Aalto University
School of Science
Degree Programme of Computer Science and Engineering

Sanja Šćepanović

Mitigating DDoS attacks
with cluster-based filtering

Master's Thesis
Espoo, June 14, 2011

Supervisor:	Professor Tuomas Aura, Aalto University
	Professor Peeter Laud, University of Tartu
Instructor:	Andrey Lukayenko, MSc. (Tech.), Aalto University

Aalto University
School of Science
Degree Programme of Computer Science and Engineering

ABSTRACT OF MASTER'S THESIS

Author:	Sanja Šćepanović		
Title: Mitigating DDoS attacks with cluster-based filtering			
Date:	June 14, 2011	**Pages:**	77
Professorship:	Data Communication Software	**Code:**	T-110
Supervisor:	Professor Tuomas Aura Professor Peeter Laud		
Instructor:	Andrey Lukayenko, MSc. (Tech.)		

Distributed Denial of Service (DDoS) attacks are considered one of the major security threats in the current Internet. Although many solutions have been suggested for the DDoS defense, real progress in fighting those attacks is still missing.

In this work, we analyze and experiment with cluster-based filtering for DDoS defense. In cluster-based filtering, unsupervised learning is used to create a normal profile of the network traffic. Then the filter for DDoS attacks is based on this normal profile. We focus on the scenario in which the cluster-based filter is deployed at the target network and serves for proactive or reactive defense. A game-theoretic model is created for the scenario, making it possible to model the defender and attacker strategies as mathematical optimization tasks. The obtained optimal strategies are then experimentally evaluated. In the testbed setup, the hierarchical heavy hitters (HHH) algorithm is applied to traffic clustering and the Differentiated Services (DiffServ) quality-of-service (QoS) architecture is used for deploying the cluster-based filter on a Linux router.

The theoretical results suggest that the cluster-based filtering is an effective method for DDoS defense, unless the attacker is able to send traffic which perfectly imitates the normal traffic distribution. The experimental outcome confirms the theoretical results and shows the high effectiveness of cluster-based filtering in proactive and reactive DDoS defense.

Keywords:	DDoS, clustering network traffic, DiffServ
Language:	English

Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan tutkinto-ohjelma

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä:	Sanja Šćepanović		
Työn nimi: Klusterointipohjainen liikenteensuodatus puoluksena hajautettuja palvelunesto-hyökkäyksiä vastaan			
Päiväys:	18. kesäkuuta 2011	**Sivumäärä:**	77
Professuuri:	Tietoliikenneohjelmistot	**Koodi:**	T-110
Valvoja:	Professori Tuomas Aura Professori Peeter Laud		
Ohjaaja:	Diplomi-insinööri Andrey Lukayenko		

Hajautetut palvelunestohyökkäykset ovat yksi nyky-Internetin suurimmista tietoturvahaasteista. Vaikkakin näitä hyökkäyksiä vastaan on kehitetty lukuisia puolustusmekanismeja, mikään näistä ei tarjoa täydellistä suojaa.

Tämä työ tutkii klusterointiin perustuvaa liikenteensuodatusta ja sen käyttöä puolustuksena palvelunestohyökkäyksiä vastaan. Klusterointipohjaisessa suodatuksessa suodatin oppii itsenäisesti normaalit liikennejakaumat. Tämän jälkeen näitä liikennejakaumia voidaan käyttää suodattamaan palvelunestohyökkäyksestä johtuvaa ylimääräistä liikennettä. Diplomityö tutkii skenaariota, jossa käytetään sekä proaktiivista, että reaktiivista klusterointipohjaista puolustusmenetelmää. Lisäksi skenaariosta formuloidaan peliteoreettinen malli, jonka avulla erilaisten hyökkäys- sekä puolustusmenetelmien analyyttinen tutkiminen on mahdollista. Analyyttisesti saatuja tuloksia evaluoidaan kokeellisesti Linux-reitittimessä hyödyntäen Hierarchical Heavy Hitter — klusterointialgoritmia sekä DiffServ-arkkitehtuuria.

Diplomityön teoreettiset tulokset osoittavat, että klusterointiin perustuva suodatus on tehokas puolustus palvelunestohyökkäyksiä vastaan ellei hyökkääjä kykene tekemään imitoimaan tavallista liikennejakaumaa palvelunestohyökkäystä tehdessään. Kokeelliset tulokset vahvistavat teoreettiset tulokset ja osoittavat klusterointipohjaisen suodatuksen tehokkuuden palvelunestohyökkäyksiä vastaan.

Asiasanat:	hajautettu palvelunestohyökkäys, klusterointi, DiffServ
Kieli:	Englanti

Contents

Abbreviations and Acronyms

ADHIC	Approximate Divisive Hierarchical Clustering
CBQ	Class Based Queuing
CERT	Computer Emergency Response Team
CIDR	Classless Inter Domain Routing
DoS	Denial of Service; Internet attacks
DDoS	Distributed Denial of Service
DiffServ	Differentiated Services
DSCP	Differentiated Services Code Point
HHH	Hierarchical Heavy Hitters; A clustering method
HTB	Hierarchical Token Bucket
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
ISP	Internet Service Provider
kbs	kilo bytes per second
KKT	Karus-Kuhn-Tucker conditions
KL-divergence	Kullback-Leibler divergence or relative entropy
NIDS	Network Intrusion Detection System
NPSR	Normal packet survival ratio
PFIFO	Packet limited First In First Out
PHB	Per Hop Behavior
RFC	Request For Comments
SFQ	Stochastic Fairness Queuing
SYN	TCP flag - synchronize sequence numbers
TBF	token bucket filter
tc	traffic conditioning
TCP	Transmission Control Protocol
qdisc	queuing discipline
QoS	Quality of Service
URL	Uniform Resource Locator

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Tuomas Aura from Aalto University. I deeply appreciate the time he has spent on discussing my topic and on supporting and guiding my work. His mentorship has been invaluable not only for this thesis, but it will stay a great inspiration for my future works. My supervisor from University of Tartu, Prof. Peeter Laud, provided constant remote support and I express my gratitude and heartiest thanks to him.

I would also like to thank my thesis instructor Andrey Lukayenko for his helpful direction and valuable ideas without which the theoretical part of my thesis could not be done. Also, I am extremely grateful to Aapo Kalliola who is a PhD researcher at Aalto University. His implementation of the HHH algorithm was greatly useful during my thesis experimental work.

I want to extend a special thanks to Eija Kujanpää, Anna Stina Sinisalo, and Vaisanen Misela as well as other administrative members of the Nord-SecMob programme for retaining the excellent quality of the program and for the two years of invaluable experience for me. A special thanks to Adolfsson Soili for all the administrative tasks she has done for me and other students to ease our thesis work.

Furthermore, I wish to thank to my friends Sajjad Rizvi, Rushil Dave, Roberto Calandra and Minel Dunar with whom my work and stay in Helsinki was so enjoyable and fun.

Last, but not the least, I address my deepest gratitude to my friends in Montenegro and my family who sincerely supported me throughout the whole duration of my Master studies.

Espoo, June 14, 2011

Sanja Šćepanović

Chapter 1

Introduction

1.1 Motivation

Among the many security threats in the current Internet, Distributed Denial of Service (DDoS) attacks are considered to be one of the most serious.

Denial of Service (DoS) attacks aim to make the resources of the computer system of the victim unavailable or unreliable in providing their intended services. In the context of this thesis, DoS attacks try to consume and exhaust the victim's *bandwidth* or the server *capacity*. In DDoS attacks, the attacker compromises a large number of hosts in Internet and instructs them to conduct a coordinated attack. The network of the compromised hosts is called a *botnet*.

While progress has been made in preventing or at least significantly lessening the impact of various security vulnerabilities, real progress in fighting DDoS is still missing. While automated software updates and antivirus programs can limit the number of compromised computers, there are still botnets comprising of millions of nodes. Another potential defense is to filter the packets sent by the DDoS attacker at a firewall after detecting the attack with and intrusion detection system (IDS). These rule-based detection and filtering techniques have not been successful in filtering DDoS traffic because the DDoS attacker can send seemingly legitimate traffic. In the case of open services, such as web servers, the DDoS attacker only needs to send large quantities of useless service requests. Thus, there might be no specific features of DDoS attack traffic that the rule-based filters can be instructed to filter. With such *malicious but legitimate* traffic, DDoS attackers are able to relatively easily bypass most means of DDoS defense.

For the reasons explained above, much effort has been put into finding new methods for defending against DDoS attacks, also in the academic community. Among the many proposed solutions, one is to cluster network packets or service request with a learning algorithm and to use the learned clusters later to classify and filter the traffic. This is the DDoS defense method on which we focus in this thesis. The *cluster-based filtering* solutions do not require changes in the existing Internet model, and show promising results in DDoS defense since the problem of legitimate but malicious traffic is tackled from a different perspective. Namely, the clustering algorithms can be applied to unlabeled data, i.e., there is the unsupervised learning phase. During such a phase, the particular normal traffic distribution can be learned. Afterwards, the filter created according to the normal profile gives precedence to the traffic matching the normal classes. In such a way, the attack traffic, unless it matches the normal profile, is filtered, without the need to explicitly identify it as malicious.

1.2 Research goals

In this thesis, we analyze and experiment with a cluster-based filtering method in DDoS defense. The idea behind this method is that the normal traffic is clustered using a data mining algorithm, and the filter for DDoS traffic is created based on the output cluster set. Some of the questions that should be resolved in the thesis are the following:

(i) For *which types of DDoS defense* the cluster-based filtering method can be employed?

(ii) What is the filtering policy when cluster-based method is used for *proactive DDoS defense* with a fixed strategy?

(iii) What is the filtering policy when cluster-based method is used for a *reactive DDoS defense* with a fast-adapting strategy?

(iv) How *effective* is cluster-based filtering in DDoS proactive or reactive defense?

(v) What is the most effective *attack strategy* when the defender is using a fixed cluster-based filtering policy?

(vi) Can *existing quality-of-service (QoS) capabilities* in standard operating systems and routers be used to implement the cluster-based filtering for DDoS defense?

We first *theoretically* model and analyze the cluster-based filtering method in DDoS prevention. Our focus is on a scenario where a web server or a firewall is applying the clustering on HTTP request data in order to defend against request flooding or packet flooding. We analyze the attack and defense strategies. Particularly, we create a game-theoretic model of the traffic filtering based on clustered traffic. Using this model, we find the optimal capacity reservations for the traffic classes in order to to maximize the amount of honest traffic served during the attack, also called the normal packet survival ratio (NPSR).

In the second part of the thesis, we *experiment* with the model using a particular clustering algorithm, hierarchical heavy hitters (HHH), and the Differentiated Services (DiffServ) quality-of-service feature in Linux routers. In this testbed, we confirm our theoretical results. The traffic data used in the experiments is a mixture of traffic logs from a web server at Aalto University and synthetic attack traffic.

1.3 Thesis outline

The thesis is structured as follows. Chapter 2 presents background about DDoS attacks and an overview of existing DDoS defense methods with particular focus on solutions based on clustering the traffic as well as solutions that employ DiffServ. In chapter 3, we present a game-theoretic model of cluster-based filtering and calculate optimal strategies for the attacker and the defender. Chapter 4 describes the testbed setup and the results of the experimental evaluation. Last, we discuss the obtained results in chapter 5 and give our conclusions in chapter 6.

Chapter 2

Background and related work

2.1 DDoS attacks

Denial of Service (DoS) attacks, as the terminology suggests, aim at disrupting the normal functioning of services and access to resources. In Internet, *network* or *bandwidth DoS attacks* are performed by sending large volumes of useless traffic in order to exhaust network capacity and deny the victim's connectivity. In *distributed DoS (DDoS)* attacks, the attacker is able to compromise a large number of computer hosts in the network and exploit them for perpetrating a coordinated attack. The compromised hosts are called zombies or bots and, collectively, a *botnet*. Such scenario is depicted in figure 2.1. While the owners of the bots are usually not aware of the whole process, the victim may be easily overwhelmed with the vast amount of malicious traffic sent to them.

The first DDoS attack conducted at a large scale has being reported more than a decade ago, on the network of University of Minnesota [18]. The fact that the CERT (Computer Emergency Response Team) has decided not to publish the number of DDoS attacks reported since 2004 due to the excessive number of such security incidents, witnesses the threat DDoS attacks are representing in the Internet world nowadays. Besides that, finding accurate data about DDoS attacks is difficult due to the fact that companies are not willing to publish the attacks that they face, and another reason is that measuring and comparing of DDoS attacks is not simple. However, some data statistics about recent DDoS attacks can be found in the World Wide Infrastructure Security Survey for the year 2010 [13]. Figure 2.2 shows the increase in the power of DDoS attacks as reported by 37 large ISPs participating in the survey.

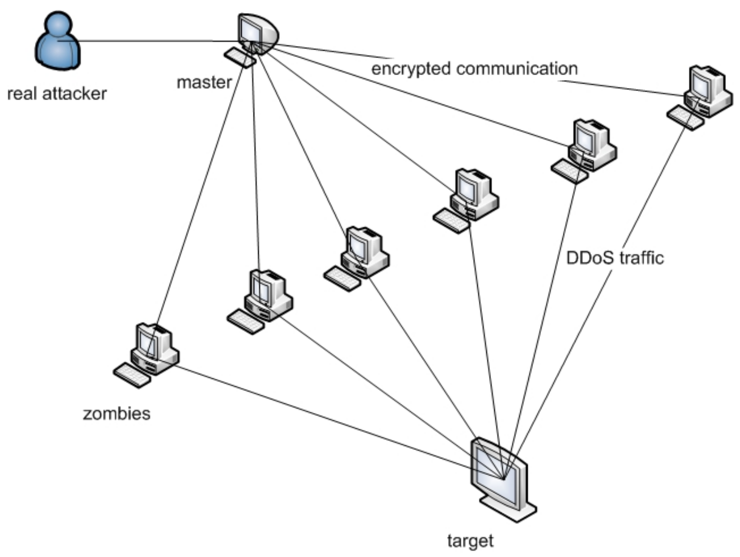


Figure 2.1: Common DDoS senario.

Current Internet architecture and DDoS

According to Peng et al. [46], the inherent design principles of the Internet architecture which made Internet so widely adopted are at the same time causes of many security threats in it, particularly DDoS attacks. For instance, Internet is a *packet-switching network* intended to provide best-effort service to users by sharing all the resources. However, this feature that makes Internet flexible, has the shortcoming that one user's service might be disrupted by the behavior of some other users. Unintended disruption appears, for example, in the case of *flash crowds*, while intended disruptions are *DoS attacks*. The *end-to-end* design principle means that the core networks should be simple while complex tasks are pushed to the end hosts. While supporting easier development of new protocols and applications, this means lack of authentication in core routers of Internet thus enabling *IP spoofing*. Similarly, *multipath routing* makes Internet efficient and scalable, but tracing of packets in the network becomes difficult. Finally, the *decentralized management* that enabled fast Internet growth, requires deployment of security solutions on many locations, which in particular leads to expensive DDoS defense mechanisms.

Taking into account the previous statements, a broad categorization of suggested solutions for defense against DDoS attacks would be:

(i) solutions that demand changes in the whole Internet model,

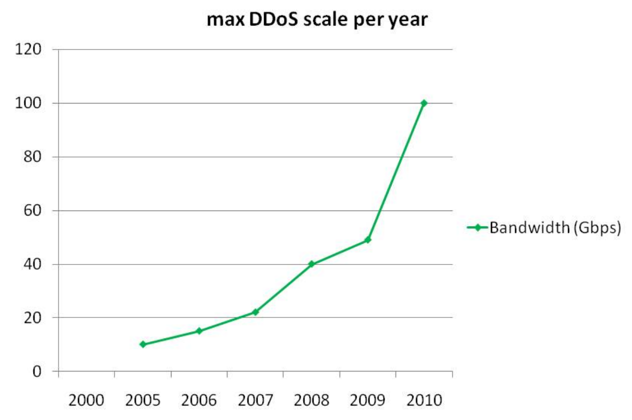


Figure 2.2: Maximum DDoS attack scale observed per year 2005-2010. Source [13].

(ii) solutions that easily integrate with the existing Internet architecture.

This work focuses on **cluster-based traffic filtering** which belongs to the second category of solutions.

2.2 DDoS defense types

DDoS defense mechanisms can be classified according to the main role they take in the process of defense as follows:

(i) detection mechanisms,

(ii) source identification techniques,

(iii) prevention and reaction mechanisms.

The distinction between DDoS prevention and reaction is not always clear, since often one mechanism serves for the both of the functions, depending on its location in the network. That is the case with the method we focus on, the cluster-based traffic filtering, as we will discuss later. Thus, it is useful to add to the previously stated categorization, the following distinction between the prevention and reaction mechanisms:

(i) prevention mechanisms — are deployed *close to the attack sources*,

(ii) reaction mechanisms — are deployed *close or at the attack target.*

2.2.1 DDoS detection mechanisms

Detection mechanisms might not obviously seem necessary, since the DDoS attacker does not focus on hiding the attack which becomes apparent as soon as the disruption of resources causes degradation of target's services. However, from the defender's point of view, the moment of detection of the attack is important. First, earlier detection will enable faster reaction to the attack and lessening its impact. Second, detection can help in finding the sources of the attack and eventually identifying the attackers.

The two main types of detection mechanisms are determined depending on the concept that detection is based on:

- detection based on specific features of the DDoS traffic,
- *anomaly-based* detection.

