used as input features (bitrate, bitrate acceleration, entropy, hurst parameter, goodput, ...).

Collect statistical information on known normal and DoS traffic

- Obtain histograms for these input features with real normal and real DoS traffic [8].
- Evaluate the likelihood ratios using these histograms. Store them in the form of a new histogram for each decision variable
- Set thresholds.

Real-time decision taking

- Measure the real-time values of the decision variables from the actual traffic
- Decide based on comparing the likelihood ratios with the thresholds.

To clarify on the details of this method:

Bitrate. It refers to the total incoming bitrate that arrives to the node through all ports.

Bitrate acceleration. This is the rate of change of the total incoming bitrate that arrives to the node through all ports.

[8] An alternative to obtaining histograms is the Parzen method, which is used to estimate the density function in one dimension given a set of observations $\{x_1, x_2, ..., x_n\}$: $p(x) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x-x_i}{h})$, where $K(z)$ is the kernel function, n is the window size and h is the smoothing parameter. Usually Gaussian functions are used for $K(z)$.

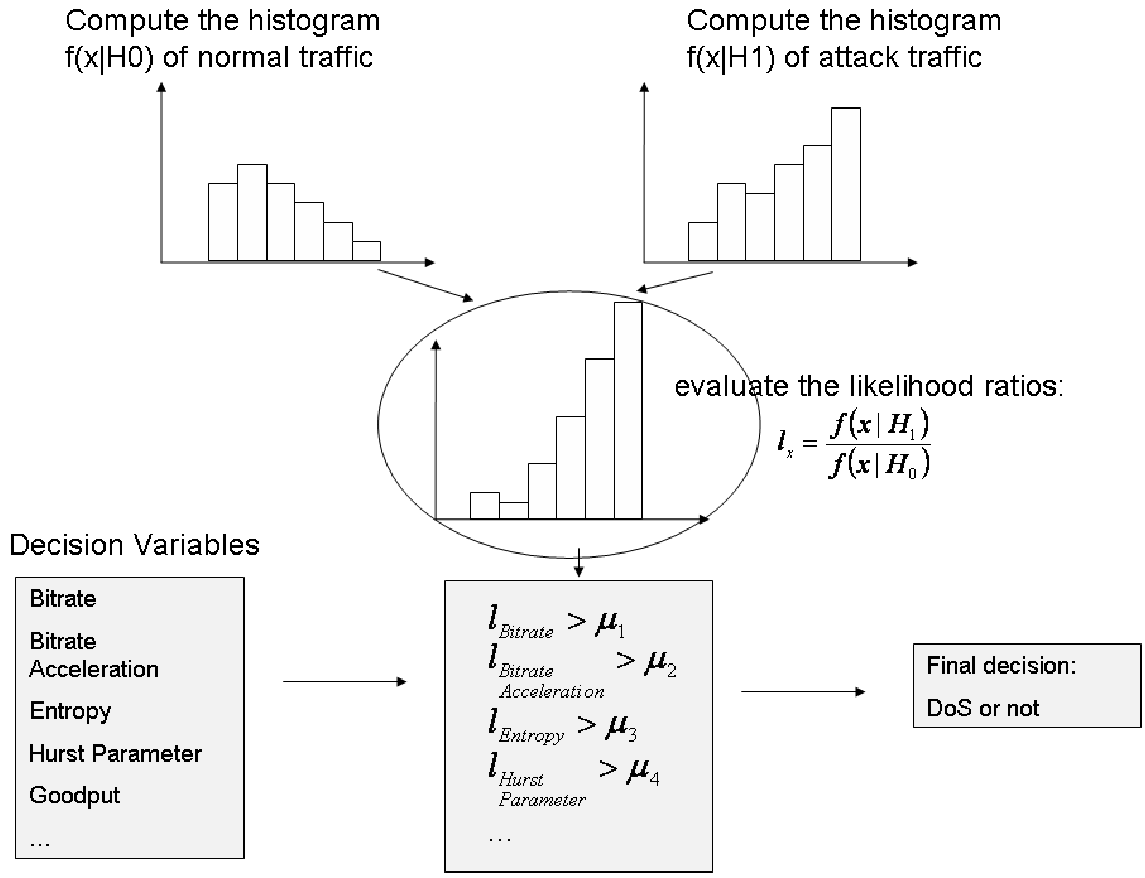

Fig. 6.6: The likelihood ratios are computed from the histograms of normal and DoS traffic, and then fed into the system to comprise its prior knowledge. With the use of this knowledge, the input variables are evaluated to produce the final decision of the detector.

Entropy. Entropy is a measure of randomness of the total incoming traffic and can be used as indication of self-similarity. In our implementation it is measured as: $S = -\sum_{i=1}^{n} P_i log_2 P_i$.

Hurst parameter. The Hurst parameter is a measure of the Long-Range Dependency (LRD) of traffic. Normal Internet traffic is supposed to be self-similar and exhibit the LRD property. Thus, entropy and the Hurst parameter may contribute to distinguishing between normal and DoS traffic. [74] and [86] suggest ways for measuring the Hurst parameter.

Goodput. As used in the rest of this thesis, the goodput is the throughput of "good" traffic, or simply the total normal incoming bitrate in a node.

Likelihood ratio. In accordance with the brief introduction to Bayesian decision of Section 6.4.1, the likelihood ratios are computed with a simple $l_x = \frac{f(x|H_1)}{f(x|H_0)}$ formula for each decision variable x.

Decision. If only one decision variable is used, then the decision is based solely on the comparison of its likelihood ratio with a corresponding pre-set threshold. If more than one decision variables are used, then a combination of them provides the final decision. In Section 6.4.4 we suggest a way of combining the partial decisions.

6.4.3 Initial numerical results and suggestions for further experimentation

In order to try this detection mechanism we set up the scenario described in Fig. 6.7. With the use of the traffic generator of Section D.3 we send for 50s normal traffic to node 110 from 107, 108 and 109. The traffic is UDP with bitrate varying in time as seen in the graphs of the same figure (the y-axis is bitrate in Mbps and the x-axis is time in sec). Then, we repeat the experiment, but in addition to this normal traffic, we also generate and send attack traffic to 110. In both cases our detection mechanism evaluates the total incoming traffic. The results of using only the bitrate, only the acceleration of the bitrate and only the entropy as decision variable, are shown in Fig. 6.8. If we consider only the bitrate to take a detection decision we have two false alarms in the "without DoS" case and several detection errors in the "with DoS" case. We obtain similar results when we consider the acceleration of bitrate as the only decision variable. The entropy, on the other hand, appears to be much more accurate, since we have no false alarms and only a slight delay before the detector starts signalling a DoS attack.

Of course, this experiment cannot be considered as a safe way to reach conclusions

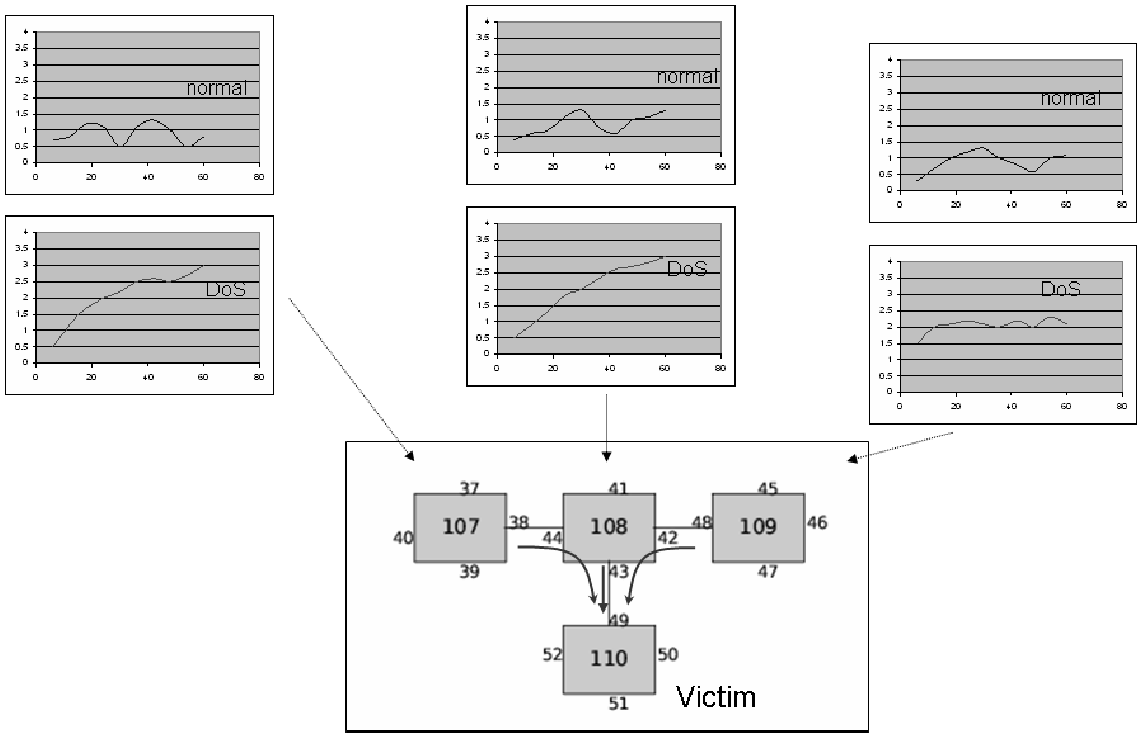

Fig. 6.7: Description of the scenario: Normal and DoS traffic directed to 110 enters the network through 107-109. The graphs show the distribution in time of the normal and DoS part of the incoming traffic for each entrance node.

on the suitability of the various detection variables, since the results are heavily based on the knowledge of the detector as obtained by the traffic used for the initial histograms. Further experimentation with various types and rates of traffic could provide valuable insights. To ensure that for these purposes we use realistic traffic, we can alter the traffic generator to use as input an actual traffic trace. Then, the difficulty is not as much in the programming side as it is in obtaining these traffic traces. We can either use traffic traces from known online sources, which are rarely available, usually limited in collected information and severely outdated, or we can develop a traffic trace tool and attempt to collect our own traffic data from computers and networks that we have access to.

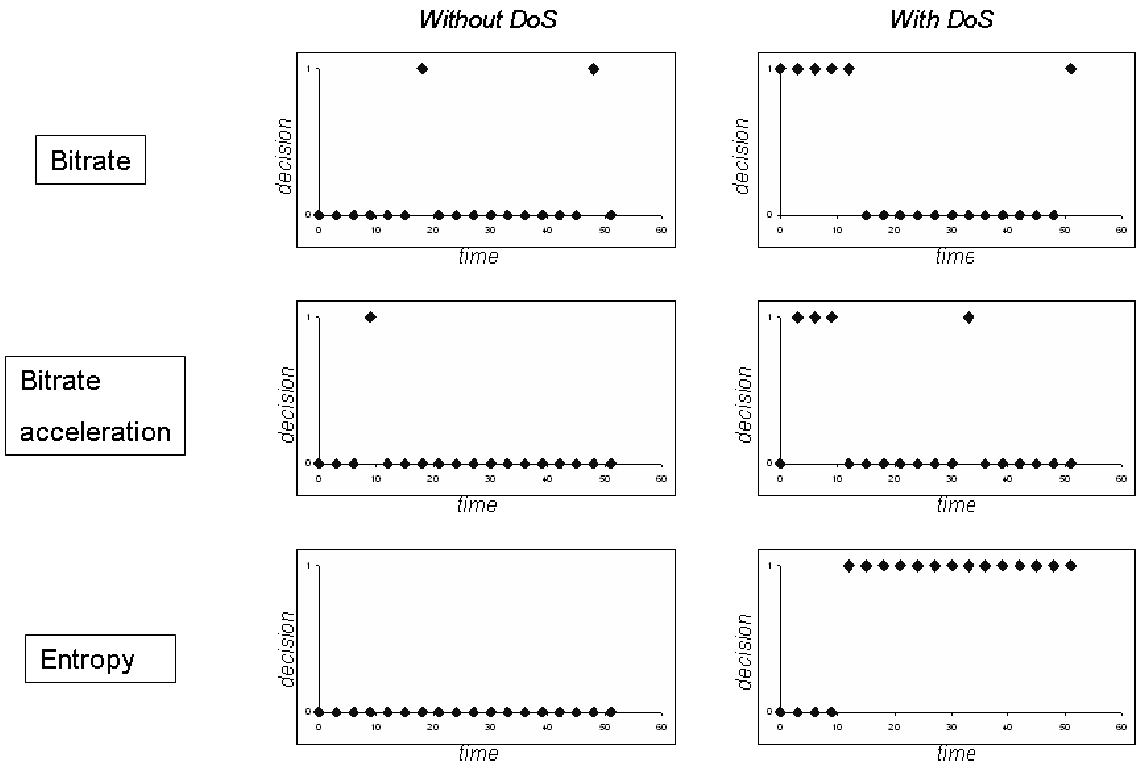

Fig. 6.8: Detection decision (1 for DoS, 0 for no DoS) Vs Time (sec). Numerical results from the application of the DoS detector on the scenario described in Fig. 6.7. The graphs on the left correspond to the case when there is no attack traffic, and the graphs on the right to the case where there is both normal and attack traffic.

