AN EFFECTIVE NETWORK FLOW PRIORITIZATION APPROACH IN DDOS SCENARIOS

BY

SHU LIU

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Adviser:

Associate Professor Yih-Chun Hu

ABSTRACT

In the modern Internet, Distributed Denial-of-Service (DDoS) has been a constant threat to all web services. An attacker may send a flood of requests with a botnet to the victim system, so that the system will waste all resources processing the huge number of unnecessary requests generated by an attacker and will not be available to any normal user. To mitigate DDoS attacks, it is crucial for a defense mechanism to have a policy that can discriminate legitimate network flows from the malicious ones. Such a fairness policy will help prioritizing the legitimate flows and prevent the system from DDoS attacks. In this thesis, we present and compare the effectiveness of four different fairness policies using data collected from real-world DDoS scenarios on a commercial web server. Unlike previous prioritization schemes, our policies study the characteristics of the network flows and are proved to be effective.

ACKNOWLEDGMENTS

I would like to express my special thanks to my adviser, Professor Yih-Chun Hu, for his guidance of this research. I also thank Tao Peng from Centaline Group for his generous offer of the DDoS network dataset, which is usually difficult to obtain. This research could not be done without their assistance.

I am grateful to my father and mother, Zhiqiang Liu and Yiping Lu, who love and support me. Their encouragement is key to my success.

.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Nowadays, as computer technologies quickly develop, the computer network has been gaining more and more importance in people's daily lives. People in modern society cannot live without the Internet. Nevertheless, the computer network abounds with malicious attackers and security issues such as the famous Distributed Denial-of-Service (DDoS) attack. In a DDoS attack, an attacker usually uses zombie machines to make requests a huge number of times to a victim and consumes the victim system's resources such as bandwidth and memory. Finally the victim system will be unavailable to serve any requests and results in loss of economic profit, delay in work progress, and many other severe effects.

Previous research has proposed several defense mechanisms to DDoS attacks such as Stateless Internet Flow Filter (SIFF) [1], TVA [2] and CRAFT [3]. However, there is still much to improve. The previous mechanisms all have disadvantages that may lead to the waste of resources and unreliable malicious flow detection. Also, little work has been done on making policies based on network trace characteristics to discriminate legitimate network flows from malicious ones on destination hosts. Thus, research on fairness policies is needed for flow discrimination and prioritization, so that the effectiveness of previous DDoS mitigation mechanisms can be improved.

In this research, we have obtained two sets of network traces from the same commercial website server on normal traffic and DDoS attack traffic respectively. We interleave both datasets together and create a synthetic network trace, where the inserted normal traffic flows are treated as ground truth. Then we have a chance to perform statistical analysis. Since both datasets come from real-world scenarios and if the ratios of their number of requests sent in the

same period of time are kept exact, the synthetic trace will be a good simulation of the real scenario and the experiment result will be reliable.

We have examined four different fairness policies: per-IP, per-AS, per-AS-per-IP and per-account-per-IP fairness. Their effectiveness is evaluated by calculating the total number of requests served when all legitimate flows are fulfilled. In other words, after a victim serves the first *n* requests, the percentage of served legitimate requests in all legitimate requests is recorded and plotted. If all legitimate requests can be fulfilled in *n* requests in total, then the policy with the smallest *n* will be the most effective one. This evaluation has been performed on all four policies, and it turns out that the per-account-per-IP fairness policy is the most effective, followed by per-AS-per-IP, per-IP and finally per-AS policy.

CHAPTER 2: BACKGROUND

2.1 DDOS ATTACK

Distributed Denial-of-Service (DDoS) is one of the most common and severe attacks in the modern Internet. In a DDoS attack, the attacker tries to use his or her machines to send a huge number of requests to a victim server, so that the server will become too busy to provide service to the rest of the world. Although the attacker cannot take advantage of the victim directly, this type of network attack can make the victim service unavailable to the rest of the world and attackers such as business rivals may gain benefits from it. The largest DDoS attack happened in 2015 to GitHub, which originated from China. A snippet of malicious JavaScript code was injected and the GitHub server received a huge number of HTTP requests from hosts with the injected code. As a result, GitHub was brought offline for several days and many project repositories were affected which brought the loss of millions of dollars to the developers and companies.

DDoS attack can be divided into several categories, depending on what type of resource the attacker will consume. Bandwidth-consuming DDoS attack is one of the major types, where the adversary tries to take over most of the victim system's bandwidth and other users cannot have enough bandwidth for the service to work. Among bandwidth-consuming attacks, flood attack is the most straightforward, where the attacker controlled zombie machines send large volumes of IP traffic to the victim [4]. The adversary can choose to send various kinds of requests such as UDP and ICMP, where source IP addresses are usually spoofed. On the other hand, amplification attack also consumes bandwidth, but is slightly smarter than UDP or ICMP flooding. In an amplification attack, the broadcast IP address features on routers are exploited [4]

and a single message can be amplified to the range of a certain network. Smurf attack is a well-known amplification attack, which makes the victim machine send echo request to a broadcast IP address and the echo response can overwhelm the victim server. There are also many other examples of amplification attacks such as DNS and NTP, but this kind of attack can be prevented by filtering the source IP address or the content of network flows. Besides the bandwidth-consuming attacks, some DDoS attacks exploit other resources such as memory, and TCP SYN flooding is a famous example. In a SYN flooding, the victim server will acknowledge every SYN that an attacker sends and wait for the attacker's non-existent ACKs. Unlike UDP and ICMP flooding, SYN flooding does not exploit network bandwidth but the limited size of the TCP state table. Then the server will be unable to process the other connections from normal users without enough memory space.

Our research will focus on bandwidth-consuming DDoS attacks, which are in fact not easy to tackle. Detection and defense of such attacks have been the two major parts that continue to be open problems despite extensive prior work (as described in Chapter 6). Also, since DDoS attacks in real-world target commercial websites such as Google or Facebook, it is hard for university researchers to find a dataset of attack network traces to work with. Furthermore, although researchers have been eagerly developing effective strategies, attackers' behaviors are also changing and they will subvert the detection and defense mechanisms after they have learned them.

