A COLLABORATIVE DEFENSE FRAMEWORK AGAINST DDOS ATTACKS IN NETWORKS

By

HAIQIN LIU

A dissertation submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

MAY 2013

To the Faculty of Washington State University:

The members of the Committee appointed to examine the dissertation of HAIQIN LIU find it satisfactory and recommend that it be accepted.

Min Sik Kim, Ph.D., Chair

Lawrence B. Holder, Ph.D.

David E. Bakken, Ph.D.

ACKNOWLEDGEMENTS

Firstly, I would like to express my gratitude to my advisor Dr. Min Sik Kim for his guidance, support, encouragement and patience throughout my graduate study. He was always there to meet and talk about my ideas and to give invaluable advice through all phases of my research. I also value his influences on my vision of my professional and personal developments.

I also want to thank my thesis committee members, Professor Larry Holder and Professor Dave Bakken, for serving on the defense committee and reviewing my thesis.

I own my thanks to all the Network Research Lab group members, past and present, for creating a friendly and inspiring atmosphere over these years.

Finally, I want to give the deepest appreciation to my parents and friends for their support.

A COLLABORATIVE DEFENSE FRAMEWORK AGAINST DDOS ATTACKS

IN NETWORKS

Abstract

by Haiqin Liu, Ph.D.
Washington State University
May 2013

Chair: Min Sik Kim

Distributed Denial of Service (DDoS) attacks pose one of the most serious security threats to the Internet. In this work, we aimed to develop a collaborative defense framework against DDoS attacks in networks. We focus on two main phases, which are anomaly detection and filtering of malicious traffic, to achieve a successful defense against DDoS attacks.

Our first accomplishment is to effectively detect DDoS traffic at local nodes. Our conducted experiments can be divided into three categories which are described as follows. Firstly, in order to detect the stealthy DDoS attack at an early stage, we proposed an effective detection scheme based on time-series decomposition method. Moreover, in order to more effectively defend against the attacks, our credit-based defense method is designed for pinpointing the malicious flows. In addition, in order to adapt to the high-speed environment, we present a two-level approach for scalable and accurate attack detection by exploiting the asymmetry in the attack traffic. At both detection levels, sketch structures are utilized to ensure the scalability of our scheme.

Secondly, current defense systems are not scalable well to high-speed networks and few of them are able to defend against attacks originated from both spoofed and genuine source addresses

effectively. Aimed at this problem, we propose a two-stage defense scheme to mitigate attacks. The main advantage of our defense approach is its space efficiency since it does not need to keep per-flow state. Moreover, both spoofed and genuine IP DDoS attacks can be well regulated. We finally extend the single-host sketch-based scheme to a distributed detection scheme and finally develop a collaborative defense scheme. In the distributed detection scheme, we deploy detectors in a certain number of edge routers at the edge side. The local analyzer periodically reports the local processed result to the global analyzer in order to infer the anomaly. The collaborative defense scheme is further developed to filter the malicious traffic. By combing both the host-based solutions with the network-wide solutions, we develop a comprehensive solution that can detect and defend against attacks more effectively. Experimental results using the real Internet traffic demonstrate its effectiveness.

TABLE OF CONTENTS

Page

APPENDICES

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

Nowadays, distributed denial of service (DDoS) attacks pose one of the most serious security threats to the Internet [4–6]. DDoS attacks can result in a great damage to the network service. The DDoS attackers usually utilize a large number of puppet machines to launch attacks against one or more targets, which can exhaust the resources of the victim side. That makes the victim lose the capability to serve legitimate customers and prevent legitimate users from accessing information or services. Since DDoS attacks can greatly degrade the performance of the network and are difficult to detect, they have become one of the most serious security challenges to the current intrusion detection systems (IDS) [4, 7, 8]. Concerning the current state of the network, every corner of the world is likely to be the target of DDoS attacks. However, as long as they are detected early, the loss can be reduced to the minimum. Therefore, DDoS attack detection and defense still attract much concern from researchers.

1.1 How Do DDoS Attacks Work?

During a typical attack period, an attacker controls the compromised hosts to send requests to a target site and those combined packet flows will overwhelm the target due to the limited resources. The target can be machine, network link or even network links of ISPs.

According to the typical communication pattern of DDoS attacks, they can be divided into two main categories, which are called direct attacks and indirect attacks, respectively. In order to launch an attack, attackers have to build a network first. Such kind of networks usually contains

three components, which are attackers, masters and agents. The attacker controls one or more masters and each master controls thousands of agents to initialize the attacks. The attacker itself does not send packets to victims directly. It makes the puppets to send attack packets in order to hide its malicious activities. By this way, it is difficult to track the attack source during attacks.

1.1.1 Direct and Reflector-based Attacks

During a direct attack, spoofed IP addresses are usually involved to prevent attackers from being discovered. As shown in Fig. 1.1, the attacker directly sends packets with forged source IP addresses to the victim side and try to periodically establish connections with the victim to exhaust the victim's resources. Such kind of attacks utilizes the inherent weaknesses of some communication protocols, which require the receiver to send feedback to the sender side when it receives packets from senders. The attacker can take advantage of such feedback mechanism to launch an attack. One of the most prevalent DDoS attacks in the past decade is SYN flood attack which belongs to

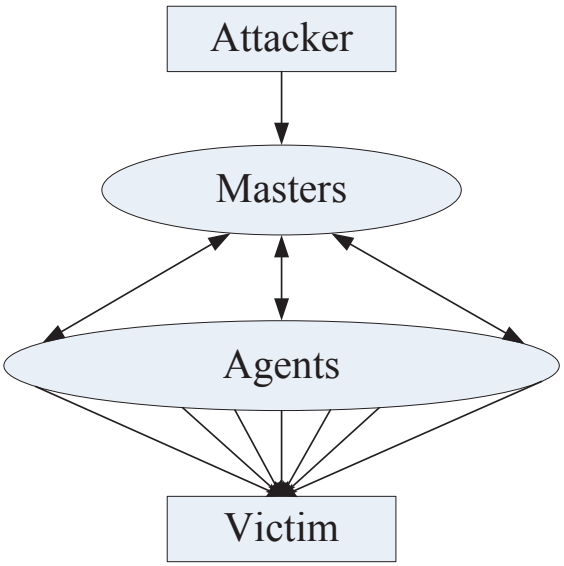


Figure 1.1: Direct attack

direct attacks. According to the three-way handshake mechanism of TCP initialization process, the victim server needs to send an acknowledge packet to the sender side. Since source IP addresses of malicious packets are spoofed, the server will never get responses from sender's side. At the same time, the victim server still keeps a large amount of memory and CPU resources for those broken connections. By exhausting the resources of the server, legitimate users cannot access normal services. Fig. 1.2 shows a typical flow distribution during a DDoS reflector attack. Compared

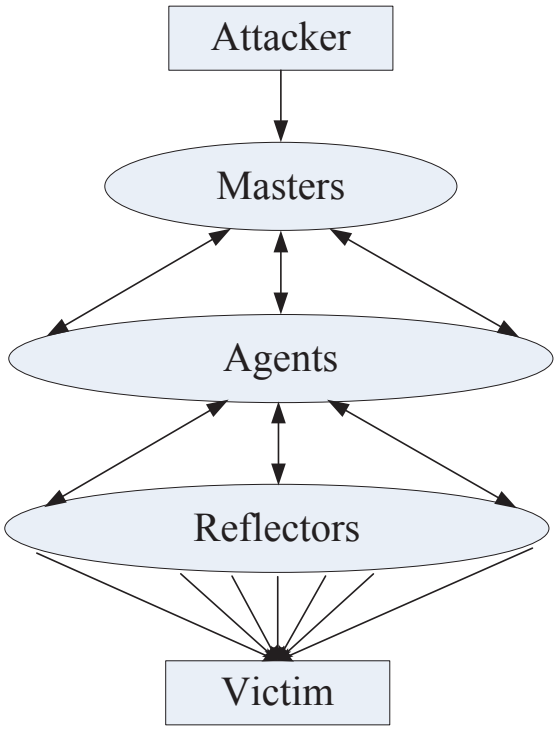


Figure 1.2: Reflector-based attack

with the direct attack, the attackers do not send packets directly to the victim but to some reflectors. Both routers and DNS servers can be utilized as the reflectors. The attacker sends packets, which are required to be responded to the reflectors. However, those packets which are sent to the reflectors contain the victims' IP addresses. The reflectors will then send a large number of packets

to the victims. The large number of packets will saturate the ingress link of the victim. Such kind of attacks is more dangerous since all the responding packets have no difference compared with legitimate packets and thus it is more difficult to detect.

1.1.2 Bandwidth and Resources Attacks

The DDoS attacks can also be divided as bandwidth attacks and resources attacks in terms of the target of DDoS attacks. For the bandwidth attack, there are usually two types of DDoS attacks, namely, denial of edge service and denial of network service attacks [9], which are shown in the Fig. 1.3 and Fig. 1.4. For the former type, the attackers usually try to saturate the ingress bandwidth of the victim side. The reflector attack belongs to the former type, which can render normal users not able to receive responses from the server on time.

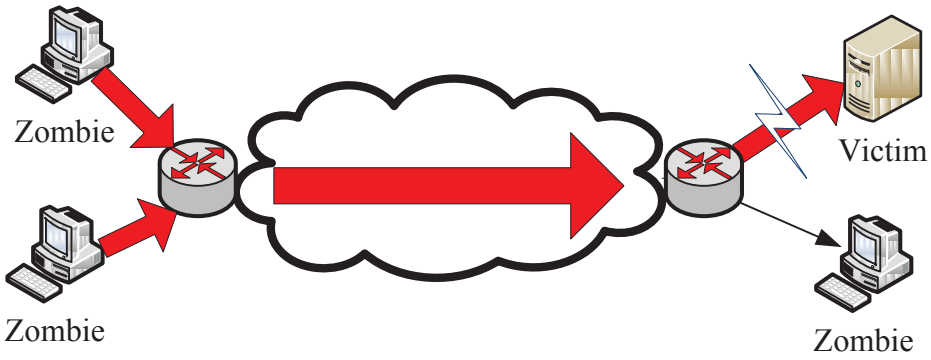


Figure 1.3: Denial of edge service

During a resource attack, the attacker mainly tries to send a large number of virtual connections in order to exhaust CPU and memory resources of the victim. Since the resource of the host is limited, a large number of broken connections will result in the disability of the server to respond to legitimate users.

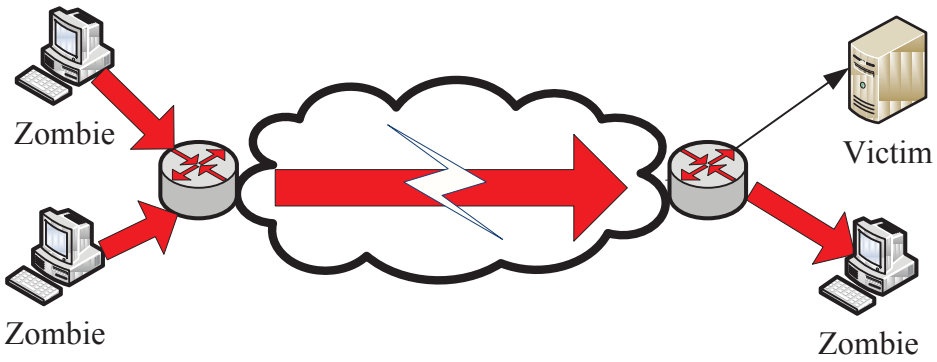


Figure 1.4: Denial of network service

1.2 Why DDoS Attacks Exist?

The prevalence of DDoS attacks today is mainly due to the design goal of the Internet [10]. In the history, the design goal of Internet mainly focused on functionality rather than security and the network tries to provide simple, fast and cheap communications. Those complicated functionalities are assigned to end hosts. Under such best efforts principle, the network itself was designed with delivery efficiency without considering security issues.

It is impossible for the today's internet to regulate the behavior of end-hosts. The attackers always try to find enough vulnerable hosts and deploy them into their BotNet. The attacks can be launched due to many reasons such as annoying, extortion, or trying to disable opponent's network operations. However, the Internet itself has no idea about the attacks and it will always try its best to forward malicious packets to the destinations. Furthermore, it only takes a little cost for attackers to cause large scale damages. That is why DDoS attacks become the most popular attack means for the attackers.

Current DoS attacks are usually extended to the distributed version of DoS. There are two main reasons for that. Firstly, the targets are often highly provisioned servers, and a single machine

usually cannot overwhelm such a server. By employing a large number of zombie machines, the attacker can easily take down a powerful server. Secondly, by using many compromised machines, it is hard for the defense scheme to trace back the source of attacks.

1.3 Motivations of our works

Currently, the large-scale denial of service (DoS) still remains as the main threat to the Internet. Although many solutions have been proposed to fight against traditional DDoS attacks, there is still no panacea for dealing with all kinds of attacks. Many sophisticated attacks keep emerging and it is more and more difficult to detect them. Also, attackers are always trying to hide their behavior by taking advantage of bugs of new proposed methods in order to achieve their goals.

Host-based methods can defend directly against attacks which can mitigate potential damages as early as possible. However, host-based solution alone is far from solving the overall problem. This is because it is usually too late to fight against attacks since damages have already been caused when we detect it at the end host. While someone may argue that we can deploy detectors near the source side in order to detect possible anomalies at an early stage. However, it is very hard to detect malicious patterns from the DDoS traffic since the traffic amount originated from those malicious sources at source-end side is usually extremely small. Furthermore, there is a new DDoS attack called denial of edge bandwidth which can congest the bandwidth of edge links of the victim side by colluding sources and some victim nodes. In this case, host-based methods are incapable to deal with such kind of attacks.

Network-wide solutions seem to be more attractive than host-based solutions. Such kind of

solutions usually places many detectors across edge routers of the core network and then collects local traffic statistics. The filtering instructions are sent back to local nodes to finally regulate the traffic. Although current network-wide solutions can detect anomalies at an early stage, they are usually too slow to mitigate the damages.

Therefore, a network-wide collaboration mechanism will be more effective and efficient to combat with current DDoS attacks. By combing both host-based solutions with network-wide solutions, we are able to develop a comprehensive solution that can detect and defend against DDoS attacks more effectively. To the best of our knowledge, we are the first one to present such collaborative mechanism to fight against DDoS attacks by making local modules and global modules work together. The global defense module collects local information which is extracted by the local defense module and makes filtering instructions in order to regulate the traffic based on a global and comprehensive decision. The local defense module we introduced does not only perform the detection work but also defend directly against some apparent anomaly events without waiting for the instructions from the global defense module to guarantee the timely reaction. We will describe the overall architecture in details in the Chapter 5.

1.4 Main Contributions

The primary contributions of this thesis lie in the design and implementation of frameworks for detecting and defending against DDoS attacks from a collaborative perspective. Theoretical analysis and extensive simulations using real-world traces are also provided to show the effectiveness of our frameworks. Some algorithms for defending against UDP-based DDoS attacks had been deployed

on the PlanetLab environment [11]. The main contributions can be summarized as follows.

- The design of scheme to accurately detect stealthy DDoS attacks which cannot be effectively detected by traditional methods.
- The design of credit-based framework to achieve a fine-grained flow-level detection of malicious flows.
- The design of sketch-based detection scheme to achieve high scalability performance by exploiting the asymmetry pattern.
- The design of sketch-based defense scheme to fight against TCP SYN flooding attacks with fine granularity.
- The design of sketch-based defense framework to mitigate UDP flooding attacks with fine granularity. This algorithm had been deployed on the PlanetLab environment.
- The design of a collaborative framework for detecting and defending against DDoS attacks from a global and comprehensive perspective to achieve better defense performance.

1.5 Thesis organization

The first three parts of our work presented in chapter 2, chapter 3 and chapter 4 all belongs to the very first step of our overall framework. That is to effectively detect DDoS traffic at local nodes. The chapter 2 is proposed to detect the stealthy DDoS attack at an early stage. In order to effectively defend against the attacks, we need to know where the attack happens. For example, if we can pinpoint those malicious flows or those victims which are under attacks earlier, we can generate

filtering rules for the defense purpose. Our credit-based defense method described in chapter 3 is exactly designed for pinpointing the malicious flows. It utilized both macro-level and micro-level features to pinpoint the malicious flows by accumulating the credits for each flow. Furthermore, out sketch-based detection method proposed in chapter 4 is developed for the victim pinpoint capability in the high-speed network environment. Although some similar credit-based and sketch-based methods have been proposed for detection, most of them have their own drawbacks, which will be presented in details in chapter 3 and chapter 4. In chapter 5, a two-stage defense scheme is firstly proposed to mitigate TCP SYN flooding attacks with low memory consumption and then a sketch-based defense method is implemented on the PlanetLab environment in order to evaluate the impact of UDP flooding attacks on the GridStat as well as the efficiency of our collaborative defense scheme. The conclusion and future work of this research work are presented in chapter 6.

1.6 Related Works

In this section, we categorize and summarize current state-of-the-art of DDoS defense schemes. According to the deployment location, the most common DDoS defense schemes proposed till now can be divided into four main categories: source-end-based, core-network-based, victim-end-based and collaborative defense schemes [4], which are described separately below.

1.6.1 Source-end based Defense Schemes

Detecting DDoS at the source-end has many advantages compared with defense at the victim-end or intermediate-network. Firstly, since the traffic volume is at a low level near the source-end, the detection overhead for monitoring traffic can also be low. Secondly, the damage can be mitigated

at the very early stage and can be reduced to the minimal level. Finally, it also prevents the overall defense system from being attacked due to the low level of the attack traffic. It is very hard for the attacker to congest the link near the source end.

However, there are two potential problems with the source-end approaches. The first one is the lack of deployment incentives. There is little motivation for the source-end ISP to protect the victims which belong to other ISPs. The second problem is it is relatively hard to detect the anomaly at the source end compared with the other two methods since the attack traffic may be still at a very low level. In fact, it is mostly deployed to filter spoofed packets at ingress routers by checking whether the packet belongs to its routing domain. D-WARD [1] is a typical source-end

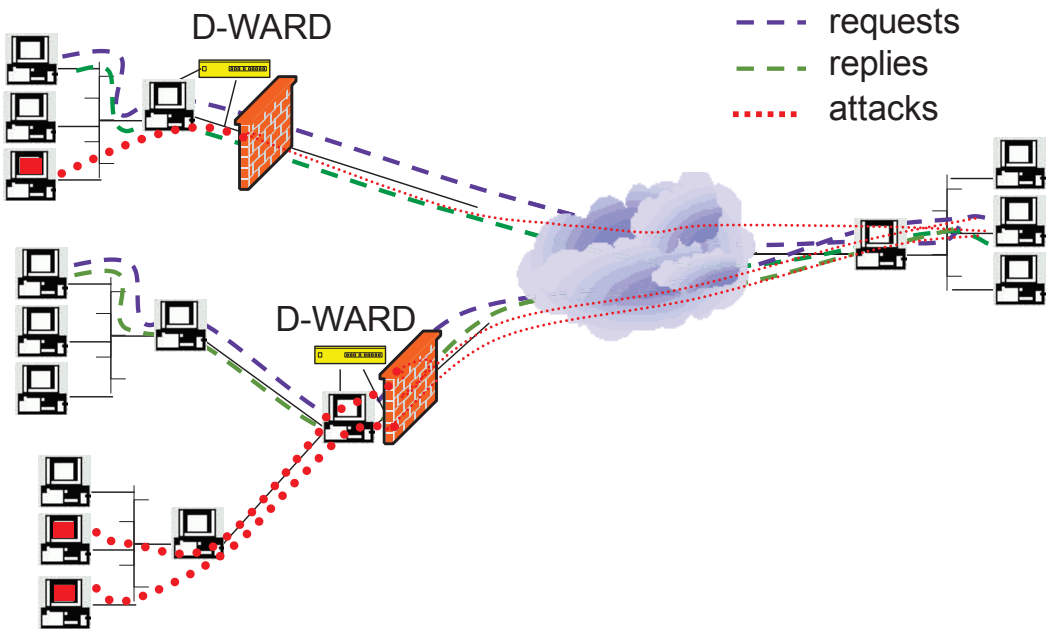


Figure 1.5: D-WARD in Action. Figure copied from [1]

defense scheme against DDoS. The main idea is to leverage a difference between DDoS and normal traffic. It autonomously detects and stops attacks originating from networks. Attacks are detected by constant monitoring of two-way traffic flows between the network and the rest of the Internet

and periodic comparison with normal flow models. Those mismatching flows are further rate-limited in proportion to their aggressiveness. Experiments conducted had shown its effectiveness in filtering DDoS traffic while it only caused little collateral damages to the legitimate traffic. Fig. 1.5 illustrates the D-WARD scheme.

The D-WARD suffers from two issues we mentioned above. Firstly, D-WARD defends other people from your network's DDoS attacks. Also, it doesn't defend your network from other people's DDoS attack. Thus, there will be little incentive for the practical deployment. Secondly, the volume of attack traffic might be very small at the source side which cannot be easily observed by the D-WARD.

In [12], researchers proposed a space and computation efficient source-end based approach to mitigate DDoS attacks by using Bloom filter. Since it only requires limited resource, this method was expected to attract more ISPs to participate the source-end detection. However, the method is only suitable to fight against the malicious traffic originated from spoofed IP addresses and cannot be used to mitigate attacks launched from genuine IP addresses.

1.6.2 Core-network based Defense Scheme

Core-network-based systems are typically deployed across the whole network which requires the global cooperation of detectors placed in the routers [13–15]. This kind of schemes can detect and counter the attack threats at a very early stage which can greatly mitigate the pressure of the victim side. However, network-based schemes suffer from two drawbacks, which can render these schemes unpractical. Firstly, due to some commercial and privacy issues, there is little incentive for cooperation among different ISP companies. Furthermore, since the large-scale deployment and

complex calculations are necessary for this kind of schemes, it is usually a resource-consuming solution.

Defense at the core-network usually requires trace back and pushback, both of which require the cooperation among various ISPs. The trace back techniques are developed to identify the real location of the attacker. Most of these schemes require marking packets along its routing path or sending some special packets. A series of marking algorithms are described in [16]. After the real path of the spoofed packets is identified, the pushback technique, which will be described below, can be applied to inform upstream routers to perform filtering.

Pushback [2] provides a mechanism that allows a router to request adjacent upstream routers to limit the rate of traffic. It extracts attacking signatures by rate-limiting the suspicious traffic destined to the congested link. This is possible since the DDoS traffic is not like the legitimate traffic following the flow control to reduce the traffic rate when the congestion happens. Fig. 1.6 illustrates the architecture of a typical Pushback-based router. The main advantage of Pushback is that even a few core routers are able to control high-volume attacks. However, it also inflicts collateral damage on legitimate traffic because traffic sharing controlled links with attack traffic is likely to also be harmed.

In [17], Seo and his colleagues propose a probabilistic filter scheduling method to mitigate DDoS attacks. In their method, filter routers identify attack paths using probabilistic packet marking technique and maintain filters using a scheduling policy to maximize the defense effectiveness. Again, such kind of methods requires the cooperation among various ISPs, which renders it infeasible to be applied in reality. Also, it will introduce extra space overhead on the header of

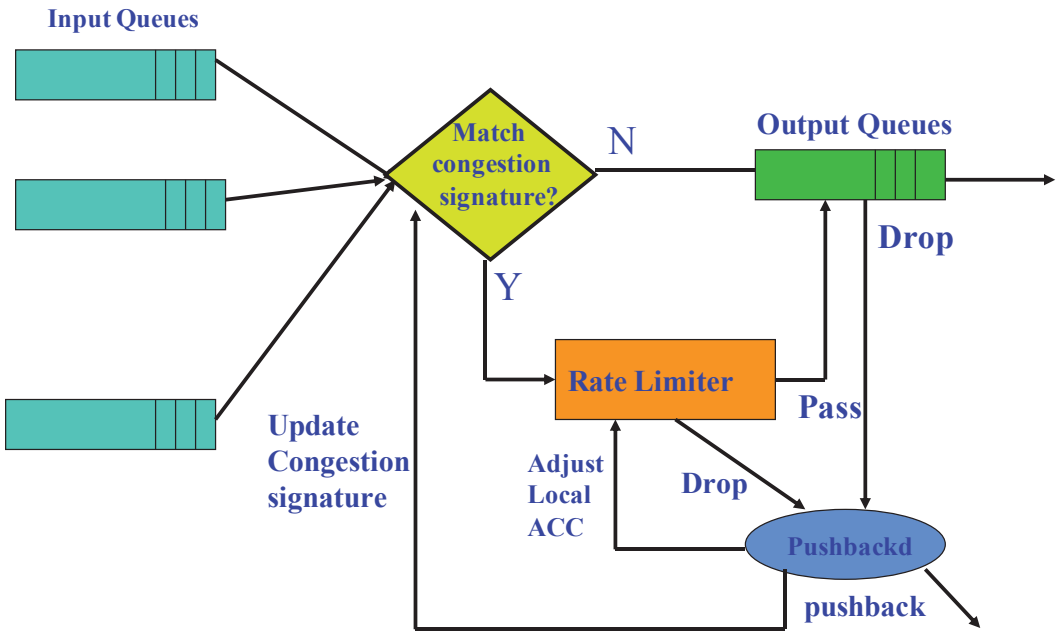


Figure 1.6: Illustration of Pushback. Figure copied from [2]

packets.

In [18], Francois proposes the architecture and algorithms for FireCol, the core of which is composed of intrusion prevention systems (IPSs) located at the ISPs level, to mitigate DDoS attacks. It makes IPSs form virtual protection rings around the hosts to defend and cooperate by exchanging traffic statistics. Again, the effectiveness of the framework depends on the cooperation among different ISPs which render it hard to deploy in reality.

1.6.3 Victim-end based Defense Scheme

There are many research works [19–21] focused on victim side since such kind of defense schemes can maximize the deployment incentives. Such systems were only deployed on the victim side to protect ISPs and enterprise networks as well as individual hosts, which are usually bound with the firewall or intrusion detection systems (IDS).

In [22], Wang proposes the SYN flooding attacks near the server side while the detectors are installed at leaf routers which connect end hosts to the Internet. The main idea of their method is to monitor abnormal SYN-FIN pairs behavior and a non-parametric cumulative sum technique is applied to analyze the anomaly pattern. In [23], Jin utilizes the TTL in the IP header to estimate the Hop-Count of each packet and detect attacks by the spoofed packet's Hop-Count deviation from normal ones. [21, 24] defend against DDoS attacks by distinguishing attacking packets from legitimate ones by a fine-grain traffic profile comparison between the current traffic profile and victim's nominal traffic profile. The effectiveness of these schemes depends on the assumption that the attackers cannot precisely mimic the victim's traffic characteristics. It suffers from two main problems. Firstly, the complex process for generating statistical filtering rules will introduce heavy computing overhead to the system. Secondly, the fact that a victim's nominal traffic profile can still be obtained by an attacker which was pointed out in [25]. In addition, this kind of systems also suffers from some unavoidable weaknesses. Firstly, the requirement of the prior nominal traffic profile render these schemes unable to well adapt to today's dynamic traffic. Therefore, it might result in a large collateral damage which can make some legitimate flows with new traffic patterns failed to pass such kind of defense systems. Furthermore, even though the nominal traffic profile can be built in a frequent and periodical manner in order to meet with the new network conditions, how to know whether the current traffic, which is used for building the new nominal traffic profile, is purely normal or not is still doubtable. In other words, it suffers from an egg-chicken problem. On one hand, the system is designed to fight against the attack. On the other hand, the key component of this system requires a mechanism to find out the period without anomaly traffic in advance.

Unless this problem can be solved properly, there is no guarantee that this kind of system can be fully reliable. We argue that these base-line based schemes can only be suitable for the relatively stable and small-scale network environment. In [26], Liu and his colleagues proposed an approach for diagnosing traffic anomalies by analyzing the behavior of network traffic. They pointed out that the traffic in communication networks has been shown to exhibit statistical self-similar phenomena which can be characterized by the so-called Hurst parameter.

1.6.4 Collaborative Defense Scheme

In [3], Jelena and her colleagues propose a distributed system for DDoS defense which is called DefCOM. Its nodes span source, victim and core networks and cooperate via an overlay network to detect and stop attacks, which is shown in Fig. 1.7. The defense nodes constrain the attack traffic to relieve victim's resources. Also, the nodes cooperate together to detect legitimate traffic within the suspicious stream and ensure its correct delivery to the victim. Furthermore, it offers a framework for existing security systems to join the overlay and cooperate in the defense. Nodes communicate with each other by using an automatically-built overlay. They collaborate by exchanging messages, marking packets as high or low priority and prioritizing traffic during attack. However, the effectiveness of DefCOM depends on several facts which might become weaknesses for this approach. Firstly, it requires the accurate anomaly detection at the victim-end in order to trigger the overall defense flow. Secondly, the nodes inside core-network should cooperate together to push the generated traffic regulation messages back to upstream nodes. Finally, it requires source-end nodes to constrain their out-going traffic which might have little incentive for them to do so.

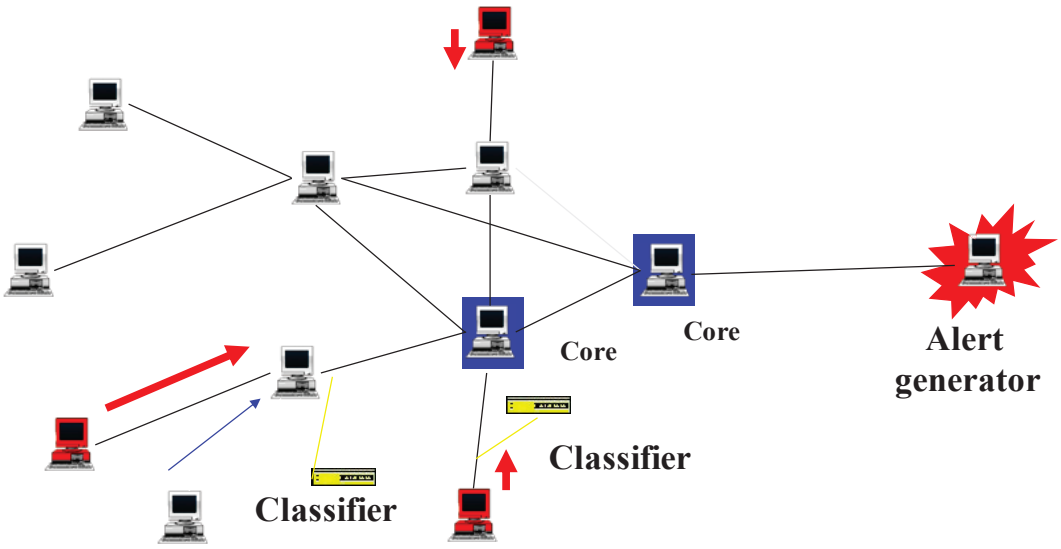


Figure 1.7: Illustration of DefCOM. Figure copied from [3]

In addition, those source networks that do not participate in DefCOM receive poor service because core nodes in DefCOM that perform filtering lack light-weight algorithms to differentiate legitimate from attack traffic at line speed. In order to overcome this problem, Mohit [27] proposed a collaborative scheme by combing DefCOM and Speak-up [28] into a synergistic defense. By integrating Speakup with core defenses in DefCOM, legitimate clients in legacy networks can thus be detected and served. However, the framework is still unable to address the three issues we pointed out above.

CHAPTER TWO

REAL-TIME DETECTION OF STEALTHY DDOS ATTACKS USING TIME-SERIES DECOMPOSITION

2.1 Motivation of this work

In this chapter, we focused on the very first step of the overall collaborative framework. That is how to detect when the attacks happen effectively and efficiently. To be specific, we aimed to detect stealthy DDoS attacks which usually have low traffic volume. Over the past decade, many efforts have been devoted to the detection of DDoS attacks. A typical approach to detecting DDoS attacks in a network is to detect whether the amount of the total flow or other similar metrics exceeds a certain threshold, which is determined based on the traffic history. However, the problem of identifying attack traffic is generally difficult because currently the pattern of the normal network flow is so dynamic and changing that those fixed thresholds can result in a high false positive rate. There are methods that utilize adaptive thresholds which can change according to the network conditions [29]. The main drawback of such methods is that those thresholds are still hard to be determined to get high detection accuracy, and the effectiveness greatly depends on the previous training by using historical data. Moreover, attackers can still manipulate their traffic and packets to defeat detection.

We introduce a new type of DDoS attacks [30] called *stealthy DDoS attacks,* which can be launched by sophisticated attackers. One special case is that a smart attacker injects the attack

traffic in a very slow speed to increase those thresholds to achieve the final attack goal. Those DDoS attacks are called shrew DDoS attacks or low-rate DDoS attacks [31] and are a subclass of the stealthy DDoS attacks. Cheng et al. firstly utilized the power spectral density (PSD) of network flow to detect general TCP SYN flood [32]. Chen and Hwang also used the power spectral density (PSD) of network flow to detect shrew attacks [33]. The attack types they focused on are stealthy, periodic, pulsing, low-rate, and embedded in TCP or UDP traffic flows. Luo and Chang studied the characteristics of shrew attack with a wavelet approach [34]. Lu and his group [15] proposed a network-wide detection scheme for the DDoS attack by exploiting spatial and temporal correlation of attack traffic. Their study mainly focused on the spoofed address attack, which will be unsuitable for detecting the attack launched by a botnet. Unfortunately, none of these defense schemes can identify and filter out the general stealthy DDoS attacks effectively and accurately. The main reason is that a new DDoS attack called slowly-increasing-intensity DDoS attack introduced in this work will be able to defeat the traditional baseline-based detection schemes by stealthily promoting those baselines. In order to detect them, new approach needs to be proposed.

2.2 Stealthy DDoS Attacks

stealthy DDoS attacks can be launched by sophisticated attackers. Such attacks are different from traditional DDoS attacks in that they cannot be detected by previous detection methods effectively. In response to this type of DDoS attacks, we propose a detection approach based on the decomposition of time series, which divides the original time series into the *trend* and *random* components according to the analysis of its characteristics. It then applies a double autocorrelation technique

to the *trend* component while an improved cumulative sum technique is adopted in the *random* component to detect anomalies in both components. By separately examining each component and synthetically evaluating the overall results, the proposed approach can greatly reduce not only false positives/negatives but also the detection latency. In addition, to make our method more generally applicable, we apply an adaptive sliding-window into the real-time algorithm. We evaluate and demonstrate the performance of the proposed approach on the real Internet traces, demonstrating its effectiveness.

We introduce the fundamentals of stealthy DDoS attacks and compare their properties with traditional DDoS attacks. Based on the comparison, we present in detail why this kind of DDoS attacks cannot be easily detected by the conventional methods.