6.4.4 Combining the decisions

Since we are in position to measure the values of several different detection variables it would be reasonable to reach a detection decision by combining the outcome of the detection process for each one. This 2nd-level decision taking can be implemented with a multi-layer neural network as seen in Fig. 6.9.

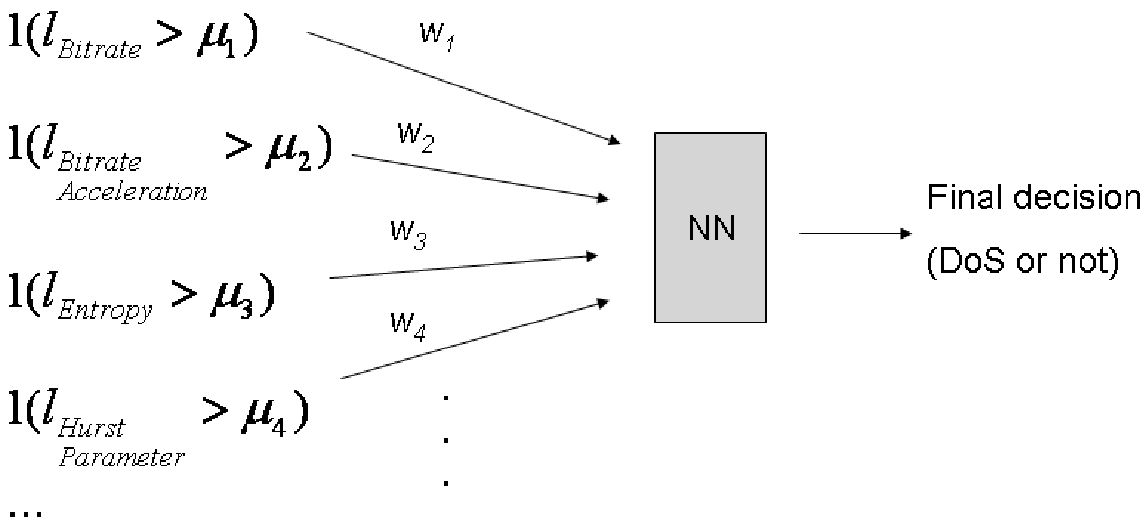

Fig. 6.9: 2nd-level decision taking with the use of a multi-layer neural network. The boolean outcomes of the comparisons of all likelihood ratios with their corresponding thresholds are combined to give a single decision outcome.

7. CONCLUSIONS

7.1 Summary of the DoS problem

As our life is becoming increasingly dependent on computer networks, the fact that malicious people can deliberately severely degrade or altogether deny us our legitimate network service is becoming more than merely a nuisance. Until a few years ago, such attacks were not involved in high-profile incidents, but today they are often used for cyber-extortion, to disrupt a market competitor's internet presence, and are feared to soon be used as means of asymmetric warfare. Denial of Service attacks are practically always distributed: the attacker takes control of a large number of lightly protected computers and orders them to simultaneously send volumes of meaningless traffic to a specific target. As a result, some routers and links in the vicinity of the target are overwhelmed, and a number of legitimate clients cannot connect to it anymore.

The aim of a DoS defence mechanism should be to retain the quality of service of the users of a network resource to the level it would be without the attack. This becomes particularly difficult when the attacker has compromised and uses a vast number of computers and manipulates its packets to make them appear as if they originate from a different source.

7.2 Summary of our contributions

7.2.1 DoS response by dropping packets which are identified as probably illegitimate

We provided a mathematical model which analyses the effectiveness of this most significant family of defence approaches and estimates the impact of the attack on the network, as a function of measurable parameters. We validated its results by comparing them with simulations and experiments. We also implemented a defence approach which exploits the unique advantages of a Self-Aware Network. To demonstrate its success we used it to retain the image quality of a sensitive video streaming application, which when left undefended dramatically deteriorated.

7.2.2 DoS defence with the combination of prioritisation and rate-limiting

We proposed a complete DoS defence architecture, in which practically all the traffic of the legitimate users arrives at the destination. Packets with higher probability of being both valid and relatively harmless for the network are served with higher priority. Packets which would be marginally classified as invalid may receive service if there is available bandwidth, this way minimising the potential collateral damage inflicted in case of false classification. Meanwhile, invalid packets or valid packets which at the specific time would be excessively harmful for the network, are served with lower priority, which essentially means that they are either delayed or dropped due to buffer overflow.

7.2.3 Allocating defence tasks in a distributed defence architecture

Using all available nodes to defend during a DoS attack is undesirable as this practice may impair the normal operation of the network, mainly due to the performance penalty that defence tasks impose on the nodes. We presented a way of numerically computing

the performance of the various possible allocations of defence tasks and suggesting the optimal one given a user-defined desired number of defenders. We also described how the same tool can be used to identify inherent weaknesses of a given topology in terms of the vulnerability of specific nodes.

7.2.4 Applications of our work on novel research problems

Based on the knowledge of the specific field that we acquired through our work and through the literature survey, we identified new problems that have not been suggested before, and described solutions based on the work presented in this thesis. These include a defence task of evidence collection for law-enforcement, the concept of charging for DoS defence, the on-demand third party defence, and a detection method based on bayesian decision.

7.3 Open issues in our work

Despite the very good performance that our approaches demonstrate, there is still a lot of space for improvement:

1. Just like the CPN protocol they were implemented in, the defence mechanisms presented in this work are oriented towards networks of relatively small size. CPN is not designed for complete deployment, but to be used in small networks distributed around the internet, which can act as islands of defence. Still, it would be interesting to investigate how the methods presented in this dissertation could be extended beyond this scope. For example, a limitation would be the scalability of the current design and implementation of the classification mechanism of Chapter 4 and the optimal distribution tool of Chapter 5.

2. Although in our mathematical model we incorporate the notion of overhead with

the variable average service time, we do not directly measure it in our experiments with the implemented defence schemes.

3. A wider variety of input traffic could be used both in terms of protocols encapsulated in CPN, such as TCP, and actual type of application, such as VoIP. The presented defence mechanisms could prove to be more effective in some types of traffic than others.

4. The stability and robustness of our mechanisms when deployed in a network should be assessed. Furthermore, an important issue that can be raised in all proposals of DoS defence, is whether and how the actual mechanism could be targeted by an attacker, with the purpose of decreasing its stability and robustness or even neutralising it altogether.

7.4 Final remark

Denial of Service attacks are expected to become even more common and threaten every aspect of communications, as with time their use is shifting from simple nuisance to acts of crime and war. Fortunately, this is followed by an explosion of scientific research to understand this intriguing phenomenon, which exploits today's imperfect automation of communications together with flaws of the human nature. The purpose of the work presented in this thesis was to design and analyse methods to fix the former.

APPENDIX

A. DETAILED HISTORIC TIMELINE OF DENIAL OF SERVICE INCIDENTS

November-December 1996. On the 6th of September 1996, a "SYN Flood" DoS attack took the New York City ISP Panix offline for a whole week, while subsequent attacks disabled the web servers of the Internet Chess Club and The New York Times. Two months later, Internet Security Systems released "Realsecure", one of the first products specifically designed to detect and halt such attacks. It watched for SYN packets coming in, and if a victim computer received more than a certain threshold in a set time period, it would reset those connections. However, against December's attack on Webcom's server, which knocked offline thousands of commercial websites, "Realsecure" failed. The attacker had randomised the IP addresses and the attack rate was the very high at the time, 200 packets/sec. The technique was around since about 1991, but the initial blame was laid on two major hacker magazines, "Phrack" and "2600", which both had published code for a simpler "SYN Flood" script. *(Source: Wired News - "Webcom Security Software Failed in Server Attack",*

http://www.wired.com/news/technology/0,1282,1052,00.html).

January 1997. For reasons of revenge and only, a Romanian Teenager attacked the IRC [1] network Undernet and several ISPs (in Norway, Romania, the United Kingdom, and the United States) with a combination of "ping attack" and "SYN Flood". At each stop, he logged onto the server, obtained root access, then deleted files and cancelled accounts. *(Source: Wired News - "Romanian Cracker Takes Down the Undernet", http://www.wired.com/news/technology/0,1282,1446,00.html).*

January 1998. DALnet and some other IRC networks became targets of "smurfing", the newest type of DoS attack. *(Source: Wired News - "Smurfing Cripples ISPs", http://www.wired.com/news/technology/0,1282,9506,00.html).*

February-March 1998. The Pentagon, NASA, several American military network systems, and hundreds of Universities were targeted by a 18-year-old Israeli, nicknamed "Analyzer". The hacker used mainly "Teardrop" and "Bonk" (or "Boink" or "new Tear") techniques, which exploited known vulnerabilities of the Microsoft Windows operating systems, and succeeded only against those computers that were not up to date with the latest security patches. *(Source: Wired News - "Pentagon Hacker Exposed by Justice Department", http://www.wired.com/news/technology/0,1282,11030,00.html).*

February 2000. The websites of Yahoo, eBay, Amazon, Datek, Buy, CNN, ETrade,

[1] Internet Relay Chat (IRC) is a form of instant communication over the Internet. It is mainly designed for group (many-to-many) communication in discussion forums called channels, but also allows one-to-one communication. An IRC server can connect to other IRC servers to form an IRC network [108]. IRC networks allow their users to create public, private and secret channels, with the latter being frequently used in denial of service attacks.

ZDNet and Dell were among the targets of a 15-year old Canadian nicknamed "Mafiaboy". The attack, which reached the rate of 1GB/s wreaked havoc in the victims' economy and changed the way people think about DoS. After that incident, DoS and in general Internet crime, started moving from the IRC networks to a world with much higher profile, the **e-Commerce**. "Mafiaboy" was sentenced in September 2001 for the total damage of $1.7 billion that he caused. *(Source: BBC - "Mafiaboy hacker jailed", http://news.bbc.co.uk/1/hi/sci/tech/1541252.stm*).

July 2000. BT.com, BTInternet.com, and Gameplay.com, were attacked by a frustrated anonymous customer, as revenge for bad service. That was one of the first occasions that DoS was used as a form of **protest**. *(Source: The Register - "Wanna know how BT.com was hacked?", http://www.theregister.co.uk/2000/07/25/wanna_know_how_bt_com/*).

January 2001. Register.com was one of the first victims of a new type of attack, the Distributed Reflector DoS attack, with a reported rate of about 60 to 90 Mbps [59]. A few months later Paxson et al published the first and definitive paper on the analysis of that new type [21]. Still, the most well-known DRDoS attack was probably the one against GRC.com in 2002, mainly because of the fact that the company's owner recorded and published the full chronicles of the attack in his website, something extremely rare on the part of the victims of such attacks. *(Source: GRC.com - "The Stranger Tale of the Denial of Service attacks against GRC.com", http://grc.com/dos/grcdos.htm*).

July-August 2001. A computer worm of Chinese origin, known as "Code Red" swept

through more than 250,000 computers in nine hours, with the infected computers being programmed to simultaneously attack the website of the White House on a specific date. The White House was forced to change its numerical IP Web address, and prompted the Pentagon to take its public websites off-line temporarily. The damage inflicted by "Code Red" was estimated around $2.6 billion. *(Source: CNN - "New 'Code Red' worm entices Web hijackers", http://edition.cnn.com/2001/TECH/internet/08/06/code.red.two/).*

September 2001. A British teenager was accused of launching a DDoS attack on the Port of Houston's IT systems, which rendered the Port's web service inaccessible for several hours. The attack, which was traced to a computer at his home, was allegedly aimed at a chatroom user, and the Port's computers were used, among others, as "zombies" for the attack. The teenager claimed that his computer had been taken over by a hacker using a Trojan Horse program and he was found not guilty. Although no injury or damage was caused, the attack could have had catastrophic repercussions, since the server contained data on navigation, tides, water depths and weather. That was supposed to be the first time that part of a country's **national infrastructure** was disabled by an electronic attack. *(Source: BBC - "Teenager cleared of hacking", http://news.bbc.co.uk/1/hi/england/hampshire/dorset/3197446.stm).*

June 2002. The website of the government of Pakistan was the victim of a **politically motivated** attack launched by Indian hackers using "YAHA", a worm which contained Denial of Service payload. Similarly to "Code Red", "YAHA" caused an infected computer to make repeated connection attempts to the Pak-

istan government site, and attempted to terminate anti-virus and firewall software. *(Source: SecurityFocus - "Yaha Worm Takes Out Pakistan Government's Site", http://online.securityfocus.com/news/501)*.