2.2 MIDDLEPOLICE

Since DDoS attacks are severe in the modern Internet and difficult to deal with, Liu et al. proposed the MiddlePolice mechanism in 2016, which seeks to marry the deployability of DDoS-protection-as-a-service solutions with the destination-based control of network capability

systems [5]. Unlike existing DDoS attack solutions such as Cloudflare, MiddlePolice can avoid the issue of information leak to the third-party solution provider. Also, most of the existing cloud-based systems use proprietary algorithms [5], so user system security completely depends on how robust and fast the third-party provider adapts to a new attack type. In other words, clever attackers may hack such services if they cannot react quickly and effectively enough. However, MiddlePolice can give the user more freedom in prioritizing network flows and thus it can be more effective than cloud-based services. Furthermore, MiddlePolice is implemented as a Linux kernel module so that it can be deployed to any computer system and evaluated with simulations.

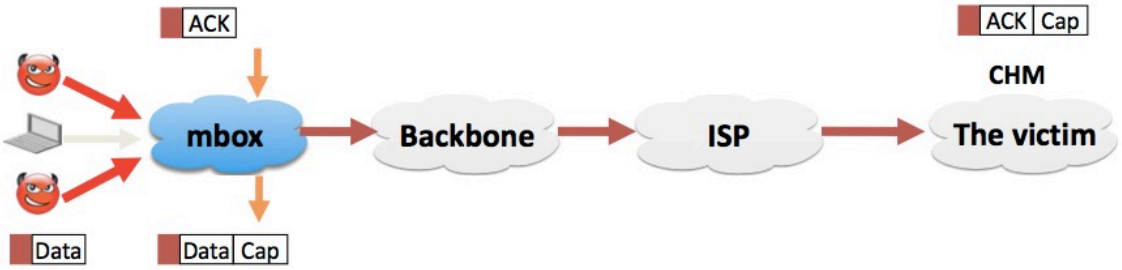

Figure 1. Feedback loop of capability in MiddlePolice mechanisms

The MiddlePolice mechanism is built on a group of well-connected "mboxes" and deployed on both the destination hosts and network routers. A victim can choose a bandwidth allocation policy and implement it on a "mbox". Such policy will rely on a feedback loop of "capabilities" defined by MiddlePolice on downstream paths as displayed in Figure 1 [5]. Specifically, upon a packet arrival, the mbox stamps a capability in the packet. The capability is a piece of cryptographically generated code by the mbox to ensure correctness. Each capability uniquely identifies a packet. The victim deploys a capability handling module (CHM). When the packet arrives at the victim server, CHM trims the capability from the packet and returns the capability back to the mbox, by piggybacking the capability to the ACK packets. If the capability

is not returned to the mbox after a sufficiently long period of time, compared to the RTT from the mbox to the victim, the mbox considers the packet carrying the capability has been lost [6]. The mboxes will also keep victim's IP as a secret to the world and the path as a secret to the victim. Moreover, even if victim's IP address is exposed and attackers can bypass mboxes, MiddlePolice can still prevent attacking traffic by a packet filtering mechanism relying on ACL on commodity routers and switches [5].

There are still several components that need to be studied and developed in the MiddlePolice mechanism. One of the most crucial parts is the destination-selected policy that should be implemented for the victims on mboxes. Since each target server may have different business models, then the policy to prioritize network flows may differ from server to server. Thus, a new network flow prioritization scheme depending on the individual business models should be created. Such a scheme should be able to block the attackers' traffic in general and serve the legitimate requests. Nonetheless, previous research has not covered characteristics such as AS name and server visiting history that can distinguish legitimate flows from malicious ones. Although mechanisms such as TVA [2] and SIFF [1] have proposed methods for distinction, they have not evaluated the effectiveness of different fairness metrics. This research compares effectiveness of different fairness metrics using DDoS traces collected from a real commercial web server.

CHAPTER 3: PROPOSED FAIRNESS POLICIES

Fairness policies are an important concept in the DDoS protection field because it can discriminate legitimate network flows from malicious ones. Different fairness policies will lead to different prioritization results and thus this is crucial to our research. In this research, we plan to perform analysis and simulations on a DDoS dataset obtained from real network attack traces. We also define total service count and loss rate as indicators of the defense effectiveness. Here is a description of our experiment design and security measurements.

3.1 FUNDAMENTAL LIMITS ON ATTACKER

Before implementing any defense mechanism, a victim server can impose some fundamental limits on the attackers as a preliminary defense, and there are various types of such defense approaches. Reverse-Turing tests are commonly used in today's commercial websites. Whenever a user tries to login and retrieve key information, a CAPTCHA is usually used to make sure a human instead of a web crawler is using the service, and CAPTCHA is very difficult for attackers to subvert. Besides, the victim server can also restrict the number of registrations by enforcing the identity information requirements such as linking phone numbers or social media accounts, because DDoS attackers usually cannot make new phone numbers for each of the zombie machines they use. Other strategies may include keeping IP address and AS domain blacklists, which can help block the primitive attackers in some way. However, these countermeasures either can easily be subverted by intelligent attackers who use more advanced strategies such as IP spoofing, or cannot apply to most websites which do not require account login.

3.2 METRICS

In order to evaluate the effectiveness of each defense policy, we evaluate the total number of requests served when a particular portion of the legitimate flows are fulfilled. If the total number of a policy is too large, then this policy identifies too many malicious requests as legitimate ones. Therefore, the smaller this total number is, the more effective a policy will be. Specifically, the total number of necessary requests can be represented this way: we define the total service count to be the total number of packets served so far, and the loss rate to be the ratio of the unserved good requests to the total number of good requests. In (1), F_{good} is the fulfilled requests so far and T_{good} is the total count of good requests. For different policies, we compare the total service number of them at the same loss rate.

$$Loss\ Rate = 100\% - \frac{F_{good}}{T_{good}} \quad (1)$$

3.3 MECHANISM DESIGN

In the ideal case where the server does not suffer from any attack, the victim only needs to fulfill all the legitimate network requests, and thus it should send the same number of responses as requests. Then we have the ground truth, and in the attacking scenarios, the server needs to send more responses in order to fulfill all requests from normal users because it cannot distinguish legitimate IPs from the pool. In the case where no defense approach is applied, the server needs to satisfy all requests, whose number is huge. Nonetheless, the purpose of this research is to figure out a way to prioritize network flows so that the total number of legitimate requests that the server is required to service is minimized. In other words, with the prioritized order of network traces, all the good requests are ranked in the first few and the server does not need to fulfill all good requests because the lower-ranked requests are all recognized as malicious.