Before describing DDoS detection based on specific features of the DDoS traffic, we should mention the security platform called *network intrusion detection system (NIDS)*. Namely, NIDS are implemented as a separate devices or software components with the function of monitoring both ingress and egress traffic in order to detect intrusions. Intrusions include different types of malicious activities and attacks, particularly DDoS attacks. The detection based on specific features of the attack traffic is often the part of *signature-based* network intrusion detection systems. Signatures are formed from specific features of the know attack types. Example assumptions that are made about specific features of DDoS traffic are:

- the DDoS traffic does not comply with the TCP flow control,
- there is a disproportion between the packet rate at the victim and close to the sources,
- the ratio of SYN packets compared to FIN and RST packets will be unbalanced in the case of SYN attacks.

However, these assumptions are not always valid. For example, an attack using a large number of bots may be conducted in such a way that each of them opens a legitimate TCP connection to the target. The attacker may send the FIN or RST packets in conjunction with the SYN.

In order to bypass the problem of finding specific features of the attack traffic, the *anomaly-based detection* schemes use different approach to tackle the challenging task of DDoS detection. Anomaly-based detection relies on finding normal traffic patterns and then classifying any traffic that does not comply with those patterns as malicious. The advantage of such approach is that previously unknown types of attacks can be detected. However, an obvious weakness of the anomaly-based detection is in *false positives*. NIDS that are based on anomaly-detection are called *anomaly-based intrusion detection systems (ADS)*. Normal traffic is analyzed using approaches such as artificial intelligence, data mining or statistical modeling for finding the normal traffic patterns. Later on, the traffic is monitored and distance from normal patterns is measured using some statistical metric. When the difference value is above a certain threshold, an alarm for a DDoS is raised.

Clustering of network traffic *in the role of DDoS detection belongs to the anomaly-based detection. Decision upon specific features to be used in the data mining is needed, as well as the choice of the most suitable clustering algorithm. The final task is to define the distance between the normal patterns, i.e, observed cluster set and the attack traffic. If the threshold for the distance is exceeded, the attack alarm is raised.*

2.2.2 DDoS source identification techniques

Source identification techniques try to identify the attack sources. In the case of a single flooding source this is still doable. In the case of DDoS, having many bots involved and with the attacker's communication to them usually being encrypted, source identification is difficult or impossible to achieve. Examples of source identification are IP traceback schemes with hash-based IP traceback scheme arguably be the most effective [48], [49].

2.2.3 DDoS prevention and reaction mechanisms

Prevention mechanisms are intended to prevent the attack traffic from reaching the target preferably close to the attack sources, while *reaction mechanisms* need to be launched as the final defense step, when prevention techniques fail and after the attack is detected. The main classes of prevention mechanisms include:

- general mechanisms which prevent host compromise,
- identifying and disrupting botnets, and
- *filtering techniques* when deployed close to the attack sources.

DDoS reaction mechanism comprise

- *filtering techniques* when they are employed for the *target bandwidth management.*

General mechanisms include common techniques that help in improving the security of the system, such as, disabling unused services, replication of resources, installing newest security updates, and disabling IP broadcast [20].

Since botnets are the most important resource for the DDoS attacker, it is crucial to identifying and disrupt the botnets. Nowadays, the attackers are able to compromise large botnets, having hundreds of thousands or even millions of bots. Thus many Community Emergency Response Teams (CERTs) and also the corporations such as Microsoft, have taken part in actions for disrupting some of the large botnets [39], [40]. Being outside of the scope of the thesis, we herein only refer to some of the academic solutions in this domain [36], [9].

Some of the filtering techniques taking the role in DDoS prevention are briefly discussed bellow. Ingress and egress filtering of traffic [17] needs to be deployed on the edge routers of the network. Essential for this type of filtering is the knowledge about which IP addresses can be expected in the packets entering or leaving the network on specific ports. One weakness of ingress and egress filtering is that providing the routers with the mentioned type of knowledge is not easy in complicated topologies existing today in Internet. Park et al. [44] proposed routers-based packet filtering (RPF) which should be deployed on the core routers and use the knowledge about BGP (Border Gateway Protocol) routing topology. The authors provide simulation results that RPF would reduce the number of spoofed packets significantly if the method is deployed by 18% of the autonomous systems (AS) that form the Internet. One weakness is that RPF thus requires changes in the whole Internet. Also, the attacker could still spoof the IP addresses from inside one domain. Source Address Validity Enforcement (SAVE) [31] demands changes in the routing protocol. Routing tables containing expected IP addresses at specific interfaces need to be created. Weaknesses are that SAVE requires changes in the Internet routing and if partially deployed, the aim of preventing spoofing will not be achieved. Keromytis et al. [25] proposed implementing a new routing protocol for the purpose of DDoS prevention named Secure Overlay Service (SOS) protocol. The aim of SOS is to provide communication between legitimate users and the victim during the DDoS attack. Again, the changes in the existing Internet model are required, with additional security treat due to introducing the new routing protocol. The common weakness applies to all the filtering methods described above. They base the prevention on the assumption that DDoS traffic is spoofed. Thanks to the possibility of compromising the whole botnets, DDoS attacks nowadays do not need to be based on the spoofed traffic. Anderson et al. [2] proposed capability based method in DDoS prevention where the destination has the ability to control the traffic directed towards itself. By sending specific tokens to the chosen hosts, the receiver grants permissions to those hosts to send the traffic to it. While addressing the problem of spoofed traffic in a novel way, the capabilities based method enables a new type of attacks in Internet — Denial of Capability (DOC). In the case of DOC, the attacker prevents the capability-setup packets from reaching the hosts.

Mere prevention mechanisms in the case of DDoS attacks are not sufficient due to the following reasons. Since the attacks are based on the quantity of the sent traffic and not on the type of packets, DDoS attacks can be conducted with the traffic that imitates the normal traffic. Due to the distributed nature of the attack, there are many compromised hosts each of them sending seemingly normal traffic. This feature of legal but malicious traffic makes DDoS traffic difficult to distinguish from normal, especially *close to the sources.* That is why reaction mechanisms in DDoS defense are also necessary.

History based filtering methods [20] suggest recording the normal traffic during some period of time and using those traffic records as baselines for filtering later on. Namely, the idea is that the structure of the DDoS traffic will differ from the structure of the normal traffic. A straightforward idea is that the set of observed IP addresses will differ during the attack. Pang et al. [45] analyze using a database of observed IP addresses (IAD) on the edge router in order to prevent requests with the addresses that are not recorded previously. The weakness of using IAD is cost of storage and, similarly to the previously described filtering methods, the assumption that the attack traffic is spoofed.

Cluster-based traffic filtering*, when implemented close to the attack sources is a DDoS prevention mechanism, while when it is deployed at the target network, serves as a target bandwidth management scheme in DDoS reaction. The method conceptually belongs to the history based prevention since it relies on finding clusters in the normal traffic, and using them as filters when the malicious traffic appears.*

Clustering of traffic as a DDoS prevention or reaction has one important advantage over the other filtering methods. While the previously described filtering methods need to distinguish is some way between the normal and the attack packets, the cluster-based filtering does not need such distinction. Filtering based on clusters prioritizes some traffic classes over others, thus increasing *normal packet survival ratio (NPSR)*, but inside each of the classes it is not necessary to know which packets are malicious. This also means that there is no distinction made between the attack and a congestion caused, for instance, by a flash crowd. Both are treated in the same way, the packets are dropped according to the normal traffic profile. *From the explained, the importance is clear of the quality of the clustering for the effectiveness of the cluster-based filtering.*

In the scenario we examine, a cluster-based filter is placed on a router between the web server, which is the target, and the Internet. The filter is used for the bandwidth management by setting different reservations of bandwidth for the traffic classes corresponding to the different clusters. We focus on the question how to create the bandwidth management policy using clusters so that the DDoS attack impact is maximally reduced. We consider first proactive and then reactive DDoS defense.

2.3 Traffic clustering in DDoS defense

In the previous section we have described which roles can take cluster-based traffic filtering in the DDoS defense. In this section we present an overview of different academic proposals that involve clustering the network traffic in DDoS defense.

2.3.1 Clustering algorithms for network traffic

As we discussed in the previous section, the important factor in DDoS defense that uses cluster-based filtering is to find 'the good set of clusters' in the traffic. Thus many academic solutions are focused solely on the question how to cluster network traffic into **qualitatively good classes**.

Erman et al. [14] examined *existing data mining algorithms*: density-based spatial clustering of applications with noise (DBSCAN) [16], K-Means [35] and AutoClass [7] [8]. The algorithms are evaluated and compared from the point of accuracy of clustering traffic into known traffic classes, then from the point of algorithm speed and number of clusters that is produced. McGregor et al. [38] also applied machine learning techniques to cluster network packet header traces. The authors showed appropriate correspondence between known traffic types in the traces, such as HTTP, SMTP, IMAP or TCP DNS, and the traffic classes obtained by the clustering.

However, the common data mining algorithms are not the best choice for the network traffic data. Firstly, the traffic datasets have *larger size* compared to the common datasets in data mining. Secondly, for the purpose of DDoS defense, network traffic often needs to be examined in real time, meaning that clustering should be applied to the *data stream.* Common data mining algorithms are slow for such task, or would require so high computing resources that are not available in networking devices. Third, the traffic features such as IP addresses and URLs, have the specific *hierarchical structure*. Thus researchers developed specific algorithms for clustering the network traffic.

Cormode et al. [10] [11] developed the *hierarchical heavy hitters algorithm (HHH)* which is an extension to the heavy hitters algorithm and takes into account the hierarchical structure of the network traffic. The authors also provide approximate versions of the HHH algorithm that are fast enough to cluster the data stream. Theoretical proofs on the error bounds for the approximate algorithms are provided. Since we use an implementation of the HHH algorithm in the evaluation phase, a more detailed description of HHH is given in chapter 4. Hijazi et al. [22][23] developed another algorithm which takes into account specific features of the network traffic. The algorithm is named *approximate divisive hierarchical clustering (ADHIC)*. ADHIC is an adaptive algorithm since the features used during the clustering, called *(p,n)-grams*, are chosen depending on the structure of the particular dataset. (p,n)-grams are defined as "a byte strings of length n located at an offset p in a packet within a data stream." The authors of ADHIC also developed a fast and lightweight algorithm implementation intended for the streaming data which is called netADICT.

2.3.2 Traffic clustering as a DDoS detection technique

Most of the prior work involving traffic clustering focuses on anomaly and, particularly, DDoS **detection**.

Lakhina et al. [29] describe an anomaly detection method that relies on changes in *the feature distributions of traffic*. In the case of DDoS, the authors find that features that are affected are source and destination addresses. *Entropy* is used as a metric when detecting changes in traffic distributions induced by anomalies. The main advantage of using entropy over volume-based metrics is that entropy can detect and classify low-volume anomalies, such as port scan, which stay undetected, or it is not possible to distinguish their structure by volume-based methods. However, entropy cannot distinguish between different distributions having the same amount of uncertainty. Gu et al. [19] create a detection model that uses relative entropy that avoids such problem in detection. Relative entropy or *Kullback-Leibler divergence (KL-divergence)* [28] [27], is a measure of the difference between two probability distributions and is often interpreted as non-symmetric metric. The authors first decide on a certain number of packet classes, precisely 2348 two-dimensional classes induced by the protocol and port numbers. Second, the *maximum-entropy configuration* is calculated by feature selection and parameter estimation giving an initial, normal traffic, configuration. This configuration is then used as a reference point to compare the monitored traffic to it. If the *KL-divergence* from the maximum-entropy distribution is found to be greater than a threshold value on any packet class, then an alarm for that packet class is raised. Stoecklin [50] suggests improving the work of Gu by using *symmetrical KL divergence* as the deviation measure. The author provides experimental evidence that such detection is resilient to periodical changes in normal traffic which reduces false positives.

Oldmeadow et al. [43] present a time-varying adaptive clustering algorithm that is based on the *fixed-width clustering* [15]. Fixed-width clustering is a clustering algorithm that is based on a geometric framework and can be applied to unlabeled data. The authors improve the detection based on fixed-width clustering by traffic feature weighting. Zhong et al. [53] apply two exisitng clustering algorithms to DDoS detection. The algorithms used are: *fuzzy c-means (FCM) algorithm* [3] and *apriori association algorithm* [1]. After creating the normal traffic profile, the authors developed the model to discern abnormal traffic and measure its duration. The evaluation in LAN network showed that the DDoS detection reaches up to 97% accuracy. Ming-Yang et al. [51] applied to DDoS detection the *k-nearest neighbor (KNN) algorithm* [21] improved by feature weighting and selection based on a genetic algorithm [24]. Overall accuracy over 97% for known DDoS attacks is achieved, and over 78% in the case of unknown attacks.

Cluster analysis of traffic is used for *proactive DDoS defense* by Lee et al. [30]. Namely, by exploiting the specific architecture of DDoS attacks, they provide evidence that it is possible to detect five different phases of the DDoS attack by looking into specific traffic features in that phase. Such early phase detection is promising since appropriate prevention and reaction techniques could be launched in the earlier time.

2.3.3 Traffic clustering in the role of DDoS prevention and reaction

As we discussed before, filters based on clustered traffic are often used for both functions, for the DDoS **prevention and reaction**.

Muhai et al. [42] present a *mathematical nonlinear defense model* against DDoS. They involve the victim, a router and an additional application server in the defense process. Using the model it is possible to control how much of the attacker and regular user traffic is served. The main weakness is in the assumption that the regular user traffic and the attacker traffic can be distinguished. However, depending on how well is the attack adapted to the normal traffic patterns (for example by spoofing addresses so that they appear legitimate) such distinction might not be possible. The authors also describe DDoS detection by the traffic analysis. After the attacks is detected, the application server sends the signal to the router to start the nonlinear traffic control system.

Similarly to our approach, Matrawy et al. [37] do not explicitly distinguish between malicious and regular traffic, but rather filter disruptive traffic according to its membership in clusters. The authors deploy traffic clustering as a reaction technique in DDoS defense in the following way. The clustering algorithm that is used is ADHIC, developed particularly for the traffic data as we described in section 2.3.1. Adaptive traffic management based on *traffic shapers* is applied after establishing the baselines. A common weakness with our approach is the assumption that the attacker cannot observe or accurately imitate honest traffic. In our work, different clustering algorithm is used, HHH algorithm, and the implementation of bandwidth management which is based on DiffServ is different. In our opinion, DiffServ enables more possibilities in DDoS defense, since among other traffic provisioning features it can, in particular, provide the traffic shaping.

Even though not employing a classical clustering mechanism, the approach by Li et al. [32] is similar to our work in building a *mathematical model* based on traffic classes and bandwidth measurements. After creating the model, the bandwidth reservations decided by the model are *allocated to the specific classes*. We follow the similar procedure in chapter 3. However, unlike in our approach, the traffic, is classified into only two classes: normal and anomalous. The traffic feature used to distinguish those two classes is the ratio of the number of common, i.e., users that have been already recorded, to the number of users that have been recorded for the first time. Instead, we apply the similar ratio of normal traffic rate to the one observed during the congestion, on each of the traffic cluster. We thus create more fine-grained filter without need to explicitly classify traffic packets into normal or malicious.

2.4 DiffServ in DDoS defense

DiffServ is a quality-of-service provisioning framework that we used in evaluation phase for creating filtering policy based on clusters. DiffServ is described in chapter 4. In this section we describe some of the academic DDoS defense solutions using DiffServ.

The idea of using DiffServ in DDoS defense is not new. We briefly describe some of the solutions bellow. An approach for adaptive traffic management is suggested by Lin et al. [34]. *The DiffServ environment with CBQ algorithm* is used for reducing disruption of normal traffic during the attack. After deciding whether the incoming packet is normal or not, it is passed to the high priority queue or low priority queue, following the queuing discipline. The decision about the packet type is based on the difference from the harmonic mean of the intervals of transmission times of incoming packets. Differing from the mentioned approach, we use HTB queuing discipline, which is an advancement of CBQ. One of the early papers with similar idea [47] describes implementing CBQ mechanism as a prevention technique against DDoS on a router. *CBQ* and *RED queuing disciplines* were shown to be successful in providing fixed bandwidth to legitimate users and reducing the impact of DDoS. Our work differs from the described proposals since we combine DiffServ with *traffic clustering*. Thus we effectively use the DiffServ capability of provisioning many different traffic classes, instead of having only two classes (normal and anomalys). Presented in the paper [33] is another similar approach to the ours, due to the use of *DiffServ* and priority scheduling. However, the authors employed DiffServ for the DDoS detection, while we focus on the DDoS reaction. Based on the mathematical analysis of traffic constraint function as introduced in another paper on DiffServ traffic provisioning [52] and by introducing the average traffic constraint, the authors describe DDoS detection model in the DiffServ environment. They provide experimental evidence of detecting DDoS attack using DiffServ, with probability guaranties on identification rate, false positives and false negatives.

Chapter 3

Theoretical model

3.1 Terminology

In order to model and analyze traffic clustering as a DDoS defense mechanism, we first chose a particular *scenario* in which traffic clustering is deployed. Since we focus on traffic clustering as a DDoS reaction mechanism, we deploy the mechanism close to the target in the scenario. Secondly, a *theoretical model* for traffic clustering is created by abstracting only relevant elements from the scenario. Afterwards, *mathematical analysis* is applied to the theoretical model. The results of the analysis provide an insight into the effectiveness of *traffic clustering as a DDoS reaction technique*.

