An Intelligent, Distributed and Collaborative DDoS Defense System

by

Xiaoyu Liang

Submitted to the Graduate Faculty of

the Department of Computer Science in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2021

UNIVERSITY OF PITTSBURGH

DEPARTMENT OF MATHEMATICS

This dissertation was presented

by

Xiaoyu Liang

It was defended on

May 25 2021

and approved by

Dr. Taieb Znati, Department of Computer Science

Dr. Milos Hauskrecht, Department of Computer Science

Dr. James Joshi, Departmental of Information Science

Dr. Youtao Zhang, Department of Computer Science

An Intelligent, Distributed and Collaborative DDoS Defense System

Xiaoyu Liang, PhD

University of Pittsburgh, 2021

The Distributed Denial-of-Service (DDoS) attack is known as one of the most destructive attacks on the Internet. With the advent of new computing paradigms, such as Cloud and Mobile computing, and the emergence of pervasive technology, such as the Internet of Things, on one hand, these revolutionized technologies enable the availability of services and applications to everyone. On the other hand, these techniques also benefit attackers to exploit the vulnerabilities and deploy attacks in more efficient ways. Latest network security reports have shown that distributed Denial of Service (DDoS) attacks have been growing dramatically in volume, frequency, sophistication and impact, making it one of the most challenging threats in the Internet. An unfortunate state of affairs is that the remediation strategies have fallen behind attackers. The severe impact caused by recent DDoS attacks strongly indicates the need for an effective DDoS defense system.

We study the current existing solution space, and summarize three fundamental requirements for an effective DDoS defense system: 1) an accurate detection with minimal false alarms; 2) an effective inline inspection and instant mitigation, and 3) a dynamic, distributed and collaborative defense infrastructure. This thesis aims at providing such a defense system that fulfills all the requirements.

In this thesis, we explore and address the problem from three directions: 1) we strive to understand the existing detection strategies and provide a survey of an empirical analysis of machine learning based detection techniques; 2) we develop a novel hybrid detection model which ensembles a deep learning model for a practical flow by flow detection and a classic machine learning model that is aware of the network status, and 3) we present the design and implementation of an intelligent, distributed and collaborative DDoS defense system that effectively mitigate the impact of DDoS attacks. The performance evaluation results show that our proposed defense system is capable of effectively mitigating DDoS attacks impacts and maintaining a limited disturbing for legitimate services.

Table of Contents

List of Tables

List of Figures

Preface

This Ph.D thesis is undertaken at the Department of Computer Science of University of Pittsburgh. This project would have been very difficult without the help and support from many people.

I am deeply grateful to my advisor, Dr. Taieb Znati. He generously provides his time, efforts, valuable advice and honest critiques at all times. He always approach research problem with great enthusiasm, which encourages us along the research career. He genuinely cares for his students' success. I am proud to be his student, and forever be grateful for what he has done for me.

I owe a deep sense of gratitude to all the faculties and staff of the Computer Science Department at University of Pittsburgh. Without their constant supports, the road to Ph.D would have been much harder. A special shout-out to all my committees, Dr. Youtao Zhang, Dr. Milos Hauskrecht and Dr. James Joshi. I also would like to express my special thanks to Keena Walker, who is always willing to help us with all sorts of un-predicted questions.

I would like to thank all my friends. I would never have made it without them. They make my life full of joy, and fill my heart with warmth. They always have time to listen to my complains, and land hand in need.

Finally, I would like to acknowledge with gratitude to my loving families. I am eternally indebted to my parents. Their love is selfless. They always encourage me to pursue any dreams I had. They never questioned my decisions, even that means I'm moving to another side of the earth. My love, Kaiyu, who loves me invariantly and accepts who I am. I truly appreciate Kaiyu for standing by me, especially when we are going through hard times, and I was irritable and depressed. Without his supports, I cannot make this far.

1.0 Introduction

Distributed Denial of Service (DDoS) attacks attempt to prevent legitimate users from accessing information or services by overwhelming the server and saturating the network connections through multiple compromised systems. DDoS attacks are typically launched from a very large number of distributed, remotely controlled devices, organized into botnets and aimed at attacking the same target [5, 36]. With the advent and the emergence of Cloud Computing and Internet of Things, on one hand, the revolutionized technology enables the availability of services and applications to everyone. On the other hand, these techniques also benefit attackers to exploit the vulnerabilities and deploy attacks in more efficient ways. A subscription-based business model, DDoS-as-a-service, also known as "booters" or "stressers", even provides DDoS attacks as a low-cost service, and causes DDoS attacks becoming accessible to the general public [33, 63].

Recent events demonstrate the severe impact of DDoS attacks. October 2016, a series of massive DDoS attacks were perpetrated against Dyn's DNS servers, and caused the disruption of multiple major websites, including Airbnb, Netflix, and Spotify [22]. This attack used the infamous Mirai, a self-propagating botnet virus, to compromise poorly protected IoT devices. February 2018, GitHub was hit by a colossal 1.35 Tbps flood of traffic, the largest by that time, resulting in major websites across large portions of the US being out of services for a number of hours [39]. September 2019, Wikipedia, suffered a disruptive DDoS attack for almost three days [25]. Although the attacking strategy was an old-style volumetric flooding attack, the attack has quantifiable exceeded 1 Tbps by leveraging the increases of the network capacity.

In fact, a Cisco study predicts that the total number of DDoS attacks would reach 14.5 million annually by 2022 [19]. An effective DDoS defense system is needed now more than ever. Despite extensive efforts have been made in both industry and academia, significant potential remains in exploring an intelligent, distributed and collaborative defense system.

1.1 Problem Statement

To successfully defend and mitigate the impact of the increasingly sophisticated and diversified DDoS attacks, the defense system must be carefully designed [58]. In this section, we describe the fundamental requirements to achieve an effective DDoS defense and discuss the challenges, the design and the deployment such a defense system entails. In summary, to be effective, DDoS defense, in large scale systems, must meet the following requirements:

1. an effective inline inspection, and a rapid respond to mitigate attack's impact;
2. an accurate attack detection with minimal false alarms;
3. a dynamic, distributed, and collaborative defense infrastructure.

1.1.1 Accurate Detection with Minimal False Alarms

Traditional DDoS detection approaches utilize stochastic analysis and exploit the entropy of network traffic to identify anomalous intrusion events. When an attack event is identified, traffic rate-limiting and filtering mechanism is applied to mitigate the impact. However, any mitigation strategies that are applied indiscriminately will cause damage to legitimate traffic. While the victim does not face overwhelming traffic, such an inappropriate response can cause denial of service to legitimate users. Thus, the detection scheme should not only be able to detect that a DDoS attack event is happening, but also should be capable of distinguishing attack traffic from legitimate traffic.

Recently, the trend of DDoS detection is to apply machine learning techniques to classify and detect malicious traffic. These techniques are capable to intelligently learn the underlying data attributes without the need to explicitly describe normal and malicious activities. While the use of machine learning based techniques hold promise, most approaches focus on offline traffic analysis and struggle to capture the ever-evolving characteristics of DDoS attacks [43].

Finally, the detection method should also minimize false alarms, which also lead to collateral damage to well-behaving sources. Hence, the defense system dose not only prevent attack traffic, but also provides reliable delivery of legitimate traffic to end users.

1.1.2 Effective Inline Inspection and Instant Mitigation

To mitigate the impact of DDoS attacks, large enterprises commonly adopt the scrubbing center strategy [31, 64]. When a suspicious DDoS attack is identified, all traffic is diverted to a designated centralized data cleansing station, called scrubbing center, where further traffic inspection and mitigation are applied [2]. Legitimate traffic is passed to the network for delivery, and malicious traffic is blocked. Once the attack stops, all traffic is redirected through routing updates. Although this solution is effective to protect victims from volumetric traffic, it has three major limitations. Firstly, it usually increases latency. All traffic sent toward the victim is detoured to the scrubbing center. In addition to the latency caused by detour, the scrubbing center could be a bottleneck and causes a significant amount of latency, especially when the bandwidth of the scrubbing center is limited. Secondly, the detouring strategy requires complicated mechanism to effectively reroute traffic, which could be extremely expensive in both financial and computational [30, 76]. Lastly, using scrubbing center only allows for monitoring inbound traffic. For enterprises and service providers, who seek to ensure they are not used as an unwitting DDoS attack platform, such a solution is inadequate.

The ideal defense system should react instantly, without introducing delays. To this end, we need an inline inspection and instant mitigation system. With inline inspection, both inbound and outbound traffic passing over the guarded network are inspected. Any identified attack traffic is then immediately blocked without the need for rerouting. While inline inspection is an optimal solution, it raises one major challenge: how to achieve traffic analysis at wire speeds and satisfy the stringent memory and complexity constraints? Currently, most inline inspections relay on packet or flow sampling algorithms to address this challenge. However, recent research has shown that sampling methods distort traffic patterns and degrade the detection accuracy [4, 28]. This major challenge remains open.