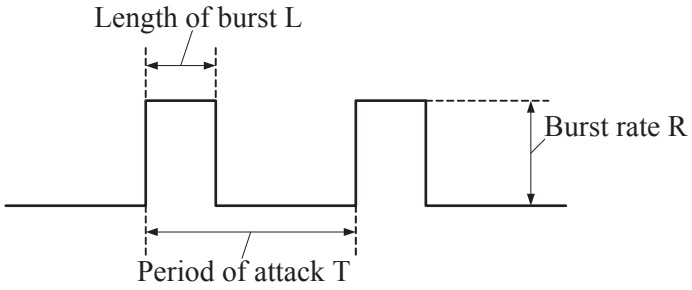


Figure 2.1: A typical low-rate DDoS attack pattern

The terminology "stealthy DDoS attacks" proposed in this work mainly refer to shrew DDoS attacks and slowly-increasing-intensity DDoS attacks (SIDA). The shrew DDoS attacks were firstly introduced in [31], which was followed by a series of related research [34–36]. Generally, a shrew DDoS attack refers to the periodic, pulsing, and low-rate attack traffic embedded in TCP or UDP flows. A typical low-rate DDoS attack is illustrated as a periodic waveform shown in Fig. 2.1, where T is the time period of an attack, L is the length of a burst period, and R is the

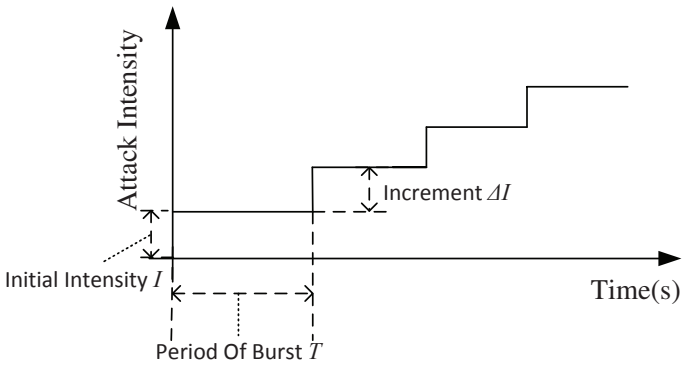


Figure 2.2: A generic example of slowly-increasing intensity DDoS attack pattern

burst rate. We consider the shrew DDoS attacks as a subclass of general stealthy DDoS attacks. In this work, our research focuses on the latter one, which can be launched by a patient, intelligent attacker in no great rush. Fig. 2.2 shows a generic example of a slowly-increasing-intensity DDoS attack, where the parameter I means the initial intensity of the attack traffic, T is the length of the period of burst and ΔI is the increment of the attack intensity each time. The value of ΔI can be manipulated by a smart attacker and be controlled in a very small range to hide the attack behavior. The overview of the SIDA curve is slowly-increasing scalariform. That means the sophisticated hacker can stealthily enhance the threshold for judging legitimate traffic and thus defeat those previous detection methods that detect based on thresholds. This type of DDoS attacks is even more dangerous than those traditional attacks because it is harder to detect, especially when embedded in a large amount of traffic, and thus can greatly delay the anomaly detection.

To the best of our knowledge, none of the previous research focus on the detection of SIDA. In order to detect the new attack type, a new approach needs to be employed.

2.3 Internet Traffic Analysis

The Internet traffic is complex network flows with strong outburst and instability. In this work, we sample the flow connection entropy (FCE) series from the Internet traffic. By calculating the distribution of the FCE series, we can obtain a coarse-grained estimation of the traffic. We define FCE, which reflects the change of the flow distribution caused by DDoS attacks, as follows.

Definition 1. *A* flow f_i *is a 3-tuple* $\{sip_i, dip_i, dport_i\}$*, where* sip_i *represents source IP address,* dip_i *destination IP address, and* $dport_i$ *destination port number.*

This definition of flow can reflect main characteristics of DDoS attack traffic, because a typical DDoS attack pattern is that each zombie machine from a BotNet launches one connection to target at a certain service port of one or more victim machines.

Definition 2. *The* flow connection entropy (FCE) *of a set of flows is defined as*

$$FCE = -\sum_{i=1}^{n} p(f_i) \log_2 p(f_i) \tag{2.1}$$

where $p(f_i)$ *is the probability of receiving a packet belonging to flow* f_i*.*

During certain time period Δt, we consider the frequency of the packets belonging to f_i as the estimation of $p(f_i)$.

From the definitions, we can see that DDoS attack will result in an abnormal increase in the FCE of the network traffic.

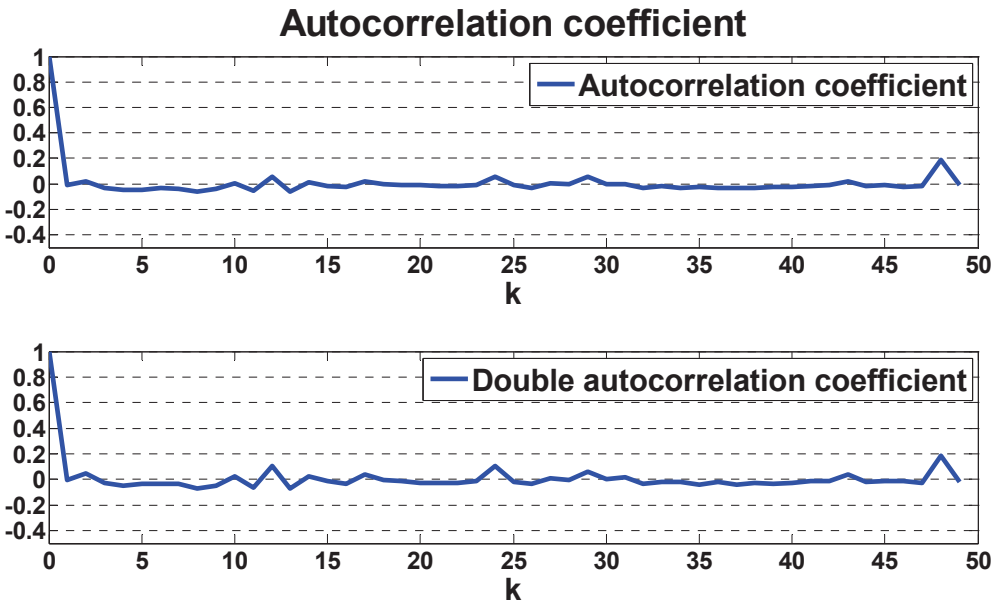


Figure 2.3: Original and double autocorrelation coefficients of normal traffic

2.3.1 Statistical Property of FCE Series

We sample network traffic with sampling period Δt and calculate FCE of every interval. Therefore, the packet arrivals are modeled by FCE time series: $Z(N, \Delta t) = \{FCE_i,\ i = 1, 2, ..., N\}$, where N is the length of the time series. The kth order auto-correlation coefficient of the series Z is then denoted as:

$$\rho_k = \frac{\sum_{i=1}^{N-k}(FCE_i - \overline{FCE})(FCE_{i+k} - \overline{FCE})}{\sum_{i=1}^{N}(FCE_i - \overline{FCE})^2} \tag{2.2}$$

where $\overline{FCE}$ is the mean of FCE series.

Fig. 2.3 shows the auto-correlation coefficient of normal traffic. From that result, we can see that ρ_k is near 0 when the order k is larger than 1. If FCE is stationary, ρ_k should decay rapidly as k increases [37]. Otherwise, ρ_k fluctuates as k increases. Since ρ_k in Fig. 2.3 stabilizes as k increases, we can conclude that the FCE series for the normal traffic are stationary. That is to say, for the

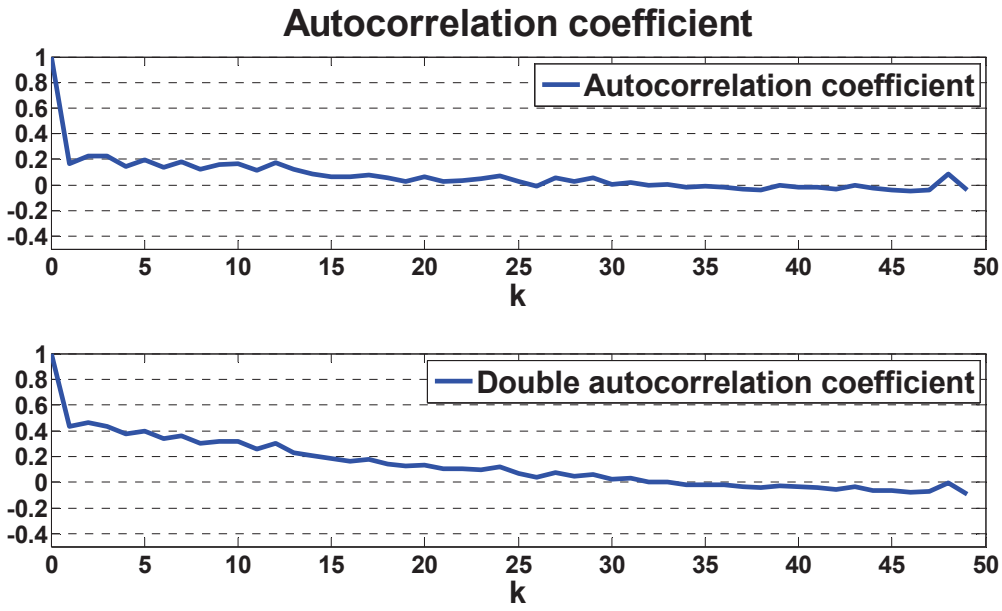


Figure 2.4: Original and double autocorrelation coefficients of traffic with SIDA attacks

normal network traffic FCE series at present time is independent of the previous network status. However, the auto-correlation coefficient of normal traffic with injected SIDA traffic, shown in the Fig. 2.4, is much larger than 0 even when the order k is greater than 1. Besides, its value decreases in a very slow pace. This point is more obvious when we obtain the double auto-correlation (DA) coefficient by re-calculating auto-correlation coefficient of the original auto-correlation coefficient series. Clearly, the injected attack traffic makes the FCE series of network traffic non-stationary. Thus, the traditional auto-regressive (AR) model does not perform well.

Since the Internet traffic today always contains attack traffic, which make the series of factual network traffic do not follow the stationary process, the decomposition of the original series into long-term component and the steady random component will greatly benefits for proposing a more effective approach.

2.3.2 Decomposition Model of Short-Term Traffic

Today's Internet traffic depends on many factors, such as user behavior, network architecture, and unexpected accidents, which make the network traffic contain both stationary and non-stationary components. According to a study conducted by Guang et al. [38], generally the long time-scale traffic exhibit the characteristics of trend, period, mutation, and randomness. Specifically, they divided the large-timescale time series such as FCE_i into trend component A_i, period component P_i, mutation component B_i, and random component R_i. Therefore, for long time-scale traffic, it can be modeled as:

$$FCE_i = A_i + P_i + B_i + R_i \tag{2.3}$$

where A_i and P_i belong to long-term changes which represents the smooth process of network traffic behavior while B_i and R_i belong to short-term changes reflecting uncertainty in network traffic.

In order to come up with a real-time detection algorithm which can be executed in an incremental way, we introduce a sliding window into our algorithm. Generally, the size of the sliding window should be kept small (10 minutes in our implementation) to adapt to real-time detection requirements. With larger window size, more memory resources will be consumed.

For a short period of time, we claim that the mutation component and the period component are negligible. In other words, the network traffic within the window can be considered as the consequence of the interaction between the trend component and random component only. This assumption greatly simplifies the real-time detection algorithm and makes our approach more

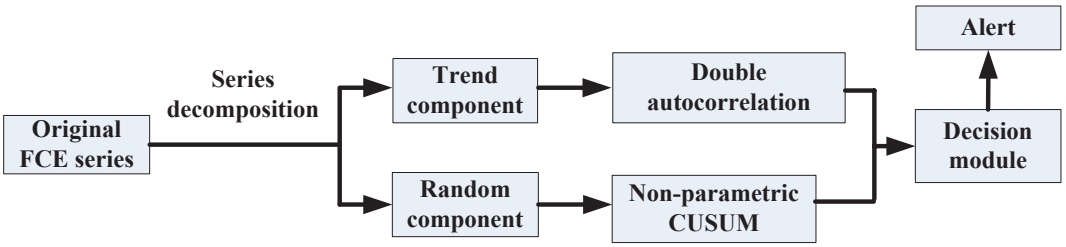


Figure 2.5: Structure and process of detection based on time series decomposition

practical. Under this assumption, the FCE series is modeled as

$$\begin{cases} FCE_i = \overline{FCE_i} + R_i \\ R_i = \eta_i + \varepsilon_i \end{cases} \tag{2.4}$$

where $\overline{FCE_i}$ denotes the trend component, R_i is the random component containing η_i and ε_i, which represents the steady random component and the measurement error (white noise) component, respectively. The value of ε_i can reflect the absolute error between the prediction sequence and the measured sequence.

Fig. 2.5 demonstrates the overview of the detection process based on the time series decomposition method. Given a series of FCE data, we first obtain the trend and random components by decomposition of the series. Then, anomaly detection methods are applied to each component separately. Results are collected at the decision module to make a comprehensive decision. Because each anomaly detection method has its own strength under certain conditions, the proposed synthetic approach has a potential to be more effective than previous approaches in detecting SIDA and general DDoS attacks.

We use a modified exponentially weighted moving average (EWMA) to produce estimate the trend component. The current estimation of FCE is obtained by combining the current FCE

with the FCE from the previous period corrected for trend as follows:

$$\begin{cases} \overline{FCE_i} = \alpha_i FCE_i + (1-\alpha_i)\overline{FCE_{i-1}} \\ \alpha_i = \alpha_{max}(1 - e^{-C\beta_i}) \\ \beta_i = \frac{|FCE_i - FCE_{i-1}|}{FCE_{i-1}} \end{cases} \tag{2.5}$$

where β_i reflects the extent of the fluctuation. Generally, if FCE series are subject to large changes, then the proportion of the history should be small so as to quickly attenuate the effect of old observations. In contrast, when FCE series smoothly change as time goes, the proportion of the history should be large to minimize random variations. The value of the parameter C is determined empirically through experiments.

After we subtract the long-term trend component from the original series, the remainder of the series can be considered as the random part of the traffic. Hence, the estimated steady random series is obtained from

$$R_i = FCE_i - \overline{FCE_i}\,. \tag{2.6}$$

2.3.3 Anomaly Detection of Each Component

By decomposing the FCE series, the trend component will reflect the majority of slowly-increasing signal while the steady random signal is contained in the random component. After we decompose the original series into two separate components, appropriate approaches will be applied to each component separately.

Conventional approaches can perform well in detecting anomaly with certain properties. For instance, the cumulative sum (CUSUM) technique, which is based on the Sequential Change Point Detection [39] can determine whether the observed time series is statistically homogeneous, and find the time when the change occurs. However, its effectiveness will be weakened when the network signal contains a large proportion of increasing- or declining- intensity signals, because those signals may result in statistical bias, causing false positives. In our approach, we apply the non-parametric CUSUM method [40] to the random component, which has all the advantages of sequential and non-parametric tests with light computational overhead.

For the trend component, we calculate the double auto-correlation coefficient series and examine the first K_{max} elements in that series. If the signal in the trend component has an evident increasing or declining tendency, then those values of the first few elements will all exceed certain threshold. Generally, since the window size used in our algorithm is small, the normal trend series should exhibit little fluctuation as time goes. The calculation of the auto-correlation coefficient series can be implemented in an incremental manner.

2.4 DDoS Detection Algorithm

The important performance metrics of DDoS detection include detection accuracy and detection latency. However, it is impossible to reduce both at the same time; we need to find out an appropriate trade-off between the two metrics. Towards this goal, we firstly discussed the following notions.

2.4.1 Self-Adaptive CUSUM Technique for Random Component

In order to accurately and timely detect the mutation point of random series R, we adopt the CUSUM technique in our approach. The basic idea of CUSUM is to accumulate those small offsets during the process to amplify the varying statistical feature and thus improve the detection sensitivity. CUSUM can detect a small deviation of the mean effectively. It is generally defined as:

$$\begin{cases} y_i = (y_{i-1} + x_i)^+ \\ y_0 = 0 \end{cases} \tag{2.7}$$

where x_i is the observed original value and Δ^+ is Δ if $\Delta > 0$ and 0 otherwise. When x_i becomes positive from a negative value, y_i becomes larger, and its exceeding a threshold TH_{CUSUM} indicates the change point of the original series. In our implementation, we use $\tilde{R}_i = R_i - R_H$ as the original series, where R_H is the upper bound of R_i series during normal process.

Although the original CUSUM algorithm can quickly and efficiently detect attacks, after an attack period ends, the alarm will remain active. To stop the alarm, we consider the attack is over if y_i does not grow for $C_{\mathrm{AttackEnd}}\Delta t$. When the alarm stops, we reset y_i to 0.

2.4.2 Double Autocorrelation Technique for Trend Component

As shown in Fig. 2.4, the double autocorrelation coefficient can be used as an indicator of the existence of SIDA in network traffic; the injected SIDA traffic increases the double autocorrelation coefficient. It results from the high internal dependency of SIDA traffic. The FCE series obtained from the same traffic also exhibit this property. Based on these observations, we can detect the

anomaly in the trend component using the following condition:

$$\rho'_k(i) > TH_{\mathrm{DA}},\ 2 \leq k \leq K_{\max} \tag{2.8}$$

where $\rho'_k(i)$ is the double autocorrelation coefficient series at phase i. If all the first $K_{\max}$ elements of the double autocorrelation coefficient series (except the very first element which always equals to 1) exceed the threshold TH_{DA}, then we conclude that there is SIDA traffic embedded in the traffic.

Considering the fact the SIDA traffic is injected to the normal traffic in a very slow pace, another condition should be introduced in order to detect the SIDA traffic at an early stage. During $\theta\Delta t$ period, if the sum of the first K_{max} element (except the very first one) keep growing $\lfloor C_{DA}\theta \rfloor$ times, then we consider a SIDA attack is initialized by attackers as follows:

$$\sum_{j=i-\theta+1}^{i} 1 * \{ \sum_{k=2}^{K_{max}} \rho'_k(j) > \sum_{k=2}^{K_{max}} \rho'_k(j-1) \} \geq \lfloor C_{DA}\theta \rfloor \tag{2.9}$$

where C_{DA} is an empirically-determined constant. The above condition can be checked by the following way. We firstly compare the sum of the first K_{max} element of the double autocorrelation coefficient series at phase $i-\theta+1$ with that at phase $i-\theta+2$. If the former is greater than latter, then 1 will be accumulated into the left side of the formula, otherwise 0. If the cumulated value during the previous θ phase is greater than $\lfloor C_{DA}\theta \rfloor$, then an anomaly is indicated.

2.4.3 Adaptive Sliding Window

We use an adaptive size sliding windows into our algorithm, which means the size of the introduced window can be automatically adjusted according to the current network traffic condition. An adaptive sliding window is necessary for improving the performance of proposed algorithm. Intuitively, when the value of double autocorrelation coefficient becomes larger than previous phases, the window size should be enlarged so as to capture a more obvious increasing or declining trend. Therefore, the following ratio R_{RA} can be used to represent that trend.

$$R_{RA}^{i} = \frac{\sum_{k=2}^{K_{max}} \rho_k'(i)}{\sum_{k=2}^{K_{max}} \rho_k'(i-1)} \tag{2.10}$$

For the random component, the performance of CUSUM technique do not require a specific length of the time series, however, the window size can be smaller in order to reduce the detecting latency.

The initial size of the sliding window can be denoted as L_0. Every time the window slides forward, the size of the window changes. Based on the discussion above, the size of the adaptive sliding window can be denoted as:

$$L_i = \begin{cases} L_{i-1} + C_1 \left\lfloor R_{DA}^{i-1} \right\rfloor - C_2 & \text{if } L_i \geq L_0 \\ L_0 & \text{otherwise} \end{cases} \tag{2.11}$$

where C_1 and C_2 are both positive integers, which can be optimally determined by experiments.

2.4.4 Overall Algorithm Description

Based on the above discussions, the proposed algorithm can be described as follows. After initializing those values of parameters, we sample the initial FCE series with window length L_0. Then, we decompose the FCE series inside the window by the improved EWMA technique proposed in the previous section. The following flow contains two branches. That is to separately apply the DA and CUSUM techniques to trend component and random component, respectively. One aspect we need to pay attention to is that the value of y_i need to be reset to 0 when the termination of the attack is verified. If one of these two branches detects an anomaly, then an appropriate attack type will be reported to the system. After computing the next $L_i - L_{i-1} + 1$ FCE series, the sliding window forwards to next, and then the algorithm repeats. As we can see, all of the techniques adopted in this algorithm only require simple processes and small computing resource.

2.5 Evaluation

We use actual network traffic downloaded from the MIT Lincoln Laboratory [41] to evaluate our approach. The traffic we tested is synthetically generated by merging the attack traffic and the normal traffic.

We developed our own tool to generate the SIDA traffic, and the increasing rate can be adjustable. The duration of one SIDA was normally distributed with mean 15 minutes and variance 1. The attack traffic is periodically generated and the duration of the period was exponentially distributed, with mean value 60 minutes. General outburst-like SYN flood attack traffic was also injected into the tested trace, and the duration of each outburst follows normal distribution with

mean 20 seconds and variance 5 seconds. The interval between each outburst was exponentially distributed with mean 1000 seconds. From Fig. 2.6, we can see that the merged traffic seems to be normal, especially we compare the traffic volume with the ground truth shown in the Fig. 2.7.

Table 2.1: The default parameter settings of stealthy DDoS detection

Item	**Parameter**	**Setting value**
Interval for Sampling FCE Series	Δt	5s
Time Series Decomposition	α_{max}	0.4
	C	10
CUSUM Technique Detection	$C_{AttackEnd}$	3
	TH_{CUSUM}	10
DA Technique	K_{max}	6
	TH_{DA}	0.4
	C_{DA}	0.8
	θ	3
Adaptive Sliding Window	L_0	50
	C_1	3
	C_2	5

We considered the detection accuracy and the detection latency as the main performance metrics. Here, the detection accuracy contains two aspects: false positive ratio (FPR) and false negative ratio (FNR). Table 2.1 shows the default parameter settings for the parameters we adopted in our experiment.

Fig. 2.6 shows that after we calculated the FCE series, the SIDA attack can be greatly amplified from the background traffic. Hence, several periodical SIDA attack can be observed from the FCE series. Such slowly-increasing trend is more obvious when we apply the time series decomposition approach into the original FCE series. Several strong long-period tendencies can

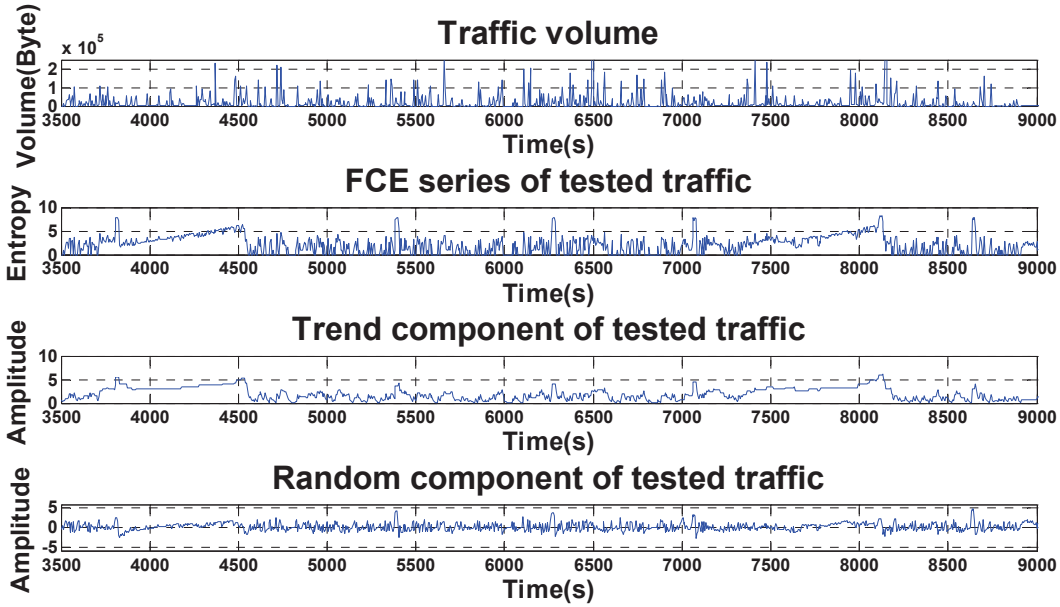


Figure 2.6: Traffic characteristic of tested traces

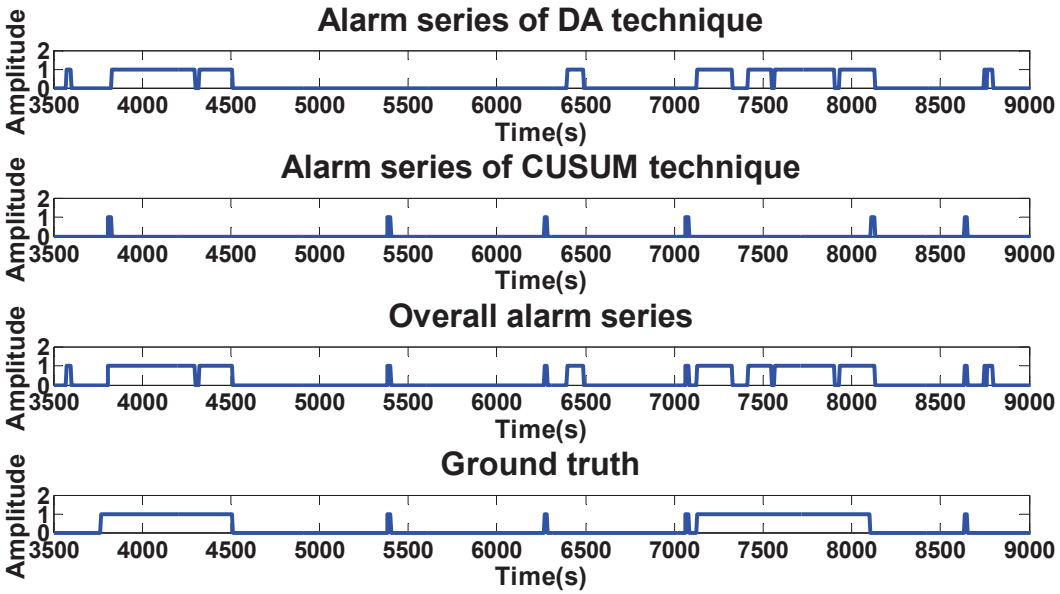


Figure 2.7: Detection result compared with ground truth

be observed from the trend component, and it provides the basis for the DA technique to detect its high internal correlation. Concerning the random component, those outbursts, which indicates general DDoS attack stream, can be quickly detected by the improved CUSUM technique.

Fig. 2.7 shows the detection result of experiments. We use alarm series to represent our detecting result, which value is equal to 1 when an attack is detected and otherwise 0. From the Fig. 2.7, nearly all the SIDA can be detected in a very early stage. The average of the detection latency for those SIDAs in our experiment is around 32 seconds, which is very small when compared with the 900 seconds whole period of SIDA. Compare with the ground truth series, the best overall FPR in our experiment is around 4.3% and the overall FNR is around 9.8%. However, the better performance of the accuracy depends on an improved combination of those parameter values, which is still our ongoing work.

2.6 Conclusion

In this chapter, we introduce a new type of DDoS attacks called stealthy DDoS attacks, which can be launched by a sophisticated attacker. Such attacks are different from traditional DDoS attacks and cannot be easily detected by traditional detection methods. For this type of DDoS attacks, we propose a detection approach based on the decomposition of time series, which divides the time series into trend and steady random components. We then analyze different components to detect the anomaly in both long-term and short-term changes of the traffic. By analyzing each component separately and evaluating results synthetically, the approach can greatly reduce both false negatives and false positives. Furthermore, to make our method more generally applicable,

we apply the adaptive sliding window to our approach. The experimental results using real Internet traces show the effectiveness of this approach.

The detection scheme developed can be used in the local detection module to detect the attacks at an early stage. In fact, it can be used to decide when to trigger the defense modules in the system in order to reduce the filtering overhead.

CHAPTER THREE

TRUSTGUARD: A FLOW-LEVEL REPUTATION-BASED DDOS DEFENSE SYSTEM

3.1 Motivation of this work

Although we are able to detect when the stealthy DDoS attacks occur in the previous work, we still have no idea where the attack happens. In order to throttle the malicious traffic, we have to know, at least estimate, the malicious flows and perform packet filtering on those suspicious flows. Aimed at this goal, we further developed the TrustGuard in this chapter to identify which flows belong to attacks.

3.2 Related works

Recently, a series of credit-model based frameworks have been proposed to defense the DDoS attacks. Since such kind of defense scheme usually requires little prior configuration and can automatically adapt to various network conditions, this direction does attract attention from researchers. Jun and his colleagues [42] introduce a message rate controlling model (MRCM) in their credit-based framework to defense against DDoS for P2P streaming system. It is essentially a network-wide solution since this scheme requires the cooperation of the nodes inside this system, which render this scheme expensive for the practical deployment. Jayashree and his group [43] propose a trust-based traffic monitoring approach for preventing DDoS attacks. Again, their framework requires the interaction among the peer nodes in the network. Furthermore, attackers can defeat

their system by overwhelming the legitimate traffic since their credit accumulation is based on clustering.

Among those previous trust-based frameworks for countering the DDoS attacks, the most similar scheme compared with our proposed system is presented by Natu and Mirkovic [44]. They propose a credit-based ticket-granting system which is deployed in the victim server. However, that framework still suffers from several deficiencies that can make it vulnerable in practice, which are addressed in this work. Firstly, since their system is deployed at the victim server, it may not be able to protect legitimate access to the server, because the ingress link to the server can be congested by high-volume attack. Secondly, their credit calculation scheme is based on the feedback of the server traffic conditions, which can usually counteract the goal of defense that to make the reaction before the damage. Moreover, since the initial credit is equally assigned to each client, attackers can easily launch attacks to defeat this scheme by using a huge number of one-off clients or random address spoofing. Finally, ticket-granting mechanism is adopted in their scheme by hash-based integrity checks, which can make the entire system failed if the secret hash method is exposed.

3.3 Credit-based DDoS defense scheme

In this work [45], we firstly provide an understanding of the existing DDoS attack pattern by analyzing recent Internet DDoS attack traffic and then point out the drawbacks of existing defense schemes. To combat these deficiencies, we propose a credit-based defense system: *TrustGuard.* Essentially, flows accumulate credit based on the diversity of their packet-size distribution. The

more diverse the flow, the more credit it has. Since DDoS attacks demonstrate low diversity they accumulate less credit and are likely to be dropped by the system. Naturally, the performance of *TrustGuard* greatly depends on the choice of credit accumulation and flow selection methods. We derive our solution by identifying the essential characteristics of DDoS attacks. Our analysis accounts for both micro and macro behaviors of DDoS attacks. The primary goal of this work is to not only detect the occurrence of a DDoS attack, but to also identify the attackers and victims involved. Experimental results demonstrate that *TrustGuard* performs admirably in both cases.

The novel features of our system are: (1) It is a light-weight and automated scheme executed in real-time manner and can be easily applied to the practical deployment without many configurations. (2) Fine-grained flow filtering capability can be provided by our scheme, which will be useful for the generation of new rule sets for the current intrusion detection system. (3) Compared with the previous filtering scheme based on the nominal traffic profile, our scheme can be more active to fight against the DDoS attack without building any traffic profile in advance. (4) The combination of the macro and micro level credit accumulation can provide the guarantee that our system can defend against DDoS attacks from both spoofed and true address.

3.3.1 DDoS Pattern Analysis

We evaluated recent Internet traffic traces from the Cooperative Association for Internet Data Analysis (CAIDA) [46, 47]. We derived normal Internet traffic from "The CAIDA Anonymized 2008 Internet Traces Dataset" (CAITD) and DDoS traffic from the "CAIDA DDoS Attack 2007 Dataset" (CDAD). The CDAD traces only contain attack traffic to the victims and any responses to that traffic. All traffic in the CAITD was collected from both directions of an OC-192 Internet backbone

link by CAIDA's equinix-chicago monitor. Both of these two traces were anonymized with the same key and the payload has been removed from all the packets for privacy reasons.

The primary goal in this section is to find those essential features of DDoS attacks which can provide critical indicators to facilitate building a more reliable and robust credit-model. We extract the key features of the DDoS attacks for both the macro and micro-levels of the traffic patterns. The terminology "macro-level" feature means those globally distinguishable traffic patterns when a DDoS attack occurs while "micro-level" represents those anomaly patterns that usually require the inspection of each individual flow. We define a flow below.

Definition 3. *A* flow f_i *is a 2-tuple* $\{sip_i, dip_i\}$*, where* sip_i *represents source IP address,* dip_i *destination IP address, for a uni-directional flow.*

The combination of the observations from both the macro and micro-level offers two benefits: (i) Macro traffic features concentrate singly undetectable events into a global event that can make detection easier and faster without the need for deep inspection of each flow. (ii) Macro traffic features alone cannot provide a high-level of granularity in filtering. Thus, combining macro-level features with the micro-level flow characteristics allows for greater refinement in filtering.

3.3.2 Macro-Level Anomaly Analysis

A number of features of DDoS traffic have been proposed as strong indicators of attack. These elements include: entropies of the individual components of the flow five-tuple (SIP, DIP, SPORT, DPORT, Protocol) [48], flow symmetry features [15], packet sizes [49], packet-to-flow ratio [50] and the ratio of SYN to SYN/ACK [51]. Our first step is to find the most descriptive features of

both spoofed and true IP DDoS attacks amongst all these candidates. In order to induce a better indicator for macro-level DDoS patterns, several features of DDoS are discussed.

[50] points out that DDoS attacks can engender an abrupt disproportion between the number of received packets and the number of IP flows. A DDoS attack from a series of zombies will have a finite number of instigating flows (i.e. the number of zombies). Thus, the ratio of packets to flow will be quite large. However, if the attacker employs spoofed IPs, then the potential range of attack addresses approaches several billion which can potentially mean many flows of few packets creating a relatively low ratio between packets and flow.

This makes sense if we consider true IP DDoS attacks where the attacker is causing many zombies to saturate a victim's resources. However, it is not necessarily the case when we consider the spoofed IP DDoS attack scenario. Since the source IP address of each packet can be randomly chosen from a large address pool, it can result in a large amount of single-packet flows in the traffic which can still maintain the proportion feature between the packet number and the flow number.

Some researchers [15, 52] try to measure the flow-level symmetry and utilize it as the basis for their detection schemes. Such a scheme is only suitable for spoofed IP attacks and requires the deployment of a detector near the source which may not be feasible in practice since little incentive exists for the source Autonomous System (AS) to protect a victim located in a different AS. Some DDoS attack traffic, generated by reflectors or zombies for example, is hard to distinguish in terms of the flow-level symmetry property as it is well-formed traffic. Our preliminary inspection of both CAIDA traces indicates that normal flows and malicious flows show approximate equality in the flow symmetry feature.

Entropy is a well-known metric for measuring the degree of randomness given a distribution. This property can be utilized for detecting a DDoS attack because DDoS traffic always comes from multiple sources and aggregate at one or a few destinations. Thus, when a DDoS attack occurs, the entropy of the source IP will become larger while the entropy of the destination IP will shrink. However, we are not going to adopt the previous entropy definitions [48] in our scheme. The main reason is that those entropy definitions usually reflect higher-level anomalies which are hard to pinpoint. In our system, the macro-level anomaly indicator should support pinpointing victim addresses, in order to facilitate the explicit identification of the misbehaving flows.

Mao et al. [49] indicate that 83% of attacks purely consist of small packets (usually less than 100 Bytes). Our inspection of the CDAD trace also shows a similar result. Fig. 3.1 illustrates an example of the packet size distribution by evaluating a single 5-minute interval of traffic from the CDAD trace compared with the trace with the same period of normal one shown in Fig. 3.2. The distributions are both plotted as histograms according to the packet size. The data in the CAIDA traces demonstrated packets between 0 and 1600 Bytes. Thus we divided the total space into 16 equal intervals. The order of the interval that a given packet size falls in denotes the size of the packet. We can see that, for each specific protocol, the distribution of attack packet size is highly concentrated in level-1, which denotes a packet size less than 100 Bytes. Over 99% of attack packets from TCP and ICMP traffic, and 90% of UDP, fall in level-1.