Many solutions based on traffic clustering use two versions of the clustering algorithm, which are often called the *offline* and *online* algorithm. The offline algorithm is applied to recorded normal traffic to obtaining the normal traffic clusters. The online algorithm is an approximation of the offline algorithm. It is faster and less demanding, so that it can be applied to the live data stream. Clustering of streaming data is necessary in DDoS detection and also for the adaptive DDoS reaction.

A **traffic cluster** is defined by ranges of *traffic feature values*. All the packets in which the feature values are inside these ranges form a traffic class. The terms traffic class and traffic cluster are often used interchangeably. Additionally, we also take as a part of the cluster definition the *amount* of the recorded traffic belonging to the class.

3.2 Scenario

In our scenario, the target is a web server, as shown in figure 3.1. The web server has exposed its services through a router on which the clustering

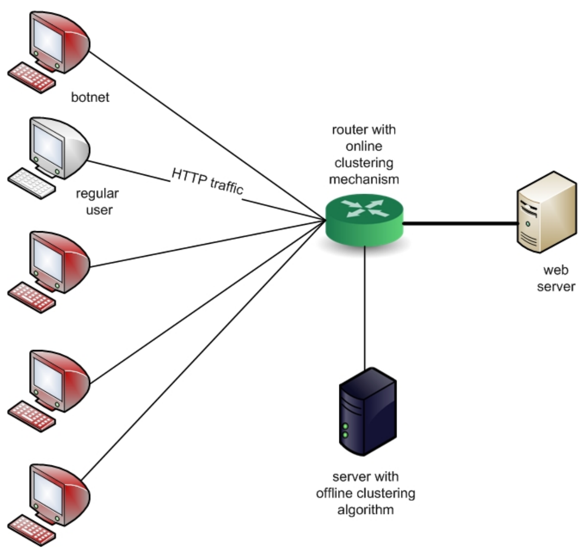

botnet
regular user
HTTP traffic
router with online clustering mechanism
web server
server with offline clustering algorithm

Figure 3.1: Scenario with clustering mechanism deployed.

mechanism is deployed. To make our model as general as possible, in the start, we do not think of a particular clustering algorithm — we just take the algorithm as a black box that outputs clusters from the network traffic.

In the scenario, the normal traffic is observed over a certain period of time. Afterwards, some offline algorithm is applied to the traffic records. The offline algorithm outputs a cluster set which represents the normal traffic profile. A filtering policy on the router is created using that profile. When the attack happens, a predefined action using the filtering policy is taken on the router as a DDoS defense reaction. The heavy task of the initial cluster computation may be allocated to a separate server which executes the offline algorithm. In that case, such a server may instruct the router about the filtering policy that the router should deploy. The online algorithm might be employed on the router to cluster the traffic stream passing through the router. In the case of an attack, the online algorithm provides information about the distribution of the attack traffic. Using the output of the online algorithm, the filter reservations may be adapted so that the DDoS reaction is improved. The success of this method, however, depends on the particular clustering algorithm used. If successfully employed, this method can be also used for DDoS detection. We analyze different possibilities when only the offline algorithm is used.

Depending whether the attacker or defender is more agile in changing its attack or defense strategy, we distinguish the following cases in our scenario:

(i) In *scenario 1*, the defender 'makes the first move', meaning that we start from a fixed defender strategy and analyze what is the **best strategy for the attacker**, who is the more agile party. We analyze such a scenario in section 3.5.1.

(ii) In *scenario 2*, the defender is the less agile party and decides to use the filtering policy that will serve as the **optimal proactive defense** against the more adaptive attack. We find out in section 3.5.2 that such optimal filtering policy should allocate cluster reservations proportionally to the normal traffic cluster sizes.

(iii) In *scenario 3*, the attacker 'makes the first move'. When the attack is launched, the changes in the amounts of traffic recorded per class will be observed by the filtering mechanism on the router. Using the data about new traffic amounts, the defender can adapt its filtering policy for the **optimal reactive defense**. Thus, in this case, the defender is the more agile party. We mathematically analyze such a scenario in section 3.5.3.

In this chapter, we find mathematical results for the optimal strategies in the described cases, and then experimentally evaluate those results in chapter 4. Since the scenario that we describe is simple, the model we create from it in the next section is general. This means that it may be applied with some modifications to other scenarios, such as when traffic clustering is deployed at other locations, for instance on the server itself or at edge routers in the network.

3.3 Problem classification and preliminaries

As stated in our definition of DDoS attacks, we focus on capacity exhaustion attacks. Thus the *capacity* of the server is the main resource we need to model. The filtering policy on the router will be based on different bandwidth allocations to traffic classes corresponding to different clusters. Our aim is to find the optimal filtering policies that will serve for DDoS defense in the different scenarios that we described above. The class of mathematical problems which deal with allocating resources in an economic way is called *resource allocation* or *knapsack problems*. Thus, the problems that will arise during the analysis in our model can be classified as knapsack problems.

The **knapsack problem** has two distinct variants, depending on the variable that is optimized:

- discrete knapsack problems, and

- continuous knapsack problems.

When commonly referring to the knapsack problems, it is usually intended to think of the *discrete knapsack* case. The knapsack problem in that context can be simply defined as follows. We have a finite number k of items available. Each item has a *value* and a *price* associated with it. The requirement is to choose a subset of the items from the available set which will give the highest value sum for a defined price that we can afford. The name for the knapsack problem comes from the most often described example where it can be applied. A person intends to go to a trip taking her stuff in one knapsack. She wants to take the most important (value) stuff that she needs and as many as possible of them. But she is bounded by the size of the knapsack (the maximum price) and the volume that each item requires (the price of the item). The knapsack problem is to choose the set of items which can be taken in the knapsack so that their value is maximal. Further on, there are different variants of the knapsack problem depending on the set of available items. Different possibilities include the cases when all the items in the set are distinct, or when there is a certain bound on the number of items of each type, or if there is an infinite number of each type of items.

However, in our model, we need to deal with the less well known variant of the knapsack problem, the **continuous knapsack** problem. In the continuous knapsack problem, a *fraction* of an item can be taken, i.e., the variable values are not constrained only to the integers, but can take the real values.

Another categorization of knapsack problems is done depending on the objective function that is optimized:

- linear knapsack problems, and
- nonlinear knapsack problems.

The knapsack problem as we described it above has the feature that the objective function in the problem is linear. Thus until this point we were discussing the class of the *linear knapsack* problems. A class of more complex resource allocation problems appears in the case when the objective function is nonlinear and they are called **nonlinear knapsack problems**.

(Coming back to the common example with the stuff that needs to be optimally packed in a knapsack, we see now that we may classify such a problem as a discrete linear knapsack problem.)

A continuous form of the linear knapsack problem is solvable more easily compared to the discrete knapsack case since the solution can be given in the form of a simple *greedy approximation algorithm*, as proposed by Dantzig

[12]. According to the algorithm, the items are sorted depending on the item value, and we keep on taking the most valuable item as long as it is available. Then the following most valuable item is being consumed and so on, until we reach the price limit or exhaust all the items available.

One of the optimization tasks that we will define for the defender strategy in our model belongs to the class of *continuous linear knapsack problems*. That optimization will be solved in section 3.5.3 using the above mentioned greedy algorithm. The optimization problem for the attacker optimal strategy and the second optimization task for the optimal defender strategy in our model will both be classified as a *continuous nonlinear knapsack problems*. However, while the optimization of the defender strategy is possible to solve using the simple derivative approach, as we show in section 3.5.2, the attacker strategy optimization in section 3.5.1 will require a more advanced algorithm. That algorithm is called *the multiplier search method for the continuous convex nonlinear knapsack problem* [5] and, since it is not as well known as the derivative approach in optimization, we describe it in more detail in the following section.

3.3.1 The multiplier search method for the continuous convex nonlinear knapsack problem

The characteristic that defines knapsack problems in the group of all optimization problems is that they have *one* linear constraint besides bounds on the variables. For that reason, each algorithm that is defined as a solution for a general optimization problem with many constraints can be applied to solving the knapsack problems. However, since such algorithms are rather complex and slow, there are specialized algorithms for solving the different variants of the knapsack problems.

Depending on the properties of the problem, different algorithms have been developed for the variants of nonlinear knapsack problems. The properties that define different variants of the problem include:

- whether it is a continuous or a discrete problem,
- whether the objective function is convex or not,
- whether the objective function is separable or not.

A function $f(x_1, ..., x_k)$ is separable if it can be represented in the form

$$f = \sum_{i=1}^{k} f_i(x_i). \tag{3.1}$$

The multiplier search method is applicable to the continuous nonlinear knapsack problems having separable and convex objective function. A general form of such a problem is given bellow:

$$\min f = \min \sum_{i=1}^{k} f_i(x_i), \tag{3.2}$$

subject to constraints:

$$\sum_{i=1}^{k} g(x_i) \leq b, \quad l_i \leq x_i \leq u_i, \qquad \text{for } i = 1, ..., k. \tag{3.3}$$

It is assumed that each function f_i is convex.

Methods for solving nonlinear optimization are in general based on Lagrange multipliers and they use Karus-Kuhn-Tucker (KKT) conditions. the multiplier search method manipulates with the KKT conditions for the nonlinear knapsack problem, in order to simplify the general solution based on solving the set of KKT. Namely, solving of a set of KKT conditions is reduced to finding only one multiplier, λ^*. The solution of the given problem using the multiplier search method follows. If the Lagrange multipliers are defined so that $\lambda \geq 0$ is the multiplier for $\sum_{i=1}^{k} g(x_i) \leq b$ and $v_i \geq 0$ is the multiplier for $l_i \leq x_i$ and $w_i \geq 0$ is the multiplier for $x_i \leq u_i$, the KKT conditions for such a problem can be written as bellow:

$$\lambda \left(\sum_{i=1}^{k} g_i(x_i) - b \right) = 0, \tag{3.4}$$

$$\frac{\partial f_i}{\partial x_i} + \lambda \frac{\partial g_i}{\partial x_i} - v_i + w_i = 0, \tag{3.5}$$

$$v_i(l_i - x_i) = 0, \tag{3.6}$$

$$w_i(x_i - u_i) = 0, \text{ for } \quad i = 1, ..., k, \tag{3.7}$$

Additional conditions for the purpose of the multiplier search method method are defined as follows:

$$x_i(\lambda) = \begin{cases} l_i & \text{if } \ \tilde{x}_i(\lambda) \leq l_i \\ \tilde{x}_i(\lambda) & \text{if } \ l_i < \tilde{x}_i(\lambda) < u_i, \\ u_i & \text{if } \ \tilde{x}_i(\lambda) > u_i \end{cases} \tag{3.8}$$

$$v_i(\lambda) = \begin{cases} \frac{\partial f_i(l_i)}{\partial x_i} + \lambda \frac{\partial g_i(l_i)}{\partial x_i} & \text{if } \ \tilde{x}_i(\lambda) \leq l_i \\ 0 & \text{if } \ \tilde{x}_i(\lambda) > l_i. \end{cases} \tag{3.9}$$

$$w_i(\lambda) = \begin{cases} 0 & \text{if} \quad \tilde{x}_i(\lambda) < u_i, \\ \frac{\partial f_i(u_i)}{\partial x_i} + \lambda \frac{\partial g_i(u_i)}{\partial x_i} & \text{if} \quad \tilde{x}_i(\lambda) \geq u_i \end{cases} \tag{3.10}$$

The multiplier search method can now be defined in the form of the following algorithm.

```
step 1: Solve the equation $\frac{\partial f_i}{\partial x_i} + \lambda \frac{\partial g_i}{\partial x_i} = 0$ finding $\tilde{x}_i$ as a function of $\lambda$.
step 2:
if $\sum_{i=1}^{k} g_i(\tilde{x}_i) \leq b$ then
  $\lambda^* = 0$
else
  calculate $\lambda^*$ from condition $\lambda^* > 0$ and $\sum_{i=1}^{k} g_i(\tilde{x}_i) = b$.
end if
step 3: Substitute $\lambda^*$ into formulas 3.8 to 3.10 to obtain the optimal solution $x_i$.
```

The multiplier search method has its name because solving of a set of KKT conditions is reduced to finding only one multiplier, λ^*.

The optimization task for the attacker strategy in our model is a continuous nonlinear knapsack problem with a separable objective function. We also find that it is possible to slightly transform the problem so that the objective function is convex. Thanks to that, our task fulfills the requirements for the application of the multiplier search method.

3.4 Mathematical model

The server bandwidth in our model is normalized to 1, corresponding to 100% utilization. We also need to model the *cluster set* and corresponding *cluster sizes* which are the output of the clustering algorithm. Finally, the *filtering policy* on the router needs to be modeled. The policy uses the knowledge about the clusters and allocates reservations to the corresponding traffic classes according to a predefined algorithm. We want to find out the best algorithm for each of the different scenarios described in the previous section.

3.4.1 Mathematical notation

For each of the identified elements to model, we introduce notation as given in table 3.1.

symbol	**conditions**	**description**
$\mathcal{C} = \{c_1, ..., c_k\}$		The set of clusters created from the normal traffic.
c_i	$c_i \in \mathcal{C}$	A cluster from the set of clusters $\mathcal{C}$.
$\rho : \mathcal{C} \to [0, 1]$ $\rho(c_i) = \rho_i$	$\sum_{i=1}^{k} \rho_i = 1$	Reservations of capacity per cluster as decided by the filter policy.
$\alpha : \mathcal{C} \to [0, 1]$ $\alpha(c_i) = \alpha_i$	$\sum_{i=1}^{k} \alpha_i = A$ $A \leq 1$	Volumes of recorded normal traffic in each cluster. We assume $A \leq 1$, i.e., normal users are not exceeding the maximum capacity.
$\beta : \mathcal{C} \to [0, +\infty)$ $\beta(c_i) = \beta_i$	$\sum_{i=1}^{k} \beta_i = B$ $A + B > 1$	Volumes of attacker traffic in each cluster. We assume $A + B > 1$ because otherwise there is no congestion and no DDoS attack.
$\tau : \mathcal{C} \to [0, +\infty)$ $\tau(c_i) = \tau_i$	$\sum_{i=1}^{k} \tau_i = T$ $T = A+B > 1$	Volumes of total observed traffic in each cluster during the attack. $T > 1$ because otherwise there is no congestion and no DDoS attack.

Table 3.1: Notation for the mathematical model.

The main function we want to observe in the model is the total amount of honest user traffic served; let us denoted it by $\mathcal{S}$. On each cluster c_i, we denote this value as f_i. Then the total amount of served honest traffic equals

$$\mathcal{S} = \sum_{i=1}^{k} f_i. \tag{3.11}$$

During the normal traffic periods, the amount of honest user traffic served on the cluster c_i equals α_i and all user traffic is served, i.e, $S = A$.

During a DDoS attack, the amount of honest user traffic served on cluster c_i will have two possible values depending on the attack traffic falling in that cluster. If the total observed traffic on cluster c_i, that is $\tau_i = \alpha_i + \beta_i$, is not exceeding the given reservation to that cluster, ρ_i, then all the honest traffic is still served since there is no need to drop packets from that class. If the attack traffic leads to higher total traffic volume in c_i than the reservation, then the assumption is that packets are randomly dropped from this cluster. The proportion of dropped normal and attack packets will correspond to the proportion of such packets in the analyzed traffic. Thus, the volume of the normal traffic served on c_i is represented by the following formula:

$$f_i = \begin{cases} \frac{\alpha_i \rho_i}{\alpha_i + \beta_i}, & \text{when} \quad \beta_i > \rho_i - \alpha_i, \\ \alpha_i, & \text{when} \quad \beta_i \le \rho_i - \alpha_i \qquad \text{for} \quad i = 1, ..., k. \end{cases}$$

Equivalently, we can represent f_i without using values β_i as bellow: [1]

$$f_i = \begin{cases} \frac{\alpha_i \rho_i}{\tau_i}, & \text{when} \quad \tau_i > \rho_i, \\ \alpha_i, & \text{when} \quad \tau_i \le \rho_i \qquad \text{for} \quad i = 1, ..., k. \end{cases}$$

Having introduced this notation and functions:

(i) The attacker's goal to maximally disrupt the normal user service, corresponds to achieving the following optimum in our mathematical model:

$$\min_{\beta} \mathcal{S} = \min_{\beta} \sum_{i=1}^{k} f_i.$$

(ii) On the other hand, the defender's (filtering policy) goal is to achieve the following optimum:

$$\max_{\rho} \mathcal{S} = \max_{\rho} \sum_{i=1}^{k} f_i,$$

[1]The capacities of attackers traffic cannot be measured directly but we can only measure the total τ_i. However, β_i are needed for modeling attacker strategy in the theoretical model.

with a natural constraint that comes in our model: $\alpha_i \leq \rho_i$, so that the filtering policy itself does not create a congestion on any of the clusters.

3.5 Analysis of the mathematical model

The introduced formal model of cluster-based traffic filtering enables us to model the strategies of the attacker and the defender in a given scenario as optimization problems. Recalling the three cases described in section 3.2, three optimization tasks need to be solved. By solving those optimization problems, we gain an insight into the impact of traffic clustering mechanism on the capabilities of the attacker and the defender in DDoS attacks.