1.1.3 Dynamic, Distributed and Collaborative Defense

A typical DDoS attack is launched from widely distributed compromised devices, which unwittingly host attack programs. These compromised devices work coordinately, so that the

attacking traffic converge at the target victim network. Therefore, the source of attacking packets are highly distributed. Nowadays, with the emergence of IoT and mobile computing, a large number of, potential insecure and vulnerable, devices are being weaponized by attackers. The attacking sources are not only spread widely, but could originate from anywhere around the world [33]. Additionally, the trend of DDoS attacks shows that the attacking traffic does not only target the victims, but also the organizations on which they depend, including Internet Service Providers (ISPs) and cloud service providers. Recent attacks exhibited the characteristics of broad-reaching, high-impact and well-coordinated. We posit that in order to effectively defense DDoS attacks before they can cause considerable damage to both the attack target and the Internet infrastructures, a distributed and collaborative defense system is needed.

According to observations of attacking paths, researchers have categorized the deployment of defense systems into three key locations: *source-end network*, *core-end network* and *victim-end network* [52, 60, 90]. Each possible deployment location has its own strengths and weaknesses. The *source-end network* is the most effective place to mitigate the attack traffic, since attack traffic can be blocked before it enters the Internet core and consumes the shared resources. However, it is difficult for differentiating the attack traffic from legitimate traffic at this location. As described previously, DDoS attack traffic is highly distributed, at the source-end, the detection scheme observes too little to perform a successful detection. In contrast to the source-end network, the *victim-end network* is a vantage point for DDoS malicious traffic detection. Close to the victim, the detection mechanism observes the aggregated attack traffic, so a malicious behavior and a degraded server performance are relatively easier to be detected, regardless of the nature of the attack and the location of the attackers. Unfortunately, at this location, it is often the case that it is too late to mitigate the impact. In the *core-end network*, deployed defense systems can also observe the aggregated traffic, and they are in a better position to effectively constrain the attack traffic. However, due to the limited resources and heavy burdens on the core routers, they cannot perform sophisticated detection and mitigation tasks.

It is clear that an effective defense system needed to have a distributed and collaborative infrastructure. The detection of DDoS attack traffic could be done close to the *victim-*

end, then the identified attack traffic signature can be propagated upstream. As such, the malicious traffic can be blocked as far away from the victim as possible. Additionally, a distributed and collaborative mechanism can support early detection by exchanging selected traffic information among different deployment locations. The ability to extend the traffic observation range also benefits the detection accuracy. Finally, a distributed defense system also provides scalability so that defenses can be quickly and affordably updated to respond to the future evolution of DDoS threats.

1.2 Research Overview

Aiming at developing a novel DDoS defense system that meets all the requirements and solves the challenges discussed above, this thesis presents an intelligent, distributed and collaborative DDoS defense system.

The core of the proposed defense system is an accurate and robust DDoS attack detection model, which combines a deep learning approach and an classic machine learning model. The proposed deep learning method has the ability to detect suspicious network flows by only examining a relatively small number of a network flow packets header information, which provides a practical solution for an effective and truly-inline inspection. The classic machine learning model provides the network status detection, which complements with the deep learning model to achieve better performance in identifying DDoS attack traffic. We further augment the detection model with communication and collaboration functionalities, so that detectors that are deployed in different locations are able to exchange information to expand their observation across the network.

Relying on the accurate and robust detection model to distinguish attack traffic from legitimate traffic, we propose a distributed and collaborative DDoS defense system. By leveraging the flexibility, programmability and maintainability provided by Software-Defined Network (SDN), we safely delegate the detection model to an application. However, naively utilizing the centralized controller could cause the defense system itself become a bottleneck, and result in paralyzation of the network. To address this challenge, we design the defense

system comprising a network of specialized SDN controllers, referred to as Sentinels, which help relieve both the computational and communication burdens from the controller. The proposed defense system monitors ongoing network traffic, distinguishes malicious traffic from legitimate traffic, and blocks attack traffic in an efficient and effective manner.

To study the viability of and provide the justification for the proposed solution, we implement the prototype of the proposed defense system in a SDN environment, and carry out a comparative analysis for assessing the performance in terms of the effectiveness of DDoS attack impact mitigation, the quality of service for legitimate users and the overheads added to the controller. The relationship between requirements and works included in this thesis is depicted in Figure 1.

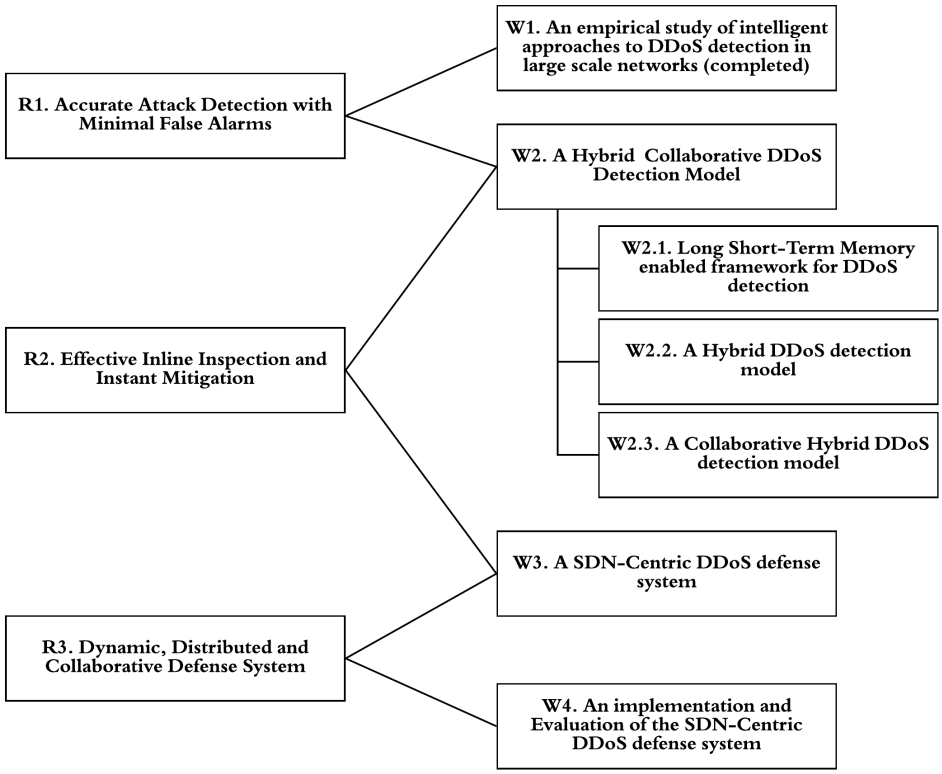

Figure 1: Fundamental requirements and thesis works

1.3 Contributions

This dissertation consists of the following main contributions:

- Detailed analysis and empirical studies of current machine learning based DDoS detection techniques, thus improving an understanding of the problem and current solution space.
- A novel detection approach using deep learning model. The proposed algorithm obviates the need for feature engineering, and successfully learns the complex flow-level feature representations embedded in raw input traffic. Most importantly, the algorithm needs to examine only a short sequence of packets to detect DDoS attack flows, which provides a practical solution for truly real-time inline inspection.
- A hybrid detection model, which combines a network flow by flow detector and a network status detector. These two detectors complement with each other, and achieve more accurate detection results in the emulation environment.
- The design of a distributed and collaborative SDN centric defense architecture to detect DDoS attacks and mitigate their impact. The proposed defense architecture comprising a network of peers, referred to as Sentinels, that dynamically and collaboratively defend against DDoS attacks. With the distributed and collaborative mechanism, the proposed defense system alleviates computational and communication burdens from the controller, then avoids causing the bottleneck.
- A prototype implementation of the proposed defense system in a SDN environment, without any modifications to the OpenFlow protocol features or OpenFlow switches. This enhances the potential of deploying the proposed defense system in a practical setting.
- The design of an evaluation framework to assess the performance of the proposed DDoS defense system and carry out a comparative analysis of its performance with other schemes. The experimental results demonstrate that the proposed defense system can effectively detect and throttle DDoS attack traffic without degrading the quality of service of legitimate users. Additionally, the overheads caused by the proposed defense system is minimal.