We argue that DDoS attack traffic tends to see smaller payloads for several reasons. First, the packet processing rate, rather than bandwidth, is the typical bottleneck for network devices. Thus, it is a better strategy for an attacker to send small packets as rapidly as possible to overload

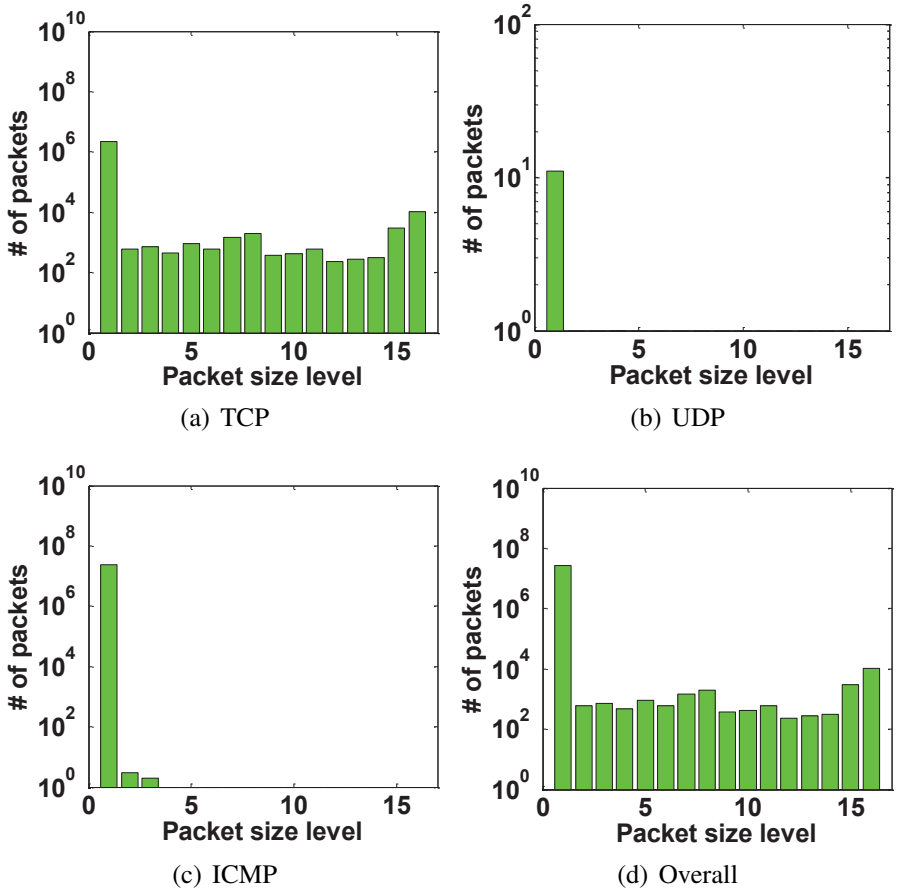
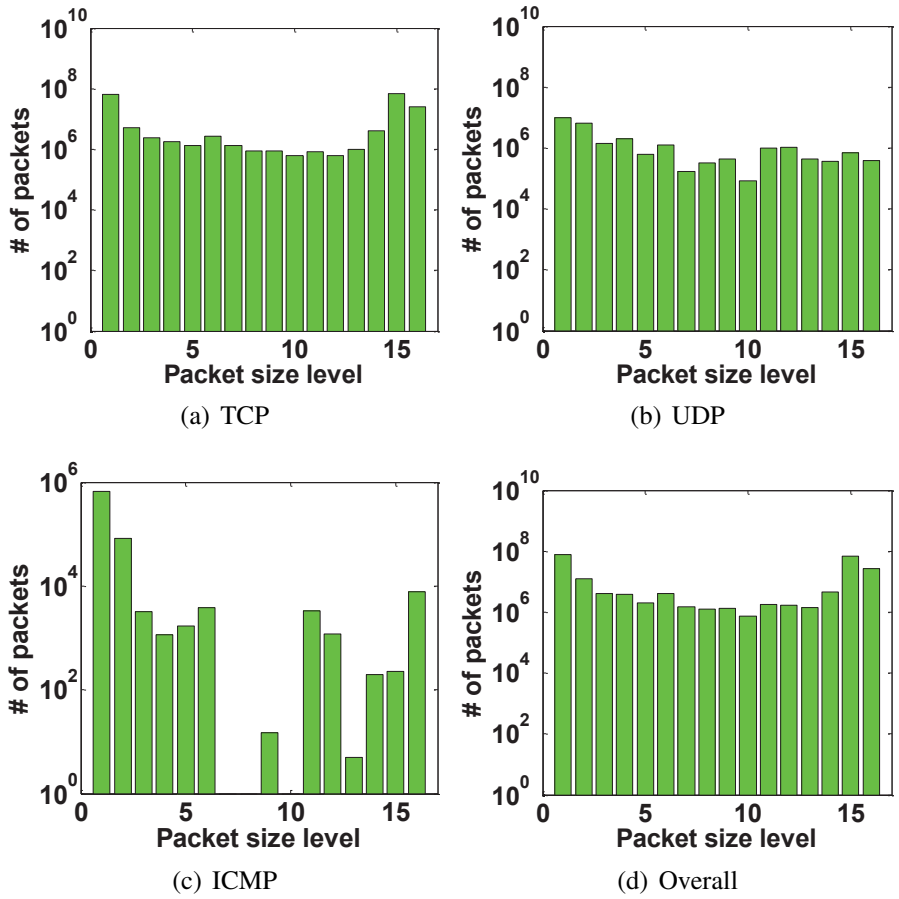


Figure 3.1: Packet size distribution of DDoS traffic over different protocols.

the victim's CPU and memory. Moreover, low-volume attack traffic by utilizing small packets can help to hide the attack behavior to circumvent volume-based IDS. Finally, attack packets originating from different sources are usually produced by the same, or at least a similar, program which leads to packet distributions that are largely deterministic. For example, the IRC robot is a popular choice for attackers and all traffic generated from that software will demonstrate similar patterns.

There is the possibility that some applications will have a high probability that most of the packets involved are smaller than 100 Bytes. In order to better target the anomaly detection, we



(a) TCP (b) UDP

(c) ICMP (d) Overall

Figure 3.2: Packet size distribution of Internet traffic over different protocols.

redefine the packet size-level scheme, as shown in Table 3.1, to further reflect the distribution of packet sizes at the lower levels.

Table 3.1: Definition of our packet size level scheme

Size level	1	2	3	4	5
Range(Bytes)	40–60	60–80	80–100	100–120	120–200
Size level	6	7	8	9	10
Range(Bytes)	200–400	400–600	600–800	800–1000	1000–1200
Size level	11	12			
Range(Bytes)	1200–1400	1400–1600			

Based on the above observation, we define the macro-level anomaly indicator $H(A)$ for a destination address A.

$$H(A) = -\sqrt{i_{\max}} \cdot \sum_{i=1}^{12} p(x_i) \log_2 p(x_i) \quad (1 \leq i \leq 12) \tag{3.1}$$

Where x_i means the i^{th} packet size-level, $p(x_i)$ represents the probability that a packet possess a size that falls in the i^{th} level and $i_{\max}$ is the packet size-level where most packets fall for one specific address A. Thus, the greater concentration of packet sizes at the lower size levels, the smaller the $H(A)$ value is, which translates to a higher probability that the victim A is under DDoS attack.

Although there might be other properties among different flows targeted at the same victim, utilizing the packet size distribution is sufficient for our detection goals.

3.3.3 Micro-Level Anomaly Analysis

Once we can identify victims based on $H(A)$, the next question is how to distinguish malicious flows from the normal flows. We call the factors that can help to indicate an anomalous flow as micro-level anomaly indicators. One micro-level indicator would be to simply build an $H(\cdot)$ function for each flow. However, this is not a light-weight solution since it would require significant resources as the number of flows increase. This could cause the system to become a bottleneck and reduce its value.

Without considering the subsequent behavior of flows, it is difficult, if not impossible, to tell whether an isolated packet is malicious or not. It is difficult to determine whether an isolated packet is malicious or not without referring to some global information. We term "malicious" to mean that

a packet contributes to a DDoS flow. We are motivated by the fact that most packets in attack flows will share a similar distribution of packet sizes at the macro-level. Based on this observation, the information $i_{\max}$ we obtained in the macro-level phase can be utilized to construct a light-weight flow level indicator at the micro-level phase. Here, we can apply a credit-based accumulation method for building the reputation of each flow targeting a particular victim. Suppose the credit is a number that ranges from LOW to $HIGH$ and a new flow is assigned a credit value of LOW. We denote f_A as a specific flow which is destined to address A. Thus, whenever there is a packet that belongs to flow f_A, the micro-level reputation of such flow can be calculated as:

$$Credit_{f_A}^{\text{new}} = \min(Credit_{f_A}^{\text{old}} + \alpha(1 - e^{-|i-i_{\max}|}), HIGH) \tag{3.2}$$

Where α is the credit increase factor, i represents the packet size-level of the incoming packet and $i_{\max}$ can be obtained from the macro-level detection phase. Since the packet size distribution of attack flows is highly correlated, their flow credits will keep constant or continue growing at a very slow pace. As discussed earlier, small packets are advantageous to attackers when instigating an attack. Thus, it is not necessarily a good strategy for an attacker to implement diverse packet sizes in their attacks. Given this we believe our accumulation scheme should maintain its effectiveness well into the future.

In contrast, normal flows will build their credit rapidly. Large flows of small packets are advantageous to attackers and changing strategy to create DDoS with different packet size distributions is not necessarily productive for the attacker. For example, if an attacker creates a DDoS with a diverse packet distribution then this will undoubtedly offset the efficiency of an attack as well

as its ability to avoid detection. Given this state our current credit accumulation scheme should maintain its effectiveness well into the future.

Other micro-level features of DDoS attacks can also be adopted into our system for building the reputation of flows. For example, as pointed out in the study [49], the fact that all packets in one TCP flow have only a single flag (such as SYN, RST, ACK in most cases) can be considered a strong indication of DDoS attacks. Obviously, more characteristics involved in the defense system can provide more accurate filtering. However, since our packet size based credit scheme is sufficient to filter the current DDoS attack flows, which will be shown in our experiments later, we are not going to adopt those additional features of DDoS attacks in order to reduce the system complexity.

3.4 System Description

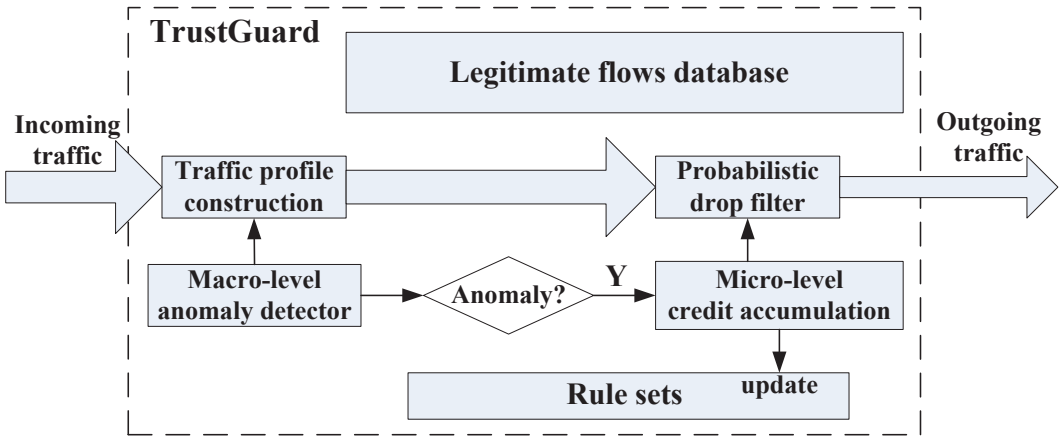


Figure 3.3: Overview of system architecture

In this section, we detail the architecture and function of our system. Fig. 3.3 illustrates the overall architecture of our defense scheme. Our system contains four main modules, namely,

Traffic Profile Construction (TPC), *Macro-level Anomaly Detector* (MAD), *Micro-level Credit Accumulation* (MCA), and *Probabilistic Drop Filter* (PDF), which are described, in detail, as follows.

3.4.1 Traffic Profile Construction

The main function of the TPC module is to build the current traffic profile. A hash table with a linked list can be adopted in this module for statistical accumulation. For a given packet, we use the corresponding destination IP of that packet as the key in the hash table. Fig. 3.4 demonstrates the typical entries in the hash table.

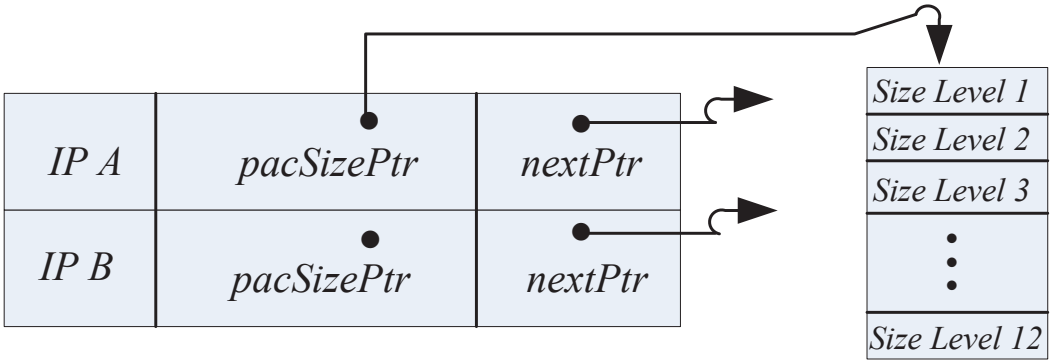


Figure 3.4: Data structure for building the traffic profile.

Besides the IP address used as the key for one entry, it also contains $pacSizePtr$ which is a pointer to an array with 12 elements, each of which maintains the number of packets that fall into the corresponding level as described in Table 3.1. The traffic profile is refreshed and built every time period T_{macro} in order to minimize memory consumption. That is to say, the hash table only maintains those statistics during a period T_{macro}.

3.4.2 Macro-level Anomaly Detector

In our scheme, the MAD module runs in an independent thread. It periodically scans the traffic profile built by the TPC module and will activate the MCA module whenever there is an anomaly event detected. To be more specific, it will report suspected victims to the MCA module for further flow-level filtering. The macro-level anomaly indicator can be utilized as the criterion for determining suspected victims. Simply, a given address A can be treated as a suspected victim whenever the following condition is satisfied.

$$H(A) < TH_H \tag{3.3}$$

Where TH_H is a pre-defined threshold that can be determined according to the practical environment.

3.4.3 Micro-level Credit Accumulation

The MCA module is only active when there is an anomaly detected by the MAD module. Those flows related to a suspected victim will be examined by the MCA module. It keeps building the credit for each flow according to equation 3.2. The flows are maintained in another hash table, each entry of which contains a 2-tuple IP pair as the key, the time stamp of the corresponding new packet, and the credit value accumulated. In addition to the credit accumulation based on equation 3.2, we evenly divide the overall credit range $[LOW, HIGH]$ into n scales. Flows in higher scales should be legitimate traffic while those in the lower scales are potentially malicious. All the flows are initially set to be in the lowest scale. In order to reduce the memory consumption,

whenever the difference between the current time and the time stamp of one flow is larger than T_{exp}, then the corresponding entry will be removed during a periodic scan of the hash table.

3.4.4 Probabilistic Drop Filter

A probabilistic drop scheme is adopted in our system, and only those packets targeting suspected victims will be filtered by the PDF module. For a specific flow, the flow's credit falls into the k^{th} interval as per:
$\left[LOW + k \cdot \frac{HIGH-LOW}{n}, LOW + (k+1) \cdot \frac{HIGH-LOW}{n}\right]$. The probability that a packet belonging to that flow will be dropped as determined by: $e^{-\beta k}$, where $0 \leq k \leq n-1$ and β is the probabilistic drop factor. Thus, flows with a low-level k will demonstrate a high probability of drop while flows with intermediate or high-level k will see limited drop.

3.4.5 Space Requirement

The primary space requirement for the system is due to the hash tables employed in the TPC and MCA modules. Let NUM_{ip} denote the number of distinct IP addresses during the hash table construction period T_{macro}. For each victim identified we must further maintain an additional hash table. Denote NUM_{flow} the average number of flows that are associated with the suspected victims detected by the MAD module. Let L_{ip} represent the length (in bytes) of each record in the TPC hash table and L_{flow} denote the same in the MCA hash table. Thus, the total memory requirement is $NUM_{\text{ip}} \cdot L_{\text{ip}} + NUM_{\text{flow}} \cdot L_{\text{flow}}$. One additional benefit by adopting the hash table is that the construction time of both the macro-level traffic profile and the micro-level credit is quite small.

3.5 Evaluation

In this section, we evaluate the performance of the proposed *TrustGuard* scheme via simulation. As we pointed out earlier, previous defense systems cannot efficiently tackle both the true IP and spoofed IP DDoS attacks. We synthetically generate traffic that can be launched from true IP addresses to evaluate our system defense against both the true and spoofed IP attacks. In order not to inject bias into our experiments, the existing Internet traces CAITD and CDAD will be utilized as much as possible and synthetically generated traffic is only employed when necessary. We chose uni-directional traffic for a 10-minute time period from the CAITD trace as our background traffic. The choice of a 10-minute interval stemmed from the fact that the CAITD has an average rate of 617594 packets per second making larger evaluations resource prohibitive. We further used only uni-directional traffic from the CDAD attack traffic in order to maintain a fair test. This traffic maintained an average rate of 79202 packets per second. Most of the traffic in the CDAD contains a single packet per flow and thus this data provided the model for spoofed IP DDoS attacks. The most significant difference we have observed between spoofed IP and true IP DDoS attacks is in the number of packets per flow. We synthetically generate true IP traffic traces in the following manner: (i) The average rate is 60000 packets per second. (ii) The source IP addresses are randomly selected from 1000 pre-defined source IP addresses. Thus, the resultant traffic will, on average, contain 60000 packets with roughly 60 packets per unique source IP. (iii) Finally, we randomly choose packet sizes to be between 40 to 50 Bytes. We set $T_{\text{macro}} = 1s$.

3.5.1 Macro-level Detection Performance

Accurate detection of suspected victims directly affects the accuracy of our approach overall as non-victims are not investigated. We inject attack traffic into the CAITD traffic in order to evaluate the detection accuracy of the MAD module. A 60 second segment of the CDAD trace is injected into the CAITD traffic at an offset of 100 seconds. Further, we synthetically generated 60 seconds of true IP DDoS attack traffic, as explained above, and insert that into the CAITD traffic at an offset of 300 seconds. Fig. 3.5 illustrates the experimental results. The minimal H series contain

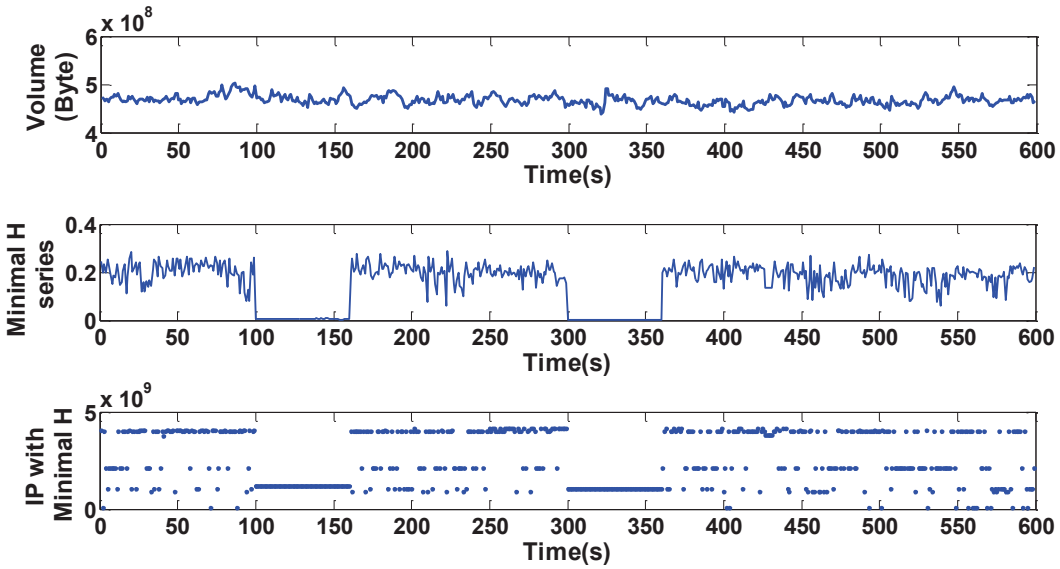


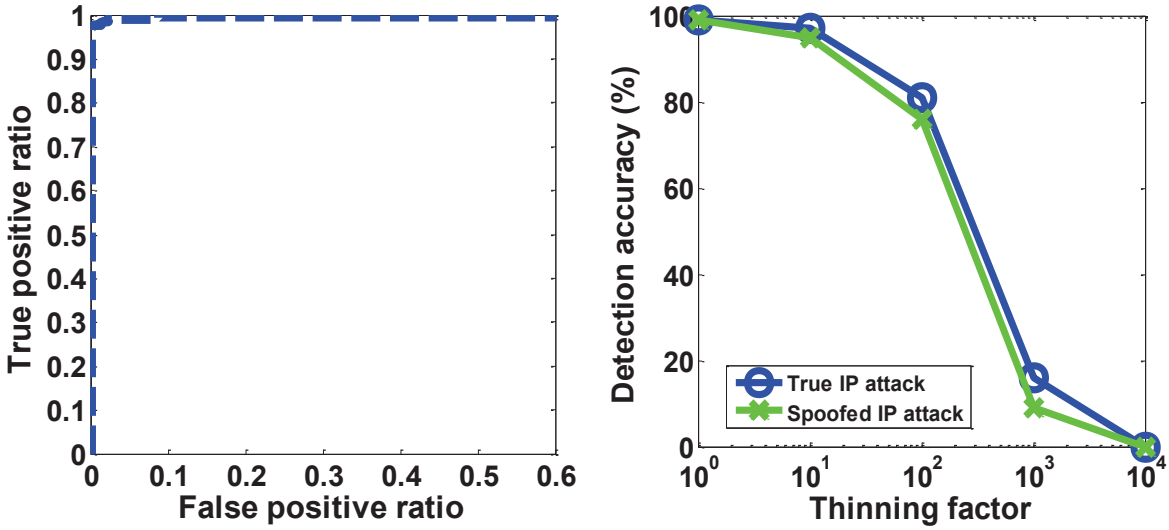
Figure 3.5: Evaluation of MAD performance.

the minimal H value amongst all the destination IP addresses for one sampling period. The IP with minimal H identifies the IP address with the lowest score for a sampling period. We can see that either the spoofed IP attack or true IP attack can result in significantly smaller values of H, which make it easy for the defense system to determine a threshold TH_H for detection. Furthermore, since those destination IP addresses can be pinpointed based on the threshold TH_H,

the countermeasures we take later can be targeted to defend the suspected victims.

The Receiver Operating Characteristic (ROC) curve of our method are depicted in Fig. 3.6(a). For a false positive rate of 1%, our system can correctly identifies over 97.68% DDoS attacks.

We also sought to determine the sensitivity of our approach to varying intensities of DDoS traffic. The attack traces are thinned by a thinning factor N such that we select 1 out of every N packets from the original attack traffic and otherwise repeat the above experiments. We chose a threshold, TH_H, of 0.1 based on the results from Fig. 3.6(a) where 0.1 marks the typical lowest level of benign traffic as well as the highest level of those malicious. We define the Detection



(a) ROC curve

(b) Sensitivity

Figure 3.6: Evaluation of detection performance.

Accuracy Ratio (DAR) as:

$$DAR = 1 - (FP + FN) \tag{3.4}$$

Where FP denotes the False Positive ratio and the FN means the False Negative ratio. Fig. 3.6(b) shows the experimental results. When the thinning factor is less than 10, the MAD module per-

forms well and can achieve over 95% DAR in both true and spoofed IP attack cases. When the thinning factor is larger than 100, which means the average rate per second of attack traffic is smaller than 0.1% of the background traffic, the detection performance rapidly degrades.

3.5.2 Micro-Level Filtering Performance

We also evaluate the micro-level filtering performance of our system. We define two metrics called the Legitimate Survival Ratio (LSR) and the Malicious Survival Ratio (MSR) to measure the performance, which are defined as follows.

$$\begin{cases} LSR = \frac{NUM_{passNormal}}{NUM_{totalNormal}} \\ MSR = \frac{NUM_{passAnomaly}}{NUM_{totalAnomaly}} \end{cases} \tag{3.5}$$

Where $NUM_{passNormal}$ and $NUM_{passAnomaly}$ are the number of legitimate and anomalous packets that successfully passed through the system (i.e. not dropped) and $NUM_{totalNormal}$ and $NUM_{totalAnomaly}$ are the total number of legitimate and anomalous packets. To evaluate the flow-level filtering performance, we chose a set of legitimate flows that are associated with a single destination from the CAITD trace and then change that destination to one that is under DDoS attack in the CDAD trace. After that, we combine the modified trace with the CDAD trace and evaluate it. The default settings for the parameters we adopted here were $T_{exp} = 10min$, $LOW = 0$, $HIGH = 5$, $\alpha = 2$, $\beta = 1$ and $n = 5$. Fig. 3.7 shows the experimental results. We can see that most normal packets (over 98%) pass through our system unmolested, while most malicious flows are filtered by our system due to the low credit they accumulate. All the first packets of flows associated with a suspected victim detected by the MAD module will be dropped during the initial phase. However, as

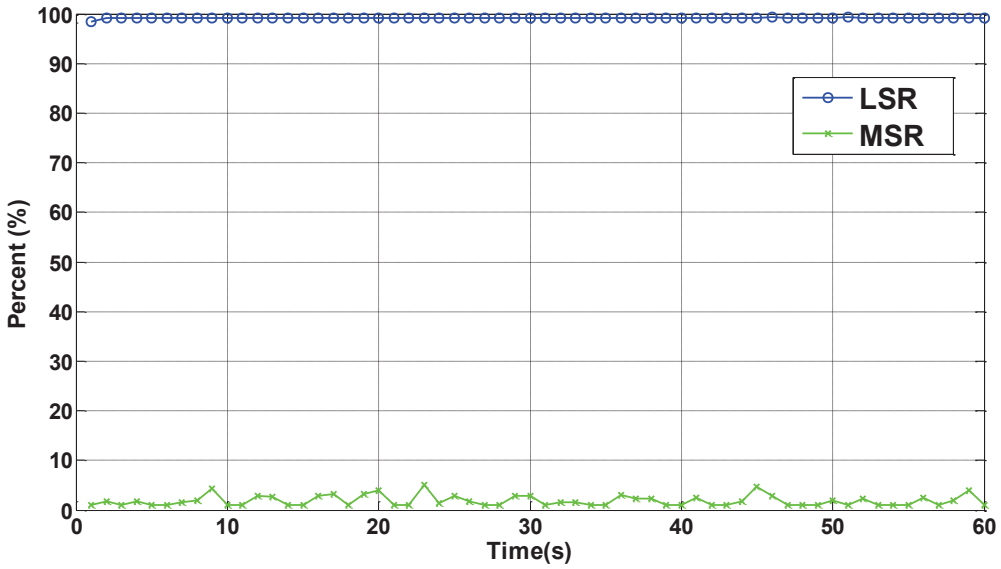


Figure 3.7: Evaluation of micro-level filtering

flows begin to move data then they have a greatly reduced chance of drop. Further, since the MAD module targets only flows impacting suspected victims then most traffic encountered is ignored by our system.

3.6 Conclusion

In this chapter, we present *TrustGuard* to counter the threat of DDoS attack. Our approach employs a two-tier model to reduce the size of the search space and make identification of specific attackers and victims possible. Our macro-level detector can accurately identify suspected victims and the micro-level detector can then confirm and refine those suspicions. We believe that this approach can accurately identify DDoS attacks down to the instigating flows and that this information can be used to improve firewall and IDS rules.

CHAPTER FOUR

A SCALABLE DDOS DETECTION FRAMEWORK WITH VICTIM PINPOINT CAPABILITY

4.1 Motivation of this work

Although TrushGuard was proposed to point out which flows belong to attacks, it cannot scale well with the large amount of traffic since it requires maintaining per-flow state to accumulate credits for each flow. In order to address this problem, we further proposed a sketch-based framework to scale with the high-speed traffic while it still provides victim pinpoint capability. The victim pinpoint capability is critical for react to the attacks at an early stage.

4.2 Related works

Over the past decades, many intrusion detection systems (IDSs) have been proposed to fight against DDoS attacks. However, those existing schemes usually present a tradeoff between scalability and accuracy. That is to say, finer grained traffic monitoring can ensure the accuracy of detection while does not scale well. For example, two most popular open-source IDSs, Snort and Bro [53, 54], keep per-flow state to detect anomalies, which makes both of them not scale well in a high-speed network. Since the volume of the Internet traffic doubles every year, how to monitor a large amount of traffic in real-time is becoming more crucial in anomaly detection. Although dimensionality reduction proposed in [55,56] may be effective in dealing with such a large data, it usually requires

complex operations, and thus is impractical in real-time detection.

Recently, a series of sketch-based approaches have been proposed for anomaly detection [57–61]. Sketch [62] is a data structure to store a summary of a large data set for space efficiency. Kompella et al. presented Partial Completion Filters (PCF) by utilizing multiple hash tables for scalable attack detection in high-speed networks [57]. As we will see later, such scheme is essentially an simplified version of the original sketch [62]. The main drawback of their scheme is that it can only tell when an attack happens without providing any hint on where the anomaly occurs; the latter is critical in mitigating the attack at an early stage. Identifying victims is also useful in responding to attacks. For instance, an IDS can generate packet classification rules automatically based on the victim information, so as to minimize future damage. In order to provide sketches with the capability that can tell those keys with heavy change, a reversible sketch framework is proposed by [58, 59]. Such feature can be used to provide the victim pinpoint capability in DDoS detection. An improved reversible sketch is proposed by [60] in DDoS detection context. They proposed a flooding attack detection method using a count-min sketch (CMS) with multi-channel nonparametric CUSUM (MNP-CUSUM) [60]. The CUSUM [39] is a change-point detection technique that can accumulate those small offsets during the process to amplify a varying statistical feature so as to improve the detection sensitivity. Although their improved scheme can detect flooding events effectively, it suffers from the following shortcomings which render it still insufficient to detect general DDoS attacks effectively. First of all, their scheme only takes the high frequency of packets in a flow as the evidence of an anomaly event. However, this feature of traffic alone is not enough to detect an anomaly. For example, a Flash Crowd event, which is caused by a

large number of legitimate users simultaneously accessing the same server during historical events, can also result in an outburst of the traffic. Their scheme will lead to a large false positive in such a case. Moreover, their scheme suffers from the scalability problem. Because of the key recovery issue in the original sketch scheme they exploited, their method has to record every incoming destination IP (DIP) for the key recovery later, which makes it unscalable to a large amount of traffic due to the huge memory consumption. Finally, it applies a multi-channel CUSUM algorithm to every bucket in the sketch, which requires heavy computations in high speed networks. In [63], researchers proposed a discrimination algorithm using the flow correlation coefficient to distinguish flash crowd events from DDoS attacks, however, their method requires complex computing work and introduces extra overhead which make the method unsuitable for real-time processing.

4.3 System Description

In this work [64,65], we propose a two-level approach for DDoS detection. We are motivated by the fact that a typical DDoS attack traffic possesses three characteristics: high frequency of incoming packets, asymmetry in interaction patterns, and high diversity of source IP (SIP) addresses. Our modified count-min sketch (MCS), bidirectional count sketch (BCS), and distinct IP addresses estimator are designed precisely for detecting these three characteristics. Although sketch is also used in our work to achieve high scalability, the differences between the previous sketch-based detection approaches and ours lie in the following ways:

Memory consumption Our scheme outperforms previous works in terms of the space requirement. By utilizing the two-level model, most of benign traffic information, which may be consid-

ered as a redundancy for IDS systems, does not need to be recorded in the system. Moreover, the traditional key recovery process in sketch [59] requires sketches to record every input keys, which will consume much space, especially when a large number of DIPs are involved in high-speed networks. By taking advantage of the high diversity of source IP addresses feature, our approach can reveal the victim set without recording every destination IPs as previous works do.

Searching time Our scheme also can achieve faster detection than previous sketch-based approaches in two aspects. Firstly, since most of traffic is benign, the adopted two-level scheme can greatly reduce the search space while the sketch adopted in [60] processes different kinds of traffic equally. Secondly, rather than applying CUSUM to multiple channels of each bucket in the high frequency anomaly detection phase, we utilize a light-weight exponentially weighted moving average (EWMA) technique to achieve the same goal while introducing much less computing overhead.

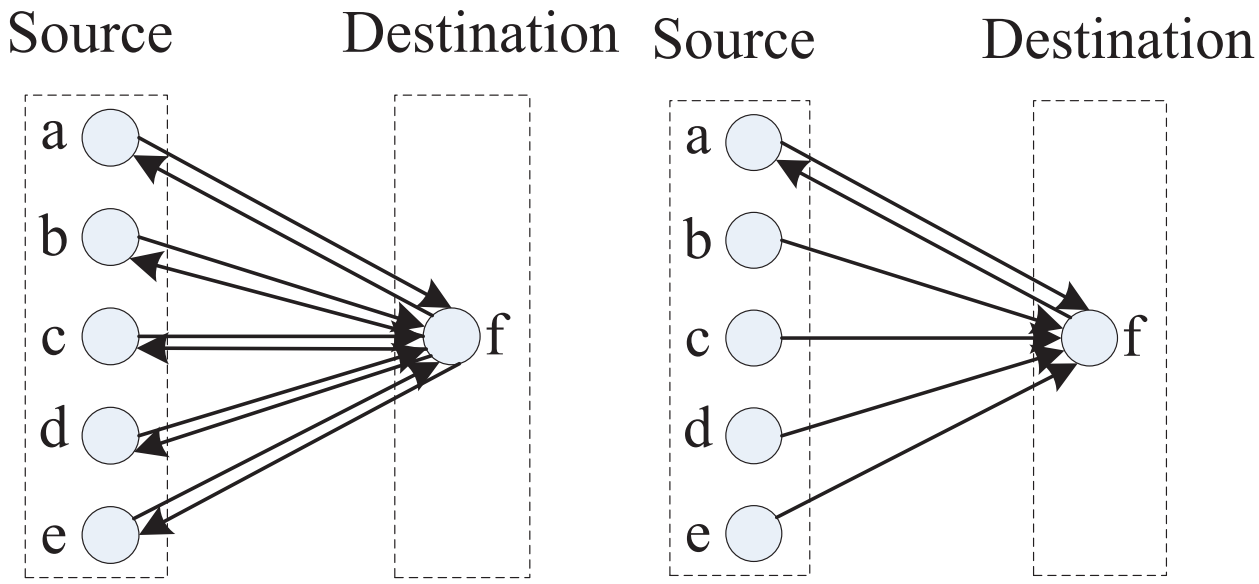
Accuracy By taking the asymmetry feature into accounts, our approach can greatly reduce the false positives by distinguishing between Flash Crowds and DDoS attacks, and thus can improve the overall accuracy.