October 2002. In October 2002 the DNS root servers [2] were under a distributed DoS attack for about an hour, causing several servers to stop being available to regular Internet traffic. However, the remaining root servers withstood the attack and ensured that the Internet's overall performance was not degraded. Although huge, the attack was hardly noticeable to the average end-user. Nonetheless, this was the most serious hacker attack ever on this key piece of the Internet infrastructure, and an eye-opener for the root-server operators, since without the DNS root servers, the Internet cannot function. *(Source: Network World Fusion - "DDoS attack highlights 'Net problems", http://www.nwfusion.com/news/2002/1028ddos.html)*.

A year later most of the 13 root servers had applied a new routing technique known as Anycast, with which their operators are replicating these servers around the world. *(Source: Network World Fusion - "Net security gets root-level boost", http://www.nwfusion.com/news/2003/1027ddos.html)*.

December 2002. "Spam King" Alan Ralsky became the victim of a unique DoS attack. During an interview, in which he supported spamming, he made the mistake to mention where he lived. His exact address was then posted on the Internet and several Internet users subscribed him to thousands of catalogs and mailing lists. For weeks he was bombarded with so much junk mail that he was unable to find

[2] Named by the letters A through M, they are operated by U.S. government agencies, universities, nonprofit organisations and companies such as VeriSign. Of the original 13 root servers, 10 are located in the U.S., one in Asia and two in Europe.

his real mail. *(Source: The Mac Observer - "Spam King Inundated By Junk Mail', http://www.macobserver.com/article/2002/12/06.11.shtml).*

A few months later Byers et al published a paper describing how to automate such an attack using simple scripts and suggested methods of protection against it, involving the relevant websites and the Post Office [65].

January 2003. The South Korean stock market reported disruptions caused by the "SQL Slammer" worm. The virus hit South Korea particularly hard because it had the world's highest penetration of broadband Internet services, while no more than 40% of South Korean firms had installed any kind of firewall. The losses were estimated around US$860,000, and there was immediate response by investors against the shares of the key Internet service provider, which fell by 3.3%. *(Source: BBC - "South Korean markets hit by net worm", http://news.bbc.co.uk/1/hi/business/2698385.stm).*

Months later (August 2003), the same worm caused a 5-hour outage to the safety monitoring system of the nuclear power plant at Ohio. *(Source: SecurityFocus - "Slammer worm crashed Ohio nuke plant network", http://www.securityfocus.com/news/6767).*

March 2003. The Arabic and English-language sites of the satellite television network Al-Jazeera suffered two days of Internet disruptions caused by American hackers as revenge for showing pictures of dead and captive American soldiers in Iraq. *(Source: BBC - "Hackers cripple Al-Jazeera sites", http://news.bbc.co.uk/2/hi/technology/2893993.stm).*

August 2004. A corporate executive in Massachussets was charged with using DoS

attacks to cause a total of $2 billion in losses to three of his main competitors. The attacks had begun in October 2003 and were mainly SYN and HTTP Floods. *(Source: SecurityFocus - "FBI busts alleged DDoS Mafia", http://www.securityfocus.com/news/9411)*.

January 2006. The www.milliondollarhomepage.com, a British teenager's novel advertising idea to earn $1m in 4 months, became very quickly famous around the world. This instant media attention drew the equally quick attention of cyber-extortionists, who bombarded the website with intense DoS attacks, initially asking for $5,000 to avert them and then $50,000 to stop them. The website could not "breathe" for a whole month. *(Source: BBC - "Blackmailers target $1m website", http://news.bbc.co.uk/1/hi/technology/4621158.stm)*.

May 2006. A 20-year old "botmaster" was sentenced to five years in prison for hijacking 500,000 computers. He was selling access to this army of bots to other hackers, who used them to launch Dos attacks and send waves of spam emails. *(Source: Reuters - "'Botmaster' gets nearly five years in prison", http://www.msnbc.msn.com/id/9918275)*.

B. THE COGNITIVE PACKET NETWORK

The Cognitive Packet Network (CPN) [119]-[124] is a packet routing protocol which addresses QoS using adaptive techniques based on on-line measurements. In CPN, users declare their QoS requirements (QoS Goals) such as minimum delay, maximum bandwidth, minimum cost, etc.

CPN makes use of three types of packets: smart packets (SP) for discovery, source routed dumb packets (DP) to carry payload, and acknowledgements (ACK) to bring back information that has been discovered by SPs which are used in nodes to train neural networks. A node in CPN acts as a storage area for packets and mailboxes (MBs). It also stores the code used to route smart packets. It has an input buffer for packets arriving from the input links, a set of mailboxes, and a set of output buffers which are associated with output links. Smart packet routing is carried out using a reinforcement learning (RL) algorithm based on Random Neural Networks (RNN). The algorithm code is stored in each router and its parameters are updated by the router. For each successive smart packet, the router computes the appropriate outgoing link based on the outcome of this computation.

Conventional IP packets may tunnel through CPN to seamlessly operate mixed IP and CPN networks. SPs are generated either by a user requesting to create a path to some CPN node, or by a user requesting to discover parts of network state, including location of certain fixed or mobile nodes, power levels at nodes, topology, paths and their QoS metrics.

SPs discover routes by using random neural networks (RNN) [117] with reinforcement learning (RL). RL is carried out using a QoS Goal (such as packet delay, loss, hop count, jitter, etc) which is defined by the user who generated a request for the connection, or by the network itself. The decisional weights of a RNN are increased or decreased based on the observed success or failure of subsequent SPs to achieve the Goal. Thus RL will tend to prefer better routing schemes, more reliable access paths to data objects, and better QoS.

When a Smart Packet arrives to its destination, an ACK is generated and heads back to the source of the request, following the reversed path of the SP. It updates mailboxes (MBs) in the CPN nodes it visits with the information which has discovered, and provides the source node with the successful path to the node. All packets have a life-time constraint based on the number of nodes visited, to avoid overburdening the system with unsuccessful requests or packets which are in effect lost. A node in the CPN acts as a storage area for packets and mailboxes (MBs). It also stores and executes the code used to route smart packets. It has an input buffer for packets arriving from the input links, a set of mailboxes, and a set of output buffers which are associated with output links. The route brought back by an ACK is used as a source route by subsequent DPs of the same QoS class having the same destination, until a newer and/or better route is brought back by another ACK.

Each node stores a specific RNN for each active source-destination pair, and each QoS goal. The number of neurons in an RNN corresponds to the number of routing outputs of a node. Each output link of a node is represented by a neuron in the RNN. The arrival of Smart Packets(SPs) triggers the execution of RNN and the routing decision is the output link corresponding to the most excited neuron. CPN reinforcement learning changes neuron weights to reward or punish a neuron according to the level of goal satisfaction measured on the corresponding output.

The level of goal satisfaction is expressed by a reward. Given some goal G that a packet has to minimize, the reward R is formulated simply as $R = 1/G$. The state q_i of ith neuron in the network is the probability that it is excited. The q_i, $1 < i < n$ satisfy the following system of nonlinear equations:

$q_i = \frac{\lambda^+(i)}{r(i)+\lambda^-(i)}$ where $\lambda^+(i) = \sum_j q_j w^+_{ji} + \Lambda_i$ and $\lambda_-(i) = \sum_j q_j w^+_{ji} + \lambda_i$

w^+_{ji} is the rate at which neuron j sends "excitation spikes" to neuron i when j is excited, w^-_{ji} is the rate at which neuron j sends "inhibition spikes" to neuron i when j is excited, and $r(i)$ is the total firing rate from the neuron i. For an n neuron network, the network parameters are these n by n "weight matrices" $W^+ = \{w^+(i,j)\}$ and $W^- = \{w^-(i,j)\}$ which need to be "learned" from input data.

The RNN weights are updated based on a threshold T:

$T_k = \alpha T_{k-1} + (1-\alpha)R_k$

where R_k, k = 1, 2, ... are successive measured values of reward R and α is some constant ($0 < \alpha < 1$) that is used to tune the responsiveness of the algorithm: for instance $\alpha = 0.8$ means that on the average five past values of R are being taken into account. Neurons are rewarded or punished based on the difference between R_k, the current reward, and T_{k-1}, the last threshold.

If for example the QoS goal G is the hop count, and the reward function R is: $R = \frac{1}{\beta \cdot G}$, G can be easily measured by SPs by incrementing a counter that increases each time the SP visits another node. The value of G related to a SPs path from a given node to the destination is brought back to that node by the SPs corresponding ACK, and is then stored in the nodes MB. Successive values of R denoted by R_k, $k = 1, 2, ...$ are used first to compute the decision threshold T.

Suppose that the RL algorithms the kth decision was to select output link (neuron) j, and that the kth reward obtained via an incoming ACK is R_k. If R_k is larger than, or equal to, the threshold T_{k-1}, then the excitatory weights going into neuron j are

increased significantly and the inhibitory weights leading to other neurons are only increased by a small amount. On the other hand if R_k is less than T_{k-1}, then we increase moderately the excitatory weights leading to all neurons other than j and increase significantly the inhibitory weight leading to neuron j, in order to punish it for not being successful:

If $T_{k-1} \leq R_k$

$w^+(i,j) \leftarrow w^+(i,j) + R_k$

$w^-(i,l) \leftarrow w^-(i,kl) + R_k/(n-1), for l \neq j$

Else

$w^+(i,l) \leftarrow w^+(i,l) + R_k/(n-1), for l \neq j$

$w^-(i,j) \leftarrow w^-(i,j) + R_k$

The probabilities q_i are then computed and the next SP will be forwarded to the output link which corresponds to the neuron which has the largest excitation probability. In CPN QoS metrics can be combined, eg. the hop count metric can be combined with the forward delay, so that the goal takes into account both the length H and the delay D of the path: $G = H + \gamma D$, where γ can be taken as: $\gamma = 10.3$ so as to bring D (μs) and H into a mutually comparable range of values. Some of the QoS Metrics which are or can be used in CPN are the following:

Delay. The sum of transmission times, queueing delays, and processing delays among all the intermediate hops of a given path. Delay is an additive metric, thus, the delay of a path is the sum of the delays on each of its hops and links. In most cases the transmission and processing delays can be considered negligible,comparing to the queueing delay.

Packet Loss. Loss of packets occurs when at least one of the packets does not reach its destination. Losses occur mainly due to congestion. Other causes are transmis-

sion or configuration errors, timeouts, the packet size, Denial os Service (DoS) Attacks, viruses, etc. Packet loss is a multiplicative metric, so, the overall loss for a path is the product of the loss of each hop and link.

Jitter (Delay Variation). Packets from the same source reach the destination with generally different delays. This variation in time intervals between packets is called jitter. Depending on the network protocol used, sudden changes, or better sudden increases, in jitter may cause packet losses. Jitter is an additive metric, thus, the delay variance of a path is the sum of the jitters on each of its hops and links.

Available Bandwidth. The minimum bandwidth a call requires or the spare bandwidth of the network which can be used by the existing or new users. The bandwidth of the path as a whole is determined by the link with the minimum available bandwidth, which makes it a concave metric.

Hop count. The number of hops a packet visits until it reaches its destination. It is used in routing policies that base their routing on finding the shortest paths. Hop count is an additive metric.

Reliability. Reliability of a network is its ability to continue provide the service that a client requires for successful operation during network failures or systematic attacks. The reliability of a link could be a function of the physical reliability of the link (eg fiber, copper), the type of the medium (e.g. wireline, wireless) or the packet loss probability due to long-term congestion. Reliability is a multiplicative metric, thus the reliability of a link is the product of each links reliability.

Security. Users, such as those that make online transaction, require a level of security in order to make a call through a network. Applications specify a minimum security level that they wish from the network. Security is a concave metric.

Cost. The cost of a link is determined by the total resource consumption at the link and the node. Cost of a path is the total cost of all links on the path, so cost is an additive metric.