1.4 Dissertation Outline

The rest of this thesis is organized as follow: Chapter 2 reviews literature. To better understand the current existing DDoS detection solutions, Chapter 3 provides a thorough study of existing DDoS detection techniques. In the study, a comparative analysis of the overall performance, and a set of sensitivity analysis of the impact factors are carried out. Inspired by the analysis result, in Chapter 4, we propose an inline hybrid detection model to achieve an accurate detection with minimal false alarms. Chapter 5 proposes a DDoS defense system, which fulfills all the requirements by leveraging a SDN-centric architecture design. Chapter 6 presents a prototype implementation of the proposed defense system as well as its performance evaluation. Finally, Chapter 7 concludes the thesis and and some of the directions for future research.

2.0 Related Works

To successfully protect a server or network from DDoS attacks, the capability of capturing malicious traffic accurately is critical. However, DDoS detection alone is not enough. A successful defense should also facilitate a real-time response so that the impact of DDoS attacks can be mitigated effectively. This chapter provides an overview of necessary backgrounds of this dissertation and discusses the related existing works from these two perspectives, one in which is DDoS detection techniques and the other is DDoS defense systems, which bring together detection approaches and mitigation strategies as a whole.

2.1 DDoS Detection Techniques

Numerous schemes have been proposed to detect DDoS attack traffic. The early schemes focus on stochastic analysis to monitor network traffic flows' behavior and exploit the entropy of network traffic to identify normal behavior and detect anomalous intrusion events. Recently, the trend to detect DDoS attack traffic is to leverage machine learning techniques to classify and detect malicious traffic. In this section, we review both traditional and intelligent based detection techniques.

2.1.1 Traditional Detection Techniques

Traditional detection techniques characterize network traffic using statistical and information theory based analysis. They usually assume a predefined model to represent normal condition. For any traffic under monitoring, the statistics of the traffic are inferred and compared with the predefined model periodically. Any non-complying traffic is considered as an attack. In [23, 51, 80], the authors proposed detection schemes that monitor the ratios between number of packets received-from and sent-to the guarded network. They believe that the monitored ratio for the legitimate traffic should below a certain threshold for a spe-

cific protocol. DDoS attack traffic is then identified accordingly. In [11, 29, 75], the authors observe the strong correlation presented by either attack or legitimate traffic from different perspectives. They apply correlation coefficient analysis to network traffic parameters, then they compare the monitored changes and make decisions. In [87], the authors assume that legitimate users' behavior, including web page request interval, browsing length, etc., follows specific distributions. The detection scheme periodically calculates the standard deviation of sampled flows. If the flow has a sufficiently large calculated value, then it will be identified as attack traffic.

Information based metrics are also very popular in DDoS detection. Entropy can be computed on several features, such as network flows, source/destination IP addresses, number of packets, source/destination IP ports, etc., with a given time window [9, 18, 73, 81, 82]. The changes on the calculated entropy values are evaluated using different information metrics, and are used to identify the presence of DDoS attack traffic.

The challenge these methods face stems from the need to reliably define a normal profile and accurately differentiate between normal and anomalous behaviors. With sufficient number of active compromised machines, the legitimate users' statistical behavior is easily to be mimicked [88]. Hence, the fundamental assumption of proposed works will be violated, and the defense schemes are deceived. Additionally, the performance of these detection schemes can be impacted severely by the location where the detection scheme is deployed [43].

2.1.2 Intelligent Based Detection Techniques

Recent DDoS detection research has shown the promising results by leveraging machine learning techniques. These techniques are capable to intelligently learn the underlying data attributes without the need to explicitly describe normal and malicious activities, thereby overcoming the limitations of traditional detection schemes. Intuitively, DDoS detection problem can be modeled as a binary classification problem, where the monitored traffic is classified as either legitimate or attack traffic. Different classification algorithms have been applied and tested on well-known benchmarks. To improve the detection accuracy while minimizing the false alarms, researchers have also explored potential feature sets [26, 56].

Deep learning, a broad family of machine learning techniques, has been successfully applied to solve challenging problems in a number of fields, including computer vision, social network filtering and video games, natural language processing and machine translation, healthcare and bioinformatics, medical image analysis and drug discovery. Its application to DDoS detection, however, has been limited. In this section, we discuss various recent notable works in this field. Several works utilize autoencoder, an unsurpervised learning technique, to discover non-linear representations from the input data, and then they apply a classification method to distinguish malicious traffic from legitimate traffic [3, 27, 54, 68].

In [27], the authors combine an auto-encoder with a soft-max regression layer for network intrusion detection. The proposed scheme exhibits promising performance in both binary and multi-class classification task. The authors then extend their work by stacking two auto-encoders to detect DDoS attacks [54]. In [68], the authors stack two autoencoders to learn the complex relationships among features. They claim that soft-max layer is weaker than classic classifiers, and combine the stacked auto-encoder with a Random Forest classifier for intrusion detection. In [3], the authors utilize autoencoder not only for feature learning but also to reduce the number of random variables under consideration. Instead of connecting a classifier with the output layer of the autoencoder, a hidden layer, which represents the compressed features, is used as input for the classifier. A Support Vector Machine (SVM) is used as the classifier. The authors claim that the SVM outperforms other classic classifiers, in terms of accuracy. Although they successfully overcome the difficulty of feature selection, the proposed schemes do not address the challenges of feature extraction.

In addition to autoencoders, RNN is a popular choice for intrusion detection [35, 70, 72, 85, 89]. RNN, an extension of a conventional neural network, is widely used for modeling sequential data to solve time-series problems. Most schemes apply RNN models to well formatted datasets, such as KDD99 and NSL-KDD, which use hand-crafted flow-level feature engineering. In these datasets, each record corresponds to a specific network flow, and is represented by a set of attributes, such as duration, number of packets, number of bytes, *etc.* Most proposed schemes treat the set of attributes for each flow as a sequential data, which is then used as input for the RNN models [35, 70, 72, 85]. To improve the classification accuracy and other evaluation metrics, these works apply different types of RNN models.

A simple RNN is adopted in [85]. In [35, 70], the authors use LSTM RNN, and focus on selecting a minimal subset of features. In [72], the authors used Gated Recurrent Unit (GRU) RNN. Although the performance of these proposed schemes surpasses traditional machine learning methods, they fail to address the challenge of feature extraction. More importantly, assuming temporal order among features in RNN models is questionable. The advantage of RNNs in modeling sequential data is that they have a memory cell which can remember data received earlier and capture the temporal dependency among data. Although features are not independent of each other, preserving order among data is not necessary.

In [89], the authors propose a RNN-based model, referred to as *DeepDefense*. Unlike previously discussed methods, the model does not use flow-level statistical features. Instead, the authors extracted 20 features from packet headers and applied sliding windows to separate continues network traffic into sequences of network packets. For a given sequence of packets, the model classifies the last packet either as a legitimate or attack traffic. Their proposed deep learning model achieves lower error rate than traditional machine learning techniques. However, reliance on packet-by-packet inspection may not be practical in a real-world setting.

Although the use of machine learning based approaches holds promise, most published works remain offline in nature, with potentially prohibitive high overhead. Additionally, improving the detection performance by tailoring features, models and parameters for specific datasets does not benefit our community any further to solve the practical problem. For detailed description of related works in this category, we refer interested readers to our recently published survey paper, in which we reviewed highly cited schemes that have been proposed over the last decade, and empirically studied the advantages and limitations exhibited by intelligent based detection techniques [44].

2.2 DDoS Defense Systems

Building an effective DDoS defense system is a non-trivial problem for both the network administrator and network security researchers. Traditionally, the defense system is deployed

on dedicated devices to monitor network traffic, identify malicious activities and mitigate the potential adversarial impacts. Recently, SDN has emerged as a powerful platform for enabling innovation in networking research and development. With the separation of the control logic from the data plane, SDN provides more flexibility to network managing. In this section, we review DDoS defense systems in both platforms to present the state-of-the-art in the field of combating DDoS attacks.

2.2.1 Router-Based Defense System

Over the past few decades, academia has proposed numerous defense systems to defend against DDoS attacks. In a conventional network, the defense system is usually deployed on routers, so that malicious network traffic can be blocked before they enter a domain.

Filtering-based systems are among the earliest works [6, 7, 10, 17, 47, 65, 79]. A filter is essentially a rule explicitly telling the router to drop traffic that is identified as undesirable. Hence, the effectiveness of these systems heavily relies on a mechanism to differentiate attack traffic and legitimate traffic. Theoretically, any detection mechanisms can be integrated into this type of system and guide the filtering rules. However, routers are facing large scale and intensive network traffic, sophisticated detection algorithms are usually not affordable to be deployed on them. Most approaches try to identify the spoofed packets, and propagate filter rules to block them close to the source. In [6, 7, 17, 47, 65], the authors employ different packet marking algorithms to trace the path of spoofed IP packets. Using marking algorithms, routers mark packets along their path to the destination. At the edge router close to the destination, the detection scheme utilizes the aggregated information to reconstruct the routing path, and identify the spoofed traffic. In [79], the authors leverage the hop counting to identify spoofed malicious packets. These schemes usually cause low router overheads and are easy to implement. However, today's DDoS attacks, which are usually well distributed with a fairly large botnet, increase the chance of wrong construction of the path. Furthermore, due to large number of compromised devices, the attackers disclose real IPs of zombie machines. The packet marking and other traceback methods would not be able to identify attack traffic.