4.3.1 Measurement on Real Traces

The effectiveness of our scheme is based on the assumption that a typical DDoS attack traffic possesses three characteristics: high frequency of incoming packets, asymmetry in the interaction patterns, and high diversity of source IP addresses.

Fig. 4.1 demonstrates the "asymmetry" in typical DDoS attacks for web services. We pay attention to the fundamental difference in flow patterns between DDoS attacks and normal traffic. In order to exhaust resources at the server side, an attacker (or more likely a large number of "puppets" in his or her botnet) tries to generate as many requests as possible. When the number of requests exceeds the capacity of the server, we observe fewer responses from the server than requests it receives. We notice that it is not always true that an IP address will serve as both source and destination of traffic, especially when the UDP or ICMP protocol is involved. Thus, in this work we only refer to the TCP flooding attack by default. Let $N_{\text{forward}}(i)$ denote the number of flows from other nodes to a server i, and $N_{\text{backward}}(i)$ the number of those flows that originate from the server i. By "flow," we mean a group of packets with the same pair of source IP (SIP) and destination IP (DIP) addresses. We do not consider the port information in the flow because we only consider the node-level interactions in our scheme. We define the asymmetry index $AI(i) = |N_{\text{forward}}(i) - N_{\text{backward}}(i)|$ for the server i. We expect that the server under a DDoS attack will exhibit a higher AI value than the one experiencing no attack. For example, Fig. 4.1(a) shows normal interactions between clients and a server, and $AI(f)$ is 0. On the other hand, in an attack scenario depicted in Fig. 4.1(b), $AI(f)$ is 4.

In order to support the above assumption, we sought to evaluate a set of public traces. We derived Internet traffic from Auckland University (AU) [66] and Washington State University (WSU) as the normal traffic and the attack traffic from "The CAIDA DDoS Attack 2007 Dataset" (CDAD) [46]. The traces are labeled AU-0, AU-1, WSU-0, WSU-1, CDAD-0, CDAD-1, where the numbers with each label are corresponding to different time periods from each trace. The default



(a) Normal (b) Attack

Figure 4.1: Flow patterns of normal traffic and DDoS attacks

time interval for the measurements is 10 minutes. We firstly measure the average number of the incoming packets per DIP in each trace for comparison. Fig. 4.2 shows the results. We notice that in Fig. 4.2 the average numbers of the incoming packets per DIP of CDAD-0 and CDAD-1 are both much larger than that of other traces (nearly 100 times larger). We also measure the average number of flows per DIP and the average traffic volume per DIP over every trace, and we get the similar results that the numbers measured from CDAD are much higher than that of other traces. Thus, rather than using all the three metrics (packet, flow, volume), we only utilize the high frequency feature of incoming packets as the anomaly indicator during the coarse-level detection since these three metrics are highly correlated with each other in our measurement. We notice that some attacks might not possess such correlations. For example, alpha flows [67] usually result in high volume of traffic while only have a small number of flows per DIP or even low-rate

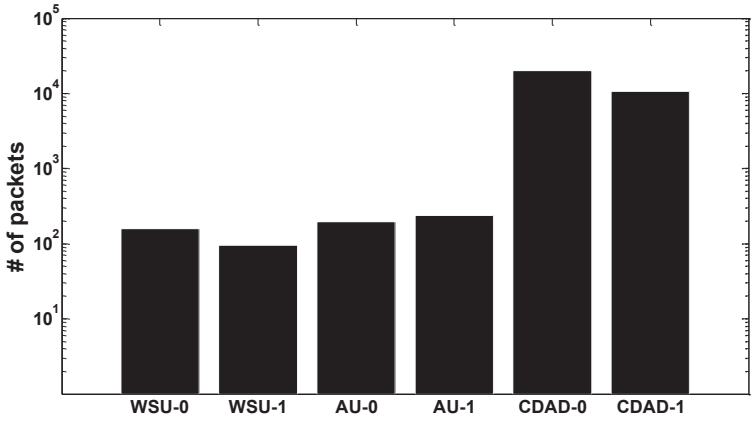


Figure 4.2: Average # of incoming packets per DIP

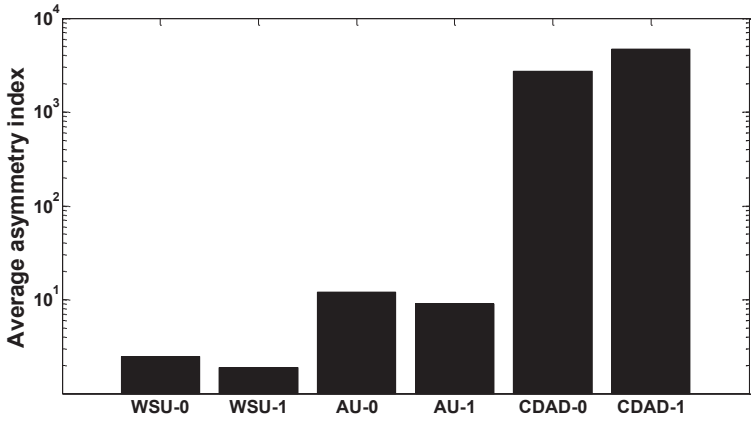


Figure 4.3: Average AI per DIP

DDoS attacks [68] can have low value of all the three metrics. In this work, we do not aim at the development of a panacea, which is very hard if not impossible, for all kinds of attacks. Instead, we only focus on the detection of the general flooding attack in high speed networks. Other types of attacks need to be further filtered by extra works. Regarding the asymmetry features, we measure the asymmetry index of DIPs of each traces, the result of which is shown in Fig. 4.3. We can see that the AI value obtained from CDAD traces is greatly larger than that of other traces. Similar results can be seen in Fig. 4.4 after we measure the number of distinct IPs that are associated with each DIP in all the traces.

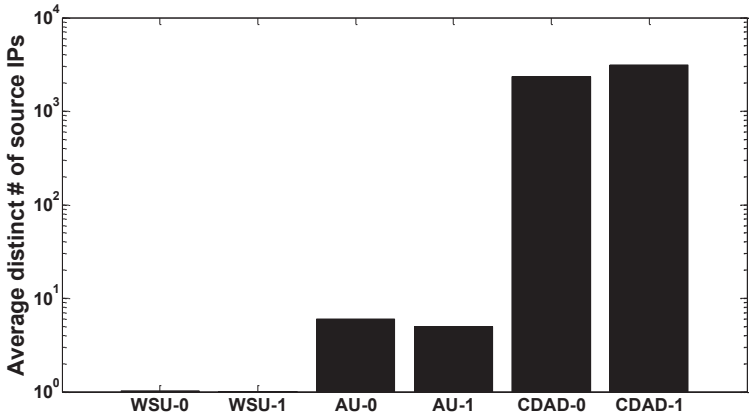


Figure 4.4: Average distinct # of source IPs

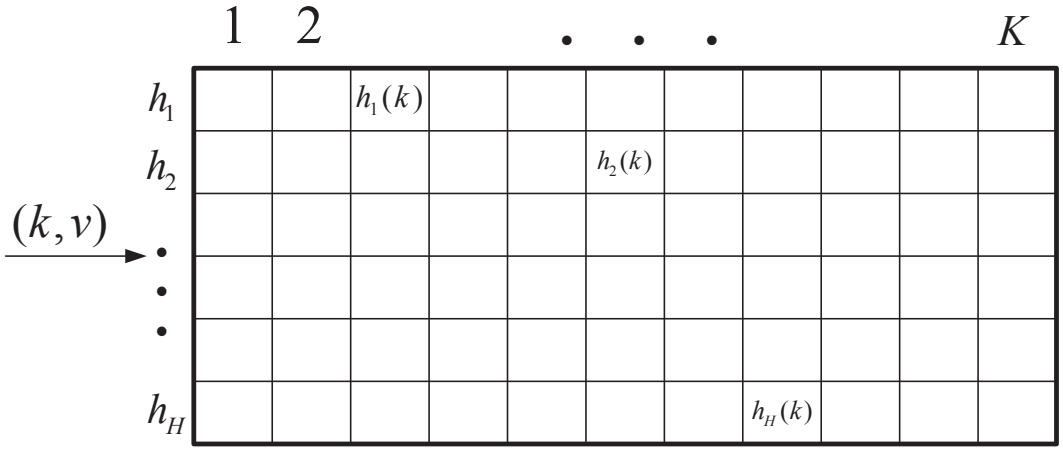


Figure 4.5: Illustration of sketch data structure

4.3.2 Data Structure

K-ary sketch is a data structure to efficiently and accurately estimate the original signals by aggregating high dimensional data streams into fewer dimensions. As shown in Fig. 4.5, it consists of H hash tables of size K. A hash function for each row is selected independently and randomly from a set of hash functions. Each data item contains a key k_i and an associated value v_i. When a new item $s_i = (k_i, v_i)$ arrives, its value v_i is added to those buckets corresponding to the key k_i. The $CMS\_Query(key)$ function can return the minimum value among all the buckets corresponding to

a specific key. In case of hash collisions, the colliding keys will be listed in the bucket for the key recovery purpose later. The key recovery process [59] can reveal those keys with high frequency in the sketch by looking into the intersection set of high value buckets across the whole sketch. The recovery process is crucial in tracking victims, which will greatly benefit in responding to attacks. Our proposed approach makes two important changes to this original sketch structure: modified count-min sketch (MCS) and bidirectional count sketch (BCS), which will be introduced in the next section.

4.3.3 System Architecture

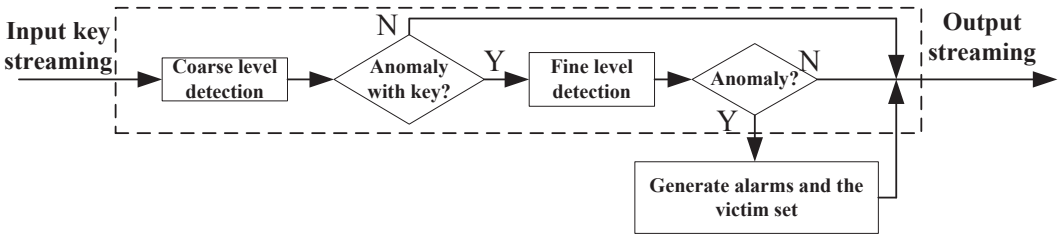


Figure 4.6: High-level view of detection process

The overall framework of our detection system is shown in Fig. 4.6. During each detection period, the related information of every incoming packet is inserted into MCS for the coarse-level detection. We use DIP as the key, and the number of packets that are destined for that IP address as the associated value. MCS only maintains counters for input IP addresses. Compared with the original CMS structure, our MCS structure utilizes space more efficiently because no information on IP addresses themselves is stored in this structure. Besides, unlike CMS, MCS does not rely on CUSUM; MCS is used only for coarse-grained filtering and a light-weight EWMA technique is applied to each bucket to determine whether to generate an alarm for this bucket or not. Whenever

an incoming packet satisfies the condition that every bucket it hashed into has an alarm signal, it will trigger the second stage for finer-grained detection, where a new structure called BCS is used.

In BCS, those suspicious flows detected in the coarse-level detection in both directions are mapped into buckets. While the general sketch structure in which a value in each bucket can increase only, a bucket value in BCS may increase or decrease. We will demonstrate how we apply BCS to exploiting the asymmetry of the attack traffic the following sections. Once an attack is detected, we use a light-weight distinct IP addresses estimator to pinpoint victims that have the most number of distinct sources, which is a strong indication of DDoS attacks.

4.4 Scheme Design and Implementation

In this section, we describe the detection processes of both coarse and fine level in details and then explain how the proposed distinct sources estimator works and why it can help to indicate DDoS attacks. Finally, a SRAM-based parallel architecture is proposed to achieve high-speed process.

4.4.1 Coarse-level Detection

In our scheme, each bucket in the MCS contains four values $(v_t, v_{t-\Delta t}, v_{\text{backup}}, Flag)$. v_t is the number of packets that are accumulated from $t - \Delta t$ to t, $v_{t-\Delta t}$ is the previous value of v_t, and v_{backup} is the value of v_t right before the alarm occurs (or null if there has been no alarm). $Flag$ is set to 1 whenever the alarm condition is satisfied; otherwise it is set to 0. Here, the definition of alarm conditions depends on the practical deployment, and we will further explain it when we describe Algorithm 1 below. For each incoming record, we update the sketch with $(k_i, 1)$ where k_i is the DIP and 1 represents the number of this incoming record. In our MCS, rather than returning

the minimum value of v_t as the original sketch does, the CMS_Query function in MCS returns the minimum value of $Flag$ among all the buckets corresponding to a specific DIP to indicate whether it is under flood attacks. As we can see, the sketch adopted here only requires $O(H \times K)$ cells, which is constant.

The main purpose of MCS is to detect items with abnormal frequency at the coarse level. It is the first stage in the system, which every packet must go through. When a new packet arrives, hash values of H hash functions are computed, and the corresponding buckets are updated; the value in each bucket is incremented by 1. This accumulation process repeats every Δt seconds. The alarm condition is tested for all $H \times K$ buckets periodically. If the alarm condition is satisfied, then the alarm flag associated with the bucket is set to 1. Whenever there is an alarm, the previous v value of the bucket is recorded in the v_{backup} for determining whether the raised alarm is terminated or not.

We use an EWMA technique to decide whether there is an anomaly in each bucket, as shown in Algorithm 1. For each bucket, if the bucket status is normal, then we estimate v_t with an EWMA parameter α. Whenever $v_t \geq (1+\theta)\overline{v_{t-\Delta t}}$, which is considered as the satisfaction of the alarm condition, an alarm is raised. θ is the parameter that represents the percentage above the estimated value that can be considered to be an indication of anomalous pattern. The procedure is different after an alarm was raised. In order to estimate when the generated alarm should be terminated, we need to compare the current value with the specific value right before the time that the alarm happened. Such specific value is recorded in v_{backup} before the alarm is generated. Also, rather than using the previous value $\overline{v_{t-\Delta t}}$, we estimate the $\overline{v_t}$ by v_{backup} in order to eliminate the

Algorithm 1: Adjustment procedure of $Flag$

for $k = 1, h = 1$ **to** K, H **do**
  if $Flag = 0$ **then**
    $\overline{v_t} \leftarrow (1-\alpha)\overline{v_{t-\Delta t}} + \alpha v_t$;
    if $v_t \geq (1+\theta)\overline{v_{t-\Delta t}}$ **then**
      $Flag \leftarrow 1$;
      $v_{\text{backup}} \leftarrow \overline{v_{t-\Delta t}}$;
    end
  else
    $\overline{v_t} \leftarrow (1-\alpha)v_{\text{backup}} + \alpha v_t$;
    if $v_t < (1+\theta)v_{\text{backup}}$ **then**
      $Flag \leftarrow 0$;
    end
  end
end

impact of the anomaly on the next following $\overline{v_t}$ series.

We can do the coarse-level detection by querying the minimal value of alarm flag for a specific key. If $CMS\_Query(key) = 1$, then there may be an anomaly associated with the key. However, the coarse-level detection would yield a certain number of false positives. There are two possible reasons for false positives. The first possibility is hash collisions, which can be reduced by carefully selecting hash functions or enlarging the size of the sketch. The second possibility is flash crowds. They can also yield many items with high frequencies in the sketch. Thus, we need to examine traffic further to separate these possibilities from true attacks, which is the goal of our next technique, BCS, which detects anomalies at the finer level.

Algorithm 2: BCS update procedure in the forward direction

```
for h = 1 to H do
    k ← BCS[h].hash(DIP) ;
    if DIP is not in BCS[h][k].list then
        insert DIP into BCS[h][k].list ;
        update BCS[h].BF by DIP|SIP ;
        BCS[h][k].counter++ ;
    else
        if DIP|SIP is not in BCS[h].BF then
            update BCS[h].BF by DIP|SIP ;
            BCS[h][k].counter++ ;
        end
    end
    DistinctSourcesEstimator(SIP) ;
end
```

4.4.2 Fine-level Detection

The objective of the fine-level detection is to find out those DIPs exhibiting high asymmetric communication patterns. A successful DDoS attack employs a large number of zombies to exhaust resources of the target side. However, they are usually unaware of the exact capacity of the server. Therefore, to guarantee to overwhelm the server, an attacker sends as much traffic as allowed, exceeding the server's capacity. This results in highly asymmetric communication patterns between clients and the server, as shown in Fig. 4.1(b). There are two reasons for such an asymmetry pattern. First, the capacity of the server, namely the victim, to respond is limited while the attacker can keep launching new connections. Second, the SIP addresses of attack traffic are forged, and thus the server has to abort communications. During some time interval ΔT, for a specific IP address, we can detect whether this address serves as both source and destination or not. If yes,

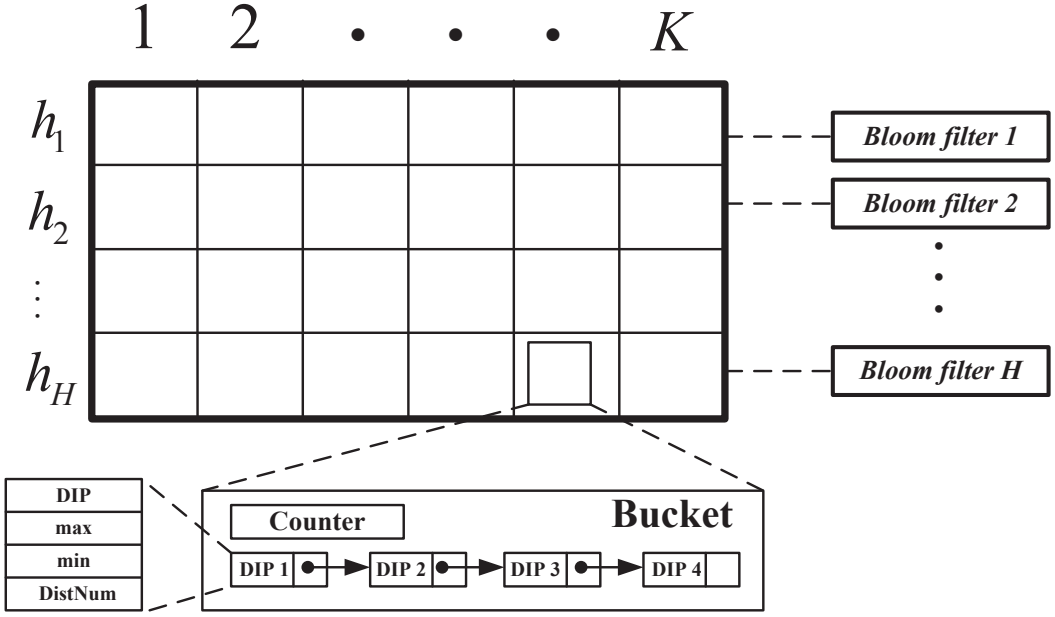


Figure 4.7: Illustration of BCS data structure

then the corresponding flow (the source and destination IP pair containing this IP address) can be considered as a normal flow. Motivated by this observation, we propose the BCS structure to monitor such kind of anomaly with fine granularity.

During one time interval, we use DIP as the key in updating the BCS structure. An illustration of BCS sketch is shown in Fig. 4.7. All the keys with hash collisions will be stored as a list in the corresponding bucket. Rather than incrementing corresponding H counters by 1 every time a new packet arrives, the counters increase only when the DIP belongs to a new flow. For example, a flow (s_i, d_i), where s_i denotes the SIP address of node i and d_i is the DIP address of node i, will contribute to the corresponding H buckets only once during a single period. On the other hand, (s_j, d_i), which is another flow with the same destination d_i, will contribute another 1 to the buckets that the key d_i is hashed into. Since we do not need to record SIP addresses in the sketch, we employ H bloom filters (BF) with m bits and k_{bf} hash functions as an ancillary structure to

estimate whether a specific flow that new packets belong to has been inserted to the BCS structure or not. Algorithm 2 presents the details of how BCS works on the forward direction of traffic. We use | as the string concatenation operator.

Algorithm 3: BCS update procedure of the backward direction

```
1 for h = 1 to H do
2 |   k ← BCS[h].hash(SIP) ;
3 |   if SIP is in BCS[h][k].list then
4 |   |   if SIP|DIP is in BCS[h].BF then
5 |   |   |   BCS[h][k].counter − − ;
6 |   |   end
7 |   end
8 end
```

The procedure for the backward traffic is shown in Algorithm 3. Whenever we find a backward flow that can be paired with an existing forward flow in the BCS structure, the corresponding counter decreases. In this way, the counters with anomalous high values indicate an anomaly event caused by asymmetric communication patterns for a specific victim.

4.4.3 Distinct Sources Estimator

In order to avoid being detected, attackers may employ a large number of SIP addresses. In such cases, those DIPs that are associated with the largest distinct SIP addresses should be a good candidate for a victim under attack. Thus, how to find the number of distinct SIPs for a victim is crucial in the DDoS defense. Without recording the SIP addresses in the system, which requires too much memory, we need to find a way to estimate this number. For each DIP that is hashed into BCS, we pick a hash function $h\colon \mathcal{N} \rightarrow [0, 1]$ which maps every number into $[0, 1]$, and then

we apply $h(\cdot)$ to all the SIP addresses that are associated with this DIP, and maintains the maximal value max and minimal value min, and then the number of distinct IP addresses, $DistNum$, can be estimated as $\frac{1}{2} \cdot \left(\frac{1}{min} + \frac{1}{1-max}\right)$. If the hash function that we choose is sufficiently random, then the above formula is a sufficiently good estimator for our purpose. In this way, each DIP which has been hashed into the BCS will be associated with a number: $DistNum$. For a specific DIP, this number $DistNum$ can be used as an indicator on how diverse the corresponding SIPs are.

4.4.4 Victims Identification

At the end of each time interval, for each row in the BCS, we compute the average counter value $\overline{C[h]}$ and the corresponding mean square deviation $D[h]$. For a specific bucket, whenever its counter value $BCS[h][k].counter$ satisfies the following condition, then it raises an alarm for an anomaly:

$$BCS[h][k].counter - \overline{C[h]} \geq \beta \cdot D[h] \tag{4.1}$$

where β is an adjustment factor that should be empirically determined. Then, we merge those DIPs that correspond to those anomalous buckets together, and sort them by their $DistNum$. In addition, we eliminate those DIPs that satisfy the condition $BCS\_Query(DIP) < TH_{counter}$ in the merged set, where BCS_Query is similar to the original CMS_Query, which returns the minimum counter value through all the hashed buckets in the sketch BCS. $TH_{counter}$ is a threshold which can be empirically determined. Finally, those victims can be chosen from the merged DIP set by picking the top few DIPs with the largest $DistNum$ value. Or, we can set a threshold $TH_{DistNum}$ to select those victims that can satisfy $DistNum \geq TH_{DistNum}$ to process the selection of victims.

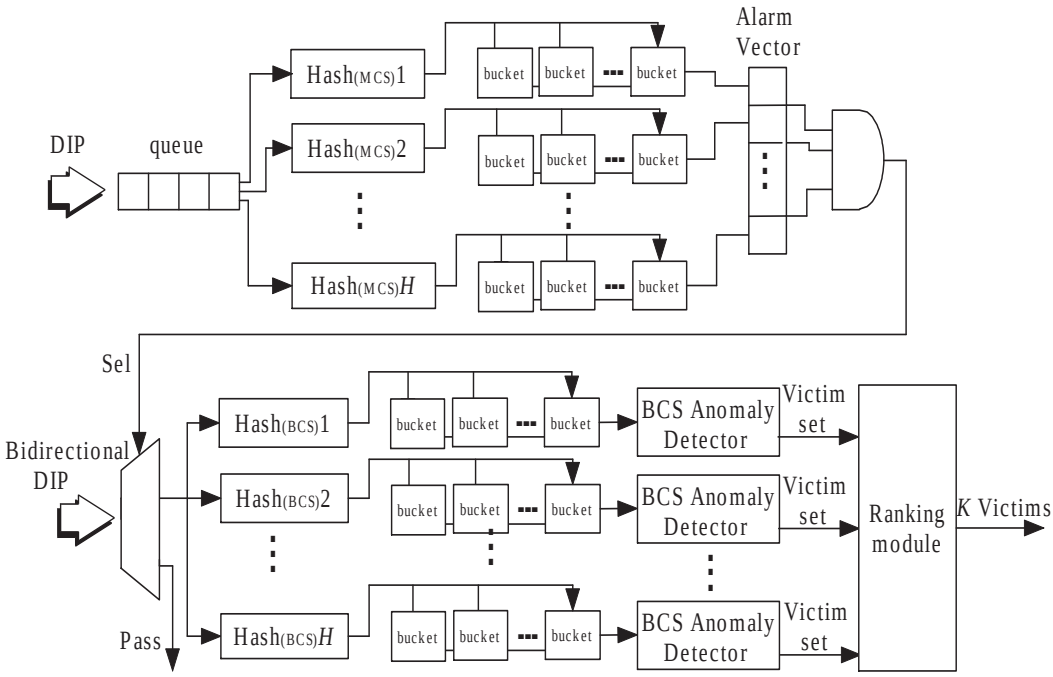


Figure 4.8: Hardware architecture for the proposed scheme

4.4.5 Hardware Architecture

Our proposed scheme can be implemented by hardware to achieve high speed process. Since the field programmable gate array (FPGA) technology has widely been utilized for real-time packet processing due to its capability of reconfigure and parallelism, we propose a SRAM-based parallel architecture as shown in Fig. 4.8. For each input DIP of incoming packets, we perform the hash computations over the H_{MCS} branches in parallel. The Carter-Wegman H3 hash function [69] can be utilized in our hardware-based scheme, since the H3 hash function mostly consists of XOR gates proportional to the number of output bits, which can make it easily implemented in hardware. The bit values in the vector are initially set to 0 and they will be periodically reset to 0 at the end of the detection interval. For each row at the first stage, whenever an alarm signal is generated, the corresponding bit in the vector will be set to 1. After doing the AND operation among all the

bit values, it can decide whether to trigger the second stage detection or not. Similarly, the H_{BCS} branches during the finer level detection phase can also be executed in parallel. The BCS anomaly detector and ranking module can be implemented by FPGAs according to the flow logic discussed in the previous sections. The overall search process can be divided into several independent parts and it can be pipelined by assigning each part to a separate memory block to accelerate the overall processing speed. For example, the hash computation of the current incoming DIP and the anomaly detection of the previous DIP are independent with each other and thus can be mapped into two different stages.

4.5 Analysis and Discussion

In this section, we firstly analyze the space requirement and then estimate the accuracy of our scheme. We further demonstrate how to extend the current scheme to a collaborative detection framework.

4.5.1 Space Requirements

The primary space consumption of the system is due to the two sketches (MCS and BCS) and H bloom filters that are employed at the finer detection stage. Let L_{mc} denote the length (in bytes) of each bucket in the sketch MCS and L_{bc} represent the same in the sketch BCS. Moreover, each bloom filter will occupy L_{bf} space. Suppose the MCS and BCS have the size $H_{mc} \cdot K_{mc}$ and $H_{bc} \cdot K_{bc}$, respectively, the total memory requirement will be:

$$H_{mc} \cdot K_{mc} \cdot L_{mc} + H_{bc} \cdot K_{bc} \cdot L_{bc} + H_{bc} \cdot L_{bf} \tag{4.2}$$

According to our method proposed above, the length L_{bc} will not be a constant value because the length of the linked list varies. In the practical deployment, we are able to limit the maximal number of the nodes in the link list. For example, for a specific link list, we can only keep those top few DIPs that are associated with the highest $DistNum$ value in the list.

4.5.2 Accuracy Estimation Analysis

We further sought to quantify the impact of the size of sketches to the overall accuracy of our framework. We estimate the accuracy of our system in terms of "false positive rate" (FPR) and "false negative rate" (FNR). In order to simplify the problem, we conduct our analysis on an assumption that the "false negative rate" of both MCS FNR_{mc} and BCS FNR_{bc} are negligible and we will demonstrate why this assumption holds in our scheme below. At the presence of certain number of malicious flows, the overall "FPR" depend on the accuracy performance of individual modules (MCS and BCS). Thus, we firstly define false positive rate of MCS FPR_{mc} and BCS FPR_{bc}, and then demonstrate how they contribute to the overall false positive rate.

Since MCS and BCS are both proposed based on sketch, we firstly conduct a general analysis of the sketch structure. Let us assume there are m different malicious keys and n buckets in each row of a sketch. Since the probability that a specific bucket is not hashed by a malicious key is $1 - \frac{1}{n}$, the probability that a specific bucket is not hashed by every malicious key is $(1 - \frac{1}{n})^m$. Therefore, the probability that a bucket in a row is hashed by at least one malicious key is $1 - (1 - \frac{1}{n})^m$ and the expectation of the number of buckets to which these m malicious keys

hash is $n(1-(1-\frac{1}{n})^m)$. When m is much less than n, we have:

$$\begin{aligned} n(1-(1-\frac{1}{n})^m) &= n(1-(1-\frac{1}{n})^{-n\cdot-\frac{m}{n}}) \\ &\approx n(1-e^{-\frac{m}{n}}) \text{ when } n >> 1 \\ &= m \cdot \frac{1-e^{-\frac{m}{n}}}{\frac{m}{n}} \\ &= m \cdot \frac{1-[1+(-\frac{m}{n})+\frac{(-\frac{m}{n})^2}{2!}+\cdots]}{\frac{m}{n}} \\ &\approx m \ \ (\text{when } n >> m \geq 1) \end{aligned} \tag{4.3}$$

We define those buckets that are hashed by malicious keys as malicious buckets. Therefore, when $n >> m$, the number of the malicious keys can be used to estimate the expected number of malicious buckets in a row. For a key that is hashed into a malicious bucket in each row of sketch, whether it is benign or not, our scheme will judge it as a malicious key, which is the main cause of false positives of sketch scheme.

We assume that there are totally N distinct incoming keys, which contains $N \cdot P_{normal}$ normal keys, $N \cdot P_{flashCrowd}$ keys associated with "Flash Crowd" events and $N \cdot P_{DDoS}$ keys with DDoS events. P_{normal}, $P_{flashCrowd}$ and P_{DDoS} are the proportion of normal keys, keys with "Flash Crowd" and keys with DDoS to the total number of different keys, respectively. Based on the above definition, we have $P_{normal} + P_{flashCrowd} + P_{DDoS} = 1$. For the analysis, we call those keys that are related to "Flash Crowd" events as "Flashcrowd" keys and keys associated with DDoS events as DDoS keys.

In the coarse level detection, both "Flashcrowd" and DDoS keys are considered to be positive (malicious) instances. Therefore, based on the Eq. 4.3, the probability that a key is hashed into

one of these malicious buckets in one row is given by $\frac{N \cdot (1-P_{normal})}{K_{mc}}$. For H_{mc} rows, the probability is:

$$FPR_{mc} = (\frac{N \cdot (1 - P_{normal})}{K_{mc}})^{H_{mc}} \tag{4.4}$$

For the fine level detection, only DDoS keys are considered to be positive (malicious) instances. Similarly, the probability that a key is hashed into one of these malicious buckets for each row in BCS is given by:

$$FPR_{bc} = (\frac{N \cdot P_{DDoS}}{K_{bc}})^{H_{bc}} \tag{4.5}$$

We define the overall false positive rate by the definition:

$$FPR_{overall} = \frac{\text{Total \# of false positive instances}}{\text{Total \# of negative instances}} \tag{4.6}$$

For the overall scheme, negative instances contains normal keys and "Flashcrowd" keys and false positives mean those negative instances that are wrongly judged as DDoS keys. Thus, we have:

$$\begin{aligned} & FPR_{overall} \\ & = \frac{\text{Total \# of false positive instances}}{\text{Total \# of negative instances}} \\ & = \frac{NP_{normal}FPR_{mc}FPR_{bc} + NP_{flashCrowd}FPR_{bc}}{N(P_{flashCrowd} + P_{normal})} \\ & = \frac{P_{normal}FPR_{mc}FPR_{bc} + P_{flashCrowd}FPR_{bc}}{P_{flashCrowd} + P_{normal}} \end{aligned} \tag{4.7}$$

From the Eq. 4.7, we can see that the overall false positive rate depends on the distribution of traffic and false positive rate of each individual module. We can estimate the overall false positive rate

$FPR_{overall}$ by Eq. 4.7. For example, suppose the total number of distinct DIPs is 1000, and there are 180 "Flashcrowd" keys and 20 "DDoS" keys. Let us assume that $K_{mc} = 1024$, $H_{mc} = 10$ for MCS and $K_{bc} = 128$, $H_{bc} = 5$ for BCS. According to the Eq. 4.4 and Eq. 4.5, we have $FPR_{mc} = ((180+20)/1024)^{10} \approx 8.08\times10^{-8}$ and $FPR_{bc} = (20/128)^5 \approx 9.31\times10^{-5}$. Therefore, $FPR_{overall}$ can be estimated as:$(0.8\times8.08\times10^{-8}\times9.31\times10^{-5}+0.18\times9.31\times10^{-5})/(0.18+0.8) \approx 1.72\times10^{-5}$.

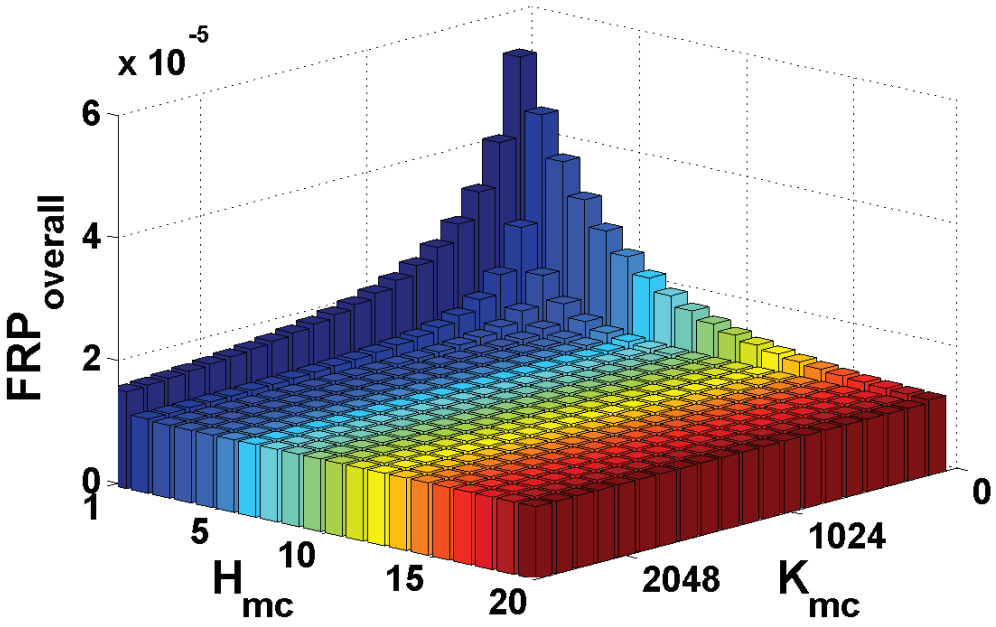


Figure 4.9: Size of MCS VS. $FPR_{overall}$

In order to further demonstrate the impact of various factors on $FPR_{overall}$, we vary different parameters and draw figures according to Eq. 4.7. The default settings for the parameters we adopted are $H_{mc} = 10$, $K_{mc} = 1024$ for the sketch size of MCS, $H_{bc} = 5$, $K_{bc} = 128$ for the size of BCS and $P_{normal} = 0.8$, $P_{flashCrowd} = 0.18$ for the traffic distribution.