We have performed experiments on several different prioritizing strategies: per-IP, per-AS, per-AS-per-IP and per-account-per-IP, each of which we describe in detail below. For each of the strategies, we can obtain a prioritized list of good requests to respond and we know that the ground truth must be a subset of it. Then we compute the relationship between loss rate and total service count as described in Section 3.2.

For per-IP fairness, we can compute the fulfilled legitimate requests step by step from our prioritization strategy. For example, if the requests are prioritized by the visiting history of their IP, each time we keep sending one request per IP. Suppose we have 3,000 legitimate IPs where 1,000 of them send one request each, 1000 send two requests each and the remaining 1000 send five requests and 3,000 bad requests which send 100 requests each. In this case, the victim starts with serving one request for each IP. Then 3,000 good and 3,000 bad requests are fulfilled. Since there are 8,000 good requests in total, then 37.5% of the good requests are served at this point. In other words, if per-IP fairness policy is enforced, when 6,000 requests are served in total, the loss rate for legitimate requests is 62.5%. Thus, the point (6000, 0.625) is recorded. Then all requests from 1,000 good IPs have been completed and in the next step, if one request per each remaining IP is served, 2,000 good requests and 3,000 bad requests are fulfilled. It can be inferred that 62.5% of the good requests are served when 11,000 requests are served in total and the point (11,000, 0.375) is recorded. Finally, three requests per each remaining IP are served and the point (11000 + 3 * 4000, 0) is recorded. After all data points are collected, a trend curve is plotted to study the relationship between loss rate and the total service count.

After all the loss rate and total service count data are collected for an experiment, a graph representing their relationship will be plotted. The ground truth, which can be treated as a golden standard, should look like a straight line because it is not affected by any other requests and the

loss rate should be proportional to the number of good requests. In the example provided previously in per-IP fairness policy, as shown in Figure 2, there are 8,000 requests treated as ground truth, and the endpoints are (0, 1) and (8000, 0), which means that whenever the loss rate decreases by $\frac{1}{8000}$, one more request will be served and this ratio is constant. For all the other experiments, the graph should decrease fast at the start and slower as the total service count gets larger. Since most good IPs or accounts send a small number of requests and bad IPs send many more, this non-uniformity leads to the slope change in graph. Furthermore, in the case where no defense mechanism is implemented, all requests need to be served in order to cover all legitimate ones. Similar with the ground truth, the graph for no defense case should also look like a straight line because good requests can be treated as uniformly distributed in the dataset. The endpoint should be (0,1) and (308000, 0), where 308,000 is the total number of requests.

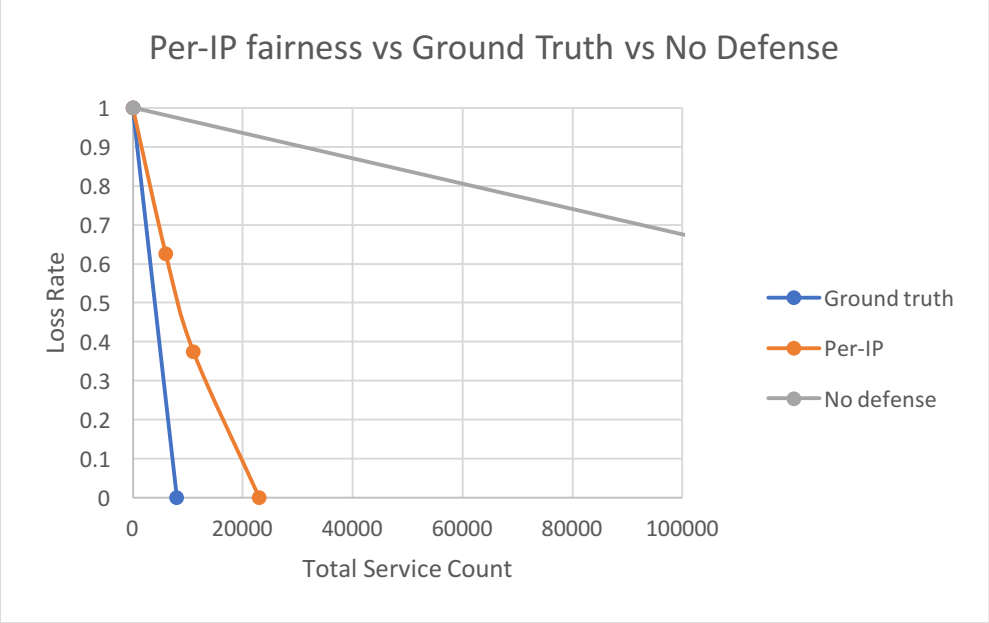

Figure 2. Relationship between loss rate and total service count in ground truth and per-IP fairness policy in an example

For per-AS fairness, the ASes of the IPs in good traces are identified as good ASes and the prioritized requests list has all requests generated by historically visited ASes. The “ipwhois”

service is used here to obtain the AS name of each IP address. For per-AS-per-IP fairness, requests sent by new ASes will be ignored and the prioritized requests list remains the same as the per-AS fairness experiment. For per-account-per-IP fairness, we assign an account to each IP that has visited the server before, and an account to 50 IPs that has not, i.e., 50 malicious IPs share one account and one legitimate IP owns one account. In total there are 50 accounts for malicious IPs. Instead of serving one request per IP, this time the victim serves one request per account. The relationship between an account's loss rate and total service count is obtained in this experiment for evaluation.