Another type of approach is called capability-based systems. In these systems, the sender has to request and receive explicit permissions from the destination, also called capabilities, before further sending out any traffic [48, 59, 83, 84]. The capabilities are used as a form of authentication, and are stamped on traffic packets. The router, which is a deployed point, will verify the authentication along the path. To avoid the collusion and attack against the request channel, researchers also proposed self-created capabilities, without cooperating with remote routers [34, 53, 77]. These systems rely on unpractical assumption and incur significant upgrade in the Internet core.

From an industrial perspective, few academic proposals have been deployed. Many security service provide companies, such as Akamai, Cloudflare, etc., provides DDoS-Protection-as-a-service. They leverage the capacity of their geographically distributed cloud servers, and use DNS or BGP to redirect traffic when under attack [16]. In section 1.1.2, we have discussed limitations of this scrubbing center technique.

In a conventional network, defense systems have to be deployed on special designed hardwares, such as network routers or middlebox devices. The need for high computational and resource requirement, due to continuous analysis of network traffic, remains a challenge. Additionally, how to deploy defense systems with little changes and cost to the current network infrastructure is still an open question.

2.2.2 SDN-Based Defense System

SDN is a new network paradigm, which physically separates the control and data planes in traditional network devices [38]. Specifically, the control plane is removed from a network device and implemented on a specialized central controller. Comparing to legacy networking architectures, SDN provides more flexibility and ease in network management, and offers new opportunities for enhancing the performance of network security.

In [66], authors developed a detection module, which continuously monitors the *Packet_In* messages rate, memory usage and CPU utilization rate of switches, and decides whether the network is under an attack accordingly. When an attack is identified, the defense system detours traffic to the neighbor switches to relieve the bandwidth pressure. Detoured traffic

will be analyzed. The identified attack packets will be filtered out, and legitimate traffic will be delivered to the destination. Although the solution could effectively protect the victim, the introduced delay and intensive communication burden between switches and the controller are not desired.

In [32], researchers proposed a security scheme using joint entropy for both DDoS detection and mitigation. When a congestion is detected, switches send packet header information to the controller. The detection module calculates the joint entropies for feature pairs and creates the current status profile. If the difference between the calculated current profile and the normal profile exceeds a certain threshold, the feature pair is identified as suspicious and sent to switches. With the given feature pair, switches calculate and update the suspicious feature pairs profile, and send back to the controller. The mitigation rules will be then generated. Before a mitigation rule is generated, switches have to communicate with the controller twice for each feature pair. Additionally, switches are required to apply extra calculations with their limited computational resources. The overheads brought by the proposed system are heavy, even disregarding the challenge of how to reliably define a normal profile.

Recently, the trend to mitigate the impact of DDoS attacks is to incorporate "intelligence" into the defense architecture, leveraging machine learning techniques to classify and detect malicious traffic [43]. In [61], researchers employed SVM and Self-Organizing Map for attack detection. The detection module periodically requires flow statistics from the switches, including flow duration, number of packets, bytes and etc. Based on the flow information, the detection model labels the inspecting flow as legitimate or attack, and a history based filtering rules are applied to mitigate the attack traffic. In [24, 41], researchers employed Convolutional Neural Networks for detecting DDoS attacks. Both papers extracted flow-level features for the detection model, such as backward packet length standard deviation, average packet size, flow duration and flow inter arrival time. In [45], the authors combined the signature attack databases and several machine learning techniques to achieve a high accurate detection results. All these proposed approaches are evaluated with well formatted datasets, in which statistical features of network flows are provided. However, in reality, flows features are not known in advance. Approaches usually periodically extract flow statistics from switches, the extracted incomplete flow statistics may distort traffic patterns

and degrade the detection accuracy. This problem is widely neglected in machine learning based approaches, which is elaborated in our results discussion.

Different from other machine learning based defense approach, in [55], the authors do not use machine learning model for detection. They utilized LSTM model for predicting network traffic status, such as bits/s, packets/s, source IP entropy, and etc. Then applied fuzzy logic to identify anomalies based on the prediction. After anomaly is detected, the mitigation rule will be installed in switches. As discussed in section 1.1, applying mitigation policy without distinguishing malicious traffic from legitimate traffic will degrade the quality of service for legitimate users, although attack target could be protected.

Characteristics of SDN architecture, such as the logically centralized control, the programmatic framework, and vendor agnostic implementation interfaces, solve the challenges that router-based defense systems are struggled to overcome. These systems capitalize on the centralized controller of SDN, and usually simply assume that the global view of the network traffic is available via the centralized controller, without considering the limited computation and communication resources host by the controller. The detection and mitigation strategies are, naturally, designed and implemented as applications attached to a centralized controller. However, naively utilizing the centralized controller could cause the defense system itself become a bottleneck. The tremendous and heterogenous data required from switches could cause the communication channel between switches and controllers under congestion, and further exhaust resources of both the controller and switches.

2.3 Summary

This chapter reviews related work in the fields of both DDoS detection and DDoS defense system. We begin with the traditional DDoS detection approaches. Then, the recent trends of leveraging machine learning techniques for DDoS detection are discussed, each with its advantage and limitations. Lastly, we survey the existing works for DDoS defense system, including router-based defense systems and SDN-based defense systems. We recognize the opportunities offered by SDN, but also point out that given the heavy duty, naively attach designed defense system as an application to the centralized controller could easily the and would result in paralyzation or misbehaving of all switches under the controller.

These existing approaches fall well short of desired capabilities as an efficient and effective DDoS defense system, which motivate this thesis to address the challenges.

3.0 An Empirical Study on Machine Learning Based DDoS Detection Techniques

An in-depth understanding of existing detection strategies is demanded for developing an effective detection approach. Traditional methods have been studied extensively. Both their advantages and disadvantages have been well analyzed, and their performances have been systematically evaluated by researchers [60, 90]. However, there is a lack of a such analysis for machine learning based techniques. Although a couple of research works have studied several published schemes, it is lack of a "holistic" analysis of the performance for these detection strategies. In this chapter, we set to investigate the performance of machine learning based DDoS detection techniques to reveal the advantages and limitations.

3.1 Introduction

A number of research works have studied DDoS attack detection with machine learning based solutions [12,57,69]. These works often focus on specific schemes, with the aim to carry out a comparative analysis of their performance. Although informative of the state-of-the-art in intelligent techniques to detect DDoS attacks, the intricacies of the studied schemes add significant complexity to the analysis, and often affect the outcome. Furthermore, the focus on specific schemes constrain the applicability of the results to other schemes. Lastly, very little research work addressed "holistically" the performance evaluation of intelligent solutions to DDoS defense. As such, the outcome of these studies remains inconclusive. A closer look at intelligent DDoS defense schemes reveals that at the core of the proposed schemes is a set of commonly used machine learning based "building blocks" for DDoS detection and prevention. To this end, a representative class of intelligent techniques, which capture main trends in DDoS detection, is carefully selected. Rather than carrying out an exhaustive comparative analysis of specific schemes, we take a different approach, focusing the impact of incorporating machine learning techniques to the DDoS defense.

3.2 Selected Techniques

To select a representative set of machine learning based techniques, a thorough search of the academic publications using Google Scholar, IEEE Xplore and Science Direct, is carried out. The search space was limited to the publishing period ranging from 2008 to 2018. Additionaly, only publications with high citation numbers were considered. Using this process, 49 academic publications are included in this work. These publications are used to extract the building blocks underlying their DDoS detection schemes. Furthermore, to gain better understanding of the capabilities of machine learning based techniques compare to traditional techniques, we include D-WARD [51], a statistics based DDoS defense scheme, into this study. Table 1 lists the evaluated techniques, and categorizes them into four groups.