First, in section 3.5.1, we analyze the capabilities of the attacker to disrupt normal users traffic when a cluster-based filtering policy is fixed on the router. Second, in section 3.5.2, the optimal filtering policy against the maximally adaptive attack is found. Third, in section 3.5.3, we describe the algorithm for adapting the filtering policy for the optimal defense against a certain DDoS traffic distribution.

3.5.1 Attacker strategy — minimization problem

In this section, we analyze *scenario 1* as described in section 3.2. The attacker wants to send malicious traffic so that regular traffic is maximally disrupted. For that he needs to decide on the amounts of different types of traffic he will send. In our model, that corresponds to deciding on the traffic amounts the attacker will send in each of the classes existing in the router's filter: $\beta_1, ..., \beta_k$. The function of honest throughput 3.11 is to be be minimized.

Lemma 3.5.1. *The best strategy for the attacker depends on the amount of traffic B he has at his disposal for conducing the attack. If B is not high enough, the best strategy is to skip attacking some of the clusters. In order for the best strategy to be to attack all of the clusters, the amount B should be high enough so that the proportion of the normal traffic in each cluster is higher than the following value*

$$\frac{\left(\sum_{j=1}^{k} \sqrt{\alpha_j \rho_j}\right)^2}{(A+B)^2} \quad .$$

Proof. The following **optimization problem** models the strategy of the attacker.

$$\min_{\beta} \mathcal{S} = \min_{\beta} \sum_{i=1}^{k} f_i.$$

The *objective function* is given by 3.11 with $\beta = (\beta_1, ..., \beta_k)$ as the function variable, and

$$f_i(\beta_i) = \begin{cases} \frac{\alpha_i \rho_i}{\alpha_i + \beta_i} & \text{if} \quad \beta_i > \rho_i - \alpha_i \\ \alpha_i & \text{if} \quad \beta_i \leq \rho_i - \alpha_i \qquad \text{for} \quad i = 1, ..., k. \end{cases} \tag{3.12}$$

The constraints for the optimization are

$$\sum_{i=1}^{k} \beta_i = B, \quad \beta_i \geq 0.$$

Similar to other resource allocation problems, our problem may be classified as *a nonlinear knapsack problem* [6]. Analysis of the task shows that we can simplify it in order to apply *the multiplier search method* as described in section 3.3. We mentioned there that we need a slight transformation to the problem. As is shown in figure 3.2, the functions f_i from formula 3.12 are not fulfilling all the requirements for the application of KKT-conditions. First, the functions are not continuously differentiable at $\beta_i = \rho_i - \alpha_i$ and, second, they are not convex. We can, however, solve the task using the functions f_i defined for $\beta_i > \rho_i - \alpha_i$ (represented by the dashed line in the figure). We should add an aditional constraint to our problem to take care of solutions $0 < \beta_i \leq \rho_i - \alpha_i$ since they represent the boundary case for the functions f_i. For such solutions, we should simply assign $\beta_i = 0$.

The multiplier search method application to our optimization task

According to the algorithm described in [5], the two sets of constraints are formalized for the optimization task. Namely, together with the common set of constraints in solving optimization problems with Lagrange multipliers, *the additional set of constraints specific for the multiplier search method* is defined.

The first set of constraints is given bellow.

$$\sum_{i=1}^{k} \beta_i = B \tag{3.13}$$

$$\beta_i > \rho_i - \alpha_i \quad \text{or} \quad \beta_i = 0 \tag{3.14}$$

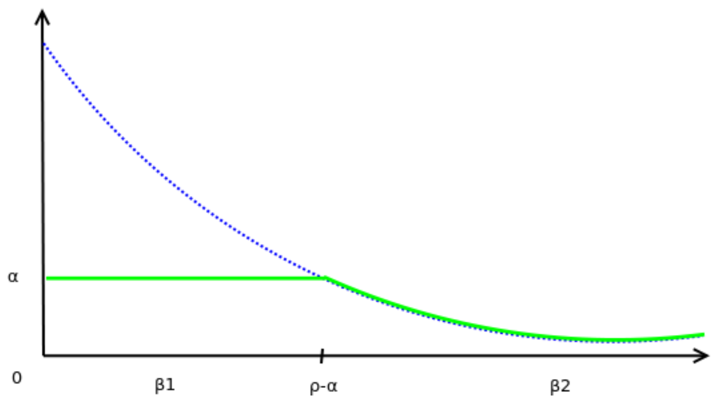


Figure 3.2: Graph of a function f_i from 3.12 is given by the green line. The function we analyse instead is represented by the dashed blue line.

$$\frac{\alpha_i \rho_i}{(\alpha_i+\beta_i)^2} - \lambda - \mu_i = 0, \quad \text{for} \quad i = 1, ..., k \tag{3.15}$$

$$\mu_i(\beta_i - \rho_i + \alpha_i) = 0 \quad \text{or} \quad \mu_i\beta_i = 0, \quad \text{for} \quad i = 1, ..., k \tag{3.16}$$

$$v\mu_i \geq 0, \quad \text{for} \quad i = 1, ..., k \tag{3.17}$$

At this step, the algorithm we apply differs from the common Lagrange multipliers approach. First we need a partial solution $\tilde{\beta}_i(\lambda)$ of the derivative Lagrangian function bellow for every i:

$$\frac{\alpha_i \rho_i}{(\alpha_i+\beta_i)^2} - \lambda = 0.$$

Then the obtained solution

$$\frac{\alpha_i \rho_i}{(\alpha_i+\beta_i)^2} = \lambda \quad \Rightarrow \quad \tilde{\beta}_i(\lambda) = \frac{\sqrt{\alpha_i\rho_i}}{\sqrt{\lambda}} - \alpha_i, \tag{3.18}$$

is used to define two *additional constraints specific to this algorithm*:

$$\beta_i(\lambda) = \begin{cases} 0 & \text{if} \quad \tilde{\beta}_i(\lambda) \leq \rho_i - \alpha_i \\ \tilde{\beta}_i(\lambda) & \text{if} \quad \tilde{\beta}_i(\lambda) > \rho_i - \alpha_i \end{cases} \tag{3.19}$$

$$\mu_i(\lambda) = \begin{cases} \frac{\alpha_i\rho_i}{(\alpha_i+\beta_i)^2} - \lambda & \text{if} \quad \tilde{\beta}_i(\lambda) \leq \rho_i - \alpha_i \\ 0 & \text{if} \quad \tilde{\beta}_i(\lambda) > \rho_i - \alpha_i. \end{cases} \tag{3.20}$$

The two additional constraints together with 3.14 to 3.17 all satisfy KKT conditions [5]. Therefore, it is only left to find λ so that the constraint 3.13 is also satisfied.

Combining $\beta_i(\lambda)$ from formula 3.19 and condition 3.14, we differentiate the cases when:

(i) $\tilde{\beta_i(\lambda)} > \rho_i - \alpha_i$ for all $i = 1, ..., k$ having

$$\sum_{i=1}^{k} \tilde{\beta}_i(\lambda) = B$$

(ii) $\tilde{\beta_i(\lambda)} > \rho_i - \alpha_i$ for $i = 1, ..., s$ and $\tilde{\beta_i(\lambda)} \leq \rho_i - \alpha_i$ for $i = s+1, ..., k$ in which case the sum becomes

$$\sum_{i=1}^{s} \tilde{\beta}_i(\lambda) + \sum_{i=s+1}^{k} 0 = B.$$

In the first case, λ is calculated easily:

$$\sqrt{\lambda} = \frac{\sum_{j=1}^{k} \sqrt{\alpha_j \rho_j}}{A+B}. \tag{3.21}$$

The assumption that all $\tilde{\beta_i(\lambda)} > \rho_i - \alpha_i$ in this case gives $\sqrt{\frac{\alpha_i}{\rho_i}} > \sqrt{\lambda}$. As we explained, if a solution is found so that $\beta_i \leq \rho_i - \alpha_i$, then such solutions should be assigned $\beta_i = 0$. That, together with λ calculated in 3.21, gives the conditions that must hold on all the clusters so that all corresponding $\beta_i > 0$, i.e., so that every cluster is attacked:

$$\frac{\alpha_i}{\rho_i} > \frac{\left(\sum_{j=1}^{k} \sqrt{\alpha_j \rho_j}\right)^2}{(A+B)^2} \quad \text{for all} \quad i = 1, ..., k.$$

With this is our lemma proved. □

One comment about the interpretation of the lemma is that by increasing the attack capacity B, the attacker may always reach the point when attacking all the clusters is his optimal strategy.

Theorem 3.5.2. *If there is enough attack capacity B, the attacker should send to each traffic class in the defender's filter the amount of traffic that makes the total amount of traffic in the class proportional to the geometric average of the capacity reservation and normal traffic volume at that class ($\sqrt{\alpha_i \rho_i}$).*

Proof. If the lemma condition is satisfied, then all $\beta_i > 0$, and inserting the result for λ from formula 3.21 in $\tilde{\beta_i}(\lambda)$ as calculated in 3.18 gives the desired solution:

$$\beta_i = (A+B)\frac{\sqrt{\alpha_i \rho_i}}{\sum\limits_{j=1}^{k} \sqrt{\alpha_j \rho_j}} - \alpha_i. \tag{3.22}$$

Such β_i are the values of the attack traffic in each cluster c_i if the attacker's best strategy is to send the traffic to all the clusters.

□

When the attacker does not have enough capacity to attack all the clusters, it is necessary to first find the number of clusters s that will be attacked for the best strategy. Also λ needs to be recalculated based on that. The solution in this case can be given in the form of an algorithm.

Algorithm 1 describes the general algorithm for the attacker to calculate his optimal strategy.

3.5.2 Proactive defense — maximization problem 1

In this section, we analyze *scenario 2* as described in section 3.2. In this scenario, the defender is the less agile party and he wants to fix the strategy that is the optimal **proactive defense** against an adaptive attacker. This kind of strategy corresponds to expecting an attacker who will always find the best attack strategy following results from theorem 3.5.2. In the worst case scenario, the attacker will have at least enough capacity B at his disposal so that all the clusters are attacked.

In our mathematical model, this corresponds to finding the optimal reservations ρ_i when the values β_i are the optimal values as calculated in formula 3.22. Since we model the defender strategy, the function of honest throughput 3.11 is to be maximized in this case.

Theorem 3.5.3. *The best proactive defense against DDoS attack when using cluster-based filtering is to assign the traffic reservations to the classes proportional to the normal traffic amounts in the classes.*

Proof. The following **optimization problem** models the strategy of the defender:

$$\max_{\rho} \mathcal{S} = \max_{\rho} \sum_{i=1}^{k} f_i.$$

Algorithm 1 Algorithm for finding attackers optimal strategy.

```
step 0: initialize $s = k$ {A variable which represents the number of clusters being attacked.}
step 1: sort the values $\frac{\alpha_i}{\rho_i}$ in decreasing order, and put them in a queue
    $$\frac{\alpha_1}{\rho_1} \geq \frac{\alpha_2}{\rho_2} \geq ...\frac{\alpha_i}{\rho_i}... \geq \frac{\alpha_k}{\rho_k}$$
step 2: calculate $\lambda$ according to
    $$\lambda = \frac{(\sum_{j=1}^{s} \sqrt{\alpha_j \rho_j})^2}{(A+B)^2}$$
step 3:
if for all $i = 1, ..., s$ holds
    $$\frac{\alpha_i}{\rho_i} > \frac{(\sum_{j=1}^{s} \sqrt{\alpha_j \rho_j})^2}{(A+B)^2}$$
 then
  return
    $$\beta_i = (A+B)\frac{\sqrt{\alpha_i \rho_i}}{\sum_{j=1}^{s} \sqrt{\alpha_j \rho_j}} - \alpha_i \quad \text{for} \quad i = 1, ..., s, \quad \text{and,}$$
    $$\beta_i = 0 \quad \text{for} \quad i = s+1, ..., k$$
else
  set $A = A - \alpha_s$ and $s = s - 1$ and go to step 2
end if
```

The *objective function* is given by 3.11 with $\rho = (\rho_1, ..., \rho_k)$ as the function variable, and the values β_i are fixed to the optimal values as calculated in 3.22.

The constraints are:

$$\sum_{i=1}^{k} \rho_i = 1, \quad \rho_i \geq 0, \quad \rho_i \leq \tau_i,$$

and each f_i equals:

$$f_i = \begin{cases} \frac{\alpha_i \rho_i}{\tau_i}, & \text{when} \quad \tau_i > \rho_i, \\ \alpha_i, & \text{when} \quad \tau_i \leq \rho_i \end{cases} \quad \text{for} \quad i = 1, ..., k. \tag{3.23}$$

The last constraint leads to reducing the domain of f_i and thus simplifying the functions to

$$f_i = \frac{\alpha_i \rho_i}{\alpha_i + \beta_i} \qquad \text{for} \quad i = 1, ..., k.$$

Using the optimal values for β_i in these functions, we get the following value of the objective function $\mathcal{S} = \mathcal{S}(\rho)$:

$$\mathcal{S}(\rho) = \sum_{i=1}^{k} \frac{\alpha_i \rho_i}{\frac{\sqrt{\alpha_i \rho_i}}{\sum_{j=1}^{k} \sqrt{\alpha_j \rho_j}} (A + B) - \alpha_i + \alpha_i} = \frac{1}{A + B} \left(\sum_{i=1}^{k} \sqrt{\alpha_i \rho_i} \right)^2.$$

The function $\mathcal{S}$ is a sum of continuously differentiable functions on the defined feasibility region. Thus we are dealing with a continuous nonlinear knapsack problem. However, it is possible to avoid the linear constraint and then transform the knapsack problem to a nonlinear optimization problem which is solvable using the simplest derivative approach. Namely, we can transform the objective function $\mathcal{S}$ to depend only on on the first $k - 1$ values ρ_i, while the last ρ_k is fixed to $\rho_k = 1 - \sum_{i=1}^{k-1} \rho_i$. Now the objective function takes one variable less:

$$\mathcal{S}(\rho) = \frac{1}{A + B} \left(\sum_{i=1}^{k-1} \sqrt{\alpha_i \rho_i} + \sqrt{\alpha_k \rho_k} \right)^2.$$

According to the derivative approach, the points that are the potential solutions (called stationary points) should have the partial derivatives $\frac{\partial \mathcal{S}}{\partial \rho_i}$ equal zero.

The stationary points have the following partial derivatives:

$$\frac{\partial \mathcal{S}}{\partial \rho_i} = \frac{1}{2(A+B)}\left(\frac{\sqrt{\alpha_i}}{\sqrt{\rho_i}} - \frac{\sqrt{\alpha_k}}{\sqrt{\rho_k}}\right)\cdot 2 \cdot \sum_{j=1}^{k} \sqrt{\alpha_j \rho_j}.$$

From the condition $\frac{\partial \mathcal{S}}{\partial \rho_i} = 0$

$$\Rightarrow \quad \frac{\sqrt{\alpha_i}}{\sqrt{\rho_i}} - \frac{\sqrt{\alpha_k}}{\sqrt{\rho_k}} = 0,$$

we easily calculate ρ_i depending on ρ_k:

$$\rho_i = \frac{\alpha_i}{\alpha_k}\rho_k.$$

Summing up obtained results in order to satisfy the previously skipped constraint $\sum_{i=1}^{k} \rho_i = 1$ gives

$$\rho_i = \frac{\alpha_i}{\sum\limits_{j=1}^{k} \alpha_j} = \frac{\alpha_i}{A}. \tag{3.24}$$

□

3.5.3 Reactive defense — maximization problem 2

In this section, we analyze *scenario 3* as described in section 3.2. In the previous section, we analyzed the possibilities for the defender when the attacker is agile. In this section we focus on the different case. The attacker 'makes the first move' and he sends fixed attack traffic. The defender is agile and tries to adapt its filtering policy for the best defense against that particular attack. Through analysis of this scenario, we want to find the optimal **reactive defense** when under the attack.

The defender, using the filtering policy deplyed on the router, may change reservations of bandwidth given to different classes so that the effects of the attack are minimized. In our mathematical model, this corresponds to knowing normal traffic capacities α_i and having measured the total traffic volumes τ_i, during the attack, and then finding the optimal reservations ρ_i. The function of honest throughput 3.11 is to be maximized in this case.

Theorem 3.5.4. *The best reactive DDoS defense when using the cluster-based filtering is to prioritize the traffic classes. The priorities are given by the ratio $\frac{\alpha_i}{\tau_i}$ of the previously observed normal traffic to the amount of traffic observed during the attack. The classes with the highest such values receive all the requested capacity, or as much capacity as is left, until there is no more server capacity remaining. The other classes are left without a traffic reservation.*

Proof. The following **optimization problem** models the strategy of the defender:

$$\max_{\rho} \mathcal{S} = \max_{\rho} \sum_{i=1}^{k} f_i.$$

The *objective function* is given by 3.11 with $\rho = (\rho_1, ..., \rho_k)$ as the function variable, and each f_i equals:

$$f_i = \begin{cases} \frac{\alpha_i \rho_i}{\tau_i}, & \text{when} \quad \tau_i > \rho_i, \\ \alpha_i, & \text{when} \quad \tau_i \leq \rho_i \qquad \text{for} \quad i = 1, ..., k. \end{cases} \tag{3.25}$$

The constraints are

$$\sum_{i=1}^{k} \rho_i = 1, \quad \rho_i \geq 0, \quad \rho_i \leq \tau_i.$$

The last constraint needs a practical explanation. Having $\rho_i = \tau_i$ means that all the traffic coming to the cluster c_i gets served. Having $\rho_i > \tau_i$ would practically mean wasting server capacity, which is obviously not desirable under congestion.