CHAPTER 4: DATASET AND DATASET PROCESSING

This experiment utilizes two datasets obtained from a real commercial server in normal and DDoS scenarios. Python and related services such as numpy, IP2location and ipwhois are used to implement the simulation. Details are provided below.

We have obtained two different sets of network traces from the same commercial website server from Centaline Group in China: normal traffic and DDoS attack traffic. The normal traffic trace is obtained several days before the attacking trace, so it can also be used as a trace of visiting history. The DDoS trace contains about 7 million requests generated by 2,532 different IPs in 24 hours, so each IP sends 2,861 requests on average, which is not normal and corresponds to the DDoS situation.

To perform statistical analysis to our network traces, we need a nominal profile, i.e. legitimate traces as ground truth in the DDoS dataset. Thus, a synthetic data trace should be created. Nevertheless, if we make up the traces and combine them together, then the result will not be reliable and probably cannot work because the data do not reflect real-world scenarios. The normal network traces we have collected is working here: each flow from that dataset is treated as ground truth and we interleave the legitimate and malicious flows in their exact ratios in the same periods of time together to make our synthetic dataset. We need to make a synthetic dataset, which consists of a group of good traces and a group of bad traces. Then, we can obtain a large number of synthetic traces which come from real-world scenarios and can be used for statistical analysis in our experiments.

CHAPTER 5: EXPERIMENT METHODOLOGY, RESULTS, AND EVALUATION

As described, our experiments are implemented using Python and related services. We have obtained results from all experiments and here we will show them and the corresponding evaluations.

5.1 PER-IP FAIRNESS

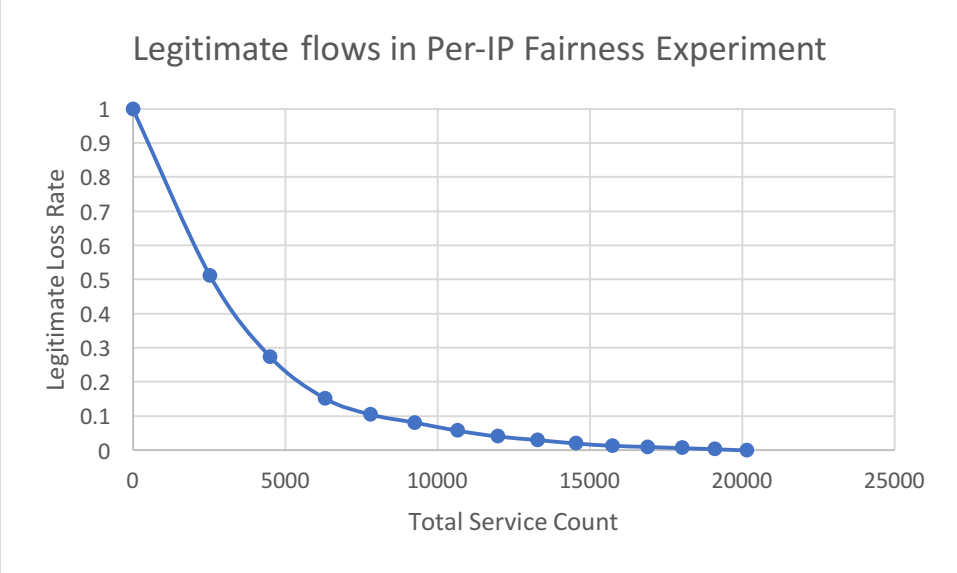

Figure 3. Relationship between loss rate per-IP and total service count on legitimate network flows in per-IP fairness experiment

As shown in Figure 3, it is easy to observe that the trend is as expected, where the graph decreases fast at the beginning, after the loss rate gets lower than 0.3 it decreases slower until all the listed requests are fulfilled at the service count 20,144. However, the service becomes inefficient in the long "tail" of this graph because it stands for the part that many responses are sent while most of them are for meaningless, malicious requests generated by the attacker's zombie machines. Nonetheless, this approach is still very effective because originally we need to send 7 million requests to cover all the legitimate ones and this mechanism has prioritized the correct requests over 99% of the unnecessary requests.

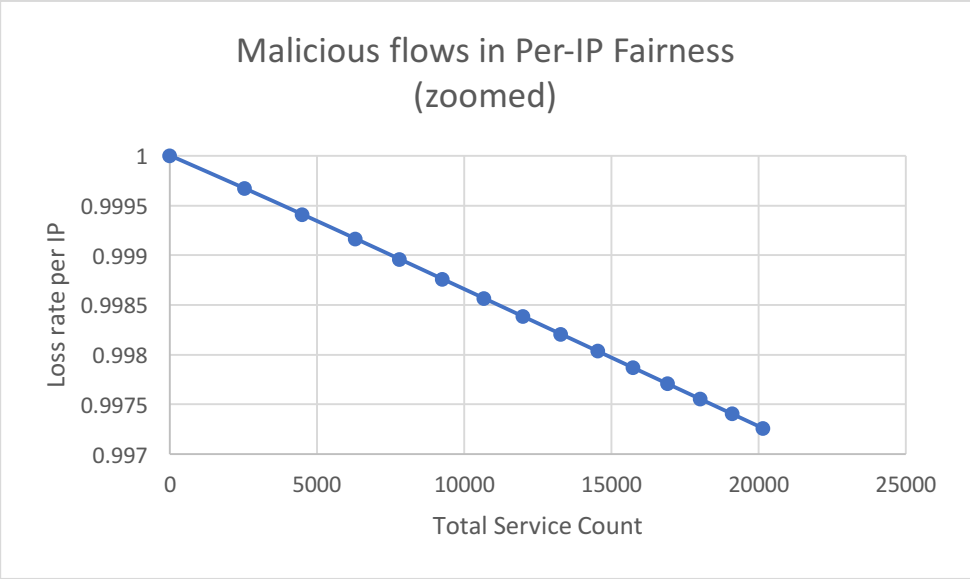

Figure 4. Relationship between loss rate per-IP and total service count on malicious network flows in per-IP fairness experiment (zoomed in for the early stage)