Fig. 4.9 and Fig. 4.10 show the impact of sizes of MCS and BCS on the overall false positive rate, respectively. From both of these two figures, we can see that by enlarging the size of sketches (either H or K), we can greatly reduce the overall false positives. Although keeping the

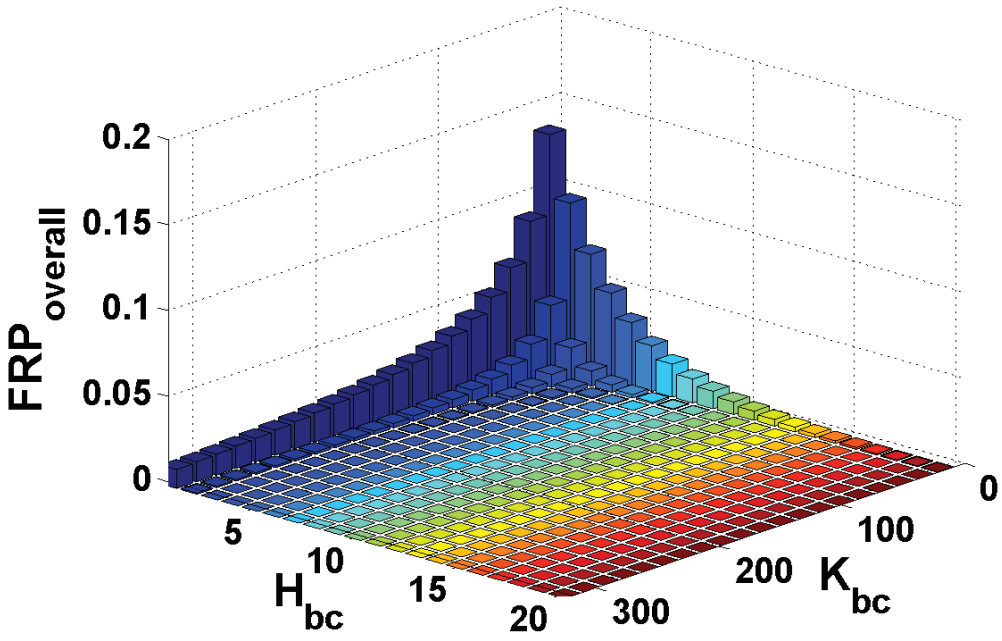


Figure 4.10: Size of BCS VS. $FPR_{overall}$

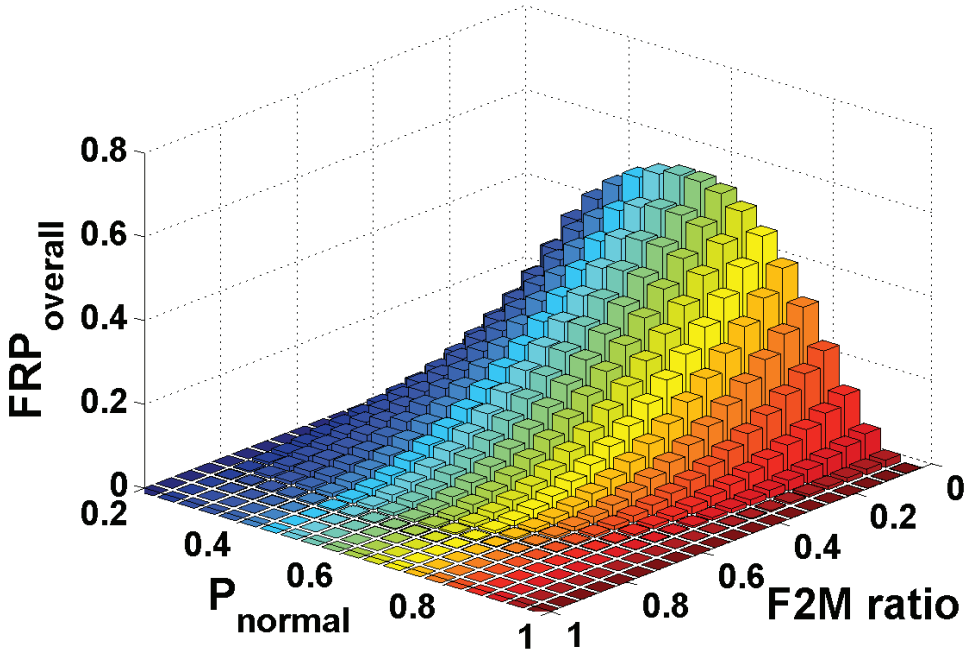


Figure 4.11: Traffic distribution VS. $FPR_{overall}$

size of sketches large will benefit the accuracy performance, it will also consume much memory space, which will be unaffordable for practical deployment. In practice, we should carefully design the size of sketches employed in the scheme based on space requirements. In Fig. 4.11, we show the impact of traffic distribution on $FPR_{overall}$, where F2M is defined as the ratio of the number of "Flashcrowd" keys to malicious keys (including "Flashcrowd" keys and DDoS keys). We can see that the $FPR_{overall}$ decreases as the proportion of "Flashcrowd" keys increases when we fix P_{normal}. This is because the probability that a normal key is hashed into a malicious bucket decreases as the number of malicious buckets diminishes.

Regarding the false negative rate, we first consider the false negative rate of MCS FNR_{mc} and BCS FNR_{bc}. Since we define those buckets that are hashed by the malicious keys as malicious buckets and we consider those keys that are hashed to malicious buckets in every row of sketches as malicious keys, there will be no false negative ideally. In our MCS stage, it is possible that some false negatives can be caused by the improved EWMA technique. For example, some of buckets, which should be considered as malicious buckets, still do not generate the corresponding alarm signal for detection. However, this case is much less often, because the malicious keys will always have much higher incoming frequency than the normal keys. Similar things happen at the BCS stage. Although there might be some false negatives in BCS due to the inherent false positive issue of bloom filters employed, such case rarely happens for the much lower false positive rate of bloom filters compared with sketches. From the perspective of the overall framework, positive instances only consist of DDoS keys and false negatives are those DDoS keys which are classified as normal

or "Flashcrowd" keys by mistake. Thus, according to the definition of false negative rate, we have:

$$
\begin{aligned}
& FNR_{overall} \\
&= \frac{\text{Total \# of false negative instances}}{\text{Total \# of positive instances}} \\
&= \frac{NP_{DDoS}FNR_{mc} + NP_{DDoS}TPR_{mc}FNR_{bc}}{NP_{DDoS}} \\
&= FNR_{mc} + TPR_{mc}FNR_{bc} \\
&= FNR_{mc} + FNR_{bc} \ \ (\text{Since } TPR_{mc} = 1)
\end{aligned}
\tag{4.8}
$$

Where TPR_{mc} is the true positive rate of MCS. The true positive rate for MCS is approximately equal to 1 based on the definition of the positive instances in MCS. Since both FNR_{mc} and FNR_{bc} are negligible, the overall false negative rate also can be neglected. Moreover, from Eq. 4.8, we can see that the false negative rate is irrelevant to the distribution of the incoming traffic.

4.5.3 Collaborative Detection Scheme

Till now, our proposed two-level framework can be categorized as a host-based system, which can be deployed at an ingress router near the victim side. The nearer the detection module from victims is, the larger amount of attack traffic we can observe. Thus, in order to reduce the difficulty of detection, one possible solution is to deploy the proposed detection module at the targeted server. However, this preliminary solution is a bad idea for two reasons. First, one deployment can only protect one victim which render it not scale well. Secondly, it cannot even well protect the victim it supposed to protect. Because the ingress bandwidth resources near the victim server can be exhausted as well by the attack traffic, which will result in the same effect to the legitimate users

since they cannot visit the victim server. Thus, a deployment that is a little far from a victim server might be a good choice.

However, a single host-based system is inherently not robust enough no matter where it is deployed. It is entirely possible that some unaware or intentional internet behaviors can damage its effectiveness. For instance, due to network device failure problems or a specific routing protocol designed for congestion avoidance, a backward traffic associated with an original forward flow might be routed by a totally different path. As a result, the traffic asymmetry feature no longer can be observed by a single router. Furthermore, such scheme can be easily fooled by a sophisticate attacker, which can be considered as an intentional internet event. Since attackers always employ a large number of zombie machines around the world to launch attacks, traffic that comes from every corner of the world can be routed by different edge routers inside an AS. Thus, if we only take a single router into accounts, the volume of attack traffic might not be aggregated at a detectable level for a detection module while the final gather of attack traffic will still cause severe damage to victim servers. Therefore, a collaborative detection approach which can comprehensively consider the global circumstance will be an attractive solution. Fortunately, our proposed approach can be easily extended to a collaborative detection scheme, which will greatly reinforce our original work. Fig. 4.12 illustrates the overall collaborative framework. The edge routers are responsible for connecting subnets (it can be customer networks or other ASes) with the core network. Our collaborative detection framework contains multiple local detectors, one global detector and a feedback loop between them. The functionality of each component is described as below.

Local Detector A local detector can be deployed at an edge router, and it is responsible for:

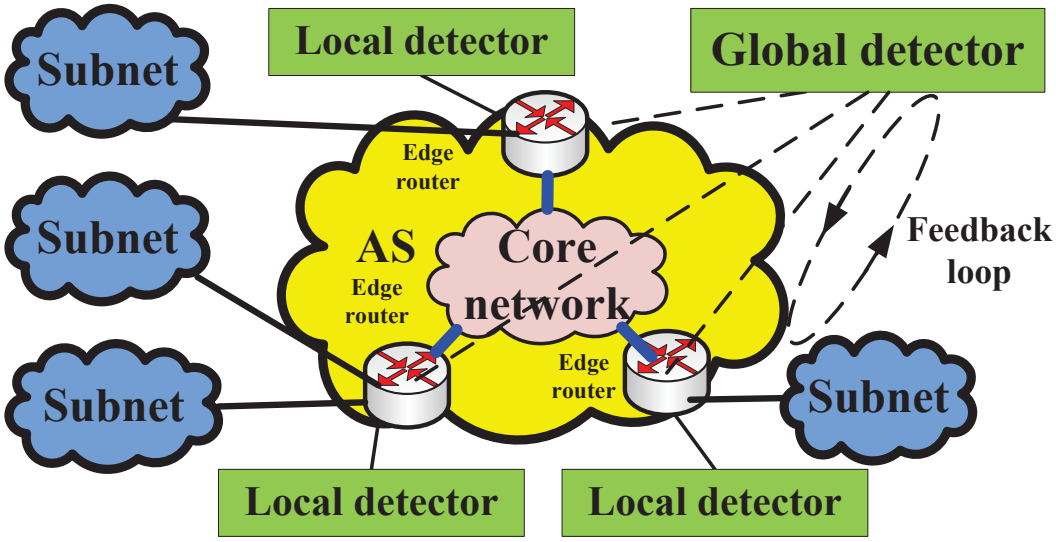


Figure 4.12: Illustration of a collaborative framework

- Summarizing traffic statistics from partial or all packets from both of two directional links
- Report the summarized traffic statistics to the global detector periodically
- Receive feedback instructions from the global detector and adjust the local information collection manner based on the feedback instructions
- Timely react to those DDoS events that can be detected at the local side

To be specific, a local detector maintains two main threads. The first thread is called "update thread", which keeps scanning every incoming packet and updates the traffic profile. The second thread, which we called as "report thread", periodically sends the built traffic profile to the global detector and finally refreshes the profile after reporting. A hash table with linked lists can be utilized for the traffic profile building. Each entry in the profile hash table contains six values $(DIP, num, suspFlag, min, max, sipPtr)$ with DIP as key value. num accumulates the number of those incoming packets associated with DIP during one period. $suspFlag$, which is set based

on the feedback instructions from the global detector, can be used to decide whether to update the remaining values in one entry or not. Whenever a DIP is suspected by the global detector due to its high packet frequency, the $suspFlag$ will be set to 1. When $suspFlag = 1$, the "update thread" will keep updating the following values $(min, max, sipPtr)$ in an entry. min and max maintain the minimal and maximal of hash value by mapping all SIPs associated with the DIP into range $(0, 1)$ as we did in the distinct sources estimator. At the same time, $sipPtr$, which is a head pointer of a linked list, will be updated by inserting those SIPs into the list. As we can see, by reporting the built traffic profile, the global detector can obtain all the necessary information for further anomaly analysis.

Global Detector The responsibility of a global detector contains:

- Receive those statistics reports from local detectors
- Perform anomaly detection based on packet frequency at coarse-level detection phase
- Perform anomaly detection based on both the distinct number of SIPs and asymmetry feature associate with each DIP at fine-level detection phase
- Send feedback instructions to local detectors based on anomaly detection results

The global detector also maintains two threads. The first thread, which we called as "MCS thread", is responsible for updating MCS and sending feedback to local detectors. The MCS update process is similar as we described above. The total incoming frequency associated with a DIP can be obtained by: $\sum_{k=1}^{M} num_k$, $(1 \leq k \leq M)$, where M is the total number of local detectors that report their local frequency num of this DIP to the global detector. When a key is detected

as suspicious key with high packet frequency during MCS detection phase in the global detector, those hash entries associated with this key at local detectors will be marked as suspicious by setting $suspFlag$ to 1. We called the second thread as "BCS thread". The "BCS thread" also does similar works as we have demonstrated above. The min and max value associated with certain DIP in BCS can be obtained by: $min = MIN(min_1, min_2, \cdots, min_M)$ and $max = MAX(max_1, max_2, \cdots, max_M)$. As we can see, our original scheme can be extended in a distributed-executing way quite smoothly. Besides those advantages we pointed out before, one great benefit by running in a distributed way is that the workload of the central global detector can be largely reduced. As a result, the scalability performance can be further improved.

We notice that the number of one packet will be counted twice in a typical AS infrastructure. One count is at the ingress router and the other one is at the egress router. Similar thing happens when we measure the count for traffic asymmetry. However, it will not impact the overall performance, because both malicious and benign traffic will be amplified by the same proportion when we measure the frequency feature. Regarding the asymmetry feature, both forward and backward traffic will be counted twice, the effect of which will be offset to each other when we measure the asymmetry feature in BCS.

4.6 Evaluation

We evaluate the performance of the proposed scheme via simulations. We use the trace data from AU [66] as the background traffic. It contains packet traces captured from the link connecting Auckland University and the Internet. This background traffic, which contains both forward and

Table 4.1: The default parameter settings of sketch-based detection

Item	Parameter	Setting value
Interval for Periodical Sketch Construction	Δt	5s
Size of MCS	H_{mc}	32
	K_{mc}	1024
Size of BCS	H_{bc}	5
	K_{bc}	128
Bloom Filters	L_{bf}	10000 Bytes
Coarse Level Detection	α	0.4
	θ	0.5
Fine Level Detection	ΔT	5s
	β	2
	$TH_{counter}$	10

reverse directions, has an average rate of 523 packets per second. We consider the accuracy of victim identification and the amount of memory consumption as two main performance metrics. The default parameter settings for the parameters we adopted in our experiment are shown in Table 4.1.

4.6.1 Detection Accuracy Evaluation

We generate the flooding traffic using attack tools we developed. The attack rates vary from 25 to 500 packets per second (25, 50, 75, 100, 200, and 500) and the duration of each the attack is 20 seconds. Those attacks are injected at the offset of every 100 seconds. Our goal is to try to gauge the detection sensitivity of our scheme under a large range of attack rates. Fig. 4.13 shows the maximal $DistNum$ series among all the detected victims in the sketch BCS as the time goes. The six spikes (excluding the smallest one) indicate all six DDoS attacks we injected. Even when

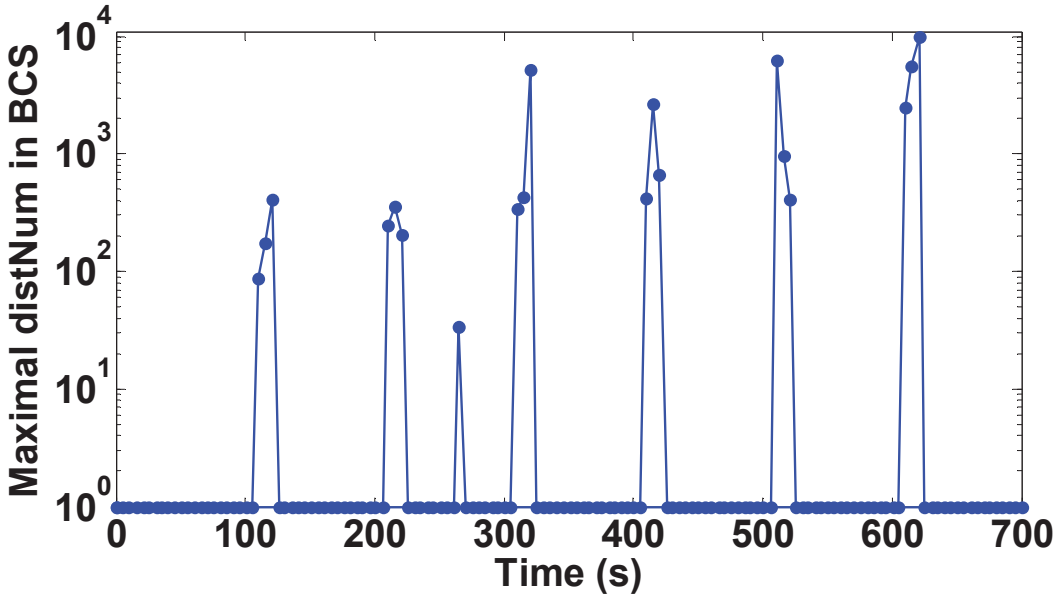


Figure 4.13: Maximal DistNum value in BCS

the attack rate is as low as 25 packets per second, which happens at the offset of 100 seconds, our scheme is still able to identify such low rate attacks while maintaining high accuracy. The maximal $DistNum$ values well reflect the rates of corresponding attacks. After we manually inspected the background traffic, we found that the remaining spike with the lowest value in Fig. 4.13 represents a low rate flooding attack in the original trace. Fig. 4.14 demonstrates the number of victims that are identified by the coarse-level and fine-level detection. On average, the coarse-level detection identify 12 victims per interval. All of those victims experience high rates of requests, which may be caused by flash crowds or DDoS attacks. However, after we further filter those potential victims using the fine-level detection, at most one victim per interval remains, which is the actual attack contained in the traffic. Moreover, the average victim number detected by the MNP-CUSUM approach [60] is around 21, which is even higher than the coarse-level detection of our approach. This is because the original CUSUM technique does not take care of the quick termination after the

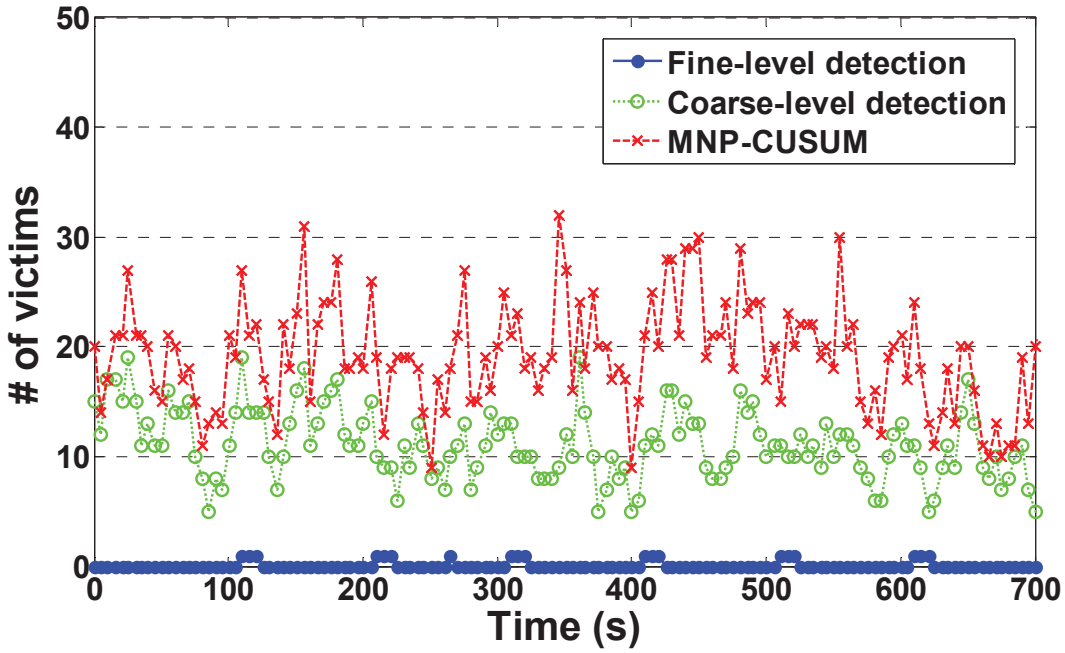


Figure 4.14: # of detected victims

alarm happens, which results in that too many buckets in the sketch remains high value for a long time. Therefore, it usually causes many false positives. After we modified the original CUSUM techniques by a method for quickly terminating alarm as proposed in [30], the average number is significantly reduced to around 12, which can be due to flash crowds.

We also measure the recall ratio under different attack rates. A recall ratio is the fraction of the true victims in the estimated victims returned by our scheme. The estimated victims identified by the coarse-level detection is the set of all DIPs which satisfy $CMS\_Query(DIP) = 1$. In Fig. 4.15, we can see that the recall ratio of the fine-level detection is very stable; nearly 100% of victims are accurately identified. Even when the attack rate is as low as 25 packets per second, the recall ratio is still over 95%. However, with the coarse-level detection only, the ratio is much lower. It requires more than 350 packets per second (about 66% of the background traffic rate)

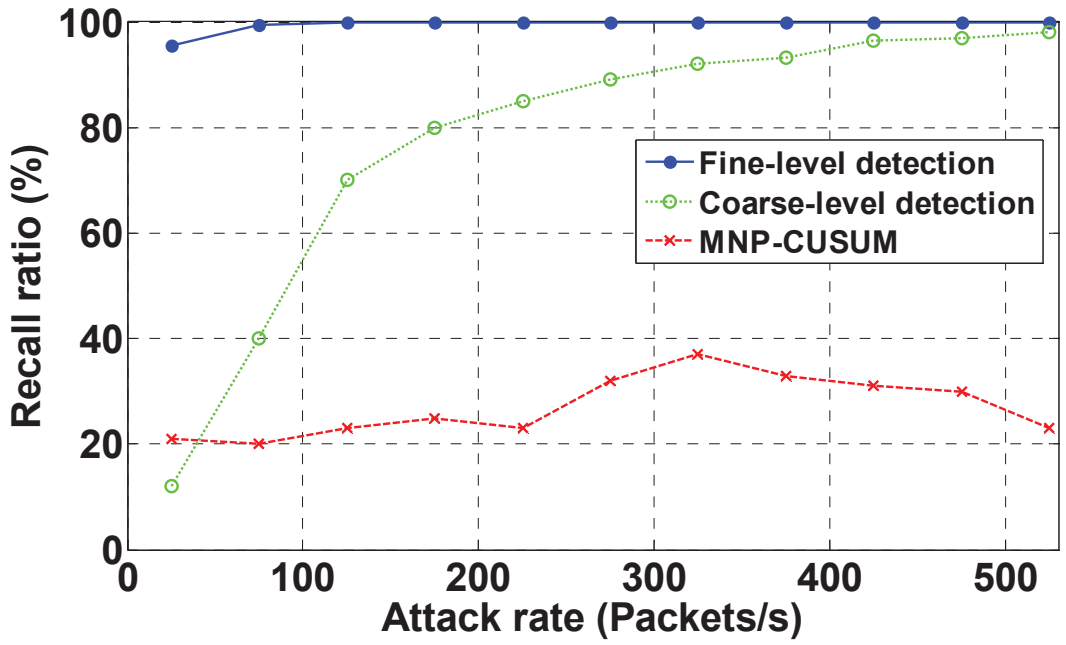


Figure 4.15: Recall ratio

to achieve the ratio over 95%. Again, due to the alarm termination problem, the MNP-CUSUM technique performs poorly here. Its recall ratio is around 23% on average.

4.6.2 Space Consumption

We also sought to measure the memory consumption. Basically, the overall space consumption of sketch-based approaches consists of two different parts. The first part, which can be attributable to the sketch structure itself, takes constant size of small space while the other part, which serves for assisting functions such as the key storage, occupies dynamic size. Since the scalability performance of sketch-based approaches greatly depends on the dynamic part, we compare our approach against [60] by measuring the number of keys that should be stored. The results are shown in Fig. 4.16. During one interval, there are 47 keys that are needed to be stored in our scheme on average while the average number of the keys of MNP-CUSUM approach is around 519. Our

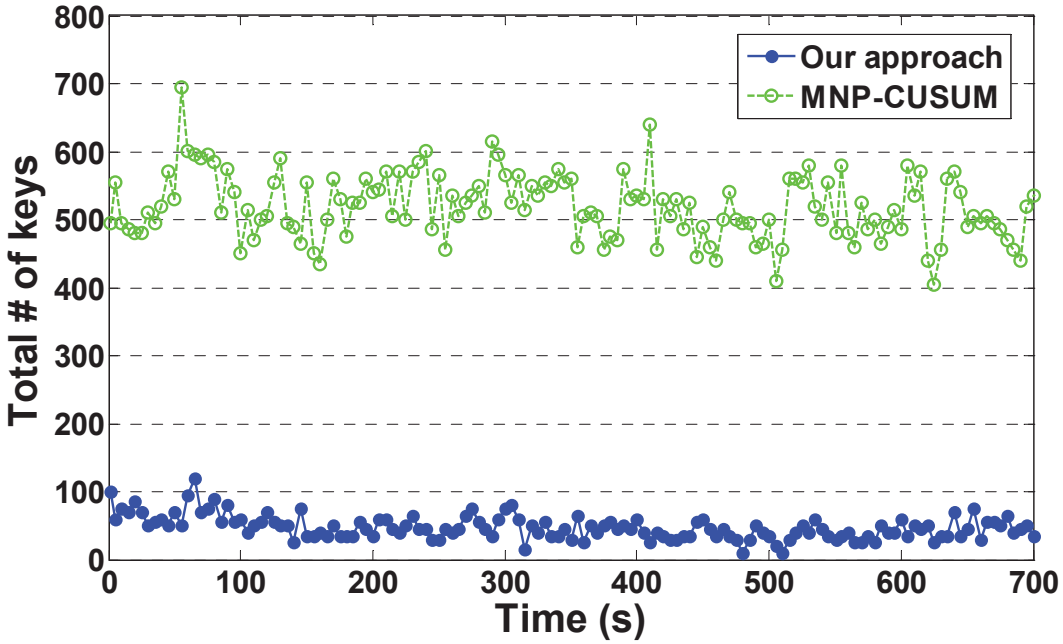


Figure 4.16: Space consumption

approach can save up to 90% keys, which translates to less memory consumption and searching space, when comparing with the previous approach. In order to evaluate the storage scalability, we shift time stamps of different periods of traces from AU and then merge them together in order to enhance the traffic intensity. We define "Merging factor" as the number of different periods, which can also reflect the intensity of the traffic. Then, we measure the required key storage over various approaches as shown in Fig. 4.17. Our method nearly keeps constant number of keys when the merging factor increases, while the MNP-CUSUM holds a linear-like trend in the same case. That is because our method only record those suspicious DIPs rather than storing every DIPs.

From the total number of keys in the sketches and the default parameter settings, the total memory consumption of our scheme can be estimated using the Eq. 4.2. The average memory cost is around 563.6 KB, which we consider can be easily accommodated in modern routers.

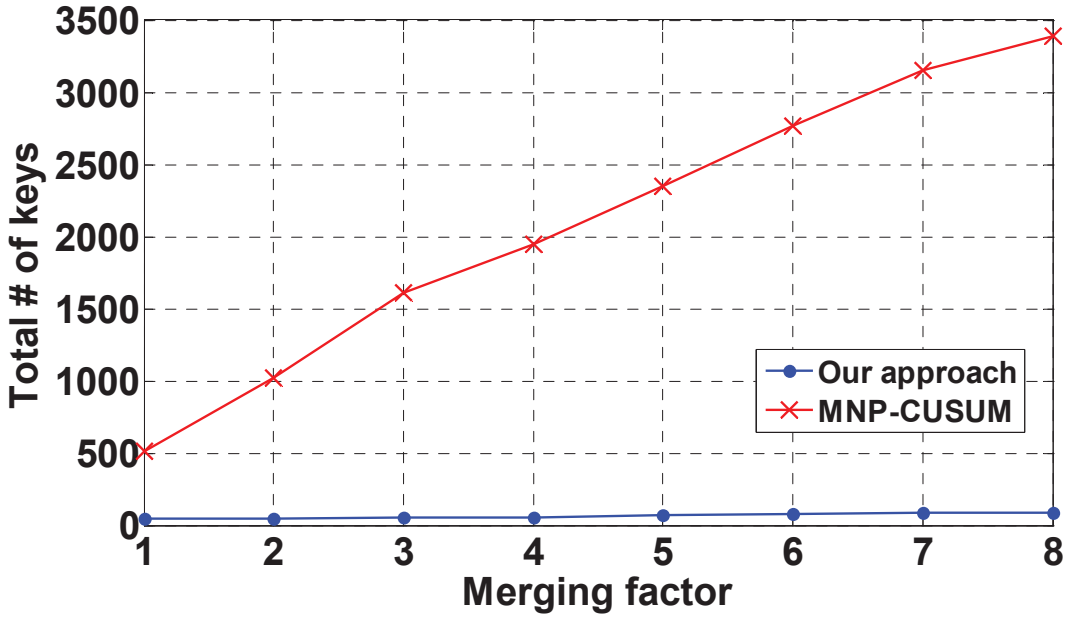


Figure 4.17: Storage scalability

4.7 Conclusion

In this work, we present a fine-grained DDoS detection scheme based on the BCS structure to counter the threat of DDoS attacks. Our approach employs a two-level model to reduce both the size of the search space and time, and further make identification of specific victims possible in the high-speed network environment. We adopt the MCS structure in coarse-level detection to achieve fast detection, and the BCS structure in the fine-level to further guarantee the accuracy. We believe that this approach can accurately identify victims of DDoS attacks with a low memory footprint and give a timely response. We also propose a SRAM-based parallel architecture to achieve high-speed process. We finally analyze accuracy estimation issue and demonstrate a collaborative detection scheme based on the original single-host detection scheme. Experimental results show that our scheme outperforms previous sketch-based methods with respect to both storage scalability and

detection accuracy.

Our future work will focus on designing a collaborative defense framework against DDoS attacks. Our proposed detection scheme can be used to facilitate defense against DDoS attacks in the following way. Since all the victims can be accurately detected by our collaborative scheme, an automatical rules generator can be developed to reinforce firewall and IDS systems in real-time.

CHAPTER FIVE

A COLLABORATIVE DEFENSE FRAMEWORK AGAINST DDOS ATTACKS

5.1 Motivation of this work

All the previous works focused on the detection of DDoS attacks, we need to have some ways to react and filter malicious traffic, which is the goal we targeted at in this work. As we pointed out in the Chapter 4, a single-host based defense system has its own weaknesses for fighting against DDoS attacks. A sophisticated attacker can easily circumvent detection by employing a large number of zombie machines around the world and make them send malicious traffic through different edge routers. As a result, if we only take a single router into accounts, the volume of malicious traffic might not be aggregated at a detectable level. The advantage of distributed defense systems over single-host systems has been recognized in [70–72]. [70, 71] deployed defense nodes at all the victim-end, source-end and core network. They can achieve higher effectiveness than source-end/victim-end solutions [72] or victim/core network schemes [73]. However, they still focus on a single approach to fight against attacks. For example, the method proposed in [70] depends on a capability mechanism and the approach proposed in [71] relies on victim-hiding technique. All of these constraints will discourage integration with other defenses and their wide deployment.

George Oikonomou and his colleagues proposed a framework called DefCOM for a collaborative DDoS defense in [3]. Nodes inside the network collaborate during the attack to spread alerts and protect legitimate traffic and then rate limiting actions are performed on the correspond-

ing nodes and routers. However, the effectiveness of their method relies on three key assumptions. (a) The victims are able to detect attacks accurately and timely. (b) The core routers in the network are willing to collaborate together to alert source-end nodes. (c) Source-end nodes have enough incentives to mitigate malicious traffic for the victim-end. However, these three assumptions are usually not easily to be satisfied in real world. Firstly, the paper did not address the key issue of how to accurately and timely detect various common DDoS attacks which will greatly affect the effectiveness of their scheme. Secondly, the collaboration of Internet-wide scale routers seems to be infeasible in reality, especially when we consider the fact that the current Internet consists of different autonomous systems which are administrated by various ISPs. Finally, source-end nodes do not have enough incentives to protect the nodes at the victim-end. Thus, their third assumption is also under questions.

In order to overcome the weaknesses of the previous framework as stated above, we propose a collaborative framework for detection and defense of DDoS attacks in this chapter. However, we are not aimed to protect the Internet-wide scale network. Rather, we proposed the collaborative framework to protect edge-networks. It can achieve the following advantages compared with the previous framework at the cost of some scalability performance which are acceptable as we consider. (a) Since the effectiveness of each individual component have been shown and proved through experiments in the previous chapters, we can guarantee victim-end nodes have the ability to detect attacks accurately and timely. (b) Since we do not focus on the protection of the Internet-wide scale, the feasibility of the implementation can be greatly increased since all the deployments only relies on the operations of a single AS. (c) Since we only protect the victim side, the defense

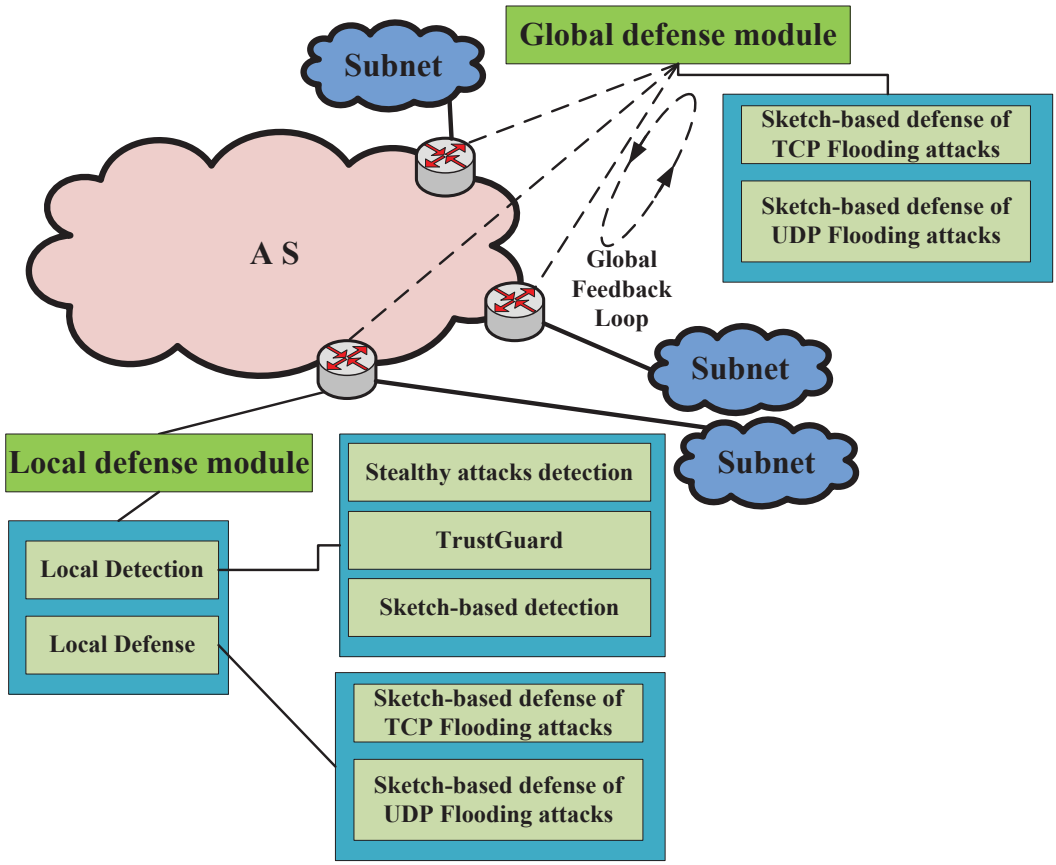


Figure 5.1: Overall system architecture

incentive of the corresponding edge network can also be maximized. We describe details of the implementation in the following sections.