Thanks to the explained constraint, the domains of functions f_i are reduced and they get the simplified form:

$$f_i = \frac{\alpha_i \rho_i}{\tau_i} \qquad \text{for} \quad i = 1, ..., k.$$

The terms $\frac{\alpha_i}{\tau_i}$ may be interpreted as fixed coefficients. We can think of those coefficients as the cost for the corresponding capacity ρ_i. Let us denote such costs by c_i. In the end the, objective function becomes

$$\mathcal{S} = \sum_{i=1}^{k} c_i \rho_i.$$

Such $\mathcal{S}$ is a linear function. Since the constraints are also linear functions, our optimization task represents a simple case of the *continuos linear knapsack optimization* [12]. The solution of such problems is based on the *greedy algorithm* and the solution in our case can be intuitively described as follows. Since, we want to maximize $\mathcal{S}$, we should take the elements of the linear sum which have the highest cost c_i and give them the maximum corresponding capacity value ρ_i. We keep choosing the elements in this way until there is no more capacity left at our disposal. Thus, the last element taken into the sum might not get the maximum capacity ρ_i, and there might be elements left over, that will not be taken into the sum.

First, the simplest, 2-dimensional case is solved bellow since it gives a good illustration of how the solution is obtained in the k-dimensional case.

2-dimensional case

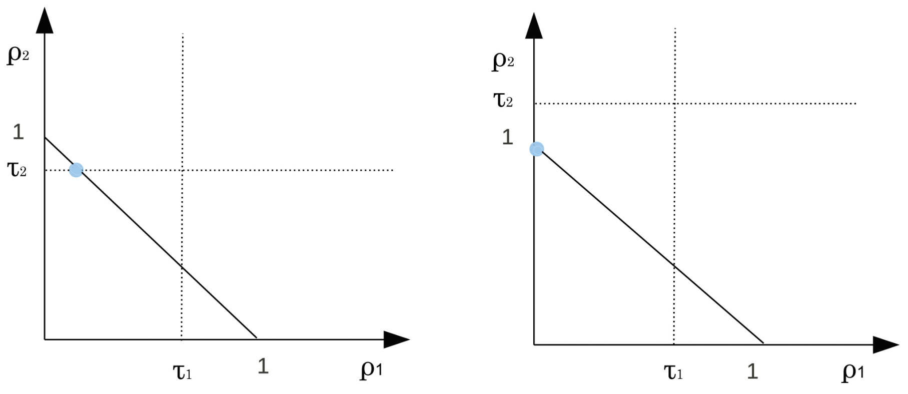


Figure 3.3: Feasibility region and possible boundary solutions (blue dots) in the 2-dimensional case.

Let the number of clusters $k = 2$. The feasibility region is the set in which the solutions of an optimization problem must belong to. Since the sum of the capacities in our problem equals 1, the solutions of our problem belong to the line $\rho_1 + \rho_2 = 1$. As shown in figure 3.3, the feasibility region in our case comprises the segment of the line $\rho_1 + \rho_2 = 1$ between its intersections with $\rho_1 = \tau_1$ and $\rho_2 = \tau_2$. The solutions are found on some of the boundary points. In the example we assume $\frac{\alpha_2}{\tau_2} \geq \frac{\alpha_1}{\tau_1}$ and the boundary point in which $\rho_2 = \tau_2$ (blue), is the solution. In this boundary point, the value of f is maximum.

Now, depending on the value of τ_2, the two possible cases are presented. In the left figure $\tau_2 \leq 1$ and the (remaining) value assigned to ρ_1 is $1 - \rho_2$. It can be also zero, if no capacity is left after assigning to ρ_2, as the right figure shows.

If the cluster c_1 gets to be served, additional subcases arise depending whether $\tau_1 \leq 1 - \rho_1$. If $\tau_1 \leq 1 - \rho_1$, the cluster c_1 is fully served; otherwise it is partially served.

Algorithm 2 describes the solution of our optimization problem in general, the k-dimensional case. The algorithm represents the optimal strategy of the defender at the time of a DDoS attack.

□

Algorithm 2 Algorithm for the optimal strategy of the defender.

```
step 0: assign value 0 to $RC$ {A variable which will represent reserved capacity. }
step 1: sort values $c_i = \frac{\alpha_i}{\tau_i}$ in increasing order and put them in a queue.
step 2: remove the maximal value $c_i$ from the queue, and assign to the corresponding $\rho_i$ the minimum of the following values: $1-RC$, $\tau_i$. Increase the value $RC$ by the value assigned to $\rho_i$.
step 3:
if $RC == 0$ then
    assign value 0 to all the unassigned $\rho_i$ in the queue and
    return 0
else
    go to step 2
end if
```

Chapter 4

Experimental setup and evaluation

4.1 Testbed methodology and requirements

In this chapter, we create an experimental testbed for evaluating the results obtained in chapter 3. The experiments will be conducted to follow the scenario and to correspond to the mathematical model presented previously. Three virtual machines are set up in a virtual network in order to serve for the router, the server and the user in the scenario, respectively. Using the real data from a web server at Aalto university, we simulate the normal traffic, and we generate different types of attack traffic for different attack scenarios. The clustering algorithm we applied to the web server traffic records in our experiments is the *hierarchical heavy hitters algorithm (HHH)*, and we describe it in more detail in section 4.1.1. Using the output from the clustering algorithm, we create the filters for the *Differentiated Services (DiffServ)* environment, particularly using its *HTB queuing discipline*. The cluster-based filter is deployed on the router in our experiments. DiffServ is a quality-of-service (QoS) provisioning environment and, for our task, it is used for defining the bandwidth management policy and for filtering the traffic according to the policy. We describe the basic principles of DiffServ, the HTB queuing discipline, and their concrete application to our requirements in section 4.1.2.

4.1.1 Hierarchical Heavy Hitters algorithm

In the *heavy hitters* data mining algorithm, the data are grouped into clusters based on the *frequency* of data items having common values on a certain subset of features. A threshold is defined, and all the feature combinations having a frequency over the threshold define clusters in the dataset. For instance, having the client IP address as the only feature in the feature set, heavy hitters would be defined as all the individual addresses that are found in the number of packets that is not smaller than the threshold value. Since the algorithm is based on finding high-frequency features, it is called heavy hitters.

In data mining *traffic* datasets, as we explained, we deal with large, sometimes live data streams, and also, as with the example of the IP address, the features of the data have a hierarchical structure. That is why Cormode at el. [10], [11] adapted the heavy hitters approach to the traffic data (or any other type of data having the hierarchical structure in features). The idea behind the *hierarchical heavy hitters (HHH)* approach is that when using individual features, such as single IP addresses, we cannot always define well the heavy hitters in the dataset. It can happen that there are many users from the campus subnet who are accessing the web server of the university. Each of the individual users is not having a number of requests to the server high enough to account his address for a heavy hitter. But all the campus users together form a subset of the requests which exceeds the threshold value and it is obviously useful to consider such a subset as a cluster since it describes a meaningful structure in the dataset. In such example, as we see, it is useful to consider the frequency of the IP address feature on the higher level of the hierarchy, i.e., on the level of subnets.

Our examples focus on the IP address as a feature because that is the feature we used for clustering our datasets. Examples of other traffic features that have a hierarchical structure are URLs, protocol types, port numbers (here the hierarchical structure can be defined in different ways, more or less artificially chosen according to the needs of the particular clustering).

Without the need, however, to consider multi-dimensional feature spaces, and thinking only of the IP address feature, we can still realize that the task of HHH clustering is complex. Namely, at the fine-grained level, the IP address feature can be considered to have 32 levels of hierarchical structure, each level being defined by one bit in the 32-bit binary representation of the IP address. A smaller number of levels inside IP addresses can be defined, for example 4 levels, each corresponding to eight bits. The decision about defining the granularity of the hierarchy is up to the implementors of HHH, and they need to decide depending on the specific application of the algorithm. A more granular hierarchy will enable finding more precise heavy hitters in the dataset, but it has the disadvantage of making the clustering algorithm computationally intensive.

For the purpose of clustering the streaming data, different solutions are proposed in order to make the HHH algorithm less computationally demanding and faster. In those approaches a common idea is *approximating* the quality of the cluster output, with certain error guaranties.

Since the traffic records, in our scenario, are clustered offline, we could afford high computational requirements as well as the algorithm taking long time to compute the clusters. For that reason, in our implementation of HHH, the IP address feature is divided into full 32 levels of hierarchy.

4.1.2 DiffServ environment

DiffServ is a networking architecture for QoS provisioning proposed by the IETF (Internet Engineering Task Force). The architecture of DiffServ is described by the RFC document [4], while other relevant RFC documents describe PHBs (per hop behaviors), DSCP (differentiated services code point) in IPv4 and IPv6, framework for Integrated Services operation using DiffServ environment etc. Kilkki, in his book [26], gives an overview of the RFC documents related to DiffServ and the applications of DiffServ. DiffServ receives special attention as nowadays many applications have emerged that require assurances of provisioning and monitoring of QoS that cannot be provided by the existing network architecture. DiffSerf is a promising architecture to support the critical requirements of the new generation applications. DiffServ also enables providing different levels of service according to service level agreement (SLA) profiles between different domains such as ISPs, companies and users.

The idea behind the DiffServ is that the differentiation of services should be based on the essential service features, such as throughput, jitter, delay, packet loss, relative priority etc. Per-hop behaviors (PHBs) are DiffServ functional units that make such differentiation possible by classifying packets into different queues in the routers. The classification is based on the bits in the differentiated services (DSCP) field or uses some other features of the traffic for differentiation, which are supported by DiffServ. The architecture document specifies three types of DiffServ forwarding nodes (corresponding to the network routers in practice): interior nodes, ingress nodes and egress nodes. The functions of the nodes are separated according to one of the main principles of the current Internet — that is keeping the core of the network simple and scalable, and moving intelligence and computational burden to the edges of the network. This means that the edge nodes do the tasks of classification, metering and marking of packets, while the core nodes are usually only classifying and forwarding or delaying and possibly dropping the packets according to the DSCP field and the respective PHBs. The DiffServ architecture fulfills our requirements for a DDoS defense mechanism since it is simple and complies with the current Internet model.

A QoS provisioning environment such as DiffServ may be used in our DDoS defense mechanism thanks to the PHB as means of resource allocation. DiffServ provides coarse-grained QoS provisioning, and PHB provides the means of resource allocation to aggregate streams. Thus, we can use PHBs for the bandwidth allocation to the different classes of traffic (corresponding to our clusters).

Practical details about the deployment of DiffServ in our testbed experiment are given in section 4.2. The current section continues by describing different DiffServ queuing disciplines and explains why we decided to use the HTB discipline.

4.1.2.1 DiffServ queuing disciplines (qdiscs)

The DiffServ forwarding nodes need to implement *queues* in which the packets are buffered in order to get delayed and prioritized according to different PHBs before being transmitted. Thus, a very important part in the implementation of DiffServ are *queuing disciplines (qdiscs)*. The two main types of the queuing disciplines are:

(a) classless queueing disciplines,

(b) classful queueing disciplines.

Common examples of *classless queuing disciplines* are:

- PFIFO (packet limited first in first out)

  packets are queued and forwarded in the order that they come in; if the queue is full as defined by the limit, packets are dropped,

- WFQ (weighted fair queuing)

  serve packets according to precomputed time they would take to complete the services under given conditions,

- SFQ (stochastic fairness queuing)

  it is intended to provide fair queuing to each of the flows, achieving the aim by creating many FIFO queues per hashed value per each flow and distributing the packets in round robin fashion from these queues,

- TBF (token bucket filter)

  works on the principle of tokens in a bucket, allowing packets to be sent by not exceeding the assigned limit; however, when it is needed and if possible, packets may be sent with short burst excesses in the rate.

However, the real power of DiffServ comes with the *classful queuing disciplines*:

- PQ (priority queuing)

  enables filtering traffic according to the DSCP field or *other features of the traffic* as supported by the traffic conditioning (`tc`) filter. The different classes are given different priority (prioritized class will have a larger buffer allocated);

- CBQ (class based queuing)

  the most complex queuing discipline available; it utilizes exponential weighted moving average (EWMA) for measuring the idle time of the link, and according to assigned bandwidth reservations, tries to make the link idle necessarily long time in order to reduce the real bandwidth to the configured rate,

- HTB (hierarchical token bucket)

  HTB can be seen as a hierarchical CBQ. Additionally, HTB is also a classful TBF discipline.

It is worth noting that, inside each individual class in a classfull discipline, we can attach some of the classless queuing disciplines that were defined previously. HTB is the queuing discipline we use, because it is created to be a simpler, yet more powerful replacement for CBQ, and it fulfills other requirements for our DDoS defense mechanism, as described in more detail in the following section.

4.1.2.2 HTB queuing discipline

HTB defines a hierarchical structure of classes, and by default, each of the classes contains a PFIFO qdisc attached. When a packet is received, the HTB qdisc starts from the root class and examines whether it should enqueue the packet by consulting the filter that is attached to the class. If not, the classes attached to the root class are examined, and so on recursively. If the process comes to examining a leaf class and the filter accepts it, the packet is enqueued there. If no class filter is found to accept the packet, it is possible that a default class is defined, in which case such packet gets enqueued in the default class, or otherwise it stays enqueued in the last examined class. Such rules ensure that each packet gets enqueued into some of the classes. The DiffServ traffic polycing is depicted in figure 4.1.

HTB has with each of the classes associated the following set of parameters:

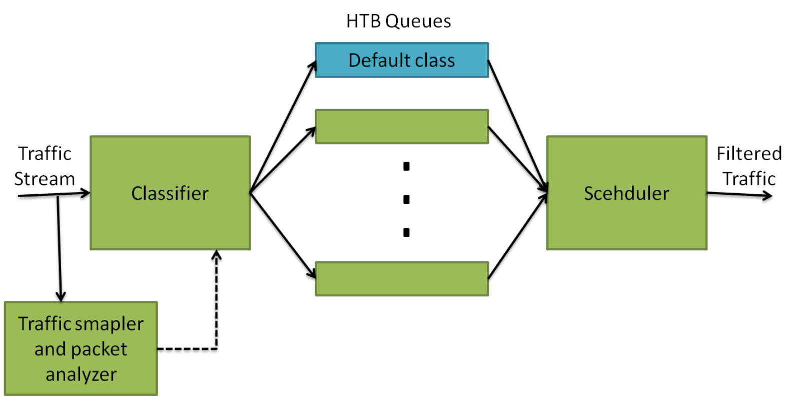


Figure 4.1: Schema representing DiffServ traffic filtering with HTB qdiscs

(i) AR — assured rate; the class should get at least this much bandwidth when needed,

(ii) CR — ceil rate; this is the maximum bandwidth the class can get despite the conditions,

(iii) P — priority; each class is assigned a priority number: lower numbers correspond to higher priority,

(iv) level — level in the hierarchical queuing discipline structure,

(v) Q — quantum; used to decide the rate the classes get in the case of borrowing, as explained bellow.

According to the requirements of our bandwidth filter, we decide to use the HTB mode with link sharing, i.e, *borrowing* in the DiffServ terminology, between the classes. In such a case, the formula for the bandwidth of a class c, i.e., its actual rate R_c, becomes:

$$R_c = \min(CR_c, AR_c + B_c),$$

where B_c is the borrowed bandwidth that comes from the ancestors classes and CR_c and AR_c are the parameters of the HTB class c as defined above. Direct ancestor (parent) of c is denoted by p, and the set of other descendants of p that want to borrow from the parent is denoted by $Desc(p)$. Then B_c is calculated according to the following formula:

$$B_c = \begin{cases} \frac{Q_c R_p}{\sum\limits_{a_i \in Desc(p)} Q_i} & \text{if} \quad \min\limits_{a_i \in Desc(p)} P_i \geq P_c, \\ 0 & \text{otherwise.} \end{cases}$$

The formula above implies that the bandwidth will be borrowed only if all the other descendant classes $a_i \in Desc(p)$ have higher priority number P_i compared to priority P_c of class c. Thus the class c needs to have the highest priority among the descendants. Otherwise the other, higher-priority classes will be served first. The formula also shows that the parent bandwidth is divided to the descendants with the same priority, in proportion with the values Q.

The described HTB behavior in combination with the clustering output offers a range of possibilities for deploying the DDoS filtering policy. Depending on the type of the DDoS defense we want to deploy, we shall assign different rate, ceil and priority values, to the different classes of traffic corresponding to the clusters. In such a way, DiffServ will do the filtering of DDoS traffic according to our cluster set.