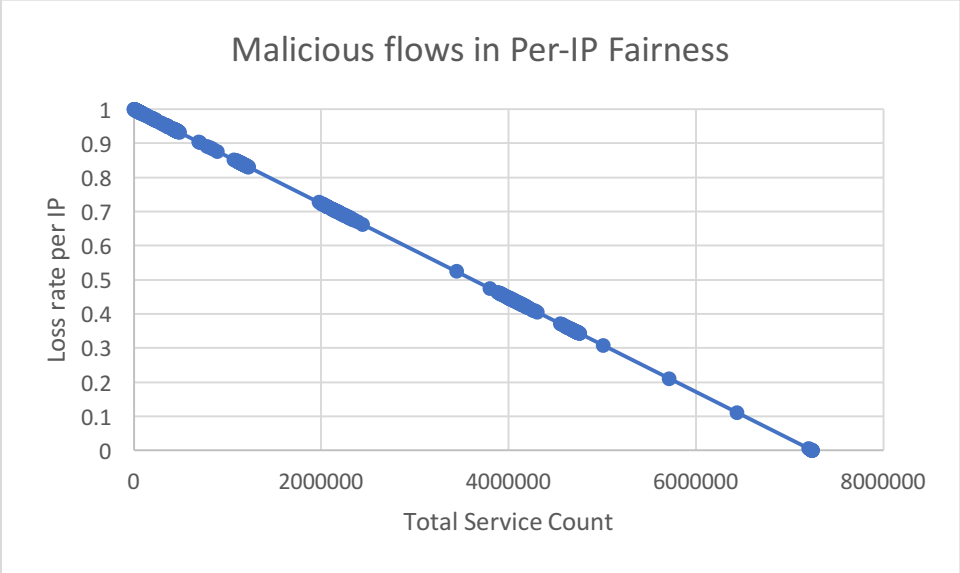

Figure 5. Overview of the relationship between loss rate per-IP and total service count on malicious network flows in per-IP fairness experiment

Beside the legitimate flows, the malicious flows' behaviors are also studied. Figure 4 shows the trend of malicious flows when they are being sent at the same time as the good flows. It looks like a straight line because the number of malicious flows are overwhelming, even at early stages. However, according to Table 1, the slope is getting steeper at a slow rate, which is opposite to the trend of legitimate flows so that it will look like a straight line when they are

combined together. After all of the good requests are fulfilled, only the rest of malicious ones need to be fulfilled. As shown in Figure 5, in this case the malicious flows are going in a straight line to the point (total number of requests, 0).

Table 1. Slope of the relationship graph at early stages for malicious flows

Total Service Count	Loss Rate	Slope
2532	0.999670358	-1.30E-07
4502	0.99940808	-1.331E-07
6301	0.999164714	-1.353E-07
7810	0.998958342	-1.368E-07
9260	0.998759149	-1.374E-07
10660	0.998566858	-1.374E-07
11987	0.998384368	-1.375E-07
13278	0.998206571	-1.377E-07
14536	0.998033329	-1.377E-07
15742	0.997867128	-1.378E-07
16902	0.997707139	-1.379E-07

5.2 PER-AS FAIRNESS

Figure 6 shows the same relationship in per-AS fairness experiment. We can see that the graph of per-AS fairness has the same changing trend as that of per-IP fairness, but is overall slower and requires sending more responses. To complete all legitimate requests in this prioritization, a server should send 78,494 responses, which is three times more than the per-IP fairness. We can conclude that per-AS fairness is not so efficient as per-IP fairness metric. However, this is also just as expected because many IPs can share the same network and both the

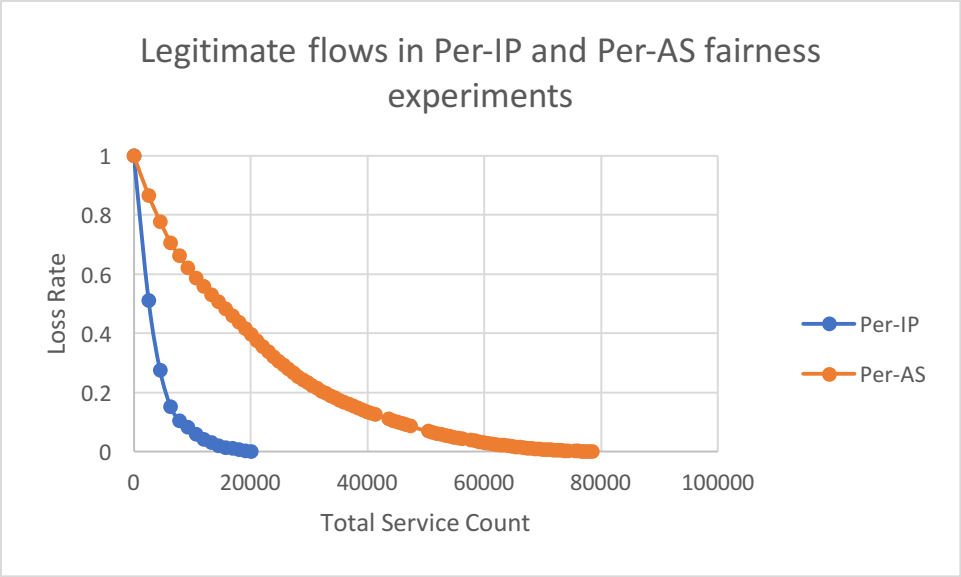

Figure 6. Relationship between loss rate and total service count in per-AS fairness experiment, in comparison with the same relationship in per-IP fairness experiment

normal user and attacker can use exactly the same AS. Thus, distinction using the AS names will obviously not be accurate enough. In fact, this result shows that the AS name is already very efficient because it has reduced 99% of the unnecessary responses to the malicious requests. The reason may lie in the fact that attackers use cloud service such as AliCloud and AWS a lot but normal user will not use those services.

5.3 PER-AS-PER-IP FAIRNESS

In per-AS-per-IP fairness experiment, we no longer use the whole dataset when computing the total service count. Instead, we will give more focus on the IPs coming from good AS domains and lower the weight of IPs from new AS domains. Instead of sending one request for all IPs at a time, we will send one request for all IPs in historically visited ASes each time, and one request for all IPs in new ASes every fifth request. According to Figure 7, this strategy yields a better result. The loss rate drops faster than both per-IP and per-AS fairness experiments because many fewer requests are sent at each step in this experiment. The slope changes in the same way as the two other graphs but finishes all necessary requests within 11,902 responses.