Fig. B.1 shows the structure of a CPN packet:

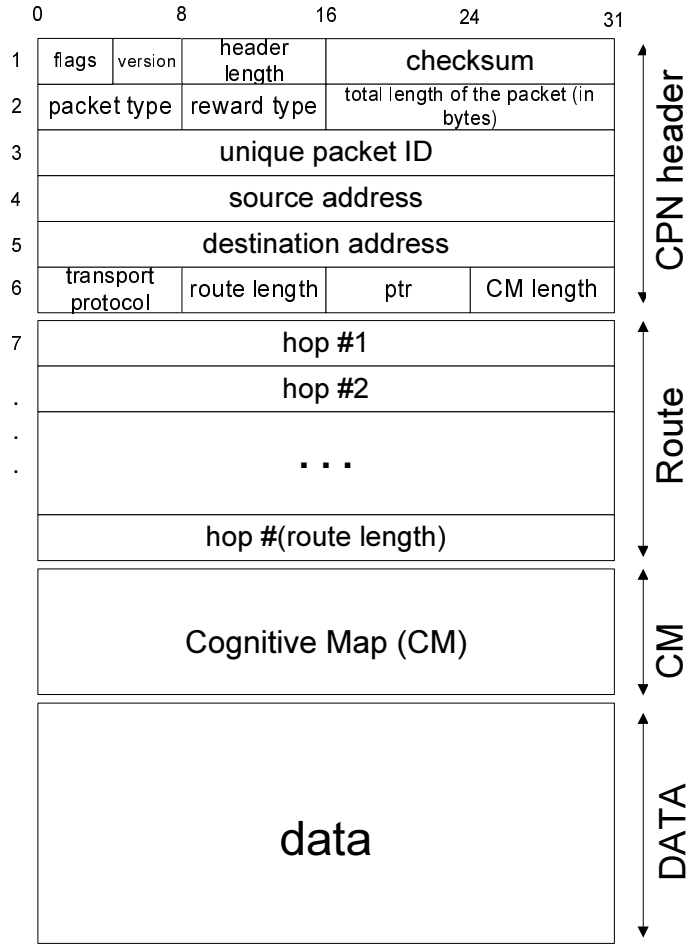

Fig. B.1: The scructure of a CPN packet

flags, version (1 byte). This is the first byte of the CPN part of the header. The flags are used by the dos mark described in Chapter 4 (first 3 bits) and the CPN tunnelling code (last bit), while the version is currently always set to 0001. An example for this first byte could be 01100001 or in hex 61, which means mark = 3, tunnelling

bit = 0, version = 1. Note that the flags field is copied to the acknowledgment (dack or sack).

header length (1 byte). It is the number of 32-bit words comprising the header (the whole header, not only the CPN header). A typical value would be 0f, which means 15 words = 60 bytes. It is used to mark the beginning of the data.

checksum (2 bytes). The checksum is the typical redundancy check for detecting errors in the transmission to protect the integrity of data.

packet type (1 byte). Set to 0 for dumb packets, 1 for smart packets, 2 for acknowledgments of dumb packets (dack), 3 for acknowledgments of smart packets (sack).

reward type (1 byte). This is the QoS metric that the CPN routing protocol will use. It is set to 0 for delay, 1 for number of hops, 6 for loss.

total length (2 bytes). The total length of the packet (in bytes). For example, a dumb packet should be about 1068 bytes (in hex 042c), while a smart packet should be something like 69 bytes (in hex 003c).

unique packet ID (4 bytes). A unique ID for each packet. Its acknowledgment (dack or sack) will have the same ID.

source address (4 bytes). E.g. 0.0.0.105 in hex will be 00000073.

destination address (4 bytes). As for the source address.

transport protocol (1 byte). The transport protocol number.

route length (1 byte). Gives the number of route entries (each one being 32 bits).

ptr (1 byte). An index pointing into the route.

CM length (1 byte). The length of the cognitive map (See Cognitive Map (CM)) in 32-bit words.

hops. The routing history of the packet is stored in the header. This is why the CPN protocol provides with automatic traceback of all packets within a network that operates it.

Cognitive Map (CM). This is where the packets store the information that they use to take their routing decisions.

data. This is the actual data payload of the packet. It comprises the largest part of the packet.

C. COMMERCIAL DOS PROTECTION SOLUTIONS

Since Denial of Service attacks are a very real threat, often causing significant financial damage to companies, they are not a field of academic only research. Since the late 1990s several commercial DoS protection products have appeared, which due to the nature of the market, were designed according to different and very specific requirements. Although not strictly mentioned anywhere they seem to be all or some of the following:

- While academic researchers try to gain further insight in the Denial of Service phenomenon, ultimately aiming at general and complete solutions, the success of a commercial product is determined "on the field". **It simply has to work** those few times that the customer's network is under attack.

- **Avoiding collateral damage** has the highest priority, since denying service to a client results in both short-term and long-term financial losses (the client will try to find the same product in another online shop or may not visit this shop in the future due to the bad quality of service he experienced). In the terminology of this thesis, the probability of false alarm needs to be minimal, possibly regardless of the probability of correct detection.

- Although particularly harmful, a DDoS attack is a relatively rare situation. Especially smaller companies are usually not willing to buy an expensive product built specifically and only to counter such attacks. At the beginning DoS attacks were usually considered by the network security industry as one more intrusion case,

for which naturally intrusion detection products should be responsible. However, partly because of their signature-based character and mainly because of the intricacies of Denial of Service, both old and newer intrusion detection solutions have proven unsuccessful at countering DoS. Thus, since it has to be DoS-specific to be effective, and since **it needs added value** to be appealing to a large enough target group, a DoS protection product often includes network monitoring and management capabilities, with all the additional research and implementation effort that these may require.

C.1 Cisco Systems: Cisco Guard

The goal of Cisco Guard [111] is to detect the presence of a potential DDoS attack, divert traffic destined for the targeted device, and identify and block malicious traffic in real time, without affecting the flow of legitimate, mission-critical transactions. The Cisco guard works together with a separate product, the Cisco Traffic Anomaly Detector (TAD) early warning system, which passively monitors network traffic, looking for any deviation from "normal" or baseline behaviour that indicates a DDoS attack. When an attack is identified, the Cisco TAD alerts the Cisco Guard, providing reports and specific alerts to react to the threat. For example, the Cisco TAD observes that the rate of UDP packets from a single source IP is out of range, even if overall thresholds are not exceeded. The Cisco Guard is deployed upstream at either the ISP data center or at the perimeter of a large enterprise to protect both the network and data center resources. When it is notified that a target is under attack (whether from a Cisco TAD or some other security-monitoring device such as an intrusion detector or firewall), traffic destined for the target is diverted to the Guard (or Guards) associated with the targeted device. The traffic is then subjected to a five-stage analysis and filtering process to

remove malicious traffic while allowing good packets to continue flowing uninterrupted. It resides adjacent to a router or switch on a separate network interface, so as to enable on-demand protection. Depending on its location, the Cisco Guard may protect routers, Web servers, DNS servers, and LAN and WAN bandwidth. The protection architecture used is the proprietary Cisco Multi-Verification Process (MVP), which features (Fig. C.1):

- Filtering. This module includes both static and dynamic DDoS filters. Static filters, which block nonessential traffic from reaching the victim under attack, are user-configurable, and they come from Cisco with preset default values. Dynamic filters are inserted by the other modules based on observed behaviour and detailed analysis of traffic flows, delivering real-time updates that either increase the level of verification applied to suspicious flows or block sources and flows that have been verified as malicious.

- Active verification. This module verifies that packets entering the system have not been spoofed. The Cisco Guard XT uses several source-authentication mechanisms to stop spoofed packets from reaching the victim and mechanisms to help ensure proper identification of legitimate traffic.

- Anomaly recognition. This module monitors all traffic that was not stopped by the filter or the active verification modules and compares it to baseline behaviour recorded over time, looking for deviations that would identify the source of malicious packets. The basic principle behind the operation of this module is that the pattern of traffic originating from a “black-hat” daemon residing at a source differs dramatically from the pattern generated by legitimate sources during normal operation. This principle is used to identify the attack source and type, as well as to provide guidelines for blocking traffic or performing more detailed analysis of

the suspected data.

- Protocol analysis. This module processes flows that anomaly recognition finds suspicious in order to identify application-specific attacks, such as HTTP error attacks. Protocol analysis then detects any misbehaving protocol transactions, including incomplete transactions or errors.

- Rate limiting. This module provides another enforcement option and prevents misbehaving flows from overwhelming the target while more detailed monitoring is taking place. The module performs per-flow traffic shaping, penalising sources that consume too many resources (for example, bandwidth or connections) for too long.

Between attacks, the Cisco Guard XT is in "learning mode", passively monitoring traffic patterns and flow for each of the different resources it protects to understand normal behavior and establish a baseline profile. This information is later used to fine-tune policies for recognizing and filtering both known and unknown attacks in real-time.

An exaple deployment for protection in an ISP environment is shown in Fig. C.2.

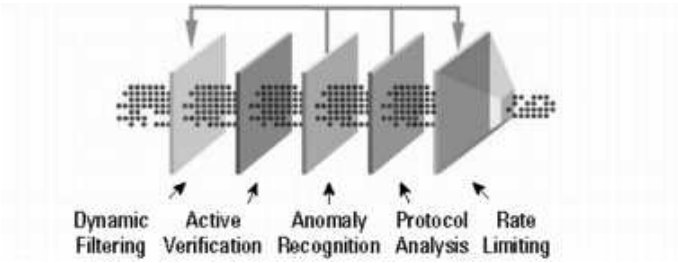

Fig. C.1: The Cisco MVP architecture

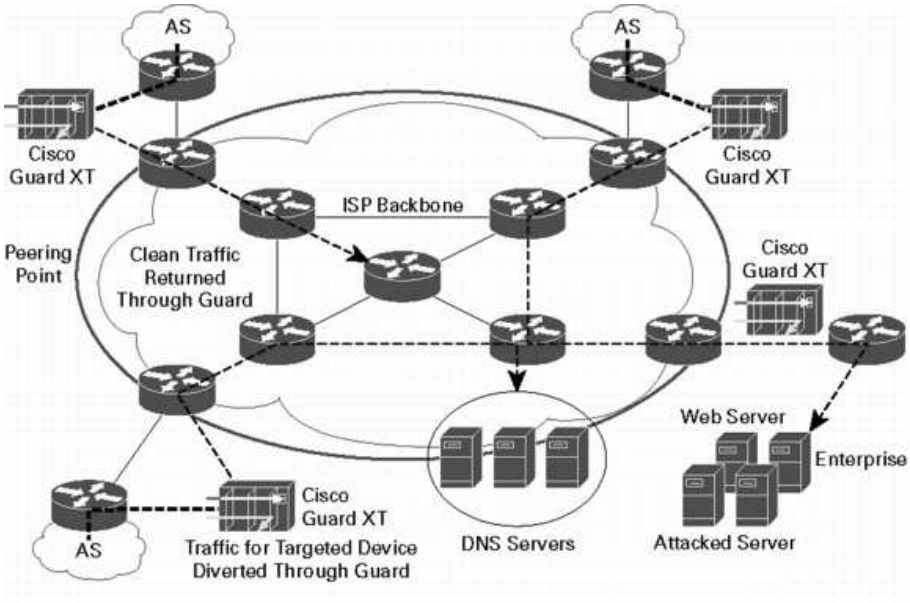

Fig. C.2: Cisco Protection in an ISP Environment. Traffic Destined for Targeted Device Is Diverted to Cisco Guard XTs; Clean Traffic Is Returned to the System.

C.2 Mazu Networks: Mazu Enforcer

Mazu Enforcer claims to be a behaviour-based system designed specifically to protect against denial of service attacks, protecting key assets while allowing legitimate traffic to flow. The Mazu Enforcer system consists of an appliance deployed at the perimeter that continuously monitors data about traffic through the existing router links. It uses the data to build and maintain a real-time baseline of filterable packet parameters and develops filter candidates for each parameter. It identifies meaningful changes in traffic to detect an attack, looking at the following three main categories of triggers:

- Bandwidth. Detect attacks that attempt to flood a network's bandwidth based on packet and byte rates.
- Suspicious traffic. Detect attacks that target resources on the network, such as TCP SYN attacks.

- Custom-defined. Monitor specified characteristics unique to a particular environment; for example, if a specific set of networks has historically been the source of DoS attacks for the network, a custom-defined trigger can be created to monitor for even a small amount of traffic coming from those networks.