4.2 Testbed setup

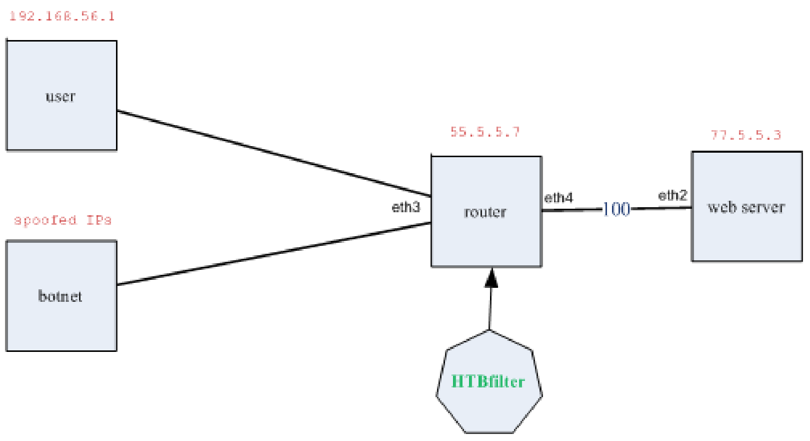


Figure 4.2: Virtual scenario for experiments corresponding to 3.2.

According to the scenario presented in section 3.2, we employ three virtual machines using VirtualBox Version 3.2.8_OSE r64453 on the Ubuntu 10.10 Maverick Meerkat host virtual machine. The host machine is used as the **user** (and as the **attacker** later) in the scenario. For the Linux **router** we use Kubuntu 10.10, the Lucid Lynx virtual image with the kernel version 2.6.35-22-generic. The **server** image is another Ubuntu 10.10 Maverick Meerkat machine. The described virtual scenario is presented in figure 4.2.

VirtualBox offers a few different virtual networking modes between the virtual machines and the host. We connect the router to the host using the *host-only* networking mode, while between the router and the server the connection is based on the *internal networking* mode. The host-only mode enables the router to be connected directly to the host, as well as to the Internet through a virtual interface created on the host. The internal networking mode corresponds to the idea of the web server being connected to the Internet only through the router. The machines on the internal network, that is the router and the server in our case, can communicate privately, hiding their traffic from the host system (the users and the attacker) and the outside world.

The Kubuntu image kernel version 2.6 enabled us to use DiffServ supported by Linux without the need for kernel recompilation, which was necessary for the versions of kernel prior to 2.4.20. In our case all the necessary modules were present and enabled, and also the HTB queuing discipline is present in kernel versions after 2.4. Thus we could just start using DiffServ after installing the `tc` command line utility, which is essential for manipulating the DiffServ qdiscs. The `tc` command is used to create the classes, to attach qdiscs to them and for defining the filters.

The HTB based DiffServ scripts that we use are created at the same time when the initial normal traffic records are clustered. The HTB classes are created to correspond to the different clusters and the parameters of the HTB class are set according to the amount of traffic in that traffic class.

Since we simulate the real traffic based on the Apache logs, a script is made to send normal traffic requests from the user machine to the server machine, with source IP addresses corresponding to the Apache log files. After testing the available bandwidth rate between the user and the server in our testbed network, we set the rate in the script to be no more than 40% of the available bandwidth. As figure 4.2 shows, the link between the server and the router is limited to 100 kbps (kilo bytes per second), while we send the normal traffic with the traffic rate not exceeding 40 kbps. For sending the traffic, the `hping3` command is used, particularly invoking it from the *TCL* scripts. The `hping3` command is invoked with the parameters `--faster` and `--flood` when sending the attack traffic. The parameter `--faster` instructs the machine to send 1000 packets per second, while the parameter `--flood` is used to send the packets at the highest possible rate by the machine.

4.3 Experiments

In the scenario described in section 3.2, we discussed the the two cases with focus on the defender strategy:

- In *scenario 2*, the defender is the less agile party and uses the filtering policy as the optimal proactive defense against an adaptive attacker.

- In *scenario 3*, the defender is more agile and adapts his filtering policy for improving the DDoS reaction to the particular attack.

In this chapter, we conduct sets of experiments to correspond to the two scenarios above. First, we apply the *HHH algorithm* to the chosen normal traffic datasets. Second, we create the *HTB filters* corresponding to the cluster sets output by HHH with different threshold values. Finally, we simulate attack scenarios in the testbed and record normal traffic rate at the server. In order to cause a congestion to happen only at the link between the router and the server during the attack, we make an outbound filter at the host machine. With the help of this filter we make sure that all the traffic sent from the host machine, the one corresponding to the user and the one corresponding to the attacker, are transmitted to the router. In this way, we model any number of independent connections that might be coming to the router. In the first set of experiments, we send the attack traffic with *randomly spoofed source IP* addresses. In the second set of experiments, we chose the attack traffic to originate from only a *certain number of different IP addresses.*

4.3.1 Applying the HHH algorithm

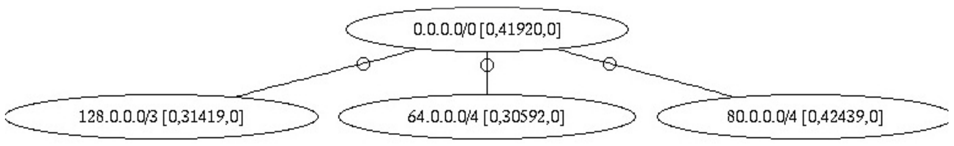


Figure 4.3: Clusters created by HHH using 5% threshold. Apache log December 2009.

To simulate the two scenarios in our testbed, we use the *real traffic datasets* from a web server at Aalto university network. The data are in the form of the whole month Apache logs with HTTP requests to the web server. We chose the log from September 2007 and another log from December 2009, to serve as the history, i.e., as the normal traffic records. The HHH algorithm is applied on the HTTP requests from the logs, based on one hierarchical feature — the *source IP address* in the requests. Afterwards, the traffic logs from the months that follow the months chosen for the history, October 2007 and January 2010, respectively, are used to generate the normal traffic stream. The attacks are conducted against those traffic streams, and we test the filter based on clusters created from the preceding months in DDoS defense. In this way, we simulate the normal web server functioning in our experiments (however, conducting the tests in rather shorter time compared to real web server functioning).

The HHH algorithm is applied to the chosen datasets using different threshold values, ranging from 20% to 0.5%. Since the HHH algorithm outputs the clusters with similar size, this implies that the number of clusters output from the traffic was 4-5 with the 20% threshold, and reached 50-100 clusters with the smallest threshold. An example cluster output from clustering the log from December 2009 using the threshold value 20% is shown in figure 4.3. We see that there are 4 clusters output, including the default cluster. For each of the clusters, the CIDR (classless inter-domain routing) number and the number of hits are shown.

4.3.2 Creating HTB filters

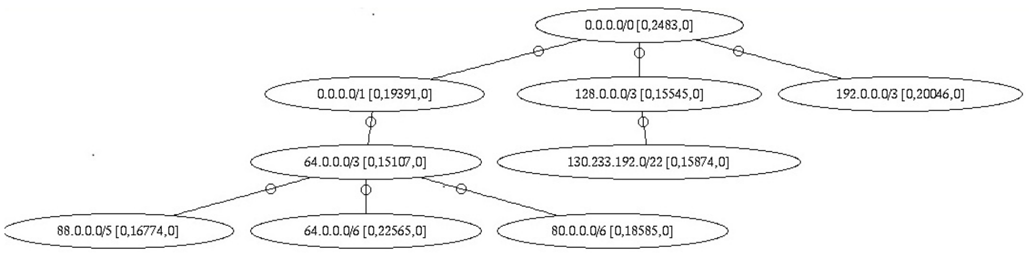


Figure 4.4: Clusters created by HHH using 10% threshold. Apache log December 2009.

Listing 4.1: Example code for creating HTB classes

```
# HTB filter for the clustering
# defintion of the root qdisc
tc qdisc add dev eth4 handle 1: root htb default 10

# definition of one root class under the root qdisc
tc class add dev eth4 parent 1: classid 1:1
htb rate 100kbps ceil 100kbps
```

```
# definition of the subclasses under the root class
# default class defintion
tc class add dev eth4 parent 1:1 classid 1:10
htb rate 1kbps ceil 100kbps
# another class
tc class add dev eth4 parent 1:1 classid 1:12
htb rate 13kbps ceil 100kbps

# setting the filter for the class
tc filter add dev eth4 parent 1: prio 1 protocol ip
u32 match ip src 0.0.0.0/1 flowid 1:12
```

The experiments are conducted using different threshold values for the HHH algorithm. For each different threshold value, we use the data from the HHH output: the number of clusters and the corresponding traffic amounts, to create the HTB filters. The number of clusters determines the number of classes in the HTB filter. The traffic amount is represented by the value `rate` in the filter command, and in value `ceil` we set the total bandwidth available for that class since borrowing between the classes is allowed. For the default cluster, there is a `default class` in HTB exactly suiting for the purpose. All the traffic that is not classified to other classes will be enqueued to the default class.

As an example, we show, in listing 4.1, the part of the HTB filter code corresponding to the clustering with 10% threshold that is shown in figure 4.4. In the example, the `root qdisc 1:` is defined and the `root class 1:1` is attached to this qdisc. Under the root class, we add two subclasses, `class 1:10`, which is defined to be the default class, and `class 1:12`, corresponding to the cluster with CIDR 0.0.0.0/1 shown in figure 4.4.

Since we conducted a number of experiments that follow using the 10% threshold clustering, we give the code for creating the whole HTB filter according to this clustering in appendix A.

4.3.3 DiffServ in proactive DDoS defense: scenario 2

We presented the mathematical analysis for scenario 2 in section 3.5.2. The results of the analysis show that the best proactive defense is to use the reservations for traffic, ρ_i, *proportional* to the amounts in the traffic classes that correspond to the clusters in the normal traffic, α_i. Thus we found that we should assign $\rho_i = \frac{\alpha_i}{A}$ for $i = 1, ..., k$.

Attacks without a filter at the router

In the first run, the normal traffic transmission is started with the rate 40kbps, and after 60s, the *randomly spoofed attack* traffic is generated with the rate 500kbps. We keep on sending until 1000000 attack traffic packets are sent, which is approximately 120s from the start of the normal transmission. The results for the NPSR and the number of normal packets served during this case can be found in table 4.1.

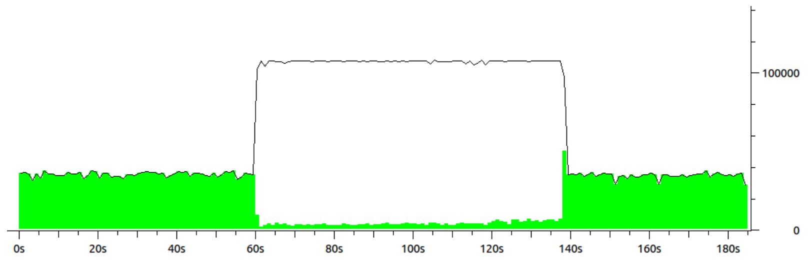
In the second run, the attack traffic *originating from ten distinct source IP addresses* is sent in a similar way. Since the link between the router and the server is limited to 100kbps, as shown in figure 4.2, the attack traffic rate recored at the server has the flat ceiling at 100kbps. This case is shown in figure 4.5(a). The damage to the normal traffic rate, represented by the green color, is obvious during the attack period. Since there is no filter deployed at the router, the damage to the normal traffic rate is similar to the previous case.

Attacks with different filters deployed on the router

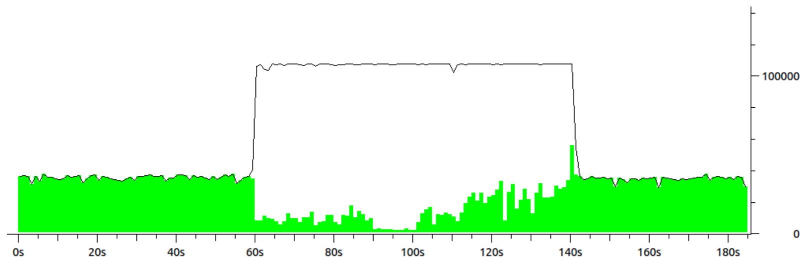
In the following round of experiments we instruct the DiffServ on the router to filter the traffic according to the normal traffic profile. Particularly, the filters with reservations proportional to the normal traffic classes are deployed. It is suggested by our mathematical analysis to be the optimal proactive DDoS defense. That mathematical result is evaluated in this section. HTB filters are created to correspond to different clustering outputs depending on the threshold. For each different threshold, we repeat the experiment after deploying the corresponding filter. An improvement is noticeable after deploying filters with the smallest number of classes, corresponding to the 20% threshold clustering.

We run the *randomly spoofed attack traffic* first. The results for all the different clusterings that we tested are summarized in table 4.1. The increase in NPSR during the attack period is significant. It reaches 89.9% with 1% threshold clustering in the experiment with Apache log from October 2007. We give graphs showing the normal traffic rate recorded at the server for this case in appendix B.

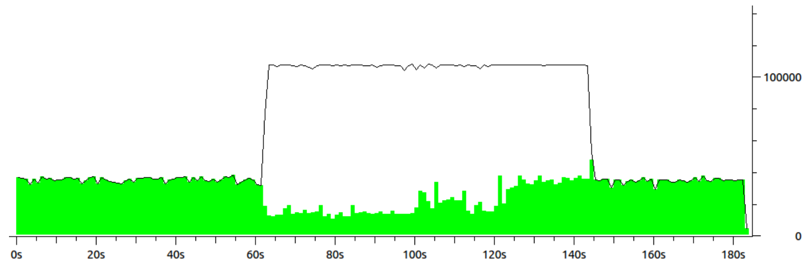
We run the experiment for the second time with the attack traffic *originating from ten distinct source IP addresses.* The traffic rate recorded at the server when the HTB filter created according to the 10% clustering is deployed is given in figure 4.5(b). The increase in the normal traffic rate is visible. However, high peaks and sudden drops in the rate are appearing during the attack, and we explain that by the relatively small number of classes



(a) *First run*: no filter is deployed at the router



(b) *Second run*: Optimal HTB filter for proactive defense corresponding to 10% clustering is deployed at the router



(c) *Third run*: Optimal HTB filter for proactive defense corresponding to 1% clustering is deployed at the router

Figure 4.5: Scenario 2: The traffic output at the server when the normal traffic is generated from the Apache log from January 2010 and the attack traffic originates from ten different source IP addresses. The black line represents the attack traffic rate. Normal traffic rate is represented by the green bars area.

threshold in %	Apache log January 2009		Apache log October 2007	
	# of packets	**NPSR**	**# of packets**	**NPSR**
no filter	4689	12.8	5895	15.1
20	6141	16.8	15740	40.4
10	11978	32.78	17711	45.4
5	17189	47.0	21204	54.4
1	23244	63.6	35035	89.9

Table 4.1: Increase in normal packet survival ratio using different clusterings. Randomly spoofed attack traffic is sent.

most of them being on the low level in HHH hierarchical output. This we could see in figure 4.4 above. Particularly, lower IP mask number means that the class includes a larger range of different IP addresses. That practically means that the attack traffic has a higher probability to match the filter of some of those classes. Thus, it is expected that a filter with a lower threshold, which implies the higher number of clusters, will provide a better protection for the normal traffic. We show the result when the experiment is repeated with the HTB filter corresponding to 1% threshold clustering in figure 4.5(c) which confirms our expectation.

4.3.4 DiffServ in reactive DDoS defense: scenario 3

In this section we describe the experiments related to the optimal strategy of the defender in the case when the attacker strategy is fixed. Mathematical analysis of this scenario and the results presented in section 3.5.3 are experimentally evaluated in this section.

For the purpose of this experiment, we divide the Apache logs from January 2010 and October 2007, used to generate the normal traffic stream, both into three approximately equal subsets. During the division, we follow the timeline of the requests in the log so that the traffic from the first subset would correspond to the earlier period of the month compared to the traffic in the second subset.

We use the first subset from an Apache log in the *first phase* of our experiment to generate the normal traffic stream. In this experiment, we are sending the attack traffic from the start of the normal transmission. Assuming that the router detects the attack and decides to deploy a filter as a reaction, in the *second phase*, when we send the traffic stream from the second subset, the HTB filter is being deployed on the server. Attack conditions remain the same, but we expect improvement due to the filter, as the previously conducted experiments in section 4.3.3 confirmed. More importantly for this experiment, the filter deployed during the second phase is used for detecting the changes in the traffic amounts in the different classes created by the attack. By applying the algorithm 2 described in section 3.5.3, the HTB filter on the router is adapted and the **new HTB filter** is created for the third phase. Thus the second phase can be considered as a learning phase for the defender before optimally adapting to the attacker strategy. In the *third phase* the new HTB filter is deployed on the router and the data from the third subset are used for generating the normal stream. We expect additional improvement since the defender strategy is now adapted to the particular attacker strategy.

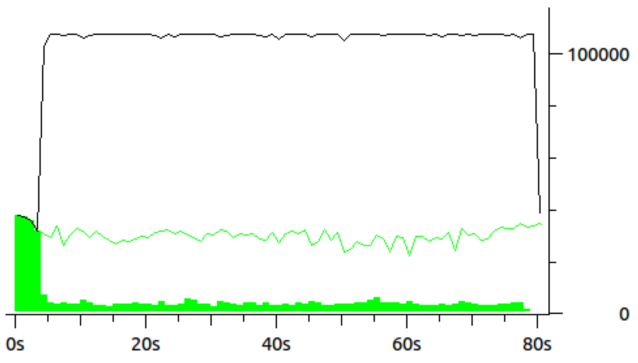
We run the experiment with *randomly spoofed* attack traffic first. The traffic rate recorded during each of the three phases described above is given in a separate figure. In the two last figures, when the HTB filters are used, they are created according to the 10% threshold clustering.