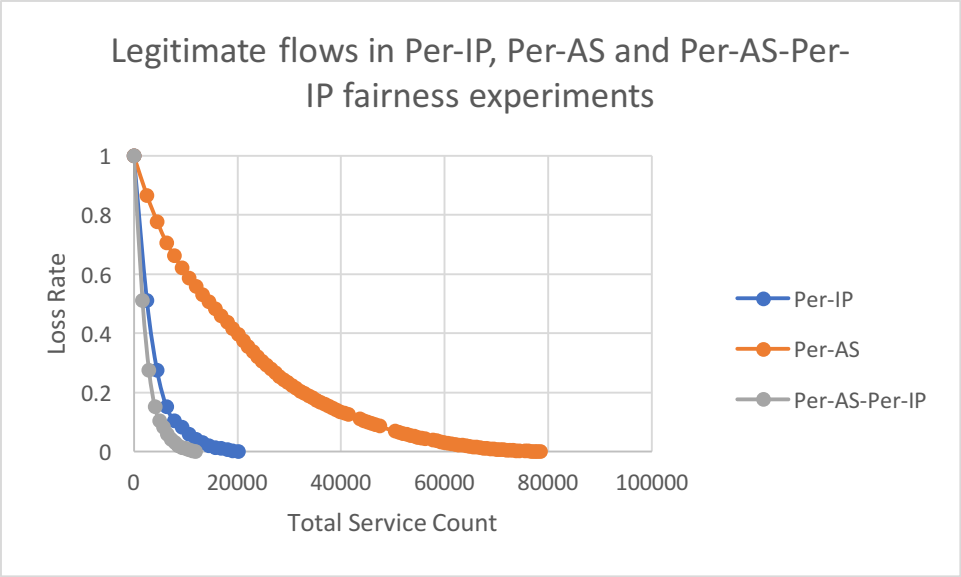

Figure 7. Relationship between loss rate and total service count in per-AS-per-IP fairness experiment, in comparison with the same relationship in per-IP and per-AS fairness experiment

5.4 PER-ACCOUNT-PER-IP FAIRNESS

In normal traffic, each of the visiting IP can be allocated to an account and if they visit the same server again in the future, they will visit on behalf of their own accounts. On the other hand, when a DDoS attack traffic is detected, every 50 different IPs will share one account because most of the IPs are malicious and generate requests that cannot bring much profit. The attacking network flows should be recognized as the same ones and not worth an account per IP. Then, our mechanism will send one request per account instead of per IP and will keep recording the loss rate and total service count whenever all requests of an account are fulfilled, until all good account's requests get served.

According to Figure 8, the per-account-per-IP method works much better than the previous strategies. Although the slope trend looks similar with previous curves, this loss rate drops much faster. In this experiment, only 831 requests need to be covered while the ground truth only has 295. This mechanism can group 50 different malicious IP addresses together as a

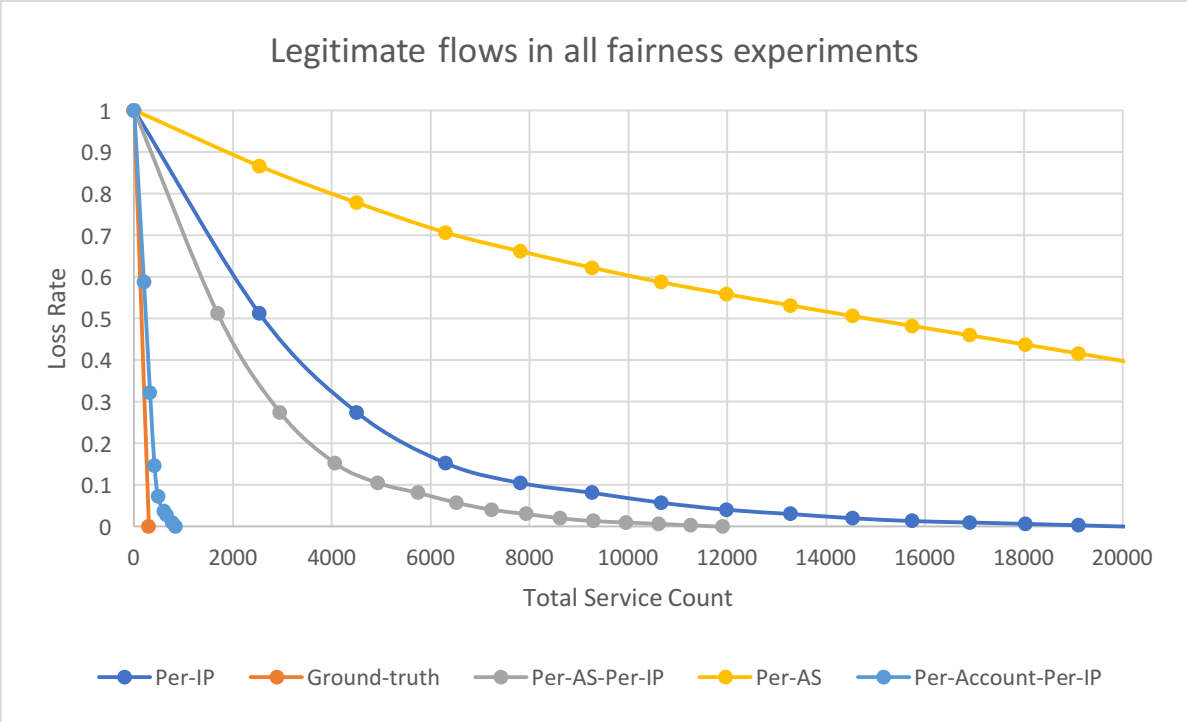

Figure 8. Relationship between loss rate and total service count in all fairness experiments in comparison with the ideal ground truth

"zombie" account and saves even 98% more computing resources from the per-IP fairness experiment. It turns out that per-account-per-IP fairness is the most efficient and effective strategy to prioritize network flows so far, and it is very close to the ground truth in our synthetic dataset.