Once an attack has been detected, the product computes candidates for each filter, evaluates each filter to assess the optimal combination of highest impact and lowest collateral damage, and recommends the broadest filter that successfully mitigates the attack while meeting user-specified impact thresholds. The available filters are:

- Packet attribute. In addition to the standard filters that are automatically created based on the characterisation of the attack, users can specify precise custom filters for highly specific conditions.
- SYN rate limiting. Optionally enables the rate of incoming SYN packets to be limited to a configurable threshold
- Behavioural. Observations about the behaviour of external host provides an additional layer of protection upstream from the packet attribute filters.
- Signature. Optionally monitors contents of packet payloads.

C.3 Arbor Networks: Peakflow

The two Peakflow products (X for enterprises and SP for Service Providers) rely on network information generated by NetFlow [1]. They include [113]:

[1] Netflow is an open protocol developed by Cisco for collecting IP traffic information and is built into routers and switches from Cisco and other vendors. It allows network components to capture information about traffic flows. For each flow, Netflow captures the source and destination IP address and port, the type of protocol used, the type of service provided, and the logical interfaces for the flow [112].

- Stateful Flow Reassembly, which manages the raw NetFlow data to provide information meaningful for DoS protection.
- Anomaly Detection. This is targeted not only against attacks from external sources, but also against employee misuse.
- Safe Quarantine, which essentially helps apply filters to allow legitimate traffic through.

An example deployment of Peakflow X is provided in Fig. C.3.

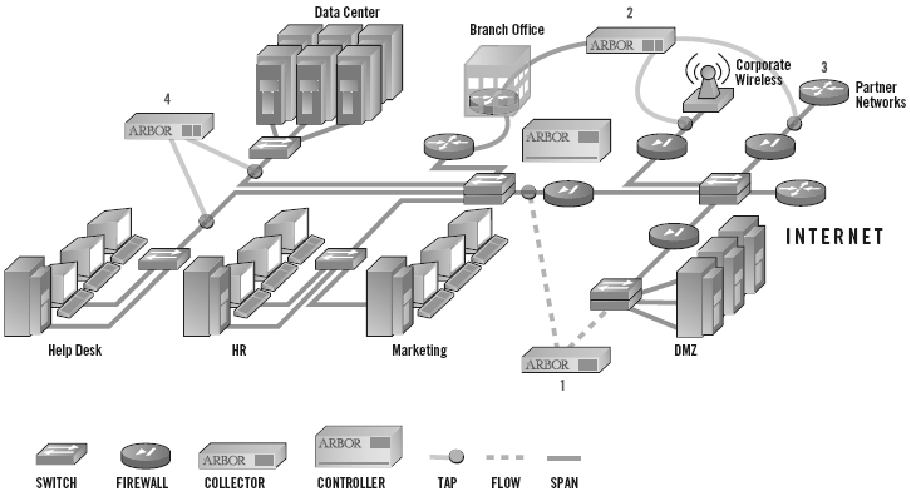

Fig. C.3: Example deployment of Arbor Networks' Peakflow X

C.4 Webscreen Technologies: Ws Series and Crossbeam X Series

Webscreen products are appliances which are deployed between a webserver and the rest of the Internet, and have been designed to detect and prevent network-level resource floods, including Denial of Service and Distributed Denial of Service. They use a proprietary heuristic algorithm, which they call CHARM, to separate good and bad traffic, and attempts to ensure that genuine users of a website have uninterrupted access,

while illegal users are blocked. As with most commercial solutions, CHARM works by analysing inbound and outbound traffic, remembering user behaviour and letting through only those it trusts the most (Fig. C.4).

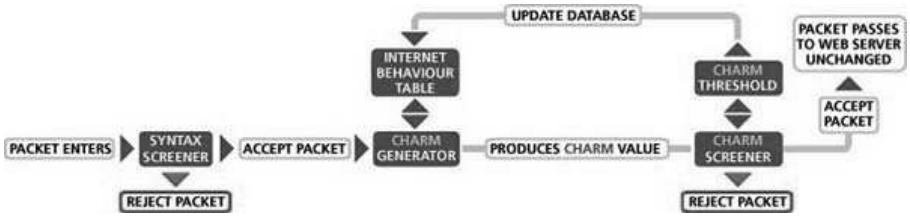

Fig. C.4: Operation of a Webscreen appliance

C.5 Captus Networks: Captus IPS

Again along the same lines as most commercial solutions, Captus IPS provides a policy language for administrators to specify rules, notifies them of ongoing DoS attacks, and attempts to mitigate them with rate-limiting and a combination of redirection and blocking for unsanctioned traffic. When conditions improve, the product re-evaluates the network and adjusts settings in conformance with the administrator's policies. Their TLIDS (Traffic Limiting Intrusion Detection Software) policy engine works with AND and OR boolean operators for rules regarding the source and destination addresses, the source and destination ports, the protocols used, flags for condition matching, rates and times [114]. Fig. C.5 shows how Captus IPS is deployed.

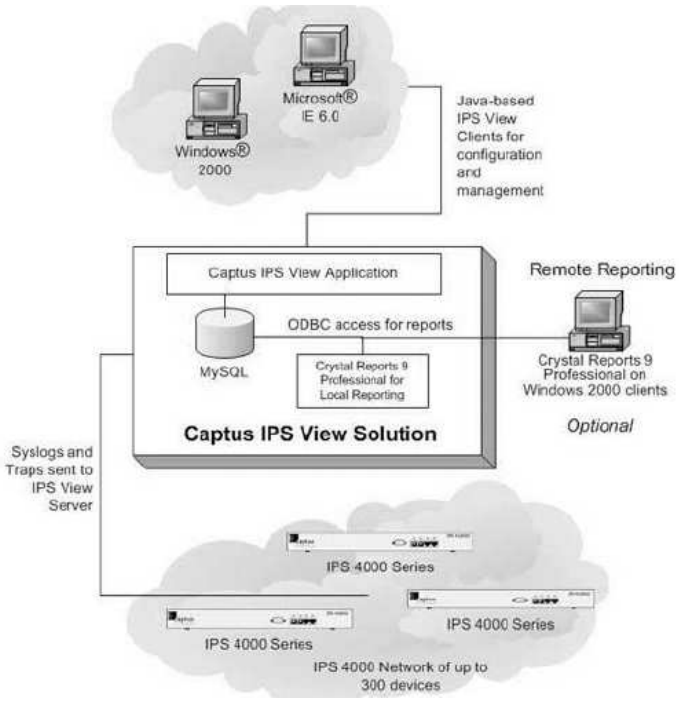

Fig. C.5: Captus IPS deployment

D. TECHNICAL DETAILS OF THE IMPLEMENTATION OF THE PRIORITISATION AND RATE-LIMITING MECHANISM

The prioritisation and rate-limiting mechanism was implemented in Linux 2.6. as an extension of the existing Cognitive Packet Network code. Most of the aspects of the CPN protocol have been developed as a kernel module, which is loaded at the startup of the machine it has been installed in. The module contains, for example, the *cpn_input.c* file which is responsible for all the processes relevant to the receival of a new packet, while *protocol.c* is responsible for the QoS metric that will be used for routing (see Appendix B for more details on the possible choices of QoS metric in CPN). We chose to implement this DoS protection mechanism in the same way, by adding a few additional processes in the core CPN module. The following is a high-level description of the details of the implementation:

D.1 Additions and modifications in the CPN code

dos_alarm.c. This is the file responsible for the detection mechanism that we used in our experiments. It consists of checks on the incoming bitrate at each port (interface) of the node and the rates of change of these bitrates (acceleration), and the same checks for the total incoming bitrate and the total acceleration of the bitrate in the node running this DoS protected version of CPN:

- Incoming bitrate at Interface IF1

- Incoming bitrate at Interface IF2
- Incoming bitrate at Interface IF3
- Incoming bitrate at Interface IF4
- Acceleration of incoming bitrate at IF1
- Acceleration of incoming bitrate at IF2
- Acceleration of incoming bitrate at IF3
- Acceleration of incoming bitrate at IF4
- Total incoming bitrate at the node
- Acceleration of total incoming bitrate at the node

For each of these metrics a threshold is set over which the network is considered to be under a DoS attack. Although most of the times this is how a DoS attack manifests itself, it is often the case that there is a sudden increase of traffic from legitimate sources without any intended DoS attack (see Section 2.2 for a description of the flash-crowd effect) or that an attack is stealthier and achieves degradation of the victim's service without volumes of traffic (see Section 2.4, Shrew attacks). However, with the mechanism for collection of the necessary statistics being added in the cpn_stats.c file of the code of the CPN module, the dos_alarm.c file can easily incorporate the additional data for a more sophisticated and more accurate detection process. See Section 6.4 for an approach that we have started implementing and testing and will consider to use in the future.

dos_mark.c. When the detection process implemented in the *dos_alarm.c* file informs of a possible ongoing DoS attack, the *dos_mark.c* is set into action. This file is responsible for the classification and marking aspects of the protection mechanism. When a packet is received in the node, then the modified *cpn_input.c* file asks

from the packet to undergo a few classification tests (the ones we have referred to as validity tests in the previous section - see Chapter 2.2 for a more detailed discussion on the concept):

- Check the source of the packet to see if it belongs to the list of frequent legitimate customers. Unless the DoS attacker has managed to acquire this list or knows personally some of them and has taken hold of their IPs, then it is extremely unlikely that an IP spoofing based on picking IPs at random would result into using IPs from the list.
- Check whether the arrival time of the flow that the packet belongs to is before or after the beginning of the attack as noticed by the detection process of the *dos_alarm.c* file. This is based on our previous discussion also from Section 2.2 on the fact that the Ramp-Up behaviour of the distributed DoS attacks means that the IP addresses which arrive after the DDoS are much more likely to be illegitimate.
- Check whether the bitrate of the flow that the packet belongs to exceeds a specified threshold.

When the packet is received, the marking process of the *dos_mark.c* file checks whether this is the first time the packet is checked by a DoS protection node, by looking at the flags field in the CPN part of its header (see Fig. D.1). If the value stored in the DoS section of the flags (the first 3 of the 4 bits are used for storing the DoS mark, while the 4th is used for other CPN functions) is 0, then the marking process sets a predetermined value as the starting mark for the packet. Then, this mark increases or decreases by 1 for each validity test if the result is positive or negative respectively.

stats.c. This file contains all the statistics collection functions in the CPN protocol. For

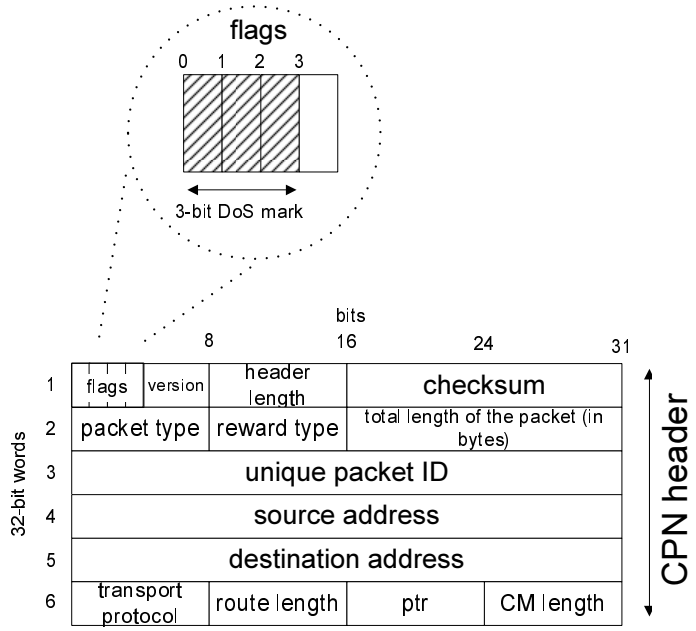

Fig. D.1: Three previously unused bits in the flags field of the standard CPN part of the packet's header are used to store the DoS mark of the prioritisation and rate-limiting mechanism. See Appendix B for a detailed description of the CPN header.

the purposes of the DoS detection process of *dos_alarm.c* we introduced a function for real-time monitoring of the incoming traffic rate per port (interface) of the node.

cpn_input.c The original file is modified to call for the initialisation of the detection process of the *dos_alarm.c* file and the marking process of the *dos_mark.c* file to start the protection mechanism.

D.2 Setting up prioritisation and rate-limiting in Linux

This additional functionality that we added at the Linux kernel level over the standard CPN code can then be used in conjunction with the routing infrastructure of Linux.