5.2 Overall system architecture

Fig. 5.1 shows the overall system architecture. We deploy local defense modules among edge routers and the global defense module is also deployed at one edge router. There exists a feedback loop between local defense modules and the global defense module. All the local defense modules periodically collect the traffic statistics, summarize the passing traffic profile and send that information to the global defense module for further analysis. At the same time, all the local defense

modules will directly defend against those attacks which are detectable at the local side by single-host-based methods demonstrated in the previous chapters. The global defense module collects the information from all local defense modules, makes comprehensive decisions according to the collected information and sends filter instructions to corresponding edge routers. In this chapter, we mostly focused on the defense of TCP and UDP flooding attacks, which can be considered as the most two representative examples of current DDoS attacks. We firstly demonstrate how the proposed defense schemes work and then show how to extend the schemes to the distributed-executing version.

5.3 Defense against TCP flooding attacks

In this section, we proposed a defense scheme against TCP flooding attacks, which is considered as the most popular DDoS attack. Then, we extended our proposed method to a distributed-executing version. The effectiveness of the proposed scheme is shown by experiments using real Internet traffic.

5.3.1 Background

Among various types of DDoS attacks, probably TCP SYN flooding is the most powerful and serious flooding method and the most common one. It has been reported that about 90% of all DoS attacks are SYN Flood attacks. According to a recent NANOG report [74], it still dominates DDoS attacks. Thus, in this work we take TCP SYN flooding attacks as a motivating example. Over the past decade, many victim-based defense systems [75–77] have been proposed to mitigate DDoS attacks. Such systems are attractive as they are able to accurately differentiate benign from

malicious traffic, and thus can perform per-flow filtering. However, most of current defense systems need to maintain per-flow state, which renders them not scalable in high-speed environment. What is worse, those systems themselves could be attack targets since maintaining per-flow state requires a significant amount of resources. Besides the scalability issues, they usually cannot effectively filter traffic launched from both genuine and spoofed IP addresses. For example, the SYN cache [75] allocates full required resources only when the connection is completed which enables it to mitigate traffic originated from spoofed sources. However, its effectiveness will be beaten by attacks using BotNet since malicious traffic in this case is launched using real addresses. On the other hand, some detection and mitigation schemes [22, 57] are based on detecting abnormal SYN/RST, SYN/FIN or SYN/SYNACK ratios and such schemes will work well when most of the traffic is launched from genuine sources. However, attackers can still avoid detection by sending crafted packets to satisfy the normal ratios and we denote this as "spoofed-packet problem".

In this work, we design and implement an innovative DDoS defense system by using a sketch and multiple bloom filters, which can be deployed at ISPs' edge routers, to protect servers. It works as follows. We firstly filter the spoofed IP attack traffic using two bloom filters. Basically, it filters spoofed traffic by pairing SYN/ACK with associated ACK packets. As a result, the remaining traffic after the first stage only contains both legitimate traffic and traffic generated by compromised nodes using genuine IP addresses. We then pass the remaining traffic to a K-ary sketch for further differentialization. The values of buckets in the sketch are built based on the number of SYN packets associated with destination IP (DIP) addresses. Then, a light-weight exponentially weighted moving average (EWMA) technique is employed for detecting anomaly

buckets. All of the source IP (SIP) addresses associated with anomaly buckets will be record in two counting bloom filters (CBF). Finally, SYN packets will be probabilistically dropped according to the associated value in the CBFs.

We are aware of recent defense systems [44, 78] that utilize users' behavior to accumulate credits and the traffic from those ill-behaved users will get punished. The main differences from previous credit-based defense schemes is that all of them require keeping per-flow state to calculate credits and thus cannot scale well with high-speed environment since the amount of required space consumption will grow linearly with the number of flows. Furthermore, previous credit-based schemes cannot fight against spoofed IP DDoS attacks since the missing interaction between source-end and victim-end which will nullify the credit profile construction.

Our defense scheme addresses these problems. Introducing sketch structure makes it scale with the high-speed environment. Also, since it filters most spoofed traffic before they go through sketch, not only false positives of sketch employed are greatly reduced but also the "spoofed-packet problem" is mitigated. To the best of our knowledge, all of the previous sketch-based schemes focus on the detection rather than defense of DDoS attacks and none of them utilize sketch for flow-level filtering. Finally, our approach filters both spoofed and genuine IP attack traffic effectively.

5.3.2 Related Works

According to the deployment location, current DDoS defense systems are classified into four categories. They are source-end, victim-end, core network, and combinations of all three. Except victim-end defense systems, none of the other three types of defense systems have been deployed

widely, although network-wide deployments have been proved to be more effective to fight against denial of network service [79]. This is mainly due to the lack of incentives for those source-end ISPs to invest on DDoS defense systems to protect servers of other ISPs. Our proposed system is designed as a victim-end defense system to maximize deployment incentives and can be used to complement the existing network-wide deployment.

In terms of defense methods, current defense systems also can be divided into two main categories. They are statistical-based and behavior-based defense systems. Statistical-based defense systems [21, 80, 81] typically rely on nominal traffic profiles built before attacks happens. The systems try to discriminate "normal" traffic from "malicious" traffic based on deviations from predefined nominal traffic profiles. However, as more and more sophisticated attacks emerging, they can bypass detection by mimicking legitimate behaviors. For example, the attacker can learn the victim's nominal traffic profiles by probing the victim and observing the corresponding responses. As a result, the systems can be beaten by sending malicious traffic that is carefully crafted to satisfy the nominal traffic profile.

On the other hand, various behavior-based approaches [42–44, 78, 79, 82] have been proposed in the literature and probably credit-based system consists of the majority of this type. Such kind of systems typically regulates the traffic based on behaviors of related sources. The ill-behaved traffic will get throttled. Yang et al. introduce a Message Rate Controlling Model (MRCM) in their credit-based framework to defend against DDoS for Peer-to-Peer (P2P) streaming systems [42]. MCRM is essentially a network-wide solution requiring the cooperation of all nodes within a system, which renders this scheme impractical for deployment. Padmanabhan et

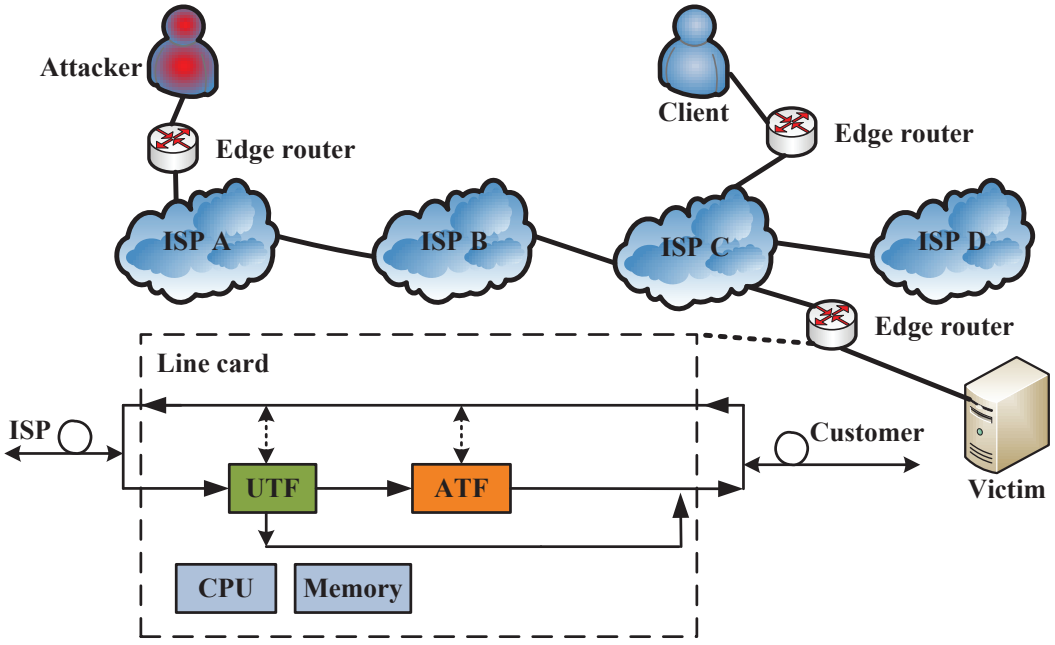


Figure 5.2: Overall architecture

al. propose a trust-based traffic monitoring approach for preventing DDoS attacks [43]. Again, their framework requires interaction among peer nodes in the network. Furthermore, all of previous behavior-based systems need to maintain per-flow state which renders it not scale well with link-speed.

5.3.3 System Description

Overall Architecture

We first describe an overview of our designed architecture, which is shown in Fig. 5.2. It can be deployed at the ISP's edge routers and serves as a supplement of the existing network filtering to protect potential victims from being flooded. It monitors both of the forward and backward passing traffic. Only traffic with the forward direction will be regulated in our scheme. The backward

traffic from servers to clients is also useful since we will utilize it for building the filters in the two modules of our method.

Unidirectional Traffic Filter

The term "unidirectional traffic" described here refers to the traffic that does not have interactions with servers during the TCP three handshake procedure. It is usually associated with spoofed IP DDoS attacks since there is no way for non-existing source nodes with bogus addresses to interact with servers. Spoofed IP DDoS attacks are usually attractive for attackers to use for launching attacks since these kinds of attacks can help attack sources to escape being traced back. The "unidirectional traffic" can also be launched by a large number of zombie machines, which indicates genuine IP DDoS attacks, by simply ignoring SYN/ACK responses from servers. In this work, the term "unidirectional traffic" and "spoofed IP attack" are exchangeable since most of "unidirectional traffic" is due to the "spoofed IP attack" in networks.

A typical TCP three-way handshake protocol works as followings. Firstly, the client sends a SYN packet to a server to ask for an open connection request. The server then reserves connection resources and then responses with a SYN/ACK packet. Finally, the client sends an ACK packet back to the server as an acknowledgement to indicate an established connection. In a TCP SYN flooding attack scenario, by not responding to the server with the expected ACK packets and making the server wait for the ACKs for some time, the attack sources can exhaust resources of the server if a large number of half-open connections are involved. As a result, the server suffers from the SYN Flood DDoS since no new connections can be made. In this case, if we deploy a detector near the victim side, we will see a large difference between the number of SYN/ACK packets

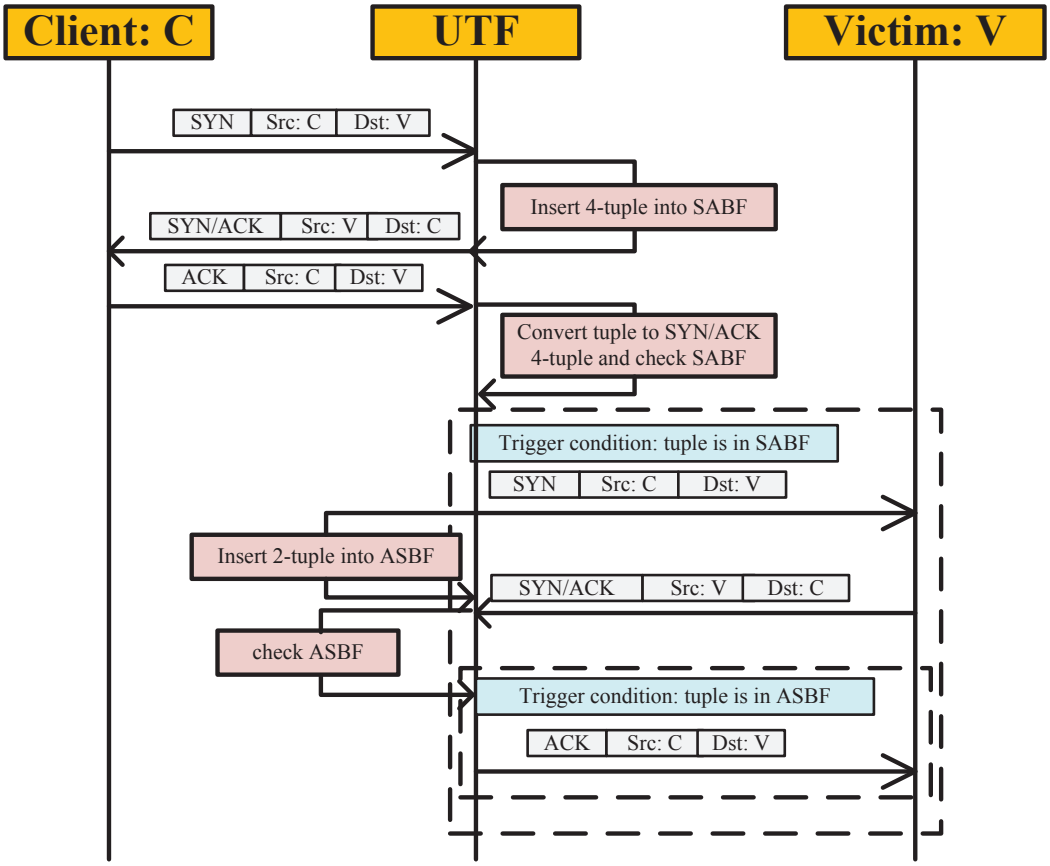


Figure 5.3: Illustration of UTF module by a sequence diagram. This figure neglects the ATF module which sits between the UTF and victims for the clear demonstration of the procedure.

sent by the server and that of ACK packets. This is because source addresses are bogus and the server will never receive ACK packets from clients. Based on this observation, researchers [83] proposed accurate detection approaches based on bloom filters (BF). However, their approach can only detect flooding attack events while it cannot filter the unidirectional traffic. In this work, we also utilize bloom filters in UTF. The main improvement is that our UTF module has the capability to filter the majority of "unidirectional traffic", which is different with previous BF-based works.

The main goal of UTF is to filter the majority of SYN packets associated with unidirectional traffic before they arrive at the victim, while not affecting SYN packets associated with

active source nodes which will finish the three handshake procedure. Fig. 5.3 illustrates the main work procedure of UTF. A typical filtering procedure is described as follows. When an attacker C sends a SYN packet to a victim V to initialize a TCP connection, the UTF will respond to C immediately by creating a corresponding SYN/ACK packet with V as SIP and sending the created SYN/ACK packet back to client C. At the same time, we extract the SIP and DIP, sequence number and ACK sequence number information of this mimic SYN/ACK packet as 4-tuple $\langle \mathrm{SIP}, \mathrm{DIP}, \mathrm{SEQ}, \mathrm{ACKSEQ} \rangle$ and insert this 4-tuple into a bloom filter called SYN/ACK Bloom Filter (SABF). Whenever the UTF observes an ACK packet in the forward direction, it will extract 4-tuple $\langle \mathrm{DIP}, \mathrm{SIP}, \mathrm{ACKSEQ} - 1, \mathrm{SEQ} \rangle$ from this ACK packet and query the SABF to see whether this ACK packet can be paired with any mimic SYN/ACK packets inserted before with a small false probability. If the tuple cannot be found in SABF, the UTF forward this ACK normally. While if the tuple is checked as positive in SABF, which means there is a paired SYN/ACK packet inserted before, the UTF will generate a new SYN packet with SIP as C for this source using the header information of this ACK packet and send it to the victim V to initialize a real three-handshake procedure. At the same time, the UTF will insert the associated SIP and source Port as 2-tuple $\langle \mathrm{SIP}, \mathrm{SP} \rangle$ into another bloom filter named Active Source Bloom Filter (ASBF). Similarly, whenever the UTF captures a SYN/ACK packet from the backward traffic, it will extract the corresponding DIP and destination port (DP) as 2-tuple $\langle \mathrm{DIP}, \mathrm{DP} \rangle$ and check whether this 2-tuple can be found in the ASBF. If yes, the UTF will generate an ACK packet with DIP in SYN/ACK as SIP and send it back to the victim V to complete the connection.

From the procedure described above, we can see that the UTF essentially serves as a dele-

gation of victim servers. The TCP connections associated with active sources will be established unaffected. However, the majority of malicious SYN packets will stop at the UTF module since actual initialization of TCP connections will not be triggered due to the lack of ACK packets that can be paired with inserted SYN/ACK packets in SABF. Someone may ask what if the UTF itself is flooded by SYN packets. Since UTF does not allocate resources to prepare for a real connection when it receives a SYN packet, this issue will not affect the performance. Also, we notice that the UTF module will introduce certain delay of the TCP three-way handshake procedure for a new connection. If we can place some existing detection modules [65] with victim pinpoint capability before the defense module, only the connections to victims which are under attacks will be affected. At the cost of some little delay to normal connections, we can avoid victims being flooded by spoofed IP attacks.

We also aware that the existing SYN cache [75] and SYN cookies [77] also utilize delegation mechanism. However, the per-flow state still needs to be maintained or state cryptographic computation is required which render it not scale well. It will also make the defense mechanism itself vulnerable to attacks. In our UTF, only two bloom filters are maintained, and it will save a lot of space consumptions compared with previous approaches. Although using bloom filters for recording will yield some false positives, the majority of "unidirectional traffic" is filtered, which greatly reduces computing overhead and false positives in the next module ATF. We will see that the remaining spoofed of traffic that pass UTF will still be well regulated by ATF as describe below. Furthermore, both of the two bloom filters should be refreshed periodically whenever the ratio of set bits to the length of bloom filters exceeds predefined threshold TH_{SABF} to avoid large false

positives.

We filter "unidirectional traffic" before passing it to the second stage for two reasons. Firstly, the ATF, which we will describe in the next section, usually requires a relatively higher computing overhead than UTF since it need to build traffic profile. Thus, by eliminating most of the "unidirectional traffic" at a light-weight UTF stage, the computing overhead of the ATF will be greatly reduced. Furthermore, since we can avoid maintaining a large number of bogus flows in sketch, the incidents of hash collisions will also be reduced in sketch, which translates much less false positives.

Aggressive Traffic Filter

After passing the UTF module, the majority of the remaining traffic should be originated from active sources using genuine SIP addresses. However, an active source is not necessarily benign. For example, a sophisticated attacker can still utilize zombies to periodically initialize and finalize TCP three handshake procedures to exhaust the resources of victims. We call this kind of attacks as genuine IP flooding attacks since it can only be launched by sources with genuine IP addresses. The genuine IP flooding attacks can pass our UTF module and exhaust connection resources of victim servers. In the sketch, we define the $Sketch\_Query(key)$ function which can return the minimum value among all the buckets corresponding to a specific key and such value can be used to estimate the frequency of occurrences of the key.

Abnormal Buckets Identification We first perform a light-weight EWMA technique to identify those abnormal buckets in the sketch. Each bucket in the sketch maintains four value $(v_t, v_{t-\Delta t}, v_{\text{backup}}, Flag)$.

v_t is the number of SYN packets that are accumulated from $t - \Delta t$ to t, $v_{t-\Delta t}$ is the previous value of v_t, and v_{backup} is the value of v_t right before the alarm occurs (or null if there has been no alarm). $Flag$ is set to 1 whenever the alarm condition is satisfied; otherwise it is set to 0. For each incoming record, we update the sketch with $(k_i, 1)$ where k_i is the DIP and 1 represents the number of this incoming SYN packet. Rather than returning the minimum value of v_t as the original sketch does, the $Sketch\_Query$ function in ATF returns the minimum value of $Flag$ among all the buckets corresponding to a specific DIP to indicate whether it is under SYN flood attacks.

When a new packet arrives, hash values of H hash functions are computed, and the corresponding buckets are updated; the value in each bucket is incremented by 1. This accumulation process repeats every Δt seconds. The alarm condition is tested for all $H \times K$ buckets periodically. If the alarm condition is satisfied, then the alarm flag associated with the bucket is set to 1. Whenever there is an alarm, the previous v value of the bucket is stored in the v_{backup} for determining whether the raised alarm is terminated or not. In our approach, we use an EWMA to generate an alarm for a bucket. We estimate v_t with an EWMA parameter α. Whenever $v_t \geq (1 + \theta)\overline{v_{t-\Delta t}}$, which is considered as the satisfaction of the alarm condition, an alarm is raised. θ is the parameter representing the percentage above the estimated value that can be considered to be an indication of anomalous pattern. There is a little difference after an alarm is generated for a bucket. Rather than using the previous value $\overline{v_{t-\Delta t}}$, we estimate $\overline{v_t}$ by v_{backup} to eliminate the impact of the anomaly on the next following $\overline{v_t}$ series. If $v_t < (1 + \theta)v_{\text{backup}}$ is satisfied, we reset the alarm flag to 0.

If $Sketch\_Query(key) = 1$ is satisfied for an incoming packet with its DIP, which indicates its suspicion, a finer-level filtering process will perform on this packet, which will be demonstrated

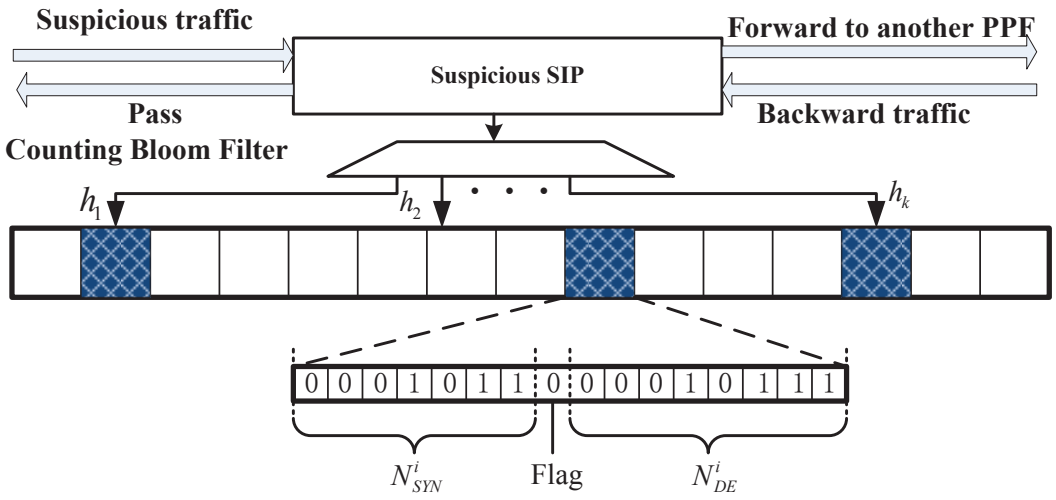


Figure 5.4: Update of bloom filters for aggressive traffic filtering.

in the next section. All the suspicious DIPs will be recorded in a bloom filter called suspicious DIP bloom filter (SDBF) and the SDBF will be placed before the sketch to reduce detection overhead. If a DIP associated with a packet is positive in the SDBF, the packet will be directly considered to be suspicious without performing $Sketch\_Query$ function.

Probabilistic Packet Filter We employ two counting bloom filters to build our probabilistic packet filter (PPF) to mitigate the flows with aggressive behavior. The first CBF builds the traffic profile for the current detection interval and the other one, which is built based on the traffic in the previous detection interval, is used for the packet filtering. The role of each of the two CBFs is altered to the other one as the time goes. When reaching the end of a detection interval, we refresh the filtering CBF and prepare it for the traffic profile construction during the next detection interval. At the same time, we used the CBF built in the last interval for filtering. We take two factors into account for identifying the aggressive behavior of a flow. Firstly, an aggressive source will send more SYN packets to the victim in order to exhaust the victim's resource compared with

a normal one. Secondly, an aggressive source will send SYN packets to victims with few real data exchanges (DEs). The attacker can be benefited from this way because it can maximize the attack power of the compromised nodes. The forward and backward update process of the CBF is described in Algorithm 4 and Algorithm 5, respectively. When a packet which is considered as malicious by sketch comes, we update the value of all the indexed N^i_{SYN} and the same of N^i_{DE}. Then, the minimal value of all the indexed N^i_{SYN} and the same of N^i_{DE} can be used to estimate the number of SYN packets and that of data exchanges associated with a key. As shown in Fig. 5.4, we use 2 bytes in each cell in the CBF where 7 bits is used for recording N^i_{SYN}, 8 bits for N^i_{DE} and 1 bit of Flag for indicating the direction of the traffic. Here, we use 0 to represent the forward direction and 1 for the reverse.

Algorithm 4: Update procedure of CBF (forward)

```
for k = 1 to K do
    i ← PPF.hash[k](SIP) ;
    if Packet.tcpFlag = SYN then
        if PPF.CBF[i].N_SYN < 128 then
            PPF.CBF[i].N_SYN++ ;
        end
        PPF.CBF[i].Flag ← 0 ;
    else
        if Packet.tcpFlag = ACK and PPF.CBF[i].Flag = 1 then
            if PPF.CBF[i].N_DE < 256 then
                PPF.CBF[i].N_DE++ ;
            end
            PPF.CBF[i].Flag ← 0 ;
        end
    end
end
```

Algorithm 5: Update procedure of CBF (backward)

for $k = 1$ **to** K **do**
  $i \leftarrow PPF.hash[k](DIP)$;
  if *Packet.tcpFlag = ACK and* $PPF.CBF[i].Flag = 0$ **then**
    $PPF.CBF[i].Flag \leftarrow 1$;
  end
end

We further define normal index (NI) as $NI^i = N^i_{DE}/N^i_{SYN}$ for a specific cell in the bloom filter. The larger NI is, the more likely the associated flows are benign. Thus, the probability of dropping packets that belong to this flow can be defined as:

$$P^i_{drop} = \frac{\left(TH_{NI} - NI^i\right)^+}{TH_{NI}} \tag{5.1}$$

Where TH_{NI} is a threshold to decide whether this flow is benign or not, and Δ^+ is Δ if $\Delta > 0$ and 0 otherwise.

Space Requirements

The primary space consumption of the system is due to the two bloom filters employed in the UTF module, one sketch in abnormal buckets identification phase of ATF module and two counting bloom filters for PPF in ATF module. Suppose each bloom filter will occupy L_{bf} space and each counting bloom filter will take L_{cbf} space. The sketch has the size of $H_{sketch} \cdot K_{sketch}$. Let L_{sketch} denote the length of each bucket in the sketch. The total memory requirement will be:

$$H_{sketch} \cdot K_{sketch} \cdot L_{sketch} + 2L_{bf} + 2L_{cbf} \tag{5.2}$$

As we can see, the space consumption is constant, which translates to the promising scalability performance.

Collaborative Defense Scheme

The proposed scheme can be extended to a distributed-executing framework. We demonstrate how to extend the proposed scheme in this way in this section. Our collaborative defense framework contains multiple local defense modules, one global defense module and a feedback loop between them. The functionality of each component is described as follows.

Local defense module A local defense module can be deployed at an edge router, and it is responsible for:

- Summarizing traffic statistics from partial or all packets from both of two directional links
- Report the summarized traffic statistics to the global defense module periodically
- Receive feedback instructions from the global defense module and perform packet filtering based on the feedback instructions
- Timely react to those DDoS events that can be detected at the local side

Since the UTF module needs to timely react to the traffic in order to reduce the TCP connection delay, we deploy the UTF module at the local side. Only the ATF module is extended to the distributed version. To be specific, a local defense module maintains two main threads. The first thread is called "update thread", which keeps scanning every incoming packet and updates the traffic profile. The second thread, which we called as "report thread", periodically sends the built

traffic profile to the global defense module and finally refreshes the profile after reporting. A hash table with linked lists can be utilized for the traffic profile building. Each entry in the profile hash table contains three values $(SIP/DIP, num, tcpFlag)$ with SIP or DIP as key value, the selection of which depends on the direction of the passing traffic. num accumulates the number of those incoming packets associated with the selected key during one period. $tcpFlag$ represent the TCP flag. As we can see, by reporting the built traffic profile, the global detector can obtain all the necessary information for further anomaly analysis.

Global defense module The responsibility of a global defense module contains:

- Receive those statistics reports from local defense modules
- Perform anomaly detection based on the collected information
- Identify those malicious flows with high level of asymmetry of data exchanges
- Send feedback instructions to local defense modules

The global defense module also maintains two threads. The first thread, which we called as "anomaly detection thread", is responsible for updating sketch and bloom filters and sending feedback to local defense modules. The process is similar as we described in the above section. The total incoming frequency associated with an IP can be obtained by: $\sum_{k=1}^{M} num_k$, $(1 \leq k \leq M)$, where M is the total number of local defense modules that report their local frequency num of this IP to the global defense module. When a flow is detected as malicious in the global detector, the corresponding dropping probability will be computed and sent back to related local defense modules for further filtering. As we can see, our original defense scheme can be extended in a

distributed-executing way quite smoothly. Besides those advantages we pointed out before, one great benefit by running in a distributed way is that the workload of the central global detector can be largely reduced. As a result, the scalability performance is further improved.

5.3.4 Evaluation

Table 5.1: The default parameter settings of sketch-based defense

Item	Parameter	Setting value
Interval for Periodical Sketch Profile Construction	Δt	5s
Interval for CBF Profile Construction	Δt	5s
Sketch Size of ATF Module	H_{sketch}	16
	K_{sketch}	1024
Abnormal Buckets Identification	α	0.3
	θ	2
Size of Counting Bloom Filters	L_{cbf}	20000 Bytes
Size of All Bloom Filters	L_{bf}	10000 Bytes
Probabilistic Dropping	TH_{NI}	2

We evaluate the performance of the proposed approach via simulations. We use the trace dataset from AU [66] as the background traffic which contains packets captured from the link connecting Auckland University and the Internet. The background traffic contains traffic of both forward and reverse directions and has an average rate of 523 packets per second. We firstly try to evaluate the effectiveness of our scheme under different attack types and finally evaluate the performance under mixed types of attacks. The filtering effectiveness Φ is defined as $\Phi = 1 - (F_n + F_p)/2$ where F_p and F_n are false positive rate and false negative rate, respectively and $0 \leq F_n, F_p \leq 1$. We consider Φ as the main performance metric. Unless otherwise noted, the

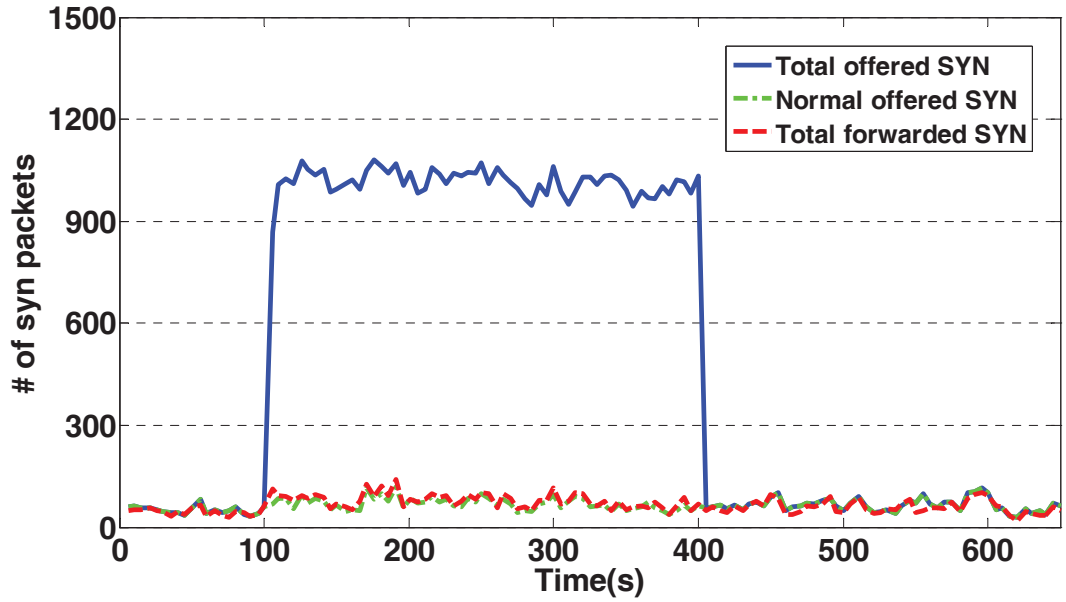


Figure 5.5: Effectiveness of mitigating spoofed IP DDoS attacks

default settings for the parameters are shown in Table 5.1.

Defense against spoofed IP flooding attacks

In order to evaluate the filtering effectiveness of the UTF module, we generated spoofed flooding traffic by attack tools we developed and then injected this traffic into the background traffic. The attack tool randomly chooses a SIP address from an IP address pool with size $N_{\text{spoof}} = 1000$ and sends SYN packets to the victim with default attack rate 200 SYN packets per second. The attack traffic is injected at the offset of 100 seconds and is terminated at the offset of 400 seconds. We set $TH_{SABF} = 0.1$, which means when the ratio of set bits in SABF is larger than 10%, we reset the SABF in order to reduce false positives. The results are shown in Fig. 5.5 by collecting statistics every 5 seconds. As we can see, nearly all of spoofed SYN packets are filtered by our UTF module without affecting normal SYN packets. In order to evaluate the accuracy performance

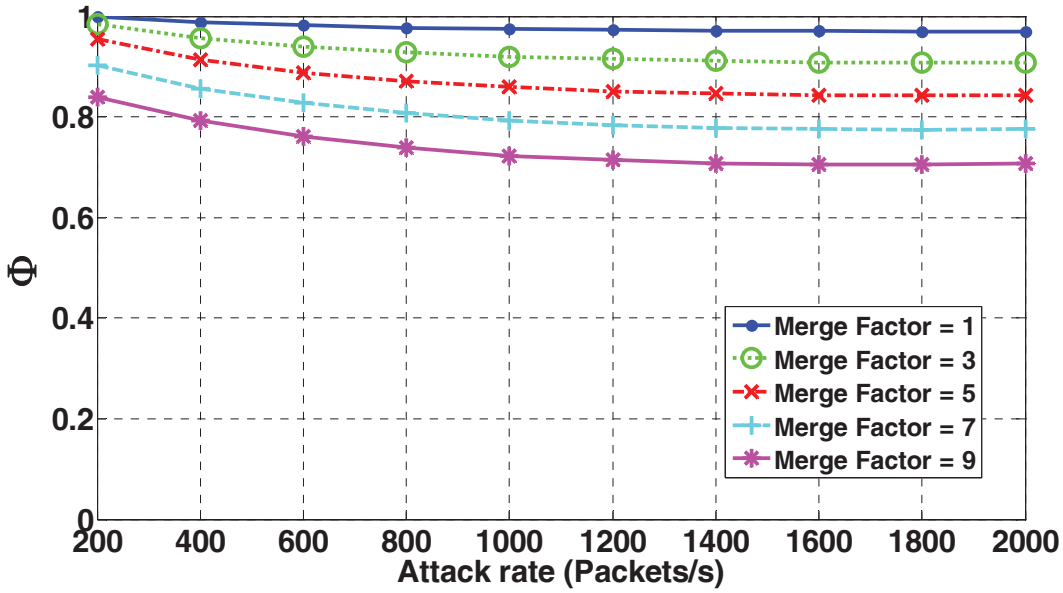


Figure 5.6: Accuracy evaluation by varying traffic scale

under different traffic scales, we shift time stamps of different periods of traces from AU and then merge them together in order to enhance the traffic intensity. We define merging factor (MF) as the number of different periods, which can also reflect the intensity of the traffic. Fig. 5.6 shows accuracy by varying the attack rate from 200 packets per second to 2000 packets per second. We can see the accuracy performance is quite stable even the traffic scale is large. For example, the Φ can still achieve over 70% under attack rate 2000 packets per second with $MF = 9$.