The HTB filter intended for the best proactive defense (first phase) is adapted for the best reactive defense against the particular attack profile (second phase). Adapting the filter is done according to algorithm 2 presented in section 3.5.2. In that case, some of the classes are left without the capacity. In the particular case, the 10-class initial HTB filter is transformed to a new HTB filter having only 5 classes. This gives an example when a filter with fewer classes is better in DDoS protection compared to the filter having twice as many classes (still, only in the particular attack case). In appendix A we provide the code of the adapted HTB filter.

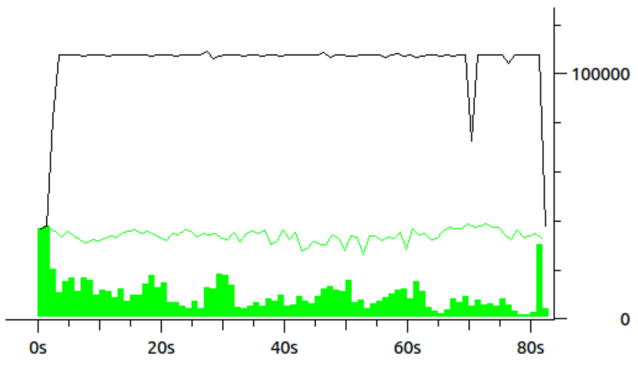
We similarly run the three experiment phases while sending the attack traffic from *ten fixed IP address sources.* The results from this set of experiments are summarized in table 4.2. Also, we give graphs with normal traffic rate recorded during such an experiment corresponding to Apache log from October 2007 in appendix B.

threshold = 10 %	Apache log January 2009		Apache log October 2007	
	# of packets	**NPSR**	**# of packets**	**NPSR**
1st phase	17029	34.32	12576	25.57
2nd phase	35510	71.55	43412	88.36
3rd phase	36114	72.77	47698	97.08

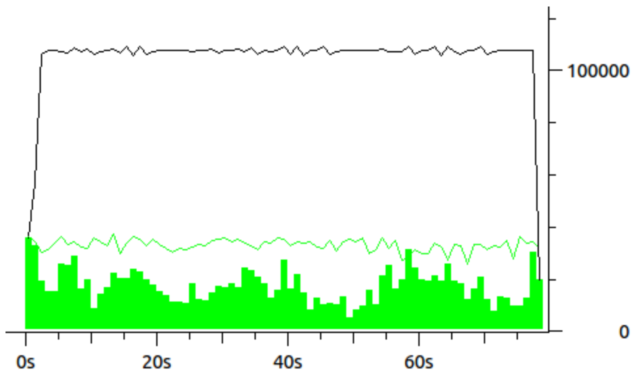
Table 4.2: Increase in normal packet survival ratio during the three experiment phases. The HBT filters are created from the 10% HHH clustering. Attack traffic from ten fixed IP address sources is sent.



(a) *First phase*: no filter is deployed at the router.



(b) *Second phase*: Optimal HTB filter for proactive defense is deployed at the router.



(c) *Third phase*: HTB filter is adapted to this particular attack distribution according to algorithm 2.

Figure 4.6: The traffic output at the server when normal traffic is generated from the three subsets of Apache log from January 2010 and attack traffic is randomly spoofed. The black line represents the attack traffic rate. Normal traffic rate served during the attack is represented by the green bars area. The green line is added for the relative estimate and shows the traffic rate recored during normal traffic conditions.

Chapter 5

Discussion

Our research goals are stated in section 1.2. In section 5.1 we discuss our results on those goals. In particular, theoretical results about the effectiveness of the cluster-based filtering in DDoS defense are discussed in chapter 3 and in chapter 4 the experimental evaluation of those theoretical results is presented. In this chapter, we provide a discussion on the theoretical results in section 5.2 and a discussion on the experimental results in section 5.3.

5.1 Research results

In this section we discuss the results on our previously stated research goals.

(i) *For which types of DDoS defense the cluster-based filtering method can be employed?*

In chapter 2, we present a number of solutions in academic literature in which the cluster-based filtering method is employed for the different roles in DDoS defense. The role depends on the location of the cluster-based filter in the network and we identified solutions that use the traffic clustering method for DDoS detection, prevention or reaction. Our study focuses on the scenario when the cluster-based filter is deployed at the target network. In such a scenario, the cluster-based filtering takes the roles in proactive and reactive target bandwidth-management defense.

(ii) *What is the filtering policy when cluster-based method is used for proactive DDoS defense with a fixed strategy?*

We resolve this research goal in section 3.5.2. Proactive defense is a technique of a less agile defender expecting a more adaptive attacker.

In such a case, the defender should deploy a filter with reservations to the traffic classes corresponding and proportional in sizes to the classes found by clustering the normal traffic.

(iii) *What is the filtering policy when cluster-based method is used for a reactive defense with a fast-adapting strategy?*

This research question is resolved in section 3.5.3. The defender using reactive defense is more agile and he adapts his existing filtering policy to serve optimally against the particular attack. For the filter adaptation the defender should apply algorithm 2.

(iv) *How effective is cluster-based filtering in DDoS proactive or reactive defense?*

The two research results above are found in the game-theoretic model. We experimentally evaluate the effectiveness of the theoretical results as described in chapter 4. The experimental results show the increase in NPSR from 72% to 97% when proactive defense is used.

(v) *What is the most effective attack strategy when the defender is using a fixed cluster-based filtering policy?*

Optimal attacker strategy when the defender is using a fixed cluster-based filtering policy is analyzed in section 3.5.1. For the optimal attack, the attacker should create his strategy according to algorithm 1.

(vi) *Can existing quality-of-service (QoS) capabilities in standard operating systems and routers be used to implement the cluster-based filtering for DDoS defense?*

The experiments presented in chapter 4 are conducted using DiffServ embedded in a standard Linux kernel. The cluster-based filter is implemented using DiffServ HTB queuing discipline. Thus, the filter implementation is very efficient and suitable for live data streams filtering. A significant increase in the number of served non-attack requests resolves this research question positively.

5.2 Discussion on the theoretical results

Talking in the terms of *game theory*, we analyzed the cases when one of the parties in the DDoS scenario ‘makes the first move’. This means that either the defender or the attacker is clearly *more agile than the other* and can adapt his strategy to the slowly-changing strategy of the other party. This corresponds to standard game-theoretic models. Quite different modeling techniques would be needed if both parties were equally adaptive or if they competed on the speed of the adaptation.

The *quality* of the DDoS defense in our analysis is measured by the amount of the normal traffic served during the attack, or *NPSR*. That is one limitation of our study, since we have not considered *traffic flows* neither the different requirements that some types of flows might demand. Namely, as described by Mirkovic et al. [41] the DDoS defense quality is not only measured by the NPSR or similar quantitative measures of traffic, but different requirements for different protocols and applications should be considered. Such protocol and application-specific requirements are obviously not covered by abstract mathematical model like the one in this thesis.

In section 3.5.1, we found how the attacker, having the amount of traffic B at his disposal, should distribute the DDoS traffic into the classes in the cluster-based filter so that his attack is most successful. If the given value B is not high enough, the attacker should attack some clusters, while if he can increase B to the sufficient level, then he achieves the goal of attacking and disrupting all the normal traffic classes.

In the analysis described above, the attacker is assumed to have *full information about the defender*. In reality such an assumption is not a simple goal for the attacker. The attacker might reason about the normal traffic profile, for example, about the regular visitors accessing a web site. But it is questionable how accurately he can create a picture of such a profile. Additionally, the attacker also needs to find out about the particular filtering policy the defender is using. One possible mean for the attacker is to use probing of different traffic classes in order to detect reservations assigned to them. If the defender is using a clustering algorithm such that it is difficult for the attacker to guess the output classes, then the attacker's goal becomes additionally complex.

In the end, even while knowing the normal profile, it might not be easy for the attacker to 'choose' which type of attack traffic he can send. The DDoS attacker might be restricted in the scope of source IP addresses or other features of traffic for conducting the attack. Such *attacker capabilities* depend on the available attack tools and the location in Internet and size of the botnets he controls. Also, the effectiveness of the clustering defense against such an attack depends on the traffic clustering algorithm and on the traffic features that the algorithm uses. If the algorithm finds cluster features that are specific to the normal traffic and difficult to imitate by the attacker, then the attacker cannot conduct the optimal attack and the impact of the attack will be less serious. This highlights the *importance of the quality of the clustering algorithm.*

Since the capabilities of DDoS attacker to adapt the attack traffic are not well known, we can keep on reasoning from the defender side about it in two directions. If we assume that it is *equally* possible for the attacker to adapt his traffic to different normal traffic classes, then the best cluster reservations for the defender should have approximately equal size. In this way, no large traffic class can be easily damaged by the attacker — he needs to adapt his traffic to many different clusters. The HHH algorithm is a good clustering choice if this is the case since it outputs approximately equal traffic amounts in each cluster class.

On the other hand, if the attacker is better able to adapt his traffic to the *certain classes*, then the analysis becomes more complex. In that case, the defender could use the knowledge about the attacker to adapt the cluster reservations accordingly, or even his choice of the clustering algorithm could be influenced by that knowledge. However, the analysis in this case would require data about real DDoS attacks or a set of experiments conducted in order to find about the real capabilities of DDoS attackers.

The results of the optimization task solved in section 3.5.2 show that the best proactive defense strategy against the more agile attacker is to initially reserve traffic amounts proportional to the normal traffic cluster sizes. This strategy does not depend upon the attacker strategy. However, it is easy to observe that using such a filtering policy, the proportion of the honest traffic served under the best attack in each of the classes is equal to $\frac{\alpha}{\alpha+\beta}$. This means that the same total amount of the normal traffic $\frac{A}{A+B}$ is served as when there was no traffic filtering policy deployed. The conclusion is that under *perfectly adapted* attack traffic, a fixed cluster-based filter does not have any effect in DDoS reaction. However, we point out again that it is not clear how well the attacker is able to adapt his traffic, especially in the case of a good clustering algorithm based on a feature space with many dimensions.

In section 3.5.3, algorithm 2 gives the best reactive strategy for the defender, who uses the clustering-based filter when under the attack. However, the algorithm does not give any guarantees on the amount of honest traffic served. It just finds the adapted defense strategy which is optimal under the current attack. The same limitation of the clustering technique applies in this case as previously: if the attack traffic *perfectly imitates* the normal traffic, i.e., if the values β_i are proportional to the values α_i, then the clustering technique does not give any benefit compared to using no clusters. However, in the other attack cases, as the experiments show, the cluster-based filters can significantly increase the NPSR.

5.3 Discussion on the experimental results

The traffic clustering can be done offline, by clustering the offline traffic records as in our experiments, or it can be done online. In the second case, the normal traffic profile is created by clustering a live data stream and such a clustering is more demanding in computational resources and time. At hight speed links, it is often necessary to sample the traffic stream in order to apply the online clustering. However, the filtering method that we implemented works with both of the described clustering types, and itself is a lightweight process that can filter live data streams.

In this thesis, we have picked one algorithm, the hierarchical heavy hitters (HHH), from the literature and do not compare it with other clusterings. We conducted the tests with the HHH clustering based only on *one traffic feature*, the client IP address. If the clustering was based on more features, for instance port numbers, it can be expected that the results in the DDoS defense would be be improved.

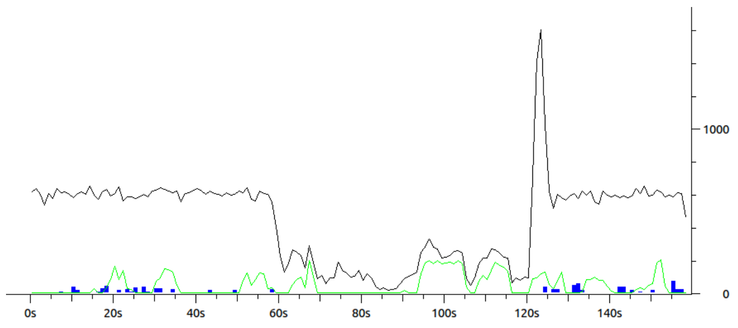


Figure 5.1: Classes inside the normal traffic that is served under randomly spoofed attack. The filter created by HHH using 10% threshold. The black line shows the total normal traffic rate. The default class is represented by the blue bars. Class with CIDR 130.233.192.0/22 is represented by the green line. Apache log January 2009.

Our results of preserving the normal traffic packets during DDoS attack using DiffServ show promising results. Using the largest number of clusters, with 1% threshold clustering, we obtain the highest NPSR. With such clustering, the NPSR corresponding to the Apache log files from January 2010 and from October 2009, respectively, equals 63% and 89% in the first scenario and, 72% and 97% in the second scenario. Without the filter the NPSR rates were between 12% and 35%. We analyze how the significant increase in NPSR is achieved. According to the filtering policy, the dropped traffic mostly belongs to the default class, since that is the class in which the most of the attack packets are queued. That means, the normal traffic belonging to the default class is the most damaged, as shown in figure 5.1. However, the normal traffic belonging to the other classes is well preserved, as can be seen from the same figure for the class with CIDR 130.233.192.0/22.

The normal requests that are classified in the default cluster correspond mostly to the 'random' users, i.e., those who visit the web server for the first time or rarely. Thus, it can be considered a good policy that those users get lower priority, compared the 'common' users of the web server.

Our virtual testbed was deployed on one machine using *virtual machines*. Thus we did not test the DDoS scenario in a real network environment and did not obtain accurate quantitative performance results. However, the *effectiveness of the filtering* was tested.

Because of the limitations of the DiffServ traffic classification in our testbed conditions, it was not possible to test the filters based on cluster set with more than 100 clusters. In reality, however, with high-bandwidth links it would make sense to have a larger number of clusters. Our assumption is that the tests with filters based on clusterings with smaller threshold than 0.1% could give valuable insight about the best number of clusters, as we will discuss in section 6.1.

The normal traffic was generated with source IP addresses based on the real traffic logs, but at a fixed rate which is many times higher than in real scenarios. Two types of synthetic attack traffic were generated: randomly spoofed attack traffic and the attack traffic having a certain number of fixed sources. It might be relevant to test different traffic distributions in future experiments.

Although in our testbed environment many real parameters are omitted or could not be tested, the results suggest possibilities of using the existing DiffServ implementation embedded in standalone routers and also in the Linux kernel for cluster-based filtering in DDoS prevention. Thus, our experiments can be taken as a *starting point* for implementing the cluster-based filtering for real network services.

Chapter 6

Conclusions

Cluster-based traffic filtering as a DDoS defense technique has received attention in the academia since the first such approaches were described a decade ago. However, after surveying the literature and readings on different solutions that involve clustering of network traffic in DDoS defense, we did not find a general analysis on the method. Developing such an analysis is the main contribution of this thesis.

We focus on solutions that use unsupervised learning to cluster normal traffic and create filters for DDoS prevention or reaction based on the clusters. Particularly, we chose a scenario in which the cluster-based filter is deployed at the target or at a router near the target. In such a scenario, the cluster-based filter can serve as a proactive or reactive DDoS defense.

The central part of our study involves creating a game-theoretic model for cluster-based filtering in the described scenario. The generality of the model makes our analysis applicable in different DDoS defense scenarios. Optimal filtering policies are modeled for the defense in the cases when the attacker is adaptive or when he uses a fixed strategy. Also, the best attack strategy is evaluated when the defender is using a fixed cluster-based filter. Our theoretical results give insight into the applicability of filters based on traffic clusters to DDoS defense. The only case when the method does not provide any benefit is when the attack traffic perfectly imitates the normal traffic distribution. However, such attacker capability is unlikely in reality, for example, for the IP address distribution.

In the second part of this thesis, we developed a testbed in order to experimentally evaluate the cluster-based filtering. We applied the HHH algorithm to model normal traffic as set of clusters and implemented the cluster-based filter using a standard quality-of-service architecture, DiffServ. Particularly, we conducted experiments to evaluate our theoretical model. The experiments confirm the theoretical results. Additionally, the experimental results suggest high effectiveness of cluster-based filtering, with NPSR reaching above 97% with the most fine-grained clusterings.

The results suggest that the cluster-based filtering can be applied for practical DDoS defense. In particular, a rather simple approach such as Diffserv would be useful in DDoS defense for a small web server because it would not require a large investment in hardware or software. Also, in cooperation with ISPs, this method could protect against bandwidth-band DDoS attacks on the ISP links for the specific domain.

6.1 Future work

As a first step in the future work, we plan to conduct the similar experiments in a real network with DiffServ filter implementation at a real router. Such experiments would provide accurate quantitative performance results. As a second step, generating and testing additional attack traffic profiles would provide more confidence about the quality of the DDoS defense results.

We used one traffic feature to cluster the traffic with the HHH algorithm. Applying the HHH algorithm to a multi-dimensional feature space is one direction for future experiments. Namely, the more features the clustering is based on, the more difficult it becomes for the attacker to adapt his traffic to the normal traffic classes.

One of the improvements for the future implementation of the DiffServ cluster-based filters would be to test attaching other queuing disciplines to the HTB classes. In the HTB filter implementation, we used the default PFIFO queuing discipline. For instance, SFQ scheduling discipline is a promising method for improving traffic throughput per flow, which could lead to qualitatively better DDoS defense, following the suggestion by Mirkovic et al. [41].

Another possible step in our future work is to develop a model for competing strategies of the attacker and defender. Such model would cover the cases when both of the parties are adapting the strategies simultaneously.