Priorities for CPN. As mentioned in the previous section, for prioritisation we used the

PRIO queueing discipline of the iproute2 infrastructure provided with Linux 2.2 and above. PRIO serves packets of a certain priority level only if all higher-priority queues are empty. It does not actually shape traffic, only subdivides it based on how its corresponding filters are configured. Each band is a separate class instead of a simple FIFO queue. When a packet is enqueued to the PRIO queueing discipline, a class is chosen based on filter commands.

Rate-limiting (throttling). For rate-limiting we used the Token Bucket Filtering (TBF) queueing discipline, as a very light-weight method of allowing packets to pass only up to a specified maximum rate. The TBF rate-limiting mechanism is triggered with the detection of a potential DoS attack and is used to rate-limit not only the total traffic that arrives at the victim, but also the traffic associated with each priority band separately.

Our prioritisation & rate-limiting architecture. With the help of [110] we set up the scheme shown in Fig. D.2. This is a technical graph of the implementation of the mechanism of the PRIO queueing discipline that we chose to use, which is why it seems complicated and containing redundant information. Practically, whenever a packet needs to be dequeued, class 1:1 (band 0) is tried first, then 1:2 and so on. Higher classes are only used if none of the lower ones contains a packet. With PRIO each class can have a separate queueing discipline instead of being a simple FIFO. As explained already, for the purposes of limiting the incoming traffic to levels that would not overwhelm the victim, each of the PRIO classes is a TBF queue with a different maximum set rate.

Practically, the mechanism we described above is initialised in Linux with commands similar to the following:

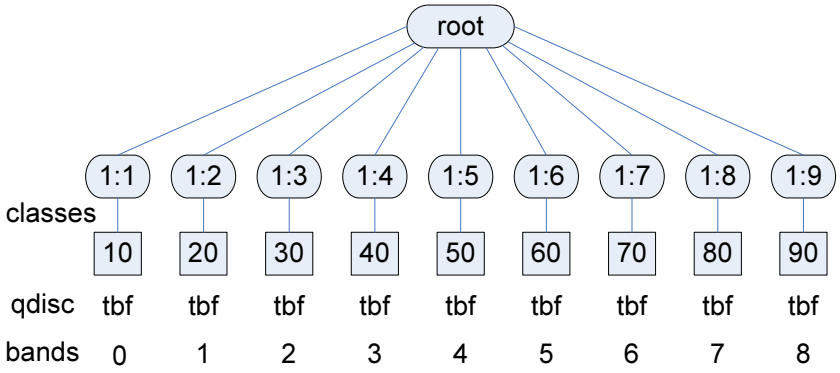

Fig. D.2: Technical description of the implementation of prioritisation: each of the classes of the prioritisation mechanism is a separate queue which operates according to the Token Bucket Filter discipline.

```
tc qdisc add dev eth3 root handle 1: prio bands 9 priomap
tc qdisc add dev eth3 parent 1:1 handle 10: tbf rate 7000kbit buffer 2000 limit 700000
tc qdisc add dev eth3 parent 1:2 handle 20: tbf rate 7000kbit buffer 2000 limit 700000
...
```

which create a root class and the subclasses for the traffic at port (interface) eth3 of the node it is installed in.

Prioritising according to the DoS mark. Having described the prioritisation architecture it only remains to explain what rules determine the actual allocation of priorities. This is done based wholly on the probability that the packet is normal or DoS, which was estimated during the classification stage by the validity tests of the processes in the dos_mark.c file, and is represented by the mark field in the CPN header of the packet (Fig. D.1).

The first class (0) is reserved for traffic of the highest priority, such as emergency messages between routers and other control packets. The next eight classes (1-8) are reserved for the rest of the traffic, which has undergone the DoS marking process. In fact, the value of the mark stored in the packet's header is the exact

class in the queue of which the packet should receive service. In order to initialise such rules, the command at the application layer of Linux would be something like this:

"All packets at port (interface) eth3 with a mark of 5 belong to priority class 1:5".

```
tc filter add dev eth3 parent 1:0 protocol all u32 match u8 0xC1 0xff at 0 flowid 1:6
```

This is the so-called u32 classifier. It is the most advanced filter available in the iproute2 infrastructure, and is entirely based on hashing tables, which make it robust when there are many filter rules. In this example, we compare the first byte of the CPN header (implied by "at 0") against the 0xC1 value (11000001 in binary). Since the first three bits comprise the DoS mark, this should be 110 in binary or simply 5 in decimal, which is the id of the class we want these packets to belong to. In a similar way we could set different priority classes for the CPN packets according to their type or some other criterion or set of criteria. For example:

```
tc filter add dev eth3 parent 1:0 protocol all u32 match u8 0x0 0xff at 4 flowid 1:6
```

which checks whether the 5th byte (the type of the packet, as seen in Fig. D.1) is 0, which corresponds to the dumb packet in the CPN protocol.

D.3 Traffic generator

For the purposes of the experiments in this chapter, we built a traffic generator utility, which we installed in the nodes that would send traffic. It works by creating every $\frac{1}{rate\_of\_traffic}$ units of time a new CPN packet of specific size, filled with zeros in the data payload part. By monitoring the rate of traffic as sent with the traffic generator and comparing it to the rate measured by the internal CPN statistics (implemented in the *stats.c*

file), we saw that the traffic generator is accurate for traffic rates up to 70 Mbits/sec, which is enough for our experiments. The utility uses as input a file containing the traffic rate instructions, and two other parameters (Fig. D.3):

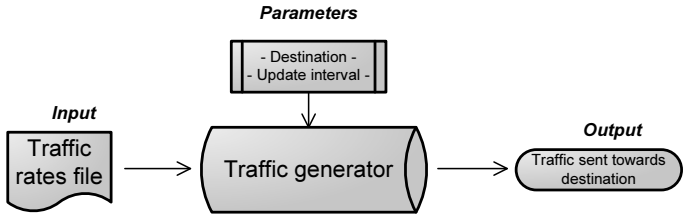

Fig. D.3: Graphical representation of the mechanism of the traffic generator used in our experiments.

Traffic rates file. This file contains a list of rate values (in Kbits/sec), which represents the changes in the sending rate of the traffic from the specific node. For example, if the input file contained the following list:

0 156.4 180 450 2000 4500 2400 800 0

the sending rate of the traffic would increase from 0 to a maximum 4.5 Mbits/s and then decrease gradually back to 0.

Destination. This parameter is simply the IP of the destination of the CPN traffic generated with this utility.

Update interval. The interval parameter determines how often the traffic generator should jump to the next rate value of the input file.

E. AVERAGE PACKET SIZES IN THE INTERNET

Throughout this work we had to use average packet sizes to compare capacities in Bytes/s or bits/s with capacities in packets/sec. Although not necessary to reach qualitative conclusions, we decided to look at this issue of the realistic average size of a packet in normal network use.

Measurements taken at an Internet backbone in 1998 [6] illustrated the predominance of small packets, with peaks at the common 44, 552, 576 and 1500 Bytes. Nearly half of the packets were 44 Bytes long, but accounted for no more than 7% of the total Byte volume. TCP traffic averaged about 95% of the bytes, 90% of the packets, and 80% of the flows, while UDP was the second largest category, with roughly 5% of the bytes, 10% of the packets, and 18% of the flows on average.

More up to date statistics can be found in the website of the Cooperative Association for Internet Data Analysis (CAIDA) [1] organisation. Table E shows measurements representative of the most popular types of traffic taken at the San Diego Network Access Point (SDNAP) [2] for the last one year (Table E.1).

From this table we can derive pretty accurate and up to date average packet sizes per traffic type in the Internet (see Table E.2).

[1] CAIDA is a collaborative undertaking among organisations in the commercial, government, and research sectors aimed at promoting cooperation in the engineering and maintenance of a robust, scalable global Internet infrastructure [116].

[2] SDNAP serves as a peering point for several Organizations and Internet Service Providers in the San Diego area.

Tab. E.1: CAIDA-SNDAP statistics for May 2005 - May 2006

Traffic type	Mbits/sec	packets/sec
HTTP (WWW)	10.14	1,890
other TCP	3.95	750
SSH	1.57	360
SMTP	0.67	140
BitTorrent File sharing	0.53	90
other UDP	0.40	160
...	...	...
Total	19.93 Mbits/sec	4400 packets/sec

Tab. E.2: Average packet sizes derived from CAIDA-SNDAP statistics for May 2005 - May 2006

Traffic type	Average packet (Bytes)
HTTP (WWW)	686.7
other TCP	674.1
SSH	558.2
SMTP	612.6
BitTorrent File sharing	753.8
other UDP	320.0
...	...
Total average	579.8

It is important to note that such average values are indicative and should be used only when there is no specific relevant information for the network at hand. Different networks may present very different characteristics, depending on their current use. For example, if a network is used mainly for online gaming applications, the average packet size for server to client traffic is usually no more than 200 Bytes, and for client to server about half of that. Using the global average packet size of 500-600 Bytes for both traffic directions would be totally unrealistic and would introduce significant inaccuracies in the estimated capacities of links and processors.

F. CURRENT LEGISLATION

As explained in Section 1.1 and Fig. 1.5 and 1.6, only a few victims of DoS attacks seek legal assistance. Then, because of the legislation complexities (Section 6.1.1) even fewer succeed in their legal actions. Although it is beyond the technical scope of this thesis, for the sake of completeness it would be interesting to look at the current legislation regarding Distributed Denial of Service attacks [59, 51, 105, 78].

F.1 UK

At the time of writing of this thesis, there is ongoing discussion from several authorities on the need for amendments to the severely outdated Computer Misuse Act 1990 of the UK parliament, mainly to increase penalties for hackers and explicitly criminalise Denial of Service attacks. This will be the third attempt to introduce a Computer Misuse Act Amendment Bill specifically for DoS attacks. The previous two (2002, 2005) failed due to wide disparity of legal opinion [100]. Although not always applicable, the following can also be used to prosecute DoS attackers:

Protection of harassment Act 1997. Can be used only if the attack consists of email messages with offensive content.

Communications Act 2003. "A person is guilty of an offence if, for the purpose of causing annoyance, inconvenience or needless anxiety to another, he ... persistently makes use of a public electronic communications network".

F.2 European Union

Since 24 February 2005 the European Justice and Home Affairs Council has adopted a common framework decision on attacks against information systems in the EU. It addresses any offence committed against a computer infrastructure with the intentions of destroying, modifying or altering the information stored on computer or networks of computers. The two key definitions in the decision are illegal access to information systems and illegal interference with the system. In both cases, intent has to be proven, to rule out gross negligence or recklessness. The decision covers not only offences affecting the Member States but also offences committed in their territory against systems located in the territory of third countries.

F.3 USA

18 U.S.C. §1030. Computer Fraud and Abuse Act. In summary, this includes accessing a computer to commit espionage, trespassing in a government computer, damaging, threatening to damage, trafficking passwords or committing a fraud an integral part of which involves unathorised access to a government computer, a bank computer, or a computer used in interstate or foreign computer.

18 U.S.C. §875. Threats transmitted in interstate commerce.

18 U.S.C. §876. Mailing threatening communications.

18 U.S.C. §877. Mailing threatening communication from a foreign country.

18 U.S.C. §880. Receipt of the proceeds of extortion.

18 U.S.C. §1343. Fraud by wire, radio or television.

18 U.S.C. §1951. Extortion that affects commerce.

BIBLIOGRAPHY

[1] L. Niven. “Flash Crowd”, short story from “The flight of the horse”. Ballantine Books, ISBN: 0345334183, 1973.

[2] V.D. Gilgor. A Note on the Denial-of-Service Problem. *Proc. of the IEEE Symposium on Computer Security and Privacy*, pp. 139-149, Oakland, California, April 1983.

[3] R.T. Morris. A Weakness in the 4.2BSD Unix TCP/IP software. *Technical Report Computer Science #117*, AT&T Bell Labs, February 1985.

[4] L.L. Scharf. Statistical Signal Processing: Detection, Estimation, and Time Series Analysis. ISBN 0-201-19038-9, Addison-Wesley Publishing Company, Inc., 1991.

[5] P. Ferguson and D. Senie. Network ingress filtering: defeating Denial of Service attacks which employ IP source address spoofing. *Tech. Rep. RFC 2267*, January 1998.

[6] K. Claffy, G. Miller, and K. Thompson. The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone. *Proc. of INET ’98*, July 1998.

[7] A. Juels and J. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. *Network and Distributed System Security Symposium (NDSS)*, pp. 151-165, February 1999.

[8] D. Dittrich. The DoS Project's Trinoo Distributed Denial of Service Attack Tool. *http://staff.washington.edu/dittrich/misc/tfn.analysis.txt*. October 1999.

[9] S. Haykin: *Neural Networks A Comprehensive Foundation*,pp. 143-146, Prentice-Hall Inc., 1999.