CHAPTER 6: REVIEW ON PREVIOUS WORKS

This research is performed based on a lot of previous works in DDoS (Distributed Denial-of-Service) protection fields, including detection and defense of such network attacks.

6.1 DDOS ATTACK DETECTIONS

The first step of DDoS prevention is to detect such attacks from active network flows and there are various ways to do it. Statistical analysis to network traces was performed and obtained a classifier that can identify the malicious and legitimate packets in the future [7]-[9]. In [7], entropy and frequency-sorted distributions of selected packet attributes are calculated and the bad packets show different characteristics with the good ones. On the other hand, [8] uses cluster analysis on network traces, which groups good and bad data points separately. This proposed system can identify the precursors of a DDoS attack because it observes trace characteristics in different attacking phases (normal, in-attack, post-attack, etc.). Furthermore, authors of [9] have obtained a covariance analysis model to detect the SYN flooding attack. Their simulations show that this model is highly effective and can even detect very subtle attacks.

DDoS attack traffic can also be detected through hardware behaviors. Carl et al. [10] observed the energy distribution with wavelet analysis on the servers and an energy spike can be found during an attack period in contrast to the stationary distribution that normal traffic yields. Similar to [8], this method can also detect the precursors of an attack in early stages. However, [11] points out that wavelet approach is difficult to implement for its complexity. Also, this approach is subject to the varying test conditions and the subverting attacker behaviors such as changing frequency after they know the defense strategy.

Not only the server side, but also the Internet Service Providers (ISP) can detect DDoS traffic. As described in [12], with the proposed distributed change-point detection system, an ISP can monitor abrupt network flow change across multiple domains at the same time. This approach can help detect a DDoS attack at the earliest time. Chen et al. [13] also applied a collaborative approach to identify shrew network flows hidden in normal traffic, which merges alerts from different routers.

As artificial intelligence and machine learning quickly develops, DDoS detection using a machine learning approach has gained more attention. The authors of [14] utilized neural network to classify network flows. Unlike traditional back-propagation (BP) neural network, Learning Vector Quantization (LVQ) neural network is applied in this research. As a result, the new LVQ neural network achieves an accuracy of over 99%, which is much better than the traditional approaches.

Since DDoS attackers usually send a flood of SYN requests to the server and the server cannot respond to all of them due to its capacity, then a new mechanism based on the difference of SYN and FIN counts is proposed [15]. Wang et al. [15] also applied a cumulative sum (CUSUM) method to make the detection mechanism more generally applicable and easier to implement and deploy.

DDoS attack can happen to any modern Internet component. Bakshi and Dujodwala [16] discussed the defense strategies that a cloud infrastructure can take. Virtualization of the machines and utilization of the Intrusion Detection System (IDS) are the proposed approaches that a cloud network should use. Mousavi and St-Hilaire [17] talked about the DDoS defense on SDN controllers, which are so crucial that may cause the backbone of the Internet to crash. To mitigate the DDoS threat, a centralized control system should be deployed on SDN. It computes

several characteristics such as information entropy and destination addresses of the first several packets of a DDoS attack, and their system can have an accuracy of over 96%.

6.2 DDOS DEFENSE MECHANISMS

The other crucial part of DDoS research is the mechanism that can effectively block the malicious traffic. According to [20], based on deployment location, defense mechanisms can be either centralized, where detection and response are deployed at the same location, or distributed, where the two parts are deployed at various locations. Centralized defense mechanisms can even be divided into three different categories: source-based, destination-based and network-based.

Distributed mechanisms are deployed at both source and destination hosts [20]. The Traffic Validation Architecture (TVA) mechanism is distributed, where all senders must gain permission in terms of capabilities from the receiver before transmission [2]. The capability information will be included in packets that the sender transmits. In the core of network, routers can check if each packet is authorized or not and will throw away the unauthorized ones to avoid large-volume DDoS attacks. This mechanism requires deployment on sender and receiver for the capability communication protocol and the network routers for traffic verification. Nonetheless, TVA assumes that receiver is able to distinguish attack traffic from the legitimate traffic, and the effectiveness of TVA will largely depend on how accurate the destination hosts discriminate the traffic [20].

Destination-based mechanisms are deployed at the destination hosts [20]. As an example, D-WARD is one of the systems that based on comparison between normal flow models and the real-time traffic [18]. It monitors two-way traffic between the target system and the rest of the Internet and periodically does statistical comparison. If any anomaly flow is found, it will limit the bad flows' rates and is presented to be effective. Similarly, the Stateless Internet Flow Filter

(SIFF) mechanism is also destination-based and divides traffic into privileged and unprivileged classes [1]. However, in SIFF, the classification is based on the dynamically updated capabilities of each host instead of statistical comparison between incoming and outgoing network traces, and SIFF provides a protocol that can calculate and exchange hosts' capabilities. Although many destination-based mechanisms are proved to be effective, one of the disadvantages that destination-based mechanisms may have is that resources are wasted on the paths to victim and the core of network still suffers from the meaningless large-scale flows [20].

Pushback [19] is another mechanism that can protect a network from DDoS attacks, but it is implemented on network routers, and thus it is a network-based defense mechanism. Each of the routers will drop malicious packets if an attacked is detected and the upstream packets will be informed to drop packets accordingly in order not to waste the Internet resources. Besides, there is also a capability-based mechanism called CRAFT implemented on network routers. CRAFT can secure all downstream links of a deploying router by enforcing TCP-fairness on all flows. Once a flow passes through a CRAFT router, it will be TCP-fair on all downstream links [3].

6.3 NETWORK PRIORITIZATION TECHNIQUES

As mentioned before, cloud environment is playing an important role in the modern Internet, but cloud infrastructures are still vulnerable to DDoS attacks. Chen et al. [21] introduced a confidence-based filtering method to classify incoming packets to a cloud service. A nominal profile will be collected when there is no attack, and based on such profile, each packet during an attack period will be awarded a score which determines how malicious it is. If the score of a packet is not legitimate enough, the system will discard it. Furthermore, this CBF method is proved to be efficient in both time and space complexity, so it is good for real-world implementation.