As explained in chapter 4, we did not test HBT filters based on clustering with threshold smaller than 1%. In the future we plan to test filters based on clusterings with smaller threshold since they could give a valuable insight into the optimal number of clusters. This idea is based on the possibility of *over-learning* by the clustering algorithm. Namely, in our results, the filters based on a smaller threshold clusterings are better. Our assumption is that bellow a certain threshold value, the clusterings would not give better results anymore. Such a threshold value would correspond to the optimal number of clusters for the particular traffic dataset.

Bibliography

[1] Agrawal, R., and Srikant, R. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases* (San Francisco, CA, USA, 1994), VLDB '94, Morgan Kaufmann Publishers Inc., pp. 487–499.

[2] Anderson, T., Roscoe, T., and Wetherall, D. Preventing internet denial-of-service with capabilities. *SIGCOMM Comput. Commun. Rev. 34* (January 2004), 39–44.

[3] Bezdek, J. C. *Fuzzy Mathematics in Pattern Classification.* PhD thesis, Applied Math. Center, Cornell University, Ithaca, 1973.

[4] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and Weiss, W. RFC 2475: An architecture for differentiated services, Dec. 1998. Status: PROPOSED STANDARD.

[5] Bretthauer, K. M., and Shetty, B. The nonlinear resource allocation problem. *Operations Research 43*, 4 (1995), 670–683.

[6] Bretthauer, K. M., and Shetty, B. The nonlinear knapsack problem - algorithms and applications. *European Journal of Operational Research 138*, 3 (May 2002), 459–472.

[7] Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., and Freeman, D. Readings in knowledge acquisition and learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993, ch. AutoClass: a Bayesian classification system, pp. 431–441.

[8] Cheeseman, P., and Stutz, J. Advances in knowledge discovery and data mining. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996, ch. Bayesian classification (AutoClass): theory and results, pp. 153–180.

[9] Cooke, E., Jahanian, F., and Mcpherson, D. The zombie roundup: Understanding, detecting, and disrupting botnets. pp. 39–44.

[10] Cormode, G., Korn, F., Muthukrishnan, S., and Srivastava, D. Finding hierarchical heavy hitters in streaming data. *ACM Trans. Knowl. Discov. Data 1* (February 2008), 2:1–2:48.

[11] Cormode, G., and Muthukrishnan, S. Diamond in the rough: Finding hierarchical heavy hitters in multi-dimensional data. In *In Proceedings of the 23rd ACM SIGMOD International Conference on Management of Data* (2004), ACM Press, pp. 155–166.

[12] Dantzig, G. *Linear programming and extensions*. Rand Corporation Research Study. Princeton Univ. Press, Princeton, NJ, 1963.

[13] Dobbins, R., and Morales, C. Worldwide infrastructure security report. Tech. rep., Arbor Networks, Corporate Headquarters, Chelmsford, Massachusetts, USA, 2010.

[14] Erman, J., Arlitt, M., and Mahanti, A. Traffic classification using clustering algorithms. In *Proceedings of the 2006 SIGCOMM workshop on Mining network data* (New York, NY, USA, 2006), MineNet '06, ACM, pp. 281–286.

[15] Eskin, E., Arnold, A., Prerau, M., Portnoy, L., and Stolfo, S. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *Applications of Data Mining in Computer Security* (2002), Kluwer.

[16] Ester, M., peter Kriegel, H., S, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. AAAI Press, pp. 226–231.

[17] Ferguson, P., and Senie, D. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing, 2000.

[18] Garber, L. Denial-of-service attacks rip the internet. *Computer 33* (April 2000), 12–17.

[19] Gu, Y., McCallum, A., and Towsley, D. Detecting anomalies in network traffic using maximum entropy estimation. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement* (Berkeley, CA, USA, 2005), IMC '05, USENIX Association, pp. 32–32.

[20] Gupta, J., and Misra. Distributed Denial of Service prevention techniques. *International Journal of Computer and Electrical Engineering 2* (April 2010), 268–276.

[21] Hastie, T., and Tibshirani, R. Discriminant adaptive nearest neighbor classification. *IEEE Trans. Pattern Anal. Mach. Intell. 18* (June 1996), 607–616.

[22] Hijazi, A., Inoue, H., Matrawy, A., van Oorschot, P., and Somayaji, A. Discovering packet structure through lightweight hierarchical clustering. In *Communications, 2008. ICC '08. IEEE International Conference on* (may 2008), pp. 33 –39.

[23] Hijazi, A., Inoue, H., Matrawy, A., van Oorschot, P., and Somayaji, A. Lightweight hierarchical clustering of network packets using (p, n)-grams. Tech. rep., Technical Report TR-09-03, School of Computer Science, Carleton University, 2009.

[24] Holland, J. H. *Adaptation in natural and artificial systems.* MIT Press, Cambridge, MA, USA, 1992.

[25] Keromytis, A. D., Misra, V., and Rubenstein, D. SOS: secure overlay services. In *Proceedings of the 2002 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '02)* (New York, NY, USA, 2002), ACM Press, pp. 61–72.

[26] Kilkki, K. *Differentiated Services for the Internet.* Macmillan Publishing Co., Inc., Indianapolis, IN, USA, 1999.

[27] Kullback, S. *Information Theory and Statistics.* Wiley, New York, 1959.

[28] Kullback, S., and Leibler, R. A. On information and sufficiency. *Annals of Mathematical Statistics 22* (1951), 49–86.

[29] Lakhina, A., Crovella, M., and Diot, C. Mining anomalies using traffic feature distributions. In *In ACM SIGCOMM* (2005), pp. 217–228.

[30] Lee, K., Kim, J., Kwon, K. H., Han, Y., and Kim, S. DDoS attack detection method using cluster analysis. *Expert Syst. Appl. 34* (April 2008), 1659–1665.

[31] Li, J., Mirkovic, J., Wang, M., Reiher, P., and Zhang, L. Save: source address validity enforcement protocol. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (June 2002), vol. 3, pp. 1557 – 1566.

[32] Li, M., Li, M., and Jiang, X. DDoS attacks detection model and its application. *W. Trans. on Comp. 7* (August 2008), 1159–1168.

[33] Li, M., and Zhao, W. Reliably identifying traffic abnormality under DDOS flood attacks in differentiated services environment based on traffic constraint, 2007.

[34] Lin, C.-H., Liu, J.-C., Huang, H.-C., and Yang, T.-C. Using adaptive bandwidth allocation approach to defend DDoS attacks. In *Proceedings of the 2008 International Conference on Multimedia and Ubiquitous Engineering* (Washington, DC, USA, 2008), IEEE Computer Society, pp. 176–181.

[35] Macqueen, J. B. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (1967), pp. 281–297.

[36] Masud, M. M., Gao, J., Khan, L., Han, J., and Thuraisingham, B. Peer to peer botnet detection for cyber-security: a data mining approach. In *Proceedings of the 4th annual workshop on Cyber security and information intelligence research: developing strategies to meet the cyber security and information intelligence challenges ahead* (New York, NY, USA, 2008), CSIIRW '08, ACM, pp. 39:1–39:2.

[37] Matrawy, A., van Oorschot, P., and Somayaji, A. Mitigating network denial-of-service through diversity-based traffic management. In *Applied Cryptography and Network Security*, J. Ioannidis, A. Keromytis, and M. Yung, Eds., vol. 3531 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2005, pp. 443–471.

[38] Mcgregor, A., Hall, M., Lorier, P., and Brunskill, J. Flow clustering using machine learning techniques. In *In PAM* (2004), pp. 205–214.

[39] Microsoft corporation:. Taking Down Botnets: Microsoft and the Rustock Botnet, May 2011.

[40] Microsoft corporation:. The official Microsoft blog, May 2011.

[41] Mirkovic, J., Hussain, A., Fahmy, S., Reiher, P., and Thomas, R. K. Accurately measuring denial of service in simulation and testbed experiments. *IEEE Trans. Dependable Secur. Comput. 6* (April 2009), 81–95.

[42] Muhai, L., and Ming, L. An adaptive approach for defending against DDoS attacks. *Mathematical Problems in Engineering 2010* (2010).

[43] Oldmeadow, J., Ravinutala, S., and Leckie, C. Adaptive clustering for network intrusion detection. In *PAKDD* (2004), pp. 255–259.

[44] Park, K., and Lee, H. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. In *In Proc. ACM SIGCOMM* (2001), pp. 15–26.

[45] Peng, T., Leckie, C., and Ramamohanarao, K. Protection from distributed denial of service attacks using history-based IP filtering. In *Communications, 2003. ICC '03. IEEE International Conference on* (may 2003), vol. 1, pp. 482 – 486 vol.1.

[46] Peng, T., Leckie, C., and Ramamohanarao, K. Survey of network-based defense mechanisms countering the DoS and DDoS problems. *ACM COMP. SURV 39*, 1 (2007).

[47] Simon, F. L., and Rubin, S. H. Distributed denial of service attacks, 2000.

[48] Snoeren, A. C., Partridge, C., Sanchez, L. A., Jones, C. E., Tchakountio, F., Kent, S. T., and Strayer, W. T. Hash-based IP traceback. *SIGCOMM Comput. Commun. Rev. 31* (August 2001), 3–14.

[49] Snoeren, A. C., Partridge, C., Sanchez, L. A., Jones, C. E., Tchakountio, F., Kent, S. T., and Strayer, W. T. Hash-based IP traceback. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2001), SIGCOMM '01, ACM, pp. 3–14.

[50] Stoecklin, M. Anomaly detection by finding feature distribution outliers. In *Proceedings of the 2006 ACM CoNEXT conference* (New York, NY, USA, 2006), CoNEXT '06, ACM, pp. 32:1–32:2.

[51] Su, M.-Y. Real-time anomaly detection systems for denial-of-service attacks by weighted k-nearest-neighbor classifiers. *Expert Syst. Appl. 38* (April 2011), 3492–3498.

[52] Wang, S., Xuan, D., Bettati, R., and Zhao, W. Providing absolute differentiated services for real-time applications in static-priority scheduling networks. *IEEE/ACM Trans. Netw. 12* (April 2004), 326–339.

[53] Zhong, R., and Yue, G. DDoS detection system based on data mining. In *Proceedings of the Second International Symposium on Networking and Network Security (ISNNS 2010)* (Jinggangshan, P. R. China, April 2010), pp. 062–065.

Appendix A

First appendix

Listing A.1: Example code: creating proactive HTB filter according to 10% clustering

```
# HTB filter for the 10pct clustering
tc qdisc add dev eth4 handle 1: root htb default 10

tc class add dev eth4 parent 1: classid 1:1
htb rate 100kbps ceil 100kbps

tc class add dev eth4 parent 1:1 classid 1:10
htb rate 1kbps ceil 100kbps

tc class add dev eth4 parent 1:1 classid 1:12
htb rate 13kbps ceil 100kbps

tc class add dev eth4 parent 1:1 classid 1:13
htb rate 10kbps ceil 100kbps

tc class add dev eth4 parent 1:1 classid 1:14
htb rate 10kbps ceil 100kbps

tc class add dev eth4 parent 1:1 classid 1:15
htb rate 13kbps ceil 100kbps

tc class add dev eth4 parent 1:1 classid 1:16
htb rate 11kbps ceil 100kbps

tc class add dev eth4 parent 1:1 classid 1:17
```

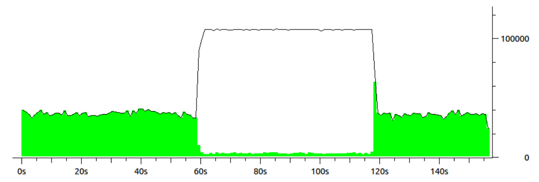
```
htb rate 15kbps ceil 100kbps

tc class add dev eth4 parent 1:1 classid 1:18
htb rate 12kbps ceil 100kbps

tc class add dev eth4 parent 1:1 classid 1:19
htb rate 10kbps ceil 100kbps


tc filter add dev eth4 parent 1: prio 1 protocol ip
u32 match ip src 130.233.192.0/22 flowid 1:19

tc filter add dev eth4 parent 1: prio 1 protocol ip
u32 match ip src 64.0.0.0/6 flowid 1:17

tc filter add dev eth4 parent 1: prio 1 protocol ip
u32 match ip src 80.0.0.0/6 flowid 1:18

tc filter add dev eth4 parent 1: prio 1 protocol ip
u32 match ip src 88.0.0.0/5 flowid 1:16

tc filter add dev eth4 parent 1: prio 1 protocol ip
u32 match ip src 64.0.0.0/3 flowid 1:13

tc filter add dev eth4 parent 1: prio 1 protocol ip
u32 match ip src 128.0.0.0/3 flowid 1:14

tc filter add dev eth4 parent 1: prio 1 protocol ip
u32 match ip src 192.0.0.0/3 flowid 1:15

tc filter add dev eth4 parent 1: prio 1 protocol ip
u32 match ip src 0.0.0.0/1 flowid 1:12
```

Listing A.2: Example code: the HTB filter above is adapted to a random attack in order to serve as the best reactive DDoS defense for such attack. label

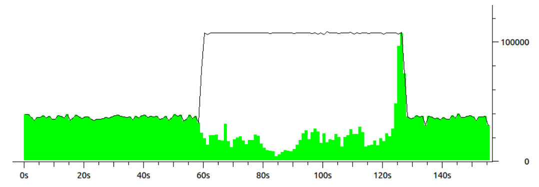
```
# HTB filter for the 10pct clustering
 adapted to the random attack.

tc qdisc add dev eth4 handle 1: root htb default 10
tc class add dev eth4 parent 1: classid 1:1
htb rate 100kbps ceil 100kbps

tc class add dev eth4 parent 1:1 classid 1:13
htb rate 35kbps ceil 100kbps

tc class add dev eth4 parent 1:1 classid 1:16
 htb rate 27kbps ceil 100kbps

tc class add dev eth4 parent 1:1 classid 1:17
 htb rate 18kbps ceil 100kbps

tc class add dev eth4 parent 1:1 classid 1:18
htb rate 20kbps ceil 100kbps


tc filter add dev eth4 parent 1: prio 1 protocol ip u32
match ip src 64.0.0.0/6 flowid 1:17

tc filter add dev eth4 parent 1: prio 1 protocol ip u32
 match ip src 80.0.0.0/6 flowid 1:18

tc filter add dev eth4 parent 1: prio 1 protocol ip u32
 match ip src 88.0.0.0/5 flowid 1:16

tc filter add dev eth4 parent 1: prio 1 protocol ip u32
 match ip src 64.0.0.0/3 flowid 1:13

tc filter add dev eth4 parent 1: prio 7 protocol ip u32
 match ip src 0.0.0.0/0 flowid 1:10
```

Appendix B

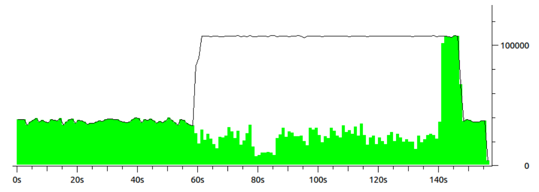
Second appendix



(a) *First run*: no filter is deployed at the router

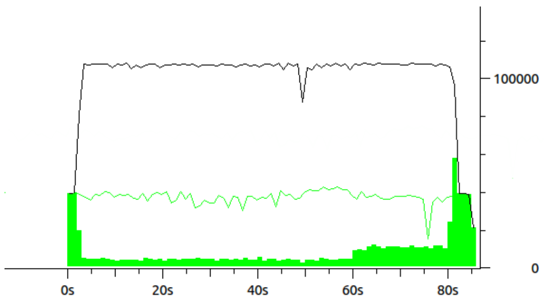


(b) *Second run*: Optimal HTB filter for proactive defense corresponding to 10% clustering is deployed at the router

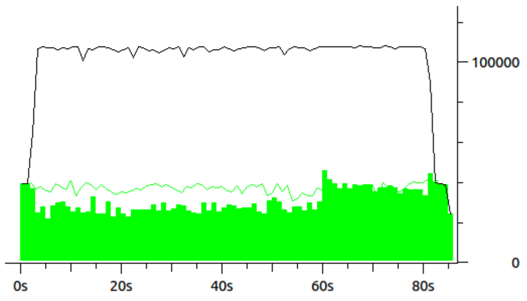


(c) *Third run*: Optimal HTB filter for proactive defense corresponding to 1% clustering is deployed at the router

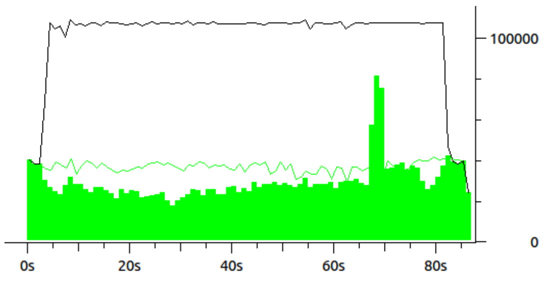
Figure B.1: The traffic output at the server when the normal traffic is generated from the Apache log from January 2010 and the attack traffic is randomly spoofed.



(a) *First phase*: no filter is deployed at the router.



(b) *Second phase*: Optimal HTB filter for proactive defense is deployed at the router.



(c) *Third phase*: HTB filter is adapted to this particular attack distribution according to algorithm 2.

Figure B.2: The traffic output at the server when the normal traffic is generated from the three subsets of Apache log from October 2007. Attack traffic is generated to originate from ten fixed IP addresses.