[10] T. Aura, P. Nikander, and J. Leiwo. DOS-resistant Authentication with Client Puzzles. *Lecture Notes in Computer Science*, 2000.

[11] S.M. Bellovin. ICMP Traceback Messages. Internet Draft: draft-bellovin-itrace-00.txt. March 2000.

[12] D. Song and A. Perrig. Advanced and Authenticated Marking Schemes for IP Traceback. Technical Report UCB/CSD-00-1107, University of California, Berkeley, 2000.

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

[14] H. Burch and B. Cheswick. Tracing anonymous packets to their approximate source. *Proceedings of USENIX LISA'00*, December 2000.

[15] R. Duda, P. Hart, and D. Stork. Pattern Classification. 2nd Edition, J. Wiley and Sons Ltd., London and New York, ISBN 0-471-05669-3, 2001.

[16] A.C. Snoeren, C. Partridge, L.A. Sanchez, W.T. Strayer, C.E. Jones, F. Tchakountio, and S.T. Kent. Hash-based IP traceback. BBN Technical Memo 1284, February 12, 2001.

[17] D. Dean, M. Franklin, and A. Stubblefield. An Algebraic Approach to IP Traceback. *Network and Distributed System Security Symposium (NDSS)*, February 2001.

[18] R. Mahajan, S.M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling High Bandwidth Aggregates in the Network. AT&T Center for Internet Research at ICSI (ACIRI) and AT&T Labs Research, Technique Report (Draft), February 2001.

[19] D. Song and A. Perrig. Advanced and authenticated marking schemes for IP traceback. *Proc. Infocom 2001*, ISBN: 0-7803-7016-3, vol. 2, pp. 878-886, Anchorage, Alaska, USA, 22-26 April 2001.

[20] L.A. Sanchez, W.C. Milliken, A.C. Snoeren, F. Tchakountio, C.E. Jones, S.T. Kent, C. Partridge, and W.T. Strayer. Hardware support for a hash-based IP traceback. *Proceedings of DARPA Information Survivability Conference & Exposition II, 2001. DISCEX '01*, Vol. 2, pp. 146-152, June 12-14, 2001.

[21] V. Paxson. An analysis of using reflectors for distributed Denial-of-Service attacks. *ACM Computer Communications Review 31(3)*, July 2001.

[22] U.S. Department of Justice. Electronic Crime Scene Investigation. A Guide for First Responders. National Institute of Justice Report, NCJ 187736, July 2001.

[23] K. Park and H. Lee. On the effectiveness of Router-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets. *Proc. ACM SIGCOMM Conference*, S. Diego, California, USA, August 2001.

[24] D. Moore, G. Voelker, and S. Savage. Inferring Internet Denial of Service Activity. *Proc. of the 10th USENIX Security Symposium*, pp. 9-22, August 2001.

[25] D. Mankins, R. Krishnan, C. Boyd, J. Zao, and M. Frentz. Mitigating Distributed Denial of Service Attacks with Dynamic Resource Pricing. *Proceedings of Computer Security Applications Conference*, pp. 411-421, Dec. 10-14, 2001.

[26] A.B. Kulkarni, S.F. Bush, and S.C. Evans. Detecting Distributed Denial-of-Service Attacks Using Kolmogorov Complexity Metrics. *Technical Report*, 2002.

[27] J. Ioannidis and S.M. Bellovin. Implementing Pushback: Router-Based Defense Against DDoS Attacks. *Proceedings of Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2002.

[28] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high-bandwidth aggregates in the network. *ACM SIGCOMM Computer Communication Review*, ISSN: 0146-4833, Vol. 32, Issue 3, pp. 62-73, July 2002.

[29] H.S. Seo, T.H. Cho. Modeling and Simulation for Detecting a Distributed Denial of Service Attack. *Lecture Notes in Artificial Intelligence*, Vol. 2557, pp. 179-190, 2002.

[30] Mitigation of Denial of Service Attacks using QoS regulation. *Proc. 10th IEEE Internation Workshop on Quality of Service (IWQOS)*, pp. 45-53, May 2002.

[31] J. Brustoloni. Protecting electronic commerce from Distributed Denial of Service Attacks. *Proc. 1st International World Wide Web Conference, ACM*, pp. 553-561, Honolulu, May 2002.

[32] J. Li, J. Mirkovic, M. Wang, P. Reiher, and L. Zhang. SAVE: Source Address Validity Enforcement Protocol. *Proc. Infocom*, Vol. 3, pp. 1557-1566, June 2002.

[33] A. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. *Proceedings of ACM SIGCOMM*, pp. 61-72, ISBN: 1-58113-570-X, Pittsburgh, PA, USA, 2002.

[34] L. Spitzner. Honeypots: Tracking Hackers. ISBN: 0321108957, Addison-Wesley Professional, September 2002.

[35] G. Rice and J. Davis. A genealogical approach to analyzing post-mortem denial of service attacks. *Secure and Dependable System Forensics Workshop*, University of Idaho, September 23-25, 2002.

[36] A. Snoeren, C. Partridge, L.A. Sanchez, C.E. Jones, F. Tchakountio, B. Schwartz, S. Kent, and W.T. Strayer. Single-packet IP traceback. *IEEE/ACM Transactions on Networking*, ISSN: 1063-6692, Vol. 10, no. 6, pp. 721-734, December 2002.

[37] J. Mirkovic, J. Martin, and P. Reiher, "A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms", University of California Los Angeles, Technical Report #020018, 2002.

[38] M. McCaughey and M.D. Ayers. Cyberactivism: Online Activism in Theory and Practice. ISBN: 0415943205, February 2003.

[39] R. Thomas, B. Mark, T. Johnson, and J. Croall. NetBouncer: client-legitimacy-based high-performance DDoS filtering. *Proc. DARPA Information Survivability Conference and Exposition*, vol. 1, pp. 14-25, April 22-24, 2003.

[40] C. Papadopoulos, R. Lindell, J. Mehringer, A. Hussain, and R. Govin. COSSACK: Coordinated Suppression of Simultaneous Attacks. *Proceedings of DARPA Information Survivability Conference and Exposition*, vol. 1, pp. 2-13, April 22-24, 2003.

[41] X. Wang and M.K. Reiter. Defending against Denial-of-Service Attacks with Puzzle Auctions. *Proceedings of IEEE Symposium on Security and Privacy*, pp. 78-92, Oakland, CA, USA, May 2003.

[42] G. Mori and J. Malik. Recognizing objects in adversarial clutter - Breaking a visual CAPTCHA. *Proc. IEEE Computer Society Conference on Computer Vision and*

Pattern Recognition 2003 (CVPR '03), ISSN: 1063-6919, ISBN: 0-7695-1900-8, vol. 1, pp. 134-141, Madison, WI, USA, June 18-20, 2003.

[43] S.M. Khattab, C. Sangpachatanaruk, R. Melhern, D. Mosse, and T. Znati. Proactive Server Roaming for Mitigating Denial-of-Service Attacks. *Proceedings of International Conference on Information Technology Research and Education*, pp. 289-290, August 11-13, 2003.

[44] J. Mirkovic, P. Reiher, and M. Robinson. Forming Alliance for DDoS Defense. in *New Security Paradigms Workshop*, Centro Stefano Francini, Ascona, Switzerland, August 18-21, 2003.

[45] A. Hussain, J. Heidermann, and C. Papadopoulos. A framework for classifying denial of service attacks. *Proc. ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication 2003*, ISBN: 1-58113-735-4, pp. 99-110, Karlsruhe, Germany, August 25-29, 2003.

[46] A. Kuzmanovic and E.W. Knightly. Low-Rate TCP-Targeted Denial of Service Attacks (The Shrew vs. the Mice and Elephants). *Proc. ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication 2003*, Karlsruhe, Germany, August 25-29, 2003.

[47] J. Mirkovic. D-WARD: Source-End Defense Against Distributed Denial-of-Service Attacks. PhD dissertation, Univ. of California Los Angeles, August 2003, http://lasr.cs.ucla.edu/ddos/dward-thesis.pdf.

[48] M. Sung and J. Xu. IP traceback-based intelligent packet filtering: a novel technique for defending against Internet DDoS attacks. *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, pp. 861-872, September 2003.

[49] W.G. Morein, A. Stavrou, D.L Cook, A.D. Keromytis, V. Mishra, and D. Rubenstein. Using graphic Turing tests to counter automated DDoS attacks against Web servers. *Proc. 10th ACM Int'l Conference on Computer and Communications Security (CCS '03)*, ISBN: 1-58113-738-9, pp. 8-19, Washington DC, USA, October 27-30, 2003.

[50] S. Jing, H. Wang, and K. Shin. Hop-Count filtering an effective defense against spoofed traffic. *Proc. ACM Conference on Computer and Communications Security (CCS '03)*, ISBN: 1-58113-738-9, pp. 30-41, , Washington DC, USA, October 27-30, 2003.

[51] C. Doyle. Cybercrime: A Sketch of 18 U.S.C. 1030 and Related Federal Criminal Laws. CRS report for Congress. Order Code RS20830, November 24, 2003.

[52] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred. Statistical Approaches to DDoS Attack Detection and Response *Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX'03)*, 2003.

[53] G. Giacinto, F. Roli, and L. Didaci. Fusion of multiple classifiers for intrusion detection in computer networks. *Pattern Recognition Letters* Vol. 24, pp. 1795-1803, 2003.

[54] T. Peng, C. Leckie, and K. Ramamohanarao. Detecting Distributed Denial of Service Attacks by Sharing Distributed Belief. *Lecture Notes in Computer Science*, Vol. 2727, pp. 214-225, 2003.

[55] S. Noh, C. Lee, K. Choi, and G. Jung. Detecting Distributed Denial of Service (DDoS) Attacks through Inductive Learning. *Lecture Notes in Computer Science*, Vol. 2690, pp. 286-295, 2003.

[56] U.S. Department of Justice. Forensic Examination for Digital Evidence. A Guide for Law Enforcement. National Institute of Justice Report, NCJ 199408, April 2004.

[57] J. Mirkovic and P. Reiher. A Taxonomy of DDoS Attack and DDoS Defence Mechanisms. *ACM Computer Communications Review*, Vol. 34, no. 2, April 2004.

[58] S. Specht and R. Lee. Taxonomies of Denial of Service Networks, Attacks, Tools and Countermeasures. *Proc. 17th International Conference on Parallel and Distributed Computing Systems*, pp. 543-550, San Francisco, September 15-17, 2004.

[59] J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher. Internet Denial of Service Attack and Defense Mechanisms. ISBN: 0-13-147573-8, PRENTICE-HALL, December 2004.

[60] M. Kim, H. Na, K. Chae, H. Bang, and J. Na. A Combined Data Mining Approach for DDoS Attack Detection. *Lecture Notes in Computer Science*, Vol. 3090, pp. 943-950, 2004.

[61] D. Andersen. Mayday: Distributed DoS Filtering for Internet Services. *ACM SIGCOMM Computer Communication Review*, Vol. 34, Issue 1, pp. 39-44, ISSN 0146-4833, 2004.

[62] S. Mukkamala and A.H. Sung. Computational Intelligent Techniques for Detecting Denial of Service Attacks. *Lecture Notes in Artificial Intelligence*, Vol. 3029, pp. 616-624, 2004.

[63] A.H. Sung and S. Mukkamala. The Feature Selection and Intrusion Detection Problems, *Lecture Notes in Computer Science*. Vol. 3321, pp. 468-482, 2004.

[64] I.A. Cheong, Y.M. Kim, M.S. Kim, and B.N. Noh. The Causality Analysis of Protocol Measures for Detection of Attacks Based on Network. *Lecture Notes in Computer Science*, Vol. 3090, pp. 962-972, 2004.

[65] S. Byers, A.D. Rubin, and D. Kormann. Defending against an Internet-based attack on the physical world. *ACM Transactions on Internet Technology (TOIT)*, Vol. 4, Issue 3, pp. 239-254, August 2004.

[66] X. Yang, Y. Liu, M. Zeng, and Y. Shi. A Novel DDoS Attack Detecting Algorithm Based on the Continuous Wavelet Transform. *Lecture Notes in Computer Science*, Vol. 3309, pp. 173-181, 2004.

[67] C. Siaterlis and B. Maglaris. Towards Multisensor Data Fusion for DoS Detection. *2004 ACM Symposium on Applied Computing*, pp. 439-446, 2004.