Table 1: Detection Techniques Taxonomy

Categories	Techniques
Classification	Support Vector Machine (SVM)
	Artificial Neural Network (ANN)
	Decision Tree (DT)
	Naive Bayes (NB)
Clustering	K-Means
Nearest-Neighbor Based	K-Nearest Neighbor (KNN)
Statistical	D-WARD

3.3 Benchmark Dataset

In order to carry a meaningful and fair comparative analysis, an ideal benchmark that closely reflects the real-life network traffic, with clearly labeled legitimate and attack traffic, is necessary. In this study, we combine two widely accepted and extensively used benchmarks. The first benchmark is CAIDA DDoS dataset, collected from an actual DDoS attack event [15]. The most significant advantage of this dataset is that the traffic consists of a real-world DDoS attack scenario. Moreover, DDoS is exclusively recorded in this dataset, thus making it appropriate for evaluating intrusion detection schemes solely for DDoS. However,

the non-attack traffic was removed from the dataset. To augment the CAIDA benchmark with legitimate traffic, a second benchmark, DARPA dataset [21], is used. The DARPA benchmark contains total of 5 weeks traffic. The first and third weeks do not contain any attacks.

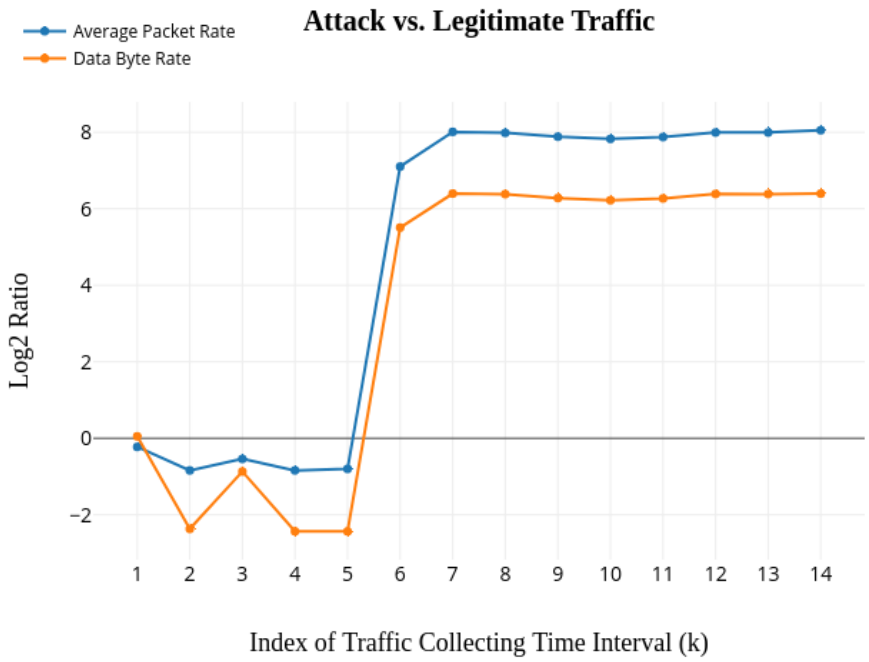

Figure 2: Packet Rates of Legitimate and Attack Traffic

The attack traffic, obtained from the CAIDA benchmark, is split into 14 sets, each containing roughly five minutes traffic traces. The legitimate traffic is selected from DARPA first week attack-free dataset. We use tcpreplay [74] to replay the attack and legitimate traffic, and recapture the combined traffic for analysis. Now, we denote the structure of our datasets, D, as:

$$D = \{d(I_k), 1 \leq k \leq 14\} \text{ where,}$$

$$d(I_k) = \{\text{traffic that collected over the } k\text{th time interval } I\}$$

$$||I|| = 5 \text{ minutes}$$

Figure 2 displays the ratio of attack traffic to legitimate traffic for each dataset $d_k(I)$, in terms of average packets rate (packets/second) and Byte rate (Bytes/second). In the figure, X-axis presents the index of the time interval (k), and Y-axis presents the average packets rate in a base-2 logarithm scale. It is clearly shown that, for $k < 6$ (the first 25 minutes), the volume of attack traffic is less than the legitimate traffic, however, starting from $k = 6$ (the 6th 5-minutes period), the attack traffic volume is dramatically increased. During $k = 10$ period, the attack traffic volume is reduced obviously, then followed by another gradually increase. We believe these patterns reflect different attacking phases of the DDoS event. We will explore the impact of these observations in the section of Experiments and Results.

3.4 Performance Evaluation

3.4.1 Comparative Analysis

In this experiment, we conduct a comparative analysis of the overall performance of the selected techniques. We assume that the entire network traffic is available for each DDoS detection technique. Although the assumption is usually impractical, it provides a strong base to evaluate and compare the detection capabilities of these techniques. The benchmark datasets, D, is divided into T_r and T_s, for training and testing purposes, respectively.

$$T_r = \{d_1(I)\};$$
$$T_s = \{d_2(I), d_3(I), d_4(I), ..., d_{14}(I)\}$$

Classification models are usually trained offline with historical data. The trained model is then applied online to classify future data. In our benchmark datasets, $d_1(I)$ contains the network traffic that is captured during the first 5 minutes of the monitoring interval. It is considered to be the historical data and used to form T_r. T_s is composed of the traffic captured over the remaining 13 5-minute intervals. The 13 traffic sets are tested independently for each evaluated technique, using the accuracy, sensitivity and specificity metrics. Boxplots are used to illustrate the assessed performance, so that the performance variation of these evaluated DDoS detection techniques is also quantified.

3.4.1.1 TCP Traffic

Results for TCP traffic using packet-header features are shown in Fig. 3. In the figure, Boxplots are used to illustrate the performance of each evaluated technique. Values of minimum, maximum, median (solid line in box), mean (dot line in box), first quartile, and third quartile are displayed by the box for the 13 test cases. X-axis presents each evaluated detection technique. Subplots present Accuracy, Sensitivity and Specificity, respectively. The results show that DT outperforms all other techniques, with respect to the defined performance metrics. Furthermore, the results show that DT performs consistently across different testing sets. These results show the high potential of rule-based strategies to efficiently detect DDoS TCP attack traffic in the packet level. In this experiment, NB and RBF-SVM exhibit the worst performance when it comes to distinguishing attack packets from legitimate traffic packets.

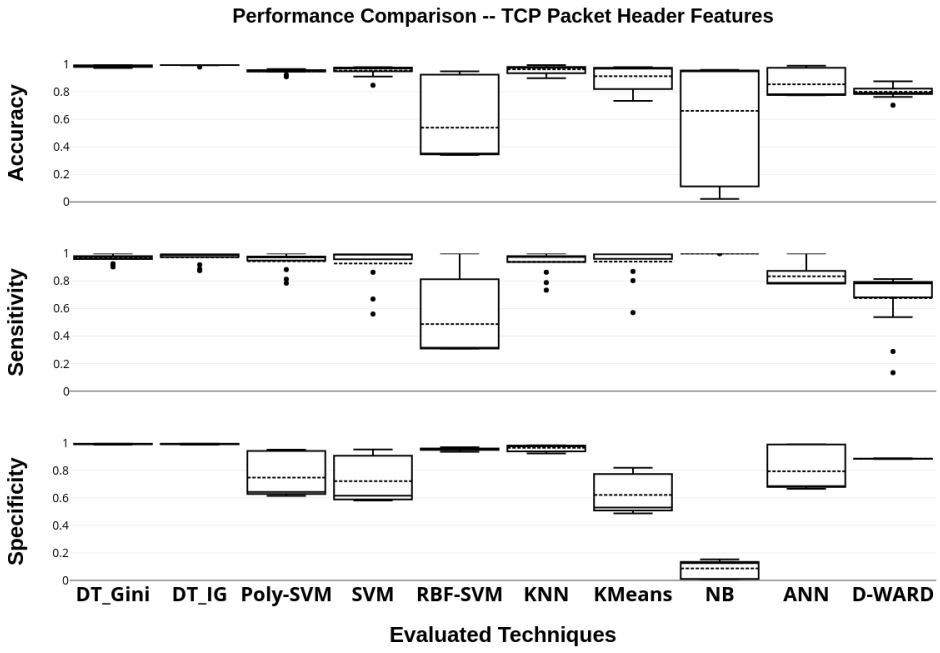

Figure 3: Performance Comparison – TCP Traffic with Packet Header Features

The NB classifier is a simple classifier based on Bayes' theorem, assuming a strong independence among the features. As such, given the packet-header features, the NB classifier is

biased towards labeling most samples as attack traffic during the testing phase. This results in an extremely high sensitivity score, but a poor specificity score.

The RBF kernel maps features into an infinite dimensional space to solve non-linearly separable samples. This may lead to a lose of generalization, if the training samples are underrepresented. In other words, the trained model fits the training samples too closely, causing the model to become very sensitive to the input data. Fig. 4 presents the performance of RBF-SVM in each test case. For the first four test cases k, $(k = 2, 3, 4, 5)$, RBF-SVM shows a good performance. However, for test cases k, $k \geq 6$, its accuracy and sensitivity scores are significantly decreased. Recall that in the attack event, presented in Fig. 2, the attack volume increases significantly starting from $k = 6$. The increase in volume affect the distribution of traffic attributes, which in turn causes the underlying patterns of the input dataset to become significantly different from that was learned during the training phase.