CHAPTER 7: CONCLUSION

In this research, we have proposed and evaluated several network bandwidth fairness policies to discriminate legitimate network flows from malicious ones in DDoS scenarios. Mechanisms such as SIFF, TVA and CRAFT from previous DDoS research have not used the characteristics we have covered in this thesis for distinction. Our research uses real-world DDoS and normal network traces from the commercial web server of Centaline Group in China and interleave them together to make a synthetic dataset for analysis. We also define loss rate and total service count as our metrics for each policy. Four different fairness policies are examined and compared: per-IP, per-AS, per-AS-per-IP and per-account-per-IP. It turns out that per-account-per-IP policy is the most effective and behaves very close to the ground truth we have in our dataset. Although the other three policies are not as good as per-account-per-IP, they are still very effective in prioritizing the legitimate flows to avoid bandwidth waste on unnecessary requests generated by attackers. Due to time constraints, we will not be able to examine other fairness policies. However, in the future, more characteristics can be taken into consideration to make the fairness policy even more effective.

REFERENCES

[1] A. Yaar, A. Perrig and D. Song, "SIFF: A stateless Internet flow filter to mitigate DDoS flooding attacks," in *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, Berkeley, CA, USA, 2004, pp. 130-143. Available doi: 10.1109/SECPRI.2004.1301320

[2] X. Yang, D. Wetherall, and T. Anderson, "TVA: A DoS-limiting network architecture," in *IEEE/ACM Transactions on Networking(TON)*. 2008, vol. 16, no. 6, pp. 1267-1280, Dec 2008. Available doi: 10.1109/TNET.2007.914506

[3] D. Kim, J. T. Chiang, Y.-C. Hu, A. Perrig, and P. R. Kumar, "CRAFT: A new secure congestion control architecture," in *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS '10)*. ACM, New York, NY, USA, pp. 705-707. Available doi: 10.1145/1866307.1866404

[4] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: Classification and state-of-the-art," *Computer Networks*, 2004, vol. 44, no. 5, pp. 643-666.

[5] Z. Liu, H. Jin, Y.-C. Hu, and M. Bailey, "MiddlePolice: Toward enforcing destination-defined policies in the middle of the Internet," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16).* ACM, New York, NY, USA, 2016, pp. 1268-1279. Available doi: 10.1145/2976749.2978306

[6] Y.-C. Hu, "MiddlePolice: Toward enforcing destination-defined policies in the middle of the Internet," presented at National Taiwan University, May 18, 2017.

[7] L. Feinstein, D. Schnackenberg, R. Balupari and D. Kindred, "Statistical approaches to DDoS attack detection and response," in *Proceedings DARPA Information Survivability Conference and Exposition*, Washington, DC, USA, 2003, pp. 303-314 vol.1. Available doi: 10.1109/DISCEX.2003.1194894

[8] K. Lee, J. Kim, K. Kwon, Y. Han and S. Kim, "DDoS attack detection method using cluster analysis," *Expert Systems with Applications*, 2008, vol. 34, no. 3, pp. 1659-1665.

[9] L. Li and G. Lee, "DDoS attack detection and wavelets," *Telecommunication Systems*, 2005, vol. 28, no. 3-4, pp. 435-451.

[10] G. Carl, G. Kesidis, R. R. Brooks and S. Rai, "Denial-of-service attack-detection techniques," in *IEEE Internet Computing*, vol. 10, no. 1, pp. 82-89, Jan.-Feb. 2006. Available doi: 10.1109/MIC.2006.5

[11] Y. Chen, K. Hwang and W. Ku, "Collaborative detection of DDoS attacks over multiple network domains," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 12, pp. 1649-1662, Dec. 2007. Available doi: 10.1109/TPDS.2007.1111

[12] Y. Chen and K. Hwang, "Collaborative detection and filtering of shrew DDoS attacks using spectral analysis," *Journal of Parallel and Distributed Computing*, 2006, vol. 66, no. 9, pp. 1137-1151.

[13] Y. Chen, K. Hwang and W. Ku, "Collaborative detection of DDoS attacks over multiple network domains," in *IEEE Transactions on Parallel and Distributed Systems*, Dec. 2007, vol. 18, no. 12, pp. 1649-1662. Available doi: 10.1109/TPDS.2007.1111

[14] J. Li, Y. Liu and L. Gu, "DDoS attack detection based on neural network," in *2010 2nd International Symposium on Aware Computing*, Tainan, 2010, pp. 196-199. Available doi: 10.1109/ISAC.2010.5670479

[15] H. Wang, D. Zhang and K. G. Shin, "Detecting SYN flooding attacks," in *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, New York, NY, USA, 2002, pp. 1530-1539. Available doi: 10.1109/INFCOM.2002.1019404

[16] A. Bakshi and Y. B. Dujodwala, "Securing cloud from DDOS attacks using intrusion detection system in virtual machine," in *2010 Second International Conference on Communication Software and Networks*, Singapore, 2010, pp. 260-264. Available doi: 10.1109/ICCSN.2010.56

[17] S. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against software defined network controllers," in *Journal of Network and Systems Management*, 2017, vol. 26, no. 3, pp. 573-591.

[18] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the source," in *Proceedings of the 10th IEEE International Conference on Network Protocols, 2002*, 2002.

[19] J. Ioannidis and S. Bellovin, "Implementing pushback: Router-based defense against DDoS attacks," in *Network and Distributed System Security Symposium: NDSS '02*, 2002

[20] S. T. Zargar, J. Joshi and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," in *IEEE Communications Surveys & Tutorials*, Fourth Quarter 2013, vol. 15, no. 4, pp. 2046-2069. Available doi: 10.1109/SURV.2013.031413.00127

[21] Q. Chen, W. Lin, W. Dou and S. Yu, "CBF: A packet filtering method for DDoS attack defense in cloud environment," *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, Sydney, NSW, 2011, pp. 427-434. Available doi: 10.1109/DASC.2011.86