Defense against genuine IP flooding attacks

We further evaluate effectiveness of filtering SYN packets of genuine IP attack. We generate the genuine IP attack traffic in the following way. Each attack source sends 20 SYN packets per second to the victim and they will finish the three-way handshake with the victim without the following data exchanges. The SIP of each attack source is generated from an IP address pool with default

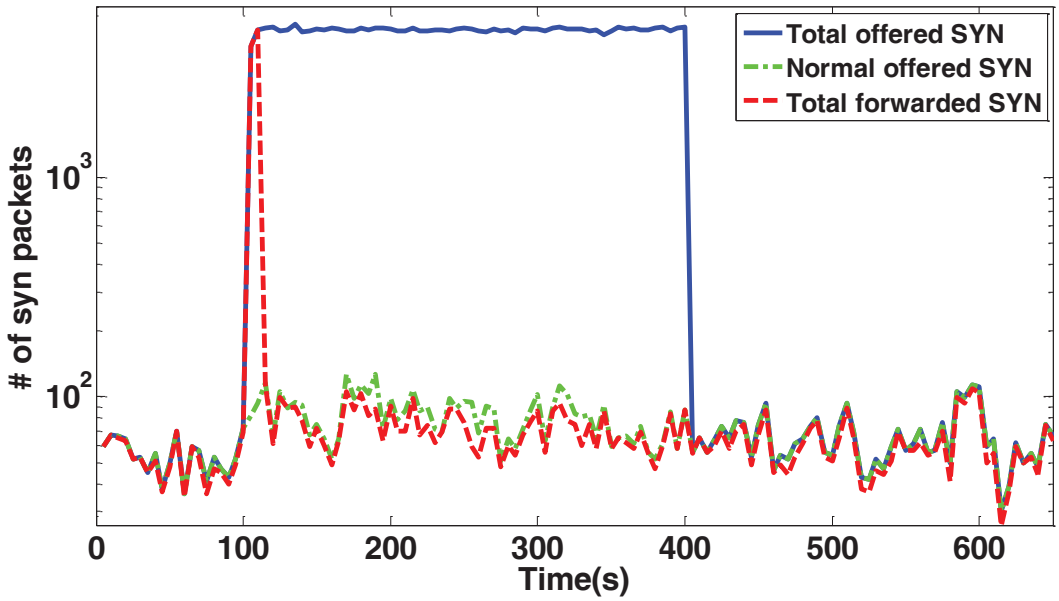


Figure 5.7: Effectiveness of mitigating genuine IP DDoS attacks

size $N_{genuine} = 1000$. Again, the attack traffic is injected at the offset of 100 seconds and is terminated at the offset of 400 seconds. Fig. 5.7 shows the experimental results when there are 100 attack sources. After the attack happens, the ATF module seems forward too many attack packets initially. This is due to the detection latency of the ATF module since it requires one detection interval to identify the victim in sketch and another detection interval for building the CBF. Our ATF module quickly filtered nearly all of the malicious SYN packets after two detection intervals from the beginning of attacks. In order to evaluate the scalability, we enhance the traffic intensity in a similar way as we do in experiments for filtering the spoofed IP attack. We vary the number of attack sources and measure the filtering effectiveness Φ as shown in Fig. 5.8. As we can see, the performance is quite stable as the traffic intensity grows. This is because we update CBFs only with the suspicious flows which are identified by the sketch. In this way, we achieve a scalable filtering. Also, the performance is also not sensitive to the increase of the number of

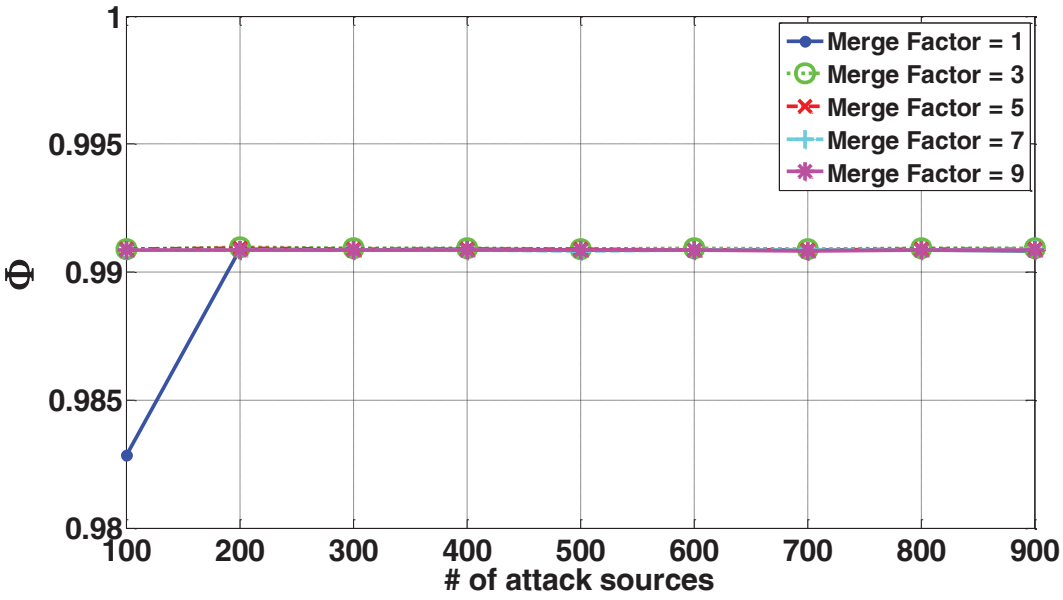


Figure 5.8: Accuracy evaluation by varying the number of attack sources

attack sources.

Defense against mixed types of attacks

The functions of UTF and ATF are not exchangeable. The UTF module can only filter spoofed IP attack effectively and genuine IP attack traffic needs to be further examined by ATF. A single ATF module can also perform filtering for the spoofed IP attack traffic, but it has relatively heavier computing overhead compared with UTF. We further evaluate the performance of accuracy by applying various defense deployments by embedding mixed attack traffic in the background traffic. We generate the mixed attack as follows. The spoofed IP attack traffic is sent at the rate of 500 SYN packets per second and the genuine IP attack traffic is sent by 50 attack sources. We obtain the filtering results by applying three deployments. They are deploying single UTF, deploying single ATF and deploying both of them. The associated numbers of forward SYN packets are

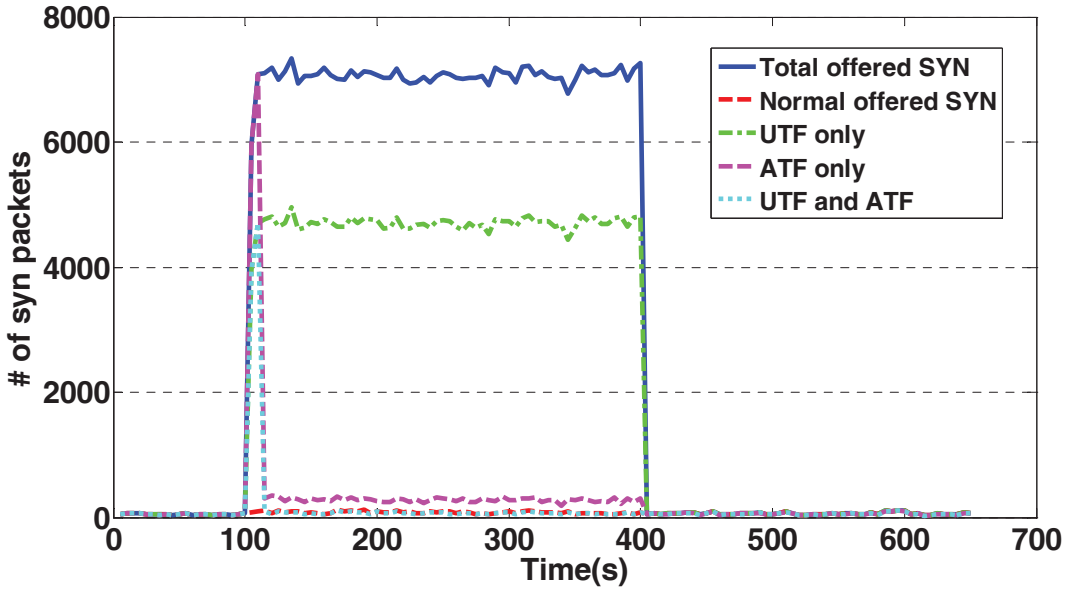


Figure 5.9: Evaluation of defense against mixed types of attacks

shown in Fig. 5.9. As we can see, by only deploying UTF in the system is not enough. Most attack traffic with genuine IPs will be forwarded normally. Also, although only placing the ATF module in the system can filter the majority of both spoofed and genuine IP attack traffic, the accuracy performance is not as good as deploying both of the two modules.

5.4 Defense against UDP flooding attacks

Various DDoS attacks are launched based on UDP protocol since UDP protocol is widely applied in the internet, such as DNS resolution, stream media, online game and so on to ensure the quality of service (QoS). It becomes one of the most effective methods for DDoS attack. The attackers usually exploit UDP's connectionless feature and its weakness in responding to the request. Such features are utilized by the attacker to submit a stream of UDP data packet to the target system which will fill the server's responding request queue. As a result, the server will refuse the new

response request and the legitimate users cannot get the response of the server as usual [84].

In this section, we evaluate the impact of UDP flooding attacks on GridStat in the "GridStat on Geni" project [85], and we further present a sketch-based defense scheme to fight against UDP Flooding attacks. The sketch structure introduced in the previous chapters is employed again to ensure the scalability performance. Our contribution mainly contains:

- We built a simulated environment for GridStat on Planet-Lab and evaluated the impact of UDP flooding attacks on GridStat in such environment. We will show that UDP flooding attacks do have a great impact on the QoS performance of GridStat, such as packet loss rate and delay.

- We further propose a solution for fighting against UDP flooding attacks. To the best of our knowledge, we are the first one to utilize sketch structure to achieve fine-grained filtering of UDP flooding traffic with small space consumption.

5.4.1 Background

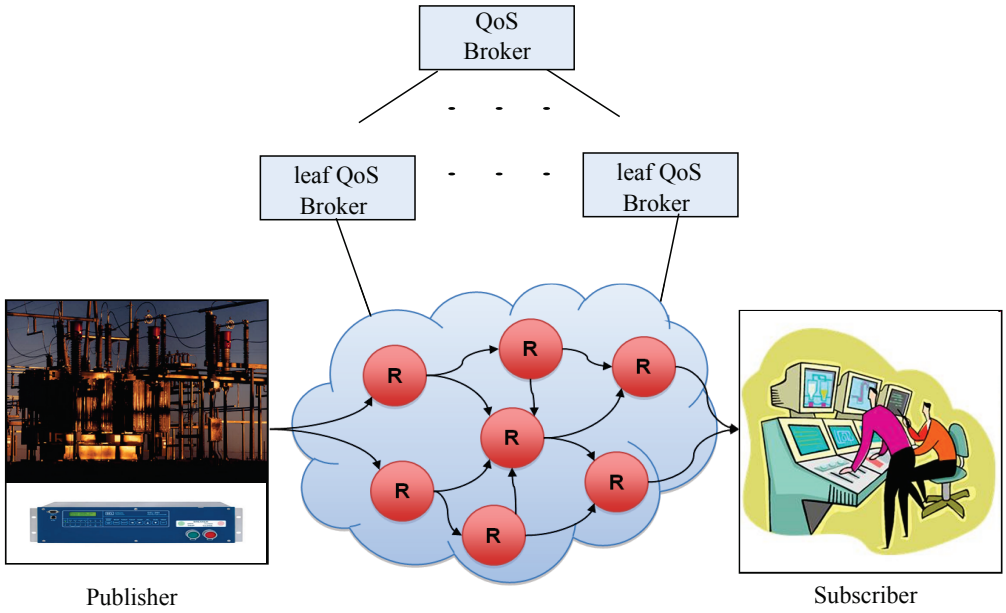


Figure 5.10: GridStat Architecture

GridStat [86, 87] is a framework for power-grid communication centered around a middleware network for power-grid data acquisition purpose. It is actually a status dissemination middleware, and it is usually a special type of publish-subscribe network. The publishers publish the measurements or signals from some electrical device in a power grid and the subscribers subscribe for data that they are interested in. The GridStat communication infrastructure consists of a number of status routers and the status routers are located between the publishers and the subscribers and are responsible for forwarding the information from the publisher side to the subscriber side. Futhermore, status routers in a single administrative domain forms a cloud and all the clouds are further governed by QoS brokers. The QoS brokers are involved in finding if certain requirements can be satisfied by the status routers. Fig. 5.10 represents the architecture of the GridStat. In order

to provides end-to-end QoS guarantees, the GridStat should be built based on protocol without congestion control such as UDP. Thus, UDP flooding attacks will become a realistic threat to the GridStat system. In this section, we evaluate the impact of UDP flooding attacks on the QoS performance of GridStat and propose a defense scheme for mitigating damage caused by the attacks.

5.4.2 Proposed Approach

Overall Architecture

As we mentioned in the previous section, a single host-based system is inherently not robust enough no matter where it is deployed. This is mainly due to the fact that a single host-based scheme can be easily fooled by a sophisticate attacker, which can be considered as an intentional internet event. Attackers will always try to employ a large number of compromised machines around the world to launch attacks and traffic that comes from every corner of the world can be routed by different edge routers inside an AS. As a result, if we only take a single router into accounts, the volume of attack traffic might not be aggregated at a detectable level for a detection module while the final gather of attack traffic will still cause serious damage to victim servers. Therefore, a collaborative approach which can comprehensively consider the global circumstance will be a more attractive solution for defense.

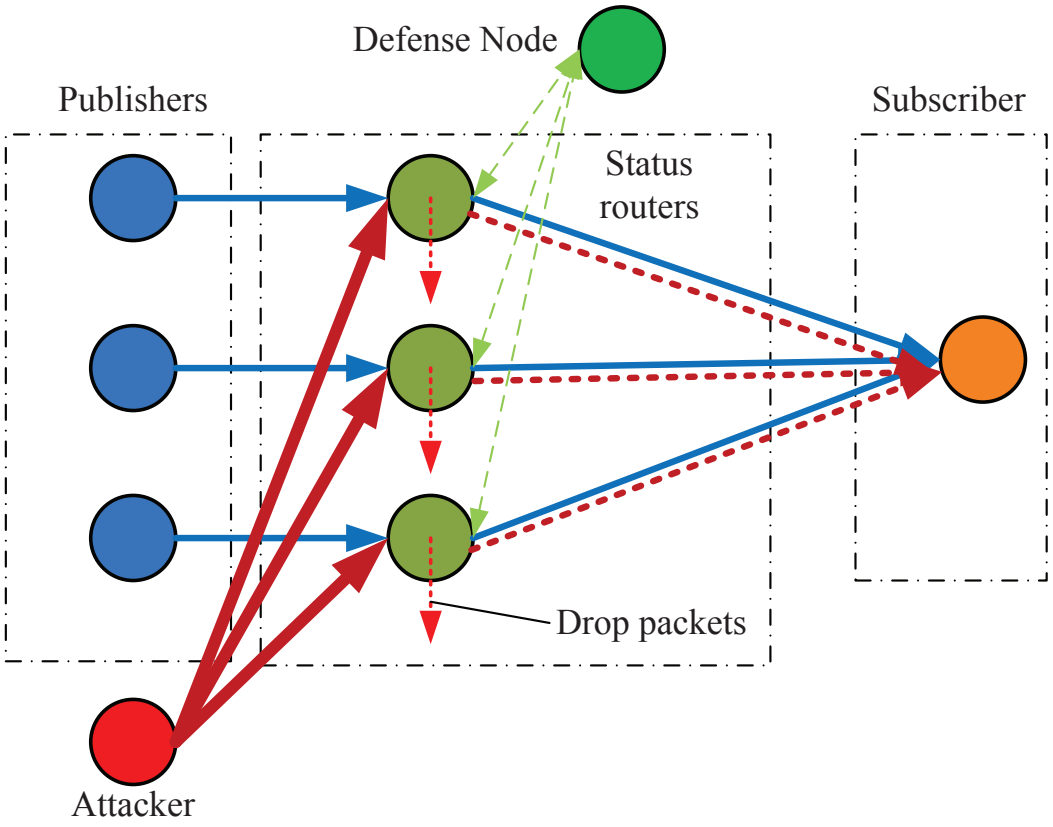


Figure 5.11: Illustration of a collaborative framework for mitigating UDP flooding attacks in GridStat

Our proposed approach is precisely designed based on a collaborative architecture. Fig. 5.11 illustrates the overall collaborative framework and we implement the prototype of GridStat on the PlanetLab environment to evaluate the performance of our detection module. The implemented GridStat prototype currently consists of publishers, status routers and subscribers which are all critical components in the GridStat network and it can well simulate the real working flow of GridStat. The publishers read data from trace files which were captured in the real GridStat environment and send status messages to subscribers through status routers. Furthermore, one defense node and one attacker node on the PlanetLab are also implemented. Each legitimate publisher

periodically sends status message to target subscribers through status routers. The status routers periodically collect and build local traffic statistics and report them to the defense node. The defense node will then send feedback messages to instruct status routers to drop those malicious flows based on the comprehensive global decision made from the reported statistics. To be specific, the functionality of each key component in the defense framework is described as below.

Local defense modules deployed in status routers The local defense modules we implemented in status routers are mainly responsible for:

- Summarizing traffic statistics from partial or all packets from both of two directional links
- Report the summarized traffic statistics to the global defense module periodically
- Receive feedback instructions from the global defense module and perform probabilistic packet filtering
- Timely react to those DDoS events that can be detected at the local side

Global defense module The responsibility of global defense module we implemented mainly contains:

- Receive those statistics reports from local defense modules
- Perform anomaly bucket detection based on packet volume
- Evaluate anomaly buckets and calculate the dropping probability for those malicious buckets in sketch

- Send feedback instructions for filtering packets to local defense modules based on anomaly detection results

Abnormal Buckets Identification

In our scheme, each bucket in the sketch contains four values $(v_t, v_{t-\Delta t}, v_{\text{backup}}, Flag)$. v_t is the number of packets that are accumulated from $t - \Delta t$ to t, $v_{t-\Delta t}$ is the previous value of v_t, and v_{backup} is the value of v_t right before the alarm occurs (or null if there has been no alarm). $Flag$ is set to 1 whenever the alarm condition is satisfied; otherwise it is set to 0. Here, the definition of alarm conditions depends on the practical deployment, and we will further explain it when we describe Algorithm 6 below. For each incoming record, we update the sketch with $(k_i, 1)$ where k_i is the SIP|DIP and 1 represents the number of this incoming record. Here, we use $|$ as the string concatenation operation. In the sketch we employed in our framework, rather than returning the minimum value of v_t as the original sketch does, the $Sketch\_Query$ function in sketch returns the minimum value of $Flag$ among all the buckets corresponding to a specific SIP|DIP to indicate whether this flow is suspicious. As we can see, the sketch adopted here only requires $O(H \times K)$ cells, which is constant.

The main purpose of sketch is to detect items with abnormal frequency. Every packet must go through. When a new packet arrives, hash values of H hash functions are computed, and the corresponding buckets are updated; the value in each bucket is incremented by 1. This accumulation process repeats every Δt seconds. The alarm condition is tested for all $H \times K$ buckets periodically. If the alarm condition is satisfied, then the alarm flag associated with the bucket is set to 1. Whenever there is an alarm, the previous v value of the bucket is recorded in the

v_{backup} for determining whether the raised alarm is terminated or not.

Algorithm 6: Adjustment procedure of $Flag$

```
for k = 1, h = 1 to K, H do
    if Flag = 0 then
        v_t ← (1 − α)v_{t−Δt} + αv_t ;
        if v_t ≥ (1 + θ)v_{t−Δt} then
            Flag ← 1 ;
            v_backup ← v_{t−Δt} ;
        end
    else
        v_t ← (1 − α)v_backup + αv_t ;
        if v_t < (1 + θ)v_backup then
            Flag ← 0 ;
        end
    end
end
```

We use a light-weight EWMA technique to decide whether there is an anomaly in each bucket, as shown in Algorithm 6. For each bucket, if the bucket status is normal, then we estimate v_t with an EWMA parameter α. Whenever $v_t \geq (1+\theta)\overline{v_{t-\Delta t}}$, which is considered as the satisfaction of the alarm condition, an alarm is raised. θ is the parameter that represents the percentage above the estimated value that can be considered to be an indication of anomalous pattern. The procedure is different after an alarm was raised. In order to estimate when the generated alarm should be terminated, we need to compare the current value with the specific value right before the time that the alarm happened. Such specific value is recorded in v_{backup} before the alarm is generated. Also, rather than using the previous value $\overline{v_{t-\Delta t}}$, we estimate the $\overline{v_t}$ by v_{backup} in order to eliminate the impact of the anomaly on the next following $\overline{v_t}$ series.

Probabilistic Packet Filtering

For each incoming SIP|DIP key, we query the minimal value of alarm flag for this specific key. If $Sketch\_Query(key) = 1$, then there may be an anomaly associated with the associated flow. The more aggressive the flow is, the higher value those hashed buckets have. We need to punish more for those flows with more aggressive behavior. Thus, we assign higher dropping probability for those malicious flows associated with anomaly buckets compared with those normal flows. We maintain a hash table named "FlowDropProbHashTable" with SIP|DIP as key and dropping probability as value for sending feedback information to the subscriber. Whenever a key is identified as abnormal, the dropping probability will be calculated based on the values in the sketch and the probability with SIP|DIP will be inserted in to the hash table "FlowDropProbHashTable". The dropping probability for an abnormal key is calculated as follows.

$$\begin{cases} \overline{V_{Normal}} = \frac{\sum_{i=1}^{M} V_{Normal}^{i}}{M} \\ P_{Drop}^{i} = \frac{\left(V_{Abnormal}^{i} - \overline{V_{Normal}}\right)^{+}}{V_{Abnormal}^{i}} \end{cases} \tag{5.3}$$

Where $\overline{V_{Normal}}$ is the average value of all the buckets with Flag 0, and Δ^{+} is Δ if $\Delta > 0$ and 0 otherwise.

Another thread is maintained to periodically traverse the hash table and send the dropping probability information to those associated status routers to mitigate the damage.

5.4.3 Evaluation

The topology deployed on the PlanetLab is also shown in Fig. 5.11. We deploy three publishers, three status routers, one subscriber, one attacker node and one defense node. The node settings for the deployed topology are shown in Table 5.2.

Table 5.2: The node settings for the topology deployed on PlanetLab

Node role	**PlanetLab Node**
Publisher	planetlab-4.eecs.cwru.edu
	planet1.cs.rochester.edu
	lefthand.eecs.harvard.edu
Status router	planetlab3.eecs.northwestern.edu
	pluto.cs.brown.edu
	planetlab04.cs.washington.edu
Subscriber	plab4.eece.ksu.edu
Malicious node	planet5.cs.ucsb.edu
Defense node	planet-lab2.cs.ucr.edu

Table 5.3: The default parameter settings of sketch-based defense for UDP flooding attacks

Item	**Parameter**	**Setting value**
Interval for Periodical Sketch Construction	Δt	10s
Interval for Periodical Instruction Feedback	$\Delta t_{feedback}$	20s
Size of Sketch	H	10
	K	1024
Anomaly Buckets Detection	α	0.3
	θ	2
Legitimate node	Packet rate	5 packets/s
Malicious node	Packet rate	2500 packets/s

Unless otherwise noted, the default parameter settings for the experiments are shown in

the Table 5.3. We consider the packet loss rate, pass ratio of legitimate traffic and pass ratio of legitimate traffic measured at the subscriber and average delay as the main performance metrics.

Evaluation of packet loss rate

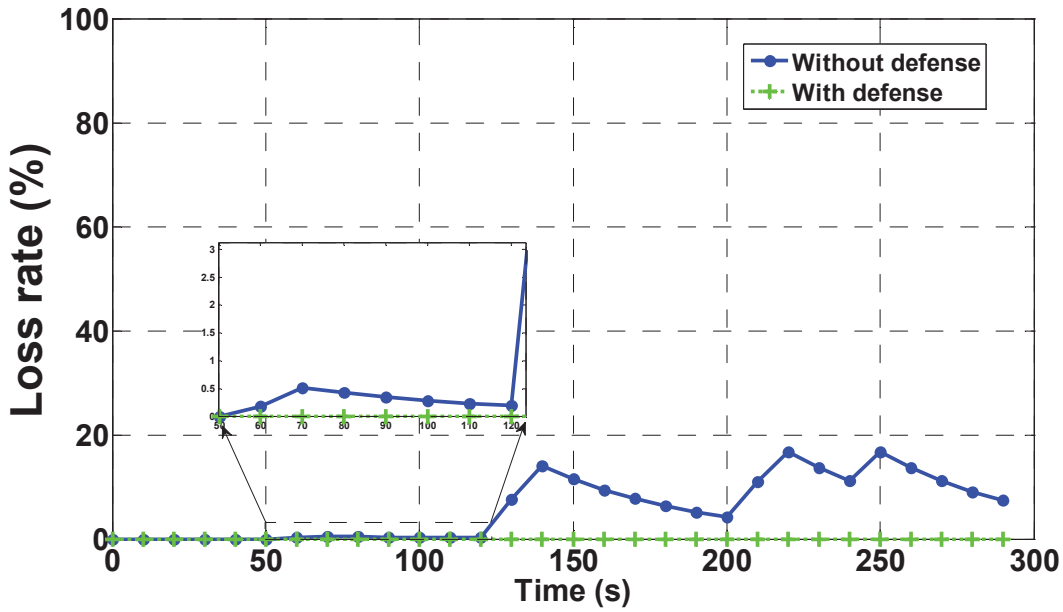


Figure 5.12: Evaluation of packet loss rate

In order to evaluate the impact of UDP flooding attacks on the packet loss, we made the malicious node send flooding traffic with the default attack rate 2500 packets per second. The packet loss rate is measured at the subscriber side. The malicious node starts sending flooding traffic at the offset of 60 seconds. We repeat the experiments after we deploy the defense node and compare the packet loss performance in order to evaluate the effectiveness of our defense method. The experimental results are shown in the Fig. 5.12. As we can see, without deploying the defense node, UDP flooding attacks do have great impact on the packet loss. The measured packet loss rate is around 18% in the worst case. Also, the packet loss rate is unstable during the attack period.

After we deploy the defense node, the average packet loss rate is greatly reduced and becomes steady.

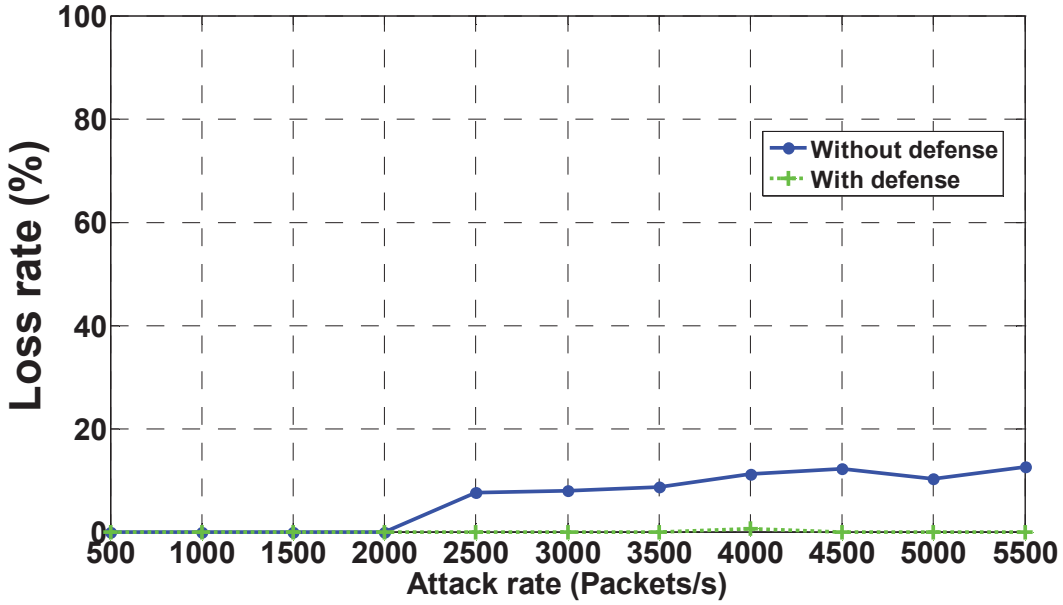


Figure 5.13: Loss rate VS. attack rate

We also measure the loss rate by varying the attack rate from 500 to 5500 packets per second which is shown in Fig. 5.13. As we can see, the loss rate slightly increases as we enhance the attack rate without deploying the defense node. On the other hand, the performance is unfluctuating if we employ the defense node.

Evaluation of accuracy

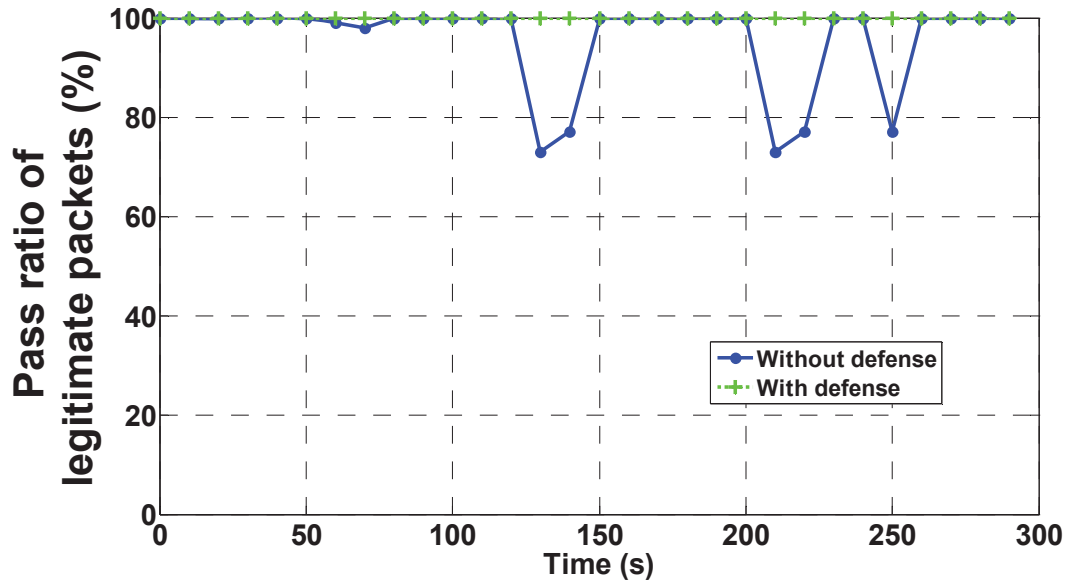


Figure 5.14: Evaluation of pass ratio of legitimate traffic

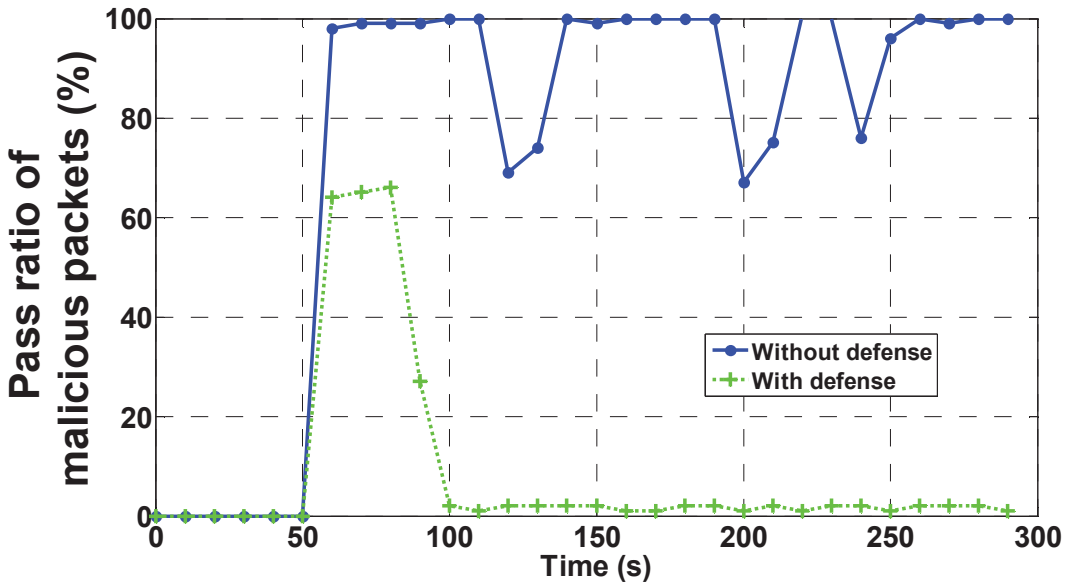


Figure 5.15: Evaluation of pass ratio of malicious traffic

We also evaluate the accuracy performance of our defense module. We measure the accuracy performance from two aspects, which are the pass ratio of legitimate traffic and malicious traffic measured at the subscriber side. Again, the malicious node starts sending flooding traffic at the offset of 60 seconds and we repeat the experiments after we deploy the defense node and compare the performance between them. The experimental results are shown in the Fig. 5.14 and Fig. 5.15. As we can see, without deploying the defense node, the majority of malicious traffic successfully arrives at the subscriber, which in turn saturates the subscriber's processing capability while the pass ratio of normal traffic becomes unstable due to the high packet loss in the network. After the defense node is deployed, most of the malicious packets are filtered by status routers which are instructed by the defense node while the normal traffic can reach the subscriber unmolested. We notice that there are some defense delays (around 30 seconds) for the defense node to take effect. This is because the defense node needs one interval for building current traffic profile and another one interval for sending feedback to status routers.