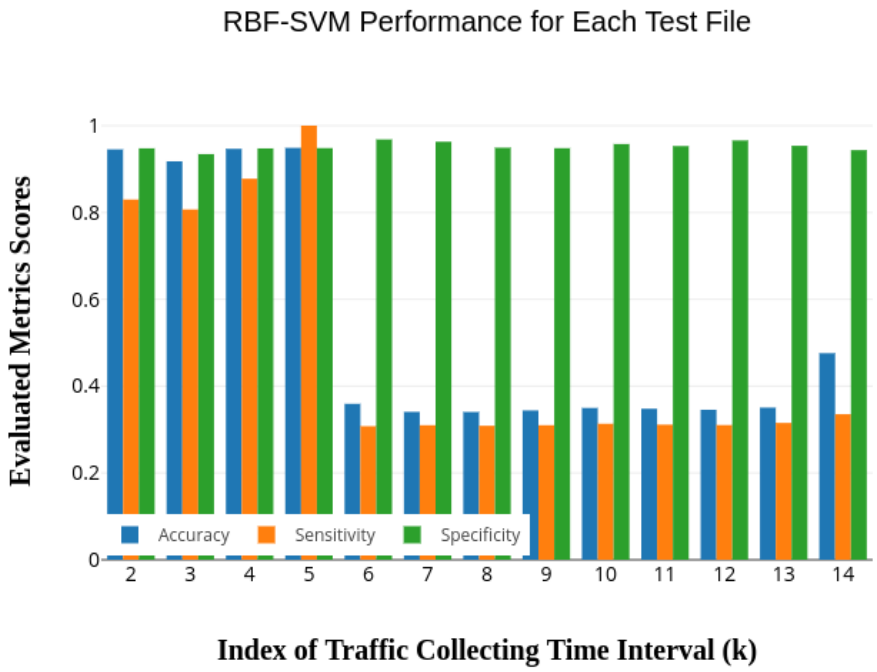

Figure 4: RBF-SVM Performance Using Packet Header Features

Focusing on the sensitivity score, the results achieved by SVM, Poly-SVM, KNN and KMeans, shown in Figure 3, are comparable to those achieved by DT. They all outperform D-

WARD. However, the poor specificity scores of the these machine learning based techniques suggest a potentially high rate of false alarms which can incorrectly prevent legitimate users from accessing resources.

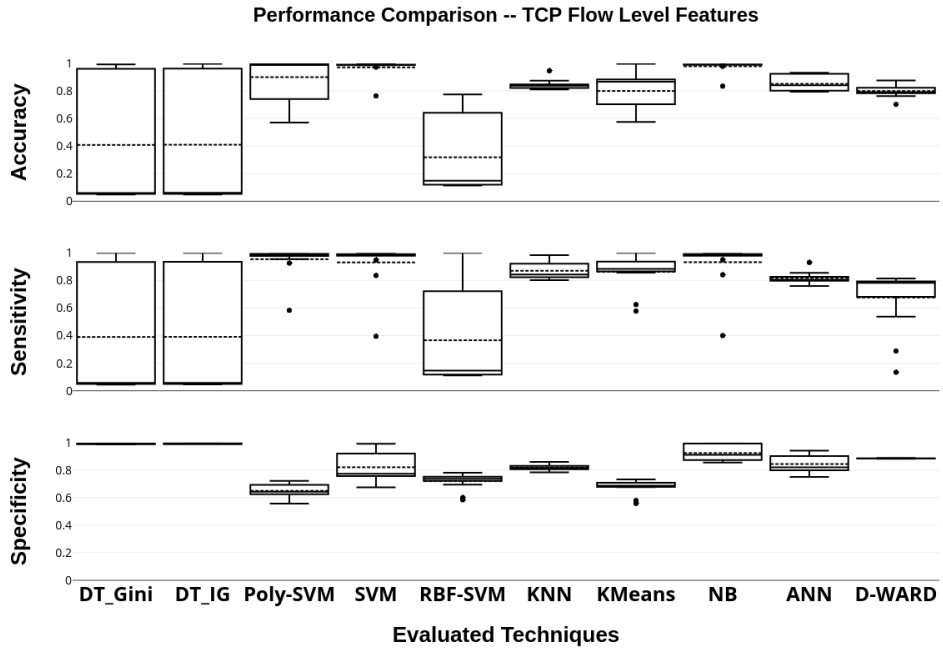

Figure 5: Performance Comparison – TCP Traffic with Flow Level Features

Results for TCP traffic using flow-level features are shown in Figure 5. DT no longer demonstrates the observed superiority over all other machine learning based techniques. Moreover, its accuracy and sensitivity scores are highly varied across all datasets. Examining closely its performance on each test case, DT suffers the over-fitting problem, similar to the pattern of the RBF-SVM behavior, shown in Figure 4. Additionally, using flow-level features, which represent features from the aggregated packet data between a source and a destination, significantly improves NB's accuracy and specificity. The performance consistency of NB across multiple test cases is also enhanced by using flow-level features. In comparison with packet-header features, flow-level features provide statistic attributes that capture the traffic behavior. Hence, the distribution assumption and the probability inferring of NB classifier is more reasonable.

Finally, it is worth noting that the traditional method, D-WARD, performs competitively

in comparison to machine learning based techniques. However, it failed to identify attack traffic in two datasets, as indicated by the poor sensitivity score for these two cases. Overall, it can be concluded that ML-based techniques can achieve high performance in detecting TCP attack traffic. Furthermore, ML-based techniques outperform D-WARD in most test cases.

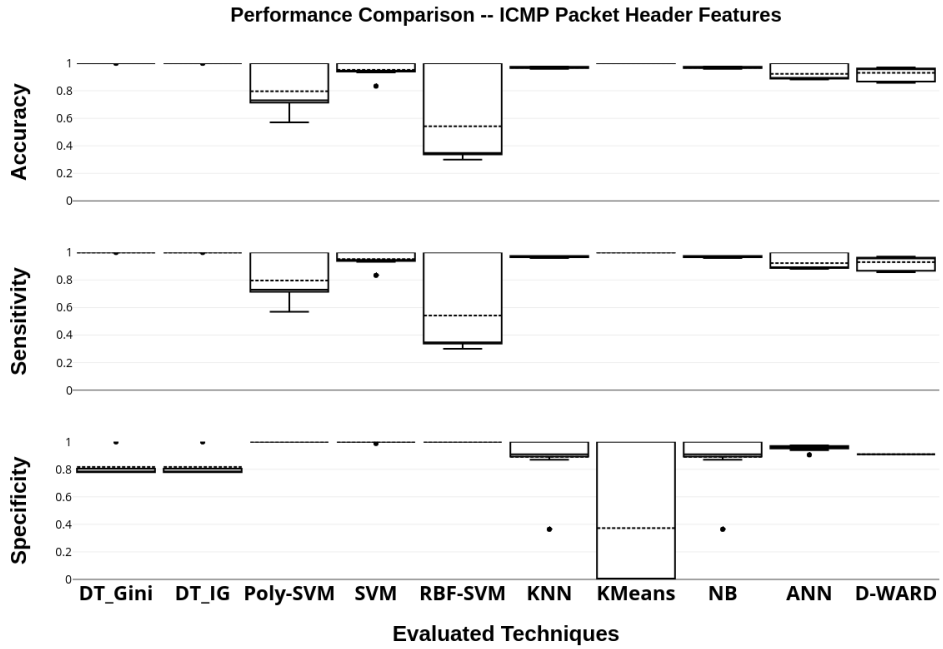

Figure 6: Performance Comparison – ICMP Traffic with Packet Header Features

3.4.1.2 ICMP Traffic

Results for ICMP traffic using packet-header features are shown in Figure 6. The results show that all ML-based techniques achieve near optimal values with respect to all performance metrics, but only for some datasets. Contrarily, the performance consistency of D-WARD is relatively higher. For ICMP traffic, D-WARD detects attack by monitoring the paired messages of ICMP requests and the corresponding relies. If the monitored ratio exceeds the threshold, the alarm is raised. The results show that this simple mechanism works effectively. Features that capture this type of information should be able to improve the performance of ML-based techniques.