[68] M.S. Shin, E.H. Kim, and K.H. Ryu. False Alarm Classification Model for Network-Based Intrusion Detection System. *Lecture Notes in Computer Science*, Vol. 3177, pp. 259-265, 2004.

[69] M. Li, C-H. Chi, and D. Long. Fractional Gaussian Noise: A Tool for Characterizing Traffic for Detection Purpose. *Lecture Notes in Computer Science*, Vol. 3309, pp. 94-103, 2004.

[70] M. Li. An Approach to Reliably Identifying Signs of DDOS Flood Attacks Based on LRD Traffic Pattern Recognition. *Computers and Security*, Vol. 23, No. 549-558, 2004.

[71] Y. Xiang, Y. Lin, W. L. Lei, and Huang S.J., Detecting DDOS attack based on Network Self-Similarity, *IEE Proceedings in Communication*, Vol. 151, No.3, pp. 292-295, 2004.

[72] K. Cetnarowicz and G. Rojek. Behavior Based Detection of Unfavourable Resource. *Lecture Notes in Computer Science*, Vol. 3038, pp. 607-614, 2004.

[73] A. Hussain, J. Heidemann, and C. Papadopoulos. Distinguishing Between Single and Multi-Source Attacks Using Signal Processing. *Computer Networks*, Vol. 46, pp. 479-503, 2004.

[74] D. Cajueiro and B. Tabak. The Hurst exponent over time: testing the assertion that emerging markets are becoming more efficient. *Physica A*, Vol. 336, Issue 3-4, pp. 521-537, Elsevier, May 2004.

[75] S.M. Khattab, C. Sangpachatanaruk, D. Moss, R. Melhem, and T. Znati. Roaming Honeypots for Mitigating Service-Level Denial-of-Service Attacks. *24th IEEE International Conference on Distributed Computing Systems (ICDCS'04)*, pp. 328-337, 2004.

[76] D. Gavrilis, I. Tsoulos, and E. Dermatas. Feature Selection for Robust Detection of Distributed Denial-of-Service Attacks Using Genetic Algorithms. *Lecture Notes in Artificial Intelligence*, Vol. 3025, pp. 276-281, 2004.

[77] L. Valeri. Update to the Handbook of Legal Procedures of Computer and Network Misuse in EU Countries for assisting CSIRTs. Study for the European Commission, 2005.

[78] Council of the European Union. Council Framework Decision on attacks against information systems. Legislative Acts and Other Instruments. 15010/04, Brussels, January 17, 2005.

[79] A. Shevtekar, K. Anantharam, and N. Ansari. Low-Rate TCP Denial-of-Service Attack Detection at Edge Routers. *IEEE Communications Letters*, Vol. 9, No. 4, April 2005.

[80] D. Gavrilis and E. Dermatas. Real-time detection of distributed denial-of-service attacks using RBF networks and statistical features. *Computer Networks*, Vol. 48, pp. 235-245, 2005.

[81] R. Jalili, F. Imani-Mehr, M. Amini, and H.-R Shahriari. Detection of Distributed Denial of Service Attacks Using Statistical Pre-processor and Unsupervised Neural Networks. *Lecture Notes in Computer Science*, Vol. 3439, pp. 192-203, 2005.

[82] S. Sarat and A. Terzis. On the Effect of Router Buffer Sizes on Low-Rate Denial of Service Attacks. *International Conference on Computer Communications and Networks ICCCN 2005*, San Diego, CA, USA, October 17-19, 2005.

[83] D.K.Y. Yau, J.C.S Lui, F. Liang, and Y. Yam. Defending against distributed Denial-of-Service attacks with max-min fair server-centric router throttles. *IEEE/ACM Transactions on Networking*, Vol. 13, No. 1, pp. 29-42, February 2005.

[84] S. Kandula, D. Katabi, M. Jacob, and A. Berger. Botz-4-Sale: surviving organized DDoS attacks that mimic flash crowds. *Proc. 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI '05)*, Boston, MA, USA, May 2-4, 2005.

[85] J. Mirkovic and P. Reiher. D-WARD: A source-end defense against flooding Denial-of-Service attacks. *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 3, pp. 216-232, July-September, 2005.

[86] R. Clegg. A Practical Guide to Measuring the Hurst Parameter (Extended version). *International Journal of Simulation: Systems, Science & Technology*, Vol. 7, No 1, 3-14, 2006.

[87] H.T. He, X.N. Luo, and B.L. Liu. Detecting Anomalous Network Traffic

with Combined Fuzzy-Based Approaches. *Lecture Notes in Computer Science*, Vol. 3645, pp. 433-442, 2005.

[88] S.Y. Lee, Y.S. Kim, B.H. Lee, S.H. Kang, C.H. Youn. A Probe Detection Model Using the Analysis of the Fuzzy Cognitive Maps. *Lecture Notes in Computer Science*, Vol. 3480, pp. 320-328, 2005.

[89] S. Mukkamala, A. H. Sung and A. Abraham. Intrusion detection using an ensemble of intelligent paradigms. *Journal of Network and Computer Applications*, Vol. 28, pp. 167-182, 2005.

[90] A.P.F. Chan, W.W.Y Ng, D.S. Yeung, E.C.C. Tsang. Multiple Classifier System with Feature Grouping for Intrusion Detection: Mutual Information Approach. *Lecture Notes in Artificial Intelligence*, Vol. 3683, pp. 141-148, 2005.

[91] F. Furuya, T. Matsuzaki, and K. Matsuura. Detection of Unknown DoS Attacks by Kolmogorov-Complexity Fluctuation. *Lecture Notes in Computer Science*, Vol. 2005, pp. 395-406, 2005.

[92] R. Gu, P. Yan, T. Zou, and C. Guo. An Automatic and Generic Early-Bird System for Internet Backbone Based on Traffic Anomaly Detection. *Lecture Notes in Computer Science*, Vol. 3420, pp. 740-748, 2005.

[93] L. Li and G. Lee. DDoS Attack Detection and Wavelets. *Telecommunication Systems*, Vol. 28:3, No. 4, pp. 435-451, 2005.

[94] X. Luo and R.K.C. Chang. On a New Class of Pulsing Denial-of-Service Attacks and the Defense. *Proceedings of the Network and Distributed System Security Symposium, NDSS '05*, San Diego, California, USA, 2005.

[95] X. Luo and R.K.C. Chang. Optimizing the Pulsing Denial-of-Service Attacks. *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2005)*, ISBN: 0-7695-2282-3, pp. 582-591, Yokohama, Japan, June 2005.

[96] Y. Kim, W. Lau, M. Chuah, and H. Chao. PacketScore: A Statistics-Based Packet Filtering Scheme against Distributed Denial-of-Service Attacks. *IEEE transactions on dependable and secure computing*, Vol. 3(2), pp. 141–155, 2006.

[97] P. Ayres, H. Sun, H. Chao, and W. Lau. ALPi: A DDoS Defence System for High-Speed Networks. *IEEE Journal on Selected Areas in Communications*, Vol. 24(10), pp. 1864–1876, 2006.

[98] U.K. Tupakula and V. Varadharajan. Analysis of Traceback Techniques. *Proc. Fourth Australasian Information Security Workshop AISW-NetSec 2006*, Hobart, Australia, 2006.

[99] Y. Chen and K. Hwang. Collaborative Detection and Filtering of Shrew DDoS Attacks Using Spectral Analysis. *Journal of Parallel and Distributed Computing*, Vol. 66, pp. 1137–1151, 2006.

[100] R. Clayton. Complexities in Criminalising Denial of Service Attacks. *http://www.cl.cam.ac.uk/ rnc1/complexity.pdf*, February 2006.

[101] CERT Coordination Center. Denial of Service attack via Ping. CERT Advisory CA-1996-26,
http://www.cert.org/advisories/CA-1996-26.html, December 1997.

[102] CERT Coordination Center. Smurf IP Denial-of-Service attacks. CERT Advisory CA-1998-01,
http://www.cert.org/advisories/CA-1998-01.html, March 2000.

[103] CERT Coordination Center. TCP SYN flooding and IP spoofing attacks. CERT Advisory CA-1996-21,
http://www.cert.org/advisories/CA-1996-21.html, November 2000.

[104] CERT Coordination Center. MS-SQL server worm. CERT Advisory CA-2003-04,
http://www.cert.org/advisories/CA-2003-04.html, January 2003.

[105] CERT Coordination Center. How the FBI Investigates Computer Crime.
http://www.cert.org/tech_tips/FBI_investigates_crime.html, June 22, 2004.

[106] UK Security Online. "Denial of Service attack threat analysed".
http://www.uksecurityonline.com/threat/dos.php, 2002.

[107] K. Hemenway and T. Calishain. Spidering hacks. ISBN: 0-596-00577-6, 1st edition, October 2003.

[108] Wikipedia. Internet Relay Chat, *http://en.wikipedia.org/wiki/Internet_Relay_Chat*.

[109] The Network Simulator NS-2, *http://www.isi.edu/nsnam/ns*.

[110] Linux Advanced Routing & Traffic Control, *http://lartc.org/*.

[111] Cisco Systems. Defeating DDoS attacks. White paper,
http://www.cisco.com/en/US/products/ps5888/products_white_paper0900aecd8011e927.shtml, 2004.

[112] Cisco Systems. Introduction to Cisco IOS NetFlow - A Technical Overview. White paper,
http://www.cisco.com/en/US/products/ps6601/products_white_paper0900aecd80406232.shtml, February 2006.

[113] Arbor Systems. Peakflow X data sheet.
http://www.arbornetworks.com/downloads/Arbor_Peakflow_X_Data_Sheet.pdf.

[114] Captus Networks. Protecting the Network from DDoS Attacks. Captus TLIDS 6.2 White Paper.
http://www.captusnetworks.com/solutions/white_papers/index.html.

[115] L.A. Gordon, M.P. Loeb, W. Lucyshyn, and R. Richardson. CSI/FBI Computer Crime and Security Survey. *www.gocsi.com*, 2005.

[116] CAIDA, the Cooperative Association for Internet Data Analysis. *http://www.caida.org*.

[117] E. Gelenbe. Learning in the recurrent random neural network. *Neural Computation*, vol. 5(1), pp. 154164, 1993.

[118] E. Gelenbe and G. Pujolle. Introduction to Queueing Networks. 2nd Edition, 2nd Printing, J. Wiley and Sons Ltd., London and New York, 1999.

[119] E. Gelenbe, R. Lent, and Z. Xu. Towards networks with cognitive packets. Opening Invited Paper *International Conference on Performance and QoS of Next Generation Networking*, Nagoya, Japan, November 2000, in K. Goto, T. Hasegawa, H. Takagi, and Y. Takahashi (eds), "Performance and QoS of next Generation Networking", Springer Verlag, London, 2001.

[120] E. Gelenbe, R. Lent, and Z. Xu. Measurement and performance of Cognitive Packet Networks. *Performance Evaluation*, Vol. 46, pp. 155-176, 2001.

[121] E. Gelenbe, R. Lent, and Z. Xu. Networking with Cognitive Packets. *Proc. ICANN.*, Madrid, August 2002.

[122] E. Gelenbe, R. Lent, A. Montuori, and Z. Xu. Cognitive Packet Networks: QoS and performance. Keynote Paper *IEEE MASCOTS*, San Antonio, TX, October 14-16, 2002.

[123] E. Gelenbe, P. Liu, and J. Lainé. Genetic algorithms for route discovery. *SPECTS'03, Summer Simulation Multiconference, Society for Computer Simulation*, Montreal, July 20-24, 2003.

[124] E. Gelenbe, M. Gellman, R. Lent, P. Liu, and P. Su. Autonomous smart routing for network QoS. *Proc. of the First International Conference on Autonomic Computing*, pp. 232-239, New York, NY, May 2004.

[125] E. Gelenbe, R. Lent, and A. Nunez. Self-aware networks and QoS. *Proceedings of the IEEE*, 92(9):1478-1489, September 2004.

[126] E. Gelenbe, M. Gellman, and G. Loukas. Defending networks against denial of service attacks. In E.M. Carapezza, editor, *Proceedings of the Conference on Optics/Photonics in Security and Defence (SPIE)*, Unmanned/Unattended Sensors and Sensor Networks, Vol. 5611, pp. 233-243, London, UK, October 2004.

[127] E. Gelenbe, M. Gellman, and G. Loukas. An autonomic approach to denial of service defence. *In Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pp. 537-541, June 2005.

[128] E. Gelenbe and G. Loukas. Self-Aware Approach to Denial of Service Defence. *Accepted for publication in special edition of Elsevier Journal on Computer Networks: Security through Self-Protecting and Self-Healing Systems*, 2006.