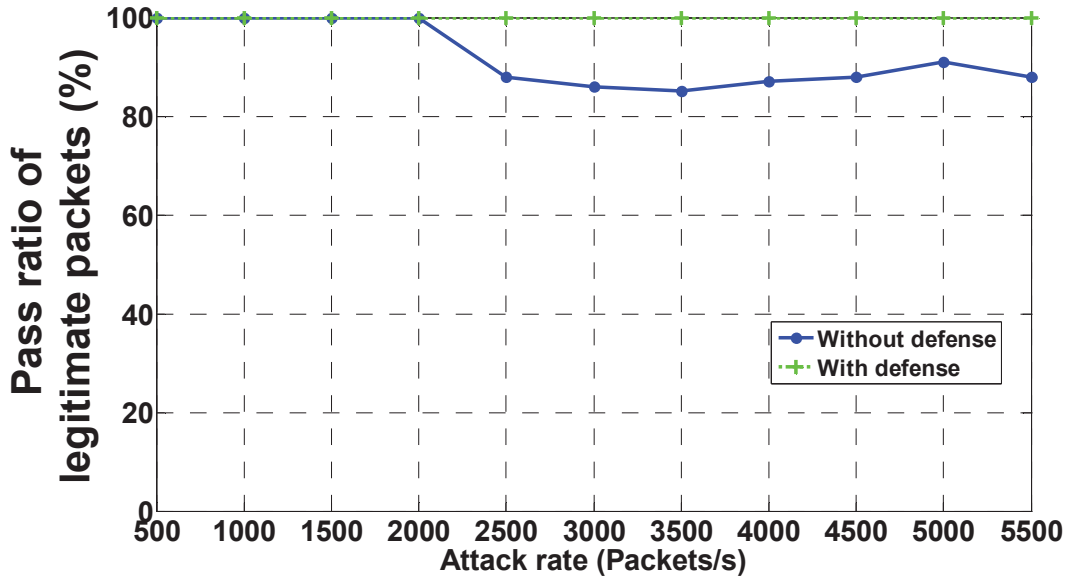


Figure 5.16: Pass ratio of legitimate traffic VS. attack rate

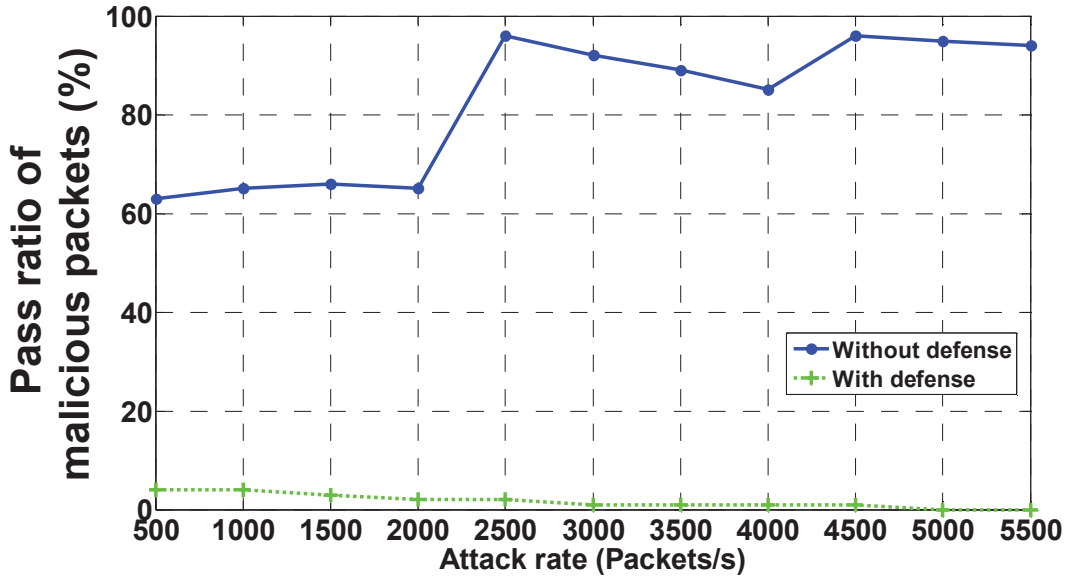


Figure 5.17: Pass ratio of malicious traffic VS. attack rate

We also measure the accuracy by varying the attack rate from 500 to 5500 packets per second which are shown in Fig. 5.16 and Fig. 5.17. As we can see, the pass ratio of legitimate traffic

will decrease as we enhance the attack rate without deploying the defense node. This is because the processing queues of status routers have been saturated by malicious packets and cannot forward all legitimate packets to the subscriber. After we deploy the defense node, the majority of malicious traffic is filtered. At the same time, the pass ratio of legitimate traffic is around 100% since the processing queues of status routers are now recovered to the normal level. We also notice that the pass ratio of malicious packets slightly decreases as the attack rate increases. This is because the drop probability of malicious flows will increase as they become more aggressive in sending packets in our detection module. The more aggressive the flow behaves, the more punishment it receives.

Evaluation of average delay

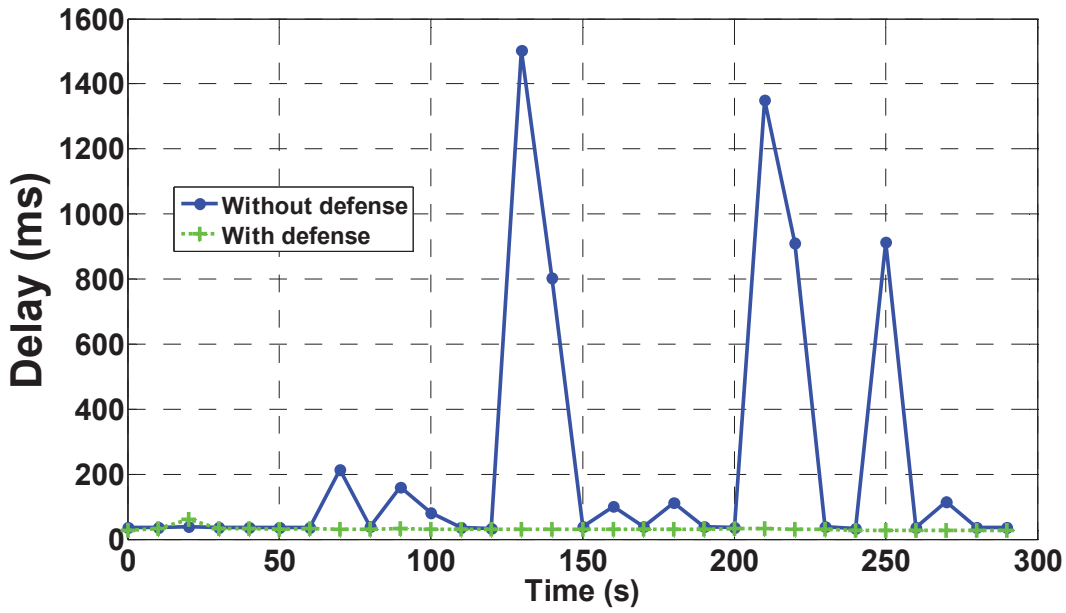


Figure 5.18: Evaluation of average delay

We finally measure the average delay at the subscriber side with the same experimental settings as described in the previous evaluation. Again, the UDP flooding attack starts at the offset of 60 seconds. As we can see from the Fig. 5.18, the average delay flitters after the attack starts without deploying the defense node. On the other hand, the delay is steady-going and kept at the low level when the defense node starts working.

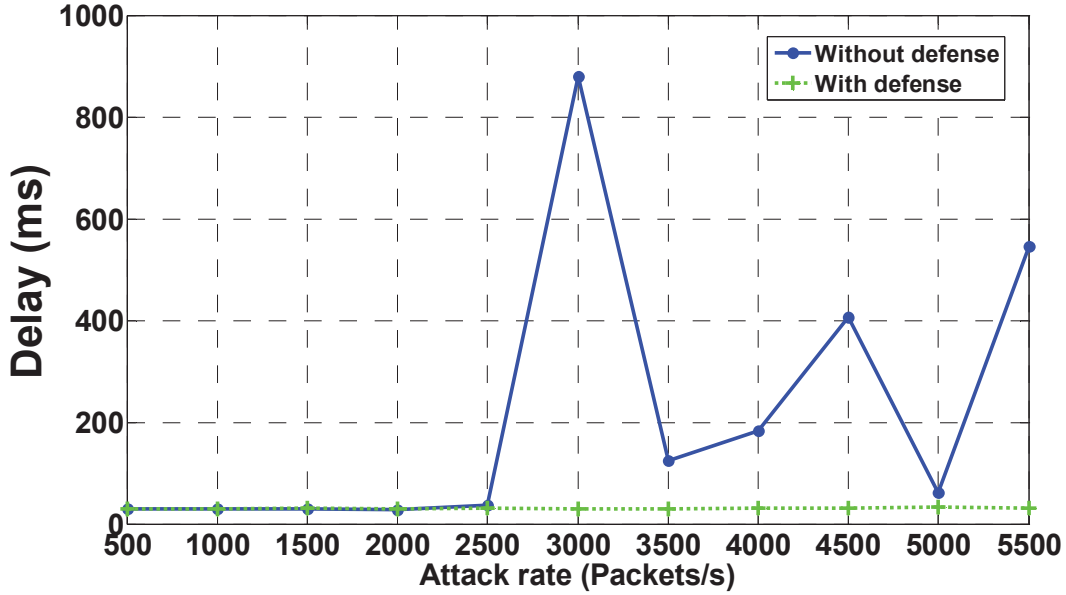


Figure 5.19: Average delay VS. attack rate

By varying the attack rate, we also evaluate the impact of attack rate on the average delay which is shown in Fig. 5.19. We can see that the average delay will increase as we enhance the attack rate without having the defense node work. The average delay becomes stable after we deploy the defense node.

5.5 Conclusion

In this chapter, we aimed at a collaborative defense scheme against both TCP and UDP flooding attacks, which are two of the most powerful and popular DDoS attacks. To be specific, we firstly present a two-stage defense scheme to mitigate TCP flooding attacks. At the UTF stage, the spoofed DDoS traffic is filtered by using two light-weight bloom filters. Then, malicious flows originated from genuine source addresses will be probabilistically dropped according to their data exchange behavior at the ATF stage. The main advantage of our approach is its space efficiency since it does not need to keep per-flow state. Moreover, both spoofed and genuine IP DDoS attacks can be well regulated. Futhermore, we built a simulated environment for GridStat on PlanetLab, evaluated the impact of UDP flooding attacks on the GridStat and proposed a sketch-based defense scheme to fight against UDP flooding attacks. Our approach employs the sketch structure to ensure the scalability performance and make identification of regulations of flows with low space consumption possible. The packets from malicious flows will be dropped more if they behave more aggressive. Experimental results demonstrate the effectiveness of the proposed framework.

CHAPTER SIX

CONCLUSION

In this thesis, an important issue is discussed: how to design an effective and efficient defense framework against DDoS attacks, which are probably the most serious threat to today's Internet. In order to address this issue, we proposed and evaluated several detection and defense schemes in this work. We do not only focus on the accuracy but also the scalability performance in our design.

6.1 Stealthy DDoS attacks detection

The very first step of our developments for the overall system is to accurately and efficiently detect when attacks happen. The research work of stealthy DDoS attack detection precisely fits into this step.

After introducing a new type of DDoS attacks called stealthy DDoS attacks, which can be launched by a sophisticated attacker, we propose a detection approach based on the decomposition of time series, which divides the time series extracted from FCE series into trend and steady random components. We then analyze different components to detect the anomaly in both long-term and short-term changes of the traffic. By applying different techniques to each component separately and evaluating results synthetically, the approach can greatly reduce both false negatives and false positives. Furthermore, to make our method more generally applicable, we apply the adaptive sliding window to our approach. The experiment results using real Internet traces show the effectiveness of this approach. To be specific, the best overall FPR in our experiment is around 4.3% and the overall FNR is around 9.8%.

6.2 TrustGuard

Only knowing when attacks happen is not enough. We need to have some ways to react to the attacks. Thus, knowing where the attacks happen is more important. In order to provide a fine-grained control over the traffic, we need to differentiate legitimate flows from malicious ones at the flow level, which is exactly the goal of development of TrushGuard.

The TrustGuard employs a two-tier model to reduce the size of the search space and make identification of specific attackers and victims possible. Our macro level detector can accurately identify suspected victims and the micro level detector can then confirm and refine those suspicions. We believe that this approach can accurately identify DDoS attacks down to the instigating flows and that this information can be used to improve firewall and IDS rules.

6.3 Sketch-based detection

Although the TrustGuard can provide flow level filtering, it still needs to keep per-flow state in order to accumulate credit for each flow. Thus, it cannot scale well with the high-speed traffic. Aimed at solving this issue, we further propose a sketch-based detection framework to achieve scalable performance in terms of space consumption while it still provides victim pinpoint capability.

The sketch-based detection scheme employs a two-level model to reduce both the size of the search space and time, and further make identification of specific victims possible in the high-speed network environment. We adopt the MCS structure in coarse-level detection to achieve fast detection, and the BCS structure in the fine-level to further guarantee the accuracy. We believe that this approach can accurately identify victims of DDoS attacks with a low memory footprint and

give a timely response. We also propose a SRAM-based parallel architecture to achieve high-speed process. We finally analyze accuracy estimation issue and demonstrate a collaborative detection scheme based on the original single-host detection scheme. Experimental results show that our scheme outperforms previous sketch-based methods with respect to both storage scalability and detection accuracy.

6.4 Sketch-based collaborative defense framework

Most of the previous works focused on the detection aspect. In this work, we developed a sketch-based collaborative defense framework to mitigate DDoS attacks. Aimed at defending against attacks launched with two main different protocols, we proposed defense schemes against both TCP and UDP flooding attacks.

We firstly present a two-stage defense scheme to mitigate TCP flooding attacks. At the UTF stage, the spoofed DDoS traffic is filtered by using two light-weight bloom filters. Then, malicious flows originated from genuine source addresses will be probabilistically dropped according to their data exchange behavior at the ATF stage. The main advantage of our approach is its space efficiency since it does not need to keep per-flow state. Moreover, both spoofed and genuine IP DDoS attacks can be well regulated.

In order to fight against threads from UDP flooding attacks, we further built a simulated environment for GridStat on PlanetLab, evaluated the impact of UDP flooding attacks on the GridStat and proposed a sketch-based defense scheme to fight against UDP flooding attacks. Our approach employs the sketch structure to ensure the scalability performance and make identification of reg-

ulations of flows with low space consumption possible. The packets from malicious flows will be dropped more if they behave more aggressive. Experimental results demonstrate the effectiveness of the proposed framework.

Compared with previous state-of-the-art defense frameworks, our framework solves the three existing key issues in previous approaches. (a) We proposed several detection schemes to accurately detect when and where attacks happen. They can be deployed at the victim side to raise alerts with small delay to ensure timely reaction to attacks. Also, they can be used as a supplement of current IDS systems to reduce the heavy weight flow-level inspections. This is because those detection modules can be placed before defense modules and the defense modules can only be triggered when the attack event is detected. Furthermore, detection at an early stage is critical for timely response to attack events at the local side. Whenever attacks happen, local detection modules will directly react to the malicious flows without interrupting the global module. (b) Our framework was proposed to protect an edge network rather than Internet scale network. By this way, the implementation feasibility can be greatly enhanced since all the edge routers belong to a single AS which make them possible to collaborate with each other. Also, the framework can be incrementally deployed across the whole Internet in the future. (c) The deployment incentives can also be maximized since the protected edge network belongs to the same AS. However, the previous approach requires source-end edge routers to throttle malicious traffic in order to protect victim-end edge networks. Thus, the deployment incentives are few which render it hard to be implemented in practice.

6.5 Future works

The future works of our research work mainly fall into four aspects which are described below.

Firstly, since there are many parameter settings in our conducted experiments and the performance of each individual detection and defense module heavily depends on the settings, how to develop an automatic and adaptive parameter adjustment mechanism is definitely a critical issue that needs to be considered when we want to deploy our modules in reality.

Secondly, there exist a number of malicious events besides TCP and UDP flooding attacks such as ICMP flooding, ACK flooding and scanning activities. Since those attacks also have high frequency of certain feature, the sketch structure, which is used to detect frequency anomalies, can definitely be applied to detect these attacks. Thus, how to extend our current sketch-based DDoS defense framework to defend against these attacks is another issue that needs to be addressed. In fact, we are currently working on developing sketch-based methods to detect scanning activities and experiments have shown the effectiveness of sketch-based methods.

Thirdly, our modules need to be integrated in the existing intrusion detection systems such as Snort or Bro. The main functionalities of current Snort and Bro focus on packet classification and deep packet inspection. How these existing modules will impact the performance of our modules still needs to be evaluated when we integrate them into the system. For example, since the existing modules require certain amount of space consumption, the remaining memory for our modules may be at a very low level.

Last but not least, current framework is still implemented on a relatively stable environment. There are several issues that need to be considered when we deploy the developed framework

on the real Internet. (a) To what extent can our system be resilient to unexpected hardware failure? (b) To what extent can the network congestion impact the performance of our framework when we deploy it in reality? (c) Since our current framework potentially suffers from a single point of failure problem, certain mechanism that is able to provide backup and recovery ability is definitely necessary for a robust overall system.

BIBLIOGRAPHY

[1] Jelena Mirkovic, Gregory Prier, and Peter Reiher. Attacking DDoS at the source. In *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP)*, pages 312–321, 2002.

[2] John Ioannidis and Steven M. Bellovin. Implementing pushback: Router-based defense against DDoS attack. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2002.

[3] George Oikonomou, Jelena Mirkovic, Peter Reiher, and Max Robinson. A framework for a collaborative DDoS defense. In *Proceedings of Computer Security Applications Conference (ACSAC)*, pages 33–42, 2006.

[4] Jelena Mirkovic and Peter Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53, Apr 2004.

[5] Chang and R.K.C. Defending against flooding-based distributed denial-of-service attacks: a tutorial. *IEEE Communications Magazine*, 40(10):42–51, Oct 2002.

[6] David Moore, Geoffrey M. Voelker, and Stefan Savage. Inferring internet denial-of-service activity. In *Proceedings of USENIX Security*, 2001.

[7] Tuomas Aura, Pekka Nikander, and Jussipekka Leiwo. DoS-resistant authentication with client puzzles. *Lecture Notes In Computer Science*, 2133:170–177, 2000.

[8] Chen Y. W, Hsiang K. S, and Hsieng T. Y. Study on the prevention of SYN flooding by using traffic policing. In *Proceedings of Network Operations and Management Symposium*, pages 593–604, Hawaii, Apr 2000.

[9] Xin Liu, Xiaowei Yang, and Yanbin Lu. To filter or to authorize: Network-layer DoS defense against multimillion-node botnets. In *Proceedings of ACM SIGCOMM*, 2008.

[10] B. Al-Duwairi. *Mitigation and Traceback Countermeasures for DDoS Attacks*. Ph.d. dissertation, Iowa State University, 2005.

[11] PlanetLab. `http://planet-lab.org/`.

[12] Yanxiang He, Wei Chen, Wenling Peng, and Bin Xiao. An efficient and practical defense method against DDoS attack at the source-end. In *Proceedings of the 11th International Conference on Parallel and Distributed Systems - Workshops - Volume 02*, pages 265–269, Washington, DC, USA, 2005. IEEE Computer Society.

[13] K. Park and H. Lee. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. In *Proceedings of ACM SIGCOMM*, 2001.

[14] S. Chen and Q. Song. Perimeter-based defense against high bandwidth DDoS attacks. *IEEE Trans. Parallel Distrib. Syst.*, 16(6):526–537, 2005.

[15] Kejie Lu, Dapeng Wu, and Jieyan Fan. Robust and efficient detection of DDoS attacks for large-scale internet. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 51(18):5036–5056, December 2007.

[16] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical network support for IP traceback. In *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 295–306, New York, NY, USA, 2000. ACM.

[17] Dongwon Seo, Heejo Lee, and A. Perrig. PFS: Probabilistic filter scheduling against distributed denial-of-service attacks. In *Proceedings of the IEEE 36th Conference on Local Computer Networks (LCN)*, pages 9–17, October 2011.

[18] J. Francois, I. Aib, and R. Boutaba. FireCol: A collaborative protection network for the detection of flooding DDoS attacks. *IEEE/ACM Transactions on Networking*, 20(6):1828–1841, December 2012.

[19] P. Ayres, H. Sun, and H. Chao. ALPi: A DDoS defense system for high-speed networks. *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, 24(10):1864–1876, 2006.

[20] L. Kencl and C. Schwarzer. Traffic-adaptive packet filtering of denial of service attacks. In *Proceedings of the 2006 International Symposium on WOWMOM*, June 2006.

[21] Huizhong Sun, Wingchiu Ngan, and H. Jonathan Chao. RateGuard: A robust distributed denial of service (DDoS) defense system. In *Proceedings of Globecom2009*, 2009.

[22] Haining Wang, Danlu Zhang, and Kang G. Shin. Detecting SYN flooding attacks. In *Proceedings of IEEE INFOCOM*, 2002.

[23] Cheng Jin, Haining Wang, and Kang G. Shin. Hop-count filtering: An effective defense against spoofed DDoS traffic. In *Proceedings of the 10th ACM conference on Computer and communications security*, pages 30–41, New York, NY, USA, 2003. ACM.

[24] Yoohwan Kim, Wing Cheong Lau, Mooi Choo Chuah, and H. Jonathan Chao. PacketScore: Statistics-based overload control against distributed denial-of-service attacks. In *Proceedings of IEEE INFOCOM*, 2004.

[25] Q Li, EC Chang, and MC Chan. On the effectiveness of DDoS attacks on statistical filtering. In *Proceedings of IEEE INFOCOM*, 2005.

[26] Lei Liu, Xiaolong Jin, Geyong Min, and Li Xu. Real-time diagnosis of network anomaly based on statistical traffic analysis. In *Proceedings of the IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 264–270, June 2012.

[27] Mohit Mehta, Kanika Thapar, George Oikonomou, and Jelena Mirkovic. Combining speak-up with defCOM for improved DDoS defense. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 1708–1714, 2008.

[28] Michael Walfish, Mythili Vutukuru, Hari Balakrishnan, David Karger, and Scott Shenker. DDoS defense by offense. In *Proceedings of ACM SIGCOMM*, 2006.

[29] Jake D. Brutlag. Aberrant behavior detection in time series for network monitoring. In *Proceedings of LISA XIV*, Dec 2000.

[30] Haiqin Liu and Min Sik Kim. Real-time detection of stealthy DDoS attacks using time-series decomposition. In *Proceedings of IEEE International Conference on Communications 2010*, May 2010.

[31] A. Kuzmanovic and E. Knightly. Low-rate TCP-targeted denial of service attacks. (the shrew vs. the mice and elephants). In *Proceedings of ACM SIGCOMM*, 2003.

[32] C.M. Cheng, H.T. Kung, and K.S. Tan. Use of spectral analysis in defense against DoS attacks. In *Proceedings of 2002 IEEE GLOBECOM*, Taipei, China, 2002.

[33] Yu Chen and Kai Hwang. Collaborative detection and filtering of shrew DDoS attacks using spectral analysis. *Journal of Parallel and Distributed Computing*, 66(9):1137–1151, Sep 2006.

[34] X. Luo and R. K. Chang. On a new class of pulsing denial-of-service attacks and the defense. In *Proceedings of Network and Distributed System Security Symposium*, San Diego, CA, Feb 2005.

[35] M. Guirguis, A. Bestavros, and I. Matta. Exploiting the transients of adaptation for RoQ attacks on internet resources. In *Proceedings of IEEE ICNP*, pages 184–195, Berlin, Germany, Oct 2004.

[36] S. Ebrahimi-Taghizadeh, A. Helmy, and S. Gupta. TCP vs. TCP: a systematic study of adverse impact of short-lived TCP flows on long-lived TCP flows. In *Proceedings of IEEE INFOCOM 2005*, pages 926–937, Miami, USA, Mar 2005.

[37] J. D. Hamilton. *Time Series Analysis*. Princeton University Press, 1994.

[38] Cheng Guang, Gong Jian, and Ding Wei. A time-series decomposed model of network traffic. *Lecture Notes in Computer Science*, pages 338–345, 2005.

[39] M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes : Theory and Application*. Prentice Hall, 1993.

[40] B.E. Brodsky and B.S. Darkhovsky. *Nonparametric Methods in Changepoint Problems*. Kluwer Academic Publishers, 1993.

[41] J. W. Haines, R. P. Lippmann, D. J. Fried, M. A. Zissman, E. Tran, and S. B. Boswell. 1999 DARPA intrusion detection evaluation: Design and procedures. Technical Report 1062, Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Massachusetts, U.S.A., February 2001.

[42] Jun Yang, Ying Li, Benxiong Huang, and Jiuqiang Ming. Preventing DDoS attacks based on credit model for P2P streaming system. In *Proceedings of the 5th International Conference on Autonomic and Trusted Computing*, 2008.

[43] Jayashree Padmanabhan, K. S. Easwarakumar, Gokul B., and Harishankar S. Trust based traffic monitoring approach for preventing denial of service attacks. In *Proceedings of the 2nd International Conference on Security of Information and Networks*, 2009.

[44] Maitreya Natu and Jelena Mirkovic. Fine-grained capabilities for flooding DDoS defense using client reputations. In *LSAD '07: Proceedings of the 2007 workshop on Large scale attack defense*, New York, NY, USA, 2007. ACM.

[45] Haiqin Liu, Yan Sun, Victor C. Valgenti, and Min Sik Kim. TrustGuard: A flow-level reputation-based DDoS defense system. In *Proceedings of the 5th IEEE International Workshop on Personalized Networks*, Las Vegas, January 2011.

[46] Paul Hick, Emile Aben, Kc Claffy, and Josh Polterock. The CAIDA DDoS Attack 2007 Dataset. `http://www.caida.org/data/passive/ddos-20070804_dataset.xml` (accessed on 2010-02-28).

[47] Colleen Shannon, Emile Aben, kc claffy, and Daniel E Andersen. CAIDA Anonymized 2008 Internet Traces Dataset (20081120). `http://www.caida.org/data/passive/passive_2008_dataset.xml` (accessed on 2010-02-28).

[48] George Nychis, Vyas Sekar, David G. Andersen, Hyong Kim, and Hui Zhang. An empirical evaluation of entropy-based traffic anomaly detection. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, New York, NY, USA, 2008.

[49] Z. Morley Mao, Vyas Sekar, Oliver Spatscheck, Jacobus van der Merwe, and Rangarajan Vasudevan. Analyzing large DDoS attacks using multiple data sources. In *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*, Pisa, Italy, 2006.

[50] H. Rahmani, N. Sahli, and F. Kammoun. Joint entropy analysis model for DDoS attack detection. In *Proceedings of the Fifth International Conference on Information Assurance and Security*, August 2009.

[51] H. Wang, D. Zhang, and K. G. Shin. Change-point monitoring for the detection of DoS attacks. *IEEE Transactions on Dependable and Secure Computing*, 1(4):193–208, October 2004.

[52] Tu Xu, Da Ke He, and Yu Zheng. Detecting DDoS attack based on one-way connection density. In *Proceedings of the 10th IEEE Singapore International Conference on Communication systems*, October 2006.

[53] M. Roesch. Snort - lightweight intrusion detection for networks. In *Proceedings of the 13th USENIX Conference on System Administration*, November 1999.

[54] V. Paxson. Bro: A system for detecting network intruders in real-time. *Computer Networks*, 31(23–24):2435–2463, December 1999.

[55] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies. In *Proceedings of ACM SIGCOMM*, August 2004.

[56] Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining anomalies using traffic feature distributions. In *Proceedings of ACM SIGCOMM*, August 2005.

[57] Ramana Rao Kompella, Sumeet Singh, and George Varghese. On scalable attack detection in the network. *IEEE/ACM Transactions on Networking*, 15(1):14–25, February 2007.

[58] Robert Schweller, Ashish Gupta, Elliot Parsons, and Yan Chen. Reversible sketches for efficient and accurate change detection over network data streams. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, pages 207–212, Taormina, Sicily, Italy, October 2004.

[59] R. Schweller, Zhichun Li, Yan Chen, Yan Gao, A. Gupta, Yin Zhang, P. Dinda, Ming-Yang Kao, and G. Memik. Reverse hashing for high-speed network monitoring: Algorithms, evaluation, and applications. In *Proceedings of IEEE INFOCOM*, April 2006.

[60] Osman Salem, Sandrine Vaton, and Annie Gravey. A scalable, efficient and informative approach for anomaly-based intrusion detection systems: theory and practice. *International Journal of Network Management*, 20:271–293, September 2010.

[61] S. Ganguly, M. Garofalakis, R. Rastogi, and K. Sabnani. Streaming algorithms for robust, real-time detection of DDoS attacks. In *Proceedings of the 27th International Conference on Distributed Computing Systems*, June 2007.

[62] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. QuickSAND: Quick summary and analysis of network data. Technical Report 2011-43, DIMACS, 2001.

[63] Shui Yu, Wanlei Zhou, Weijia Jia, Song Guo, Yong Xiang, and Feilong Tang. Discriminating DDoS attacks from flash crowds using flow correlation coefficient. *IEEE Transactions on Parallel and Distributed Systems*, 23(6):1073–1080, June 2012.

[64] Haiqin Liu, Yan Sun, and Min Sik Kim. Fine-grained DDoS detection scheme based on bidirectional count sketch. In *Proceedings of IEEE International Conference on Computer Communication Networks*, Hawaii, August 2011.

[65] Haiqin Liu, Yan Sun, and Min Sik Kim. A scalable DDoS detection framework with victim pinpoint capability. *Journal of Communications*, 6(9):660–670, December 2011.

[66] Auckland-IV trace data, 2001. `http://wand.cs.waikato.ac.nz/wand/wits/auck/4/`.

[67] S. Sarvotham, R. Riedi, and R. Baraniuk. Network traffic analysis and modeling at the connection level. In *Proceedings of Internet Measurement Workshop*, San Francisco, November 2001.

[68] Kuzmanovic Aleksandar and Knightly Edward W. Low-rate TCP-targeted denial of service attacks and counter strategies. *IEEE/ACM Transactions on Networking*, 14:683–696, August 2006.

[69] Carter J. Lawrence and Wegman Mark N. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.

[70] Xiaowei Yang, David Wetherall, and Thomas Anderson. A doS-limiting network architecture. In *Proceedings of SIGCOMM*, 2005.

[71] Angelos D. Keromytis, Vishal Misra, and Dan Rubenstein. SOS: An architecture for mitigating dDoS attacks. *IEEE Journal on Selected Areas in Communications*, 22(1), January 2004.

[72] Christos Papadopoulos, Robert Lindell, John Mehringer, and Alefiya Hussain. COSSACK: Coordinated suppression of simultaneous attacks. In *Proceedings of DISCEX*, pages 2–13, 2003.

[73] D.K.Y. Yau, J. C. S. Lui, and F. Liang. Defending against distributed denial of service attacks with max-min fair server-centric router throttles. *IEEE/ACM Transactions on Networking*, 13(1):29–42, 2005.

[74] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Survey of network-based defense mechanisms countering the DoS and DDoS problems. *ACM Computing Surveys*, 39(1), 2007.

[75] Jonathan Lemon. Resisting SYN flood DoS attacks with a SYN cache. In *Proceedings of the USENIX BSDCon*, 2002.

[76] Yuichi Ohsita, Shingo Ata, and Masayuki Murata. Deployable overlay network for defense against distributed SYN flood attacks. In *Proceedings of Internatonal Conference on Computer Communications and Networks*, 2005.

[77] SYN cookies.

[78] Ping Du and Akihiro Nakao. Overcourt: DDoS mitigation through credit-based traffic segregation and path migration. *Computer Communications*, 33(18):2164–2175, 2010.

[79] Xin Liu, Xiaowei Yang, and Yong Xia. NetFence: Preventing internet denial of service from inside out. In *Proceedings of ACM SIGCOMM*, 2010.

[80] Yoohwan Kim, Wing Cheong Lau, Mooi Choo Chuah, and H. Jonathan Chao. PacketScore: Statistical-based overload control against distributed denial-of-service attacks. In *Proceedings of IEEE INFOCOM*, 2004.

[81] Qiming Li, Ee-Chien Chang, and Mun Choon Chan. On the effectiveness of DDoS attacks on statistical filtering. In *Proceedings of IEEE INFOCOM*, 2005.

[82] Xiaowei Yang, David Wetherall, and Thomas Anderson. TVA: A DoS-limiting network architecture. *IEEE/ACM Transactions on Networking*, 16(6):1267–1280, 2008.

[83] Changhua Sun, Chengchen Hu, Yi Tang, and Bin Liu. More accurate and fast SYN flood detection. In *Proceedings of the 18th Internatonal Conference on Computer Communications and Networks*, 2009.

[84] Juniper Networks. Denial of service and attack protection. white paper, 2006.

[85] Ruma Paul, Divya Giri, Haiqin Liu, Victor Valgenti, Carl Hauser, and Min Sik Kim. GridStat on GENI: Simulating a smart power grid infrastructure over GENI. First DFG/GENI Doctoral Consortium, 2011.

[86] GridStat. `http://www.gridstat.net`.

[87] Carl H. Hauser, David E. Bakken, Ioanna Dionysiou, K. Karalrd Gjermud, Venkata S. Irava, Joel Helkey, and Anjan Bose. Security, Trust, and QoS in Next-Generation Control and Communication for Large Power Systems. *International Journal of Critical Infrastructures*, 2008.

Appendix ONE

PUBLICATIONS

A.1 Journal

- **Haiqin Liu**, Yan Sun, Min Sik Kim, "A Scalable DDoS Detection Framework with Victim Pinpoint Capability", Journal of Communications, Vol. 6(9), pp. 660-670. Dec. 2011.
- **Haiqin Liu**, Min Sik Kim, "Fine-Grained Defense against DDoS Attacks Using Sketch and Bloom Filters", prepared to be submitted to a journal.

A.2 Conference

- **Haiqin Liu**, Yan Sun, Min Sik Kim, "Fine-Grained DDoS Detection Scheme Based on Bidirectional Count Sketch", In Proceeding of the International Conference on Computer Communication Networks (ICCCN'11), Aug. 2011.
- Yan Sun, **Haiqin Liu**, Min Sik Kim, "Using TCAM Efficiently for IP Route Lookup", In Proceeding of the 8th IEEE Consumer Communications & Networking Conference (CCNC'11), Jan. 2011.
- **Haiqin Liu**, Yan Sun, Victor Valgenti, Min Sik Kim, "TrustGuard: A Flow-level Reputation-based DDoS Defense System", In Proceedings of the 5th IEEE International Workshop on Personalized Networks (PerNets 2011) , Jan. 2011.
- **Haiqin Liu**, Yan Sun, Min Sik Kim, "Provider-Level Content Migration Strategies in P2P-Based Media Distribution Networks", In Proceedings of the 3rd IEEE International Work-

shop on Digital Entertainment, Networked Virtual Environments, and Creative Technology (DENVECT 2011) , Jan. 2011.

- Yan Sun, **Haiqin Liu**, Victor Valgenti, Min Sik Kim, "Hybrid Regular Expression Matching for Deep Packet Inspection on Multi-core Architecture", In Proceeding of the International Conference on Computer Communication Networks (ICCCN'10), Aug. 2010.
- **Haiqin Liu**, Min Sik Kim, "Real-Time Detection of Stealthy DDoS Attacks Using Time-Series Decomposition", In Proceedings of IEEE International Conference on Communications 2010 (ICC'10), May 2010.
- Yan Sun, **Haiqin Liu**, Min Sik Kim, "Energy-Efficient Routing Protocol in Event-Driven Wireless Sensor Networks", In Proceeding of the IEEE ICC Workshop on Energy Efficiency in Wireless Networks & Wireless Networks for Energy Efficiency (E2Nets 2010), May. 2010.

A.3 Poster

- **Haiqin Liu**, Min Sik Kim, "TrustGuard: a flow-level reputation-based DDoS Defense System", Poster in Academic Showcase of WSU, Pullman, WA, 2010.
- **Haiqin Liu**, Min Sik Kim, "A new approach based on FFT for path selection in networks", Poster in Academic Showcase of WSU, Pullman, WA, 2009.