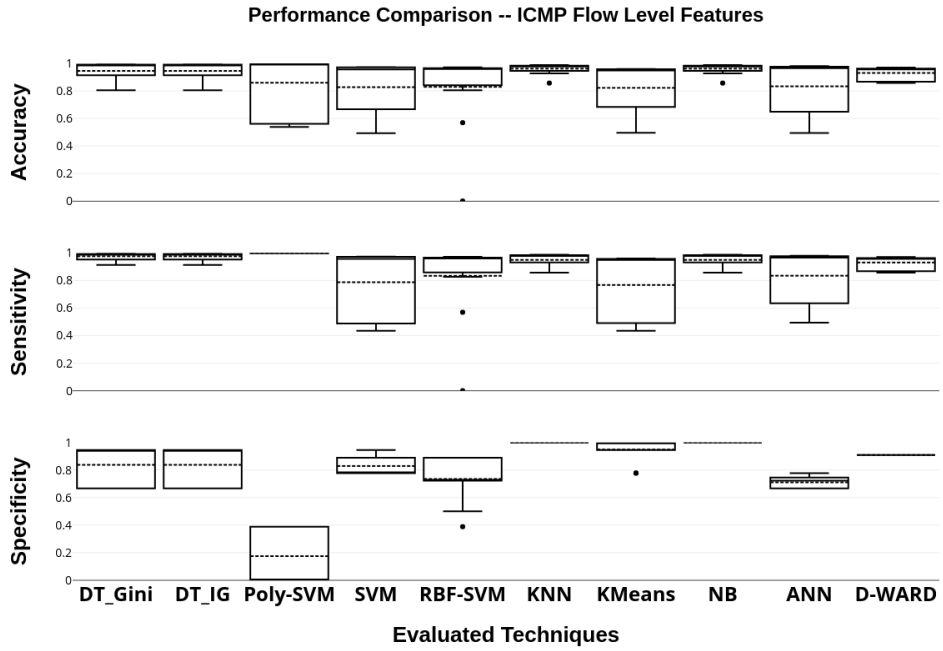

Figure 7: Performance Comparison – ICMP Traffic with Flow Level Features

Results for ICMP traffic using flow-level features are shown in Figure 7. The results exhibit a similar performance pattern as the one in TCP traffic, underscoring the fact that using sophisticated features does not necessarily improve the performance of all ML-based techniques. As indicated by the results, only specific ML-based techniques, such as Poly-SVM and NB, are improved. The use of these features did not significantly improve the performance of the other schemes. In some cases, the performance of these latter schemes has decreased.

In summary, the outcome of this experiment shows that it is not clear that a single technique outperforms all others in all test cases, especially when focusing on the ICMP traffic. The experiment shows that different techniques perform better when using certain types of features, suggesting that feature selection should be method specific. Furthermore, the capability of detecting attack traffic shown by ML-based techniques is evident. On the other hand, the performance inconsistency exhibited by ML-based techniques in dealing with different types of attack traffic raise doubts about their ability to efficiently detect DDoS attack in real world scenarios.

3.4.2 Impact of Observable Traffic Proportions

In the comparative experiment, the entire network traffic is assumed to be available for each DDoS detection scheme. In practice, however, it is infeasible for a detection scheme to have access to the entire network traffic. Actually, the detection scheme is usually deployed on, or attached with, routers or switches. Consequently, a detection scheme can only observe the network traffic passing through the network device on which it is deployed. The purpose of this experiment is to investigate the ability of a detection technique to only access a limited portion of network traffic.

To emulate a realistic network environment, we randomly select a proportion p of the total traffic for testing. The selected traffic is then analyzed by the detection techniques, and the performance for each metric is evaluated. Two selection criteria, namely packet- and flow-level, are used to generate a specified portion, p, of the network traffic, For packet-level, a proportion, p, of the total traffic packets is randomly selected without any consideration of the flows to which the selected packets belong. For flow-level, however, a proportion, p, of the total traffic flows is randomly selected and only packets belonging to these flows are made available to the DDoS detector. It is to be noted that the total number of packets selected is different for each case.

The same training data set T_r, used in experiment 1, is also used for this experiment. The traffic proportion p is selected from a set $P = \{1\%, 5\%, 10\%, 20\%, 50\%, 75\%\}$. We do not use separate symbols to differentiate packet- and flow-level datasets, since they are structurally the same. The random traffic selection of DDoS detector observed traffic is repeated 10 times, with different random seeds, to avoid data bias. The dataset used for testing is denoted by D_{exp2}, where $D_{exp2} = d^p_{k,r}(I)$, $1 \leq r \leq 10$, $p \in P$ and $2 \leq k \leq 14$. In total, 780 test cases are used to assess the performance of each technique, for both packet- and flow-level selections.

To quantitatively evaluate the impact of the observable traffic proportions, Pearson Correlation Coefficient test is applied to measure the strength of the linear correlation between each evaluated metric and the observed proportion. For both packet- and flow-level traffic selections, the performance of D-WARD presents a strong correlation with the increasing

observed traffic proportion. Conversely, only a weak correlation between traffic proportions and performance is exhibited by ML-based techniques. Table 2 shows results for packet- and flow-level traffic selections.

Table 2: Correlation Coefficient Scores with Observable Traffic Proportions

Techniques	Packet-Level Correlation Score with			Flow-Level Correlation Score with		
	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
DT-Gini	-0.0024	-0.0123	-0.0117	-0.2698	0.0232	-0.3496
DT-IG	-0.0103	-0.0103	0.0	0.0549	-0.0159	0.0253
SVM	-0.0002	-0.0074	0.0006	-0.0177	0.2045	0.0147
RBF-SVM	-0.0002	-0.0022	0.0057	-0.2732	-0.2384	0.2769
Poly-SVM	-0.0003	-0.0065	0.0008	-0.2473	0.0666	-0.0661
KNN	-0.0016	-0.0088	-0.0024	-0.1760	0.0885	0.01744
KMeans	-0.0011	-0.0058	-0.0008	-0.0912	0.0300	-0.1349
NB	<0.0001*	0.0215	-0.0011	-0.0946	0.1870	-0.4023
ANN	0.0001	-0.0012	0.0008	-0.1283	-0.0456	-0.0857
D-Ward	0.5458	0.8008	-0.5519	0.7703	0.8055	0.6329

*The value is negative, and $\|value\| < 0.0001$

Recall that traditional detection techniques, represented by D-WARD, usually infer network status through monitoring two-way traffic. As such, D-WARD gains a relatively complete picture of the network status given a higher proportion of observed traffic packets or flows. This leads to more accurate attack detection. Ideally, D-WARD should be deployed at the only boarder router, so that both directions of flows can be observed by the detector. However, this is impractical, a limitation also reported by the authors of D-WARD [51]. Comparing to the traditional detection method, the weak correlation, presented by ML-based techniques, shows that the deployment location does not impact the performance of ML-based detection techniques.

3.4.3 Impact of Attack Intensities

The focus of the previous experiment was on the proportion of network traffic observed by a detector. In this experiment, we further refine the previous experiment to focus exclusively on the intensity of the attack traffic observed by the detector. Consequently, legitimate traffic

is kept unchanged, and attack traffic is increasingly injected into the network. Specifically, we randomly select $a\%$ of total attack traffic, measured in both packet- and flow-level. As the value of a increased, so does the attack intensity.

The same training dataset T_r, used in previous experiments, is also used in this experiment. The increase of the attack traffic is cumulatively achieved, whereby the increased attack traffic contains the previous attack traffic. The traffic injection procedure is repeated 10 times with different random seeds. We do not use separate symbols to differentiate packet- and flow-level datasets, since they are structurally the same. Hence, dataset used in this experiment is denoted as $D_{exp3} = d^a_{k,r}(I)$, where $1 \leq r \leq 10$, $a \in \{1\%, 5\%, 10\%, 25\%, 50\%, 75\%\}$ and $2 \leq k \leq 14$. In total, 780 test cases are used to assess the performance of each DDoS detection technique, for both packet- and flow-level attack traffic injections.

Similarly as in experiment 2, we use Pearson Correlation Coefficient test to quantitatively evaluate the impact of the attack intensities. Results of packet- and flow-level injections are shown in Table 3. As expected, D-WARD presents a strong positive correlation with the increasing attack intensities in terms of accuracy and sensitivity. With the attack intensity increases, the amount of observed traffic increases as well, which is critical for D-WARD to detect attack traffic correctly.

Table 3: Correlation Coefficient Scores – Attack Intensity

Techniques	Packet-Level Correlation Score with			Flow-Level Correlation Score with		
	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
DT-Gini	-0.2880	-0.0098	<0.0001	-0.2862	-0.0759	<0.0001
DT-IG	0.2901	-0.0103	<0.0001	0.2856	-0.0657	<0.0001
SVM	0.5777	-0.0039	<0.0001	0.5777	-0.0100	<0.0001
RBF-SVM	-0.4309	-0.0023	<0.0001	-0.4309	-0.0045	<0.0001
Poly-SVM	0.5314	-0.0026	<0.0001	0.5314	-0.0070	<0.0001
KNN	-0.0013	-0.0079	<0.0001*	-0.0008	-0.0255	<0.0001*
KMeans	0.6209	-0.0003	<0.0001*	0.6209	-0.0033	<0.0001*
NB	0.3873	0.0117	<0.0001*	0.3875	0.0568	<0.0001*
ANN	0.0795	-0.0010	<0.0001	0.0795	-0.0069	<0.0001
D-Ward	0.5728	0.7936	<0.0001*	0.5636	0.8101	<0.0001*

*The value is negative, and $\|value\| < 0.0001$

3.4.4 Impact of the Class Imbalance Problem

The class imbalance problem is frequently encountered in practice, where the number of observations of one class is far less than the other class. When this problem occurs in the testing phase, accuracy alone is no longer enough to assess the performance of the detection scheme. Different types of evaluation metrics, such as sensitivity and specificity used in this work, need to be used to complement the accuracy to better assess performance. When the class imbalance problem occurs in the training dataset, it may hinder the learning process of classification algorithms [37]. Practically, if the imbalanced class distribution in the training dataset matches the native class prevalence in the test scenario, then the dataset bias in the learning process can be neglected. Yet, the described scenario does not apply to DDoS attack detection.

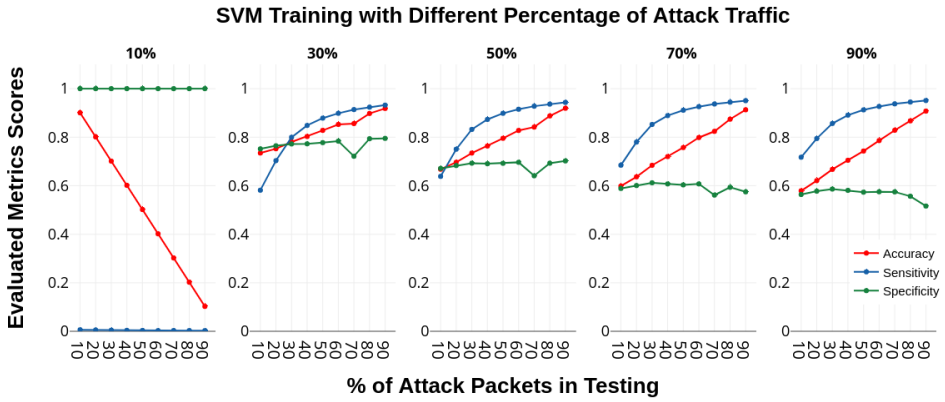

Figure 8: Class Imbalance Problem Analysis – Linear SVM

[1] The subtitle of each figure depicts the ratio of attack and legitimate traffic in the training sets. X-axis present the ratio of attack and legitimate traffic in the testing case.

From the perspective of a DDoS attack detector, attack traffic usually represents a very small subset of all network traffic it observed, particularly in stealth attacks. However, when an attack happens, the attack traffic may become the majority class among the traffic that is observed by the detector. Hence, it is common that the detection scheme is dealing with highly imbalanced dataset, and the dominant class is non-stationary. Represented in our

benchmark, the attack traffic is the minority class during the training phase, but it becomes the majority class after the attacker increases the attacking volume, shown in Figure 2.

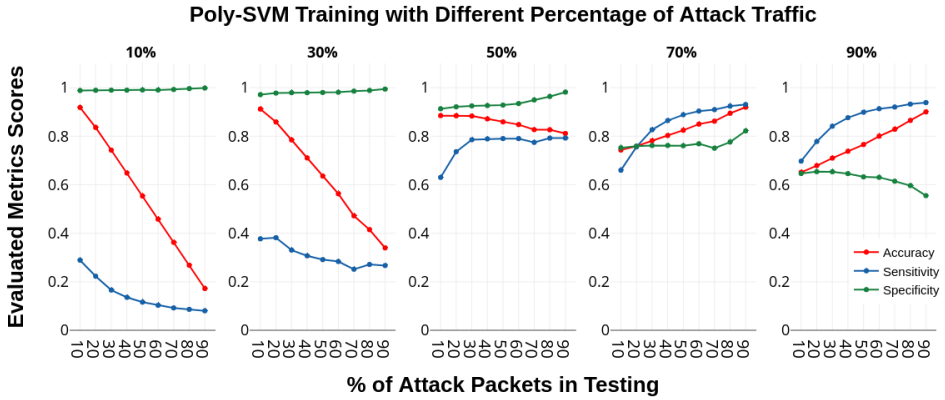

Figure 9: Class Imbalance Problem Analysis – SVM with Polynomial Kernel

To assess the impact of the class imbalance problem in the training datasets, we generate a set of training data with different degrees of imbalance. We apply a simple random under-sampling method to create five subsets from the training dataset T_r. Each subset contains 70,000 packets, and the percentage of attack traffic in each subset is drawn from $\{10\%, 30\%, 50\%, 70\%, 90\%\}$. All ML-based techniques are trained with each subset independently. Five models are then built for each technique. Due to the limited number of ICMP legitimate traffic, only TCP traffic is considered in this experiment. For testing, we apply five trained models on the testing dataset T_s from Exp.1. The results show that the family of SVM techniques exhibits the strongest sensitivity to the imbalanced training datasets, while other techniques are affected slightly.

To further study the correlation between the class imbalance of the training data and the performance, we generate nine subsets from T_s with different ratios of attack to legitimate traffic packets. Each subset contains 100,000 packets, and the percentage of attack traffic in each subset is drawn from $\{10\%, 20\%, 30\%, ..., 90\%\}$. The procedure of generating testing samples is repeated 10 times. In total, 90 test cases are used to assess the impact of the imbalanced and balanced training models on the performance of each technique. Figure 8- 10 displays the results of linear SVM, Poly-SVM and RBF-SVM. Trained with small percentage

of attack traffic, all SVM models tend to label most samples as legitimate traffic. Using a relatively balanced training dataset, the results show that the robustness of the model is improved. It is worth noting, however, that a balanced dataset does not necessarily achieve the best performance. A sophisticated kernel, such as RBF-SVM, can identify the hidden information by mapping features into higher dimensions. Specifically, RBF-SVM outperforms linear SVM when extremely imbalanced training data is used, although its overall performance remain less than optimal. On the other hand, the sophisticated kernel exhibits higher sensitivity to the data balancing, making it difficult to optimize its performance without degrading the robustness of the model.

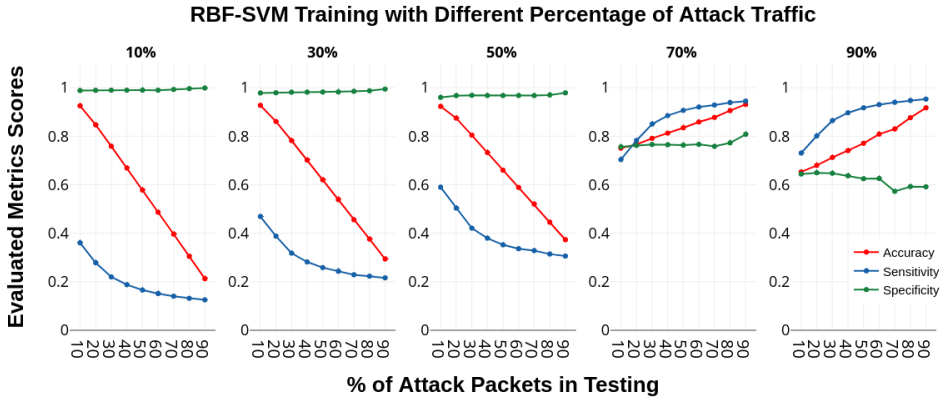

Figure 10: Class Imbalance Problem Analysis – SVM with RBF Kernel

In summary, the results clearly show that the impact of the class imbalance problem in datasets should not be neglected. Carefully designing the training process, analyzing the application scenario and choosing the appropriate method are critical for a successful intelligent DDoS detection scheme. Additionally, detection DDoS attacks in dynamically changing environment remains a challenge for ML-based detection methods.

3.5 Summary

In this chapter, we conduct a series of experiments to explore the advantages, limitations and influential factors for ML-based DDoS detection techniques. The detection capabilities exhibited by ML-based techniques are evident, although no single technique that outperforms all others in all test cases. Additionally, different techniques exhibit different preferences over feature types, emphasizing the significance of feature selection and suggesting that feature selection should be model oriented.

The sensitivity analysis illustrates the observed traffic proportions severely impact the performance of traditional detection methods that rely on monitoring the two-way traffic. Although ML-based techniques display weak correlation with the proportion of the observed traffic, the observed portion does indeed cause higher performance variance.

Lastly, we explored the impact of the class imbalance problem on the performance of ML-based techniques. The results show that the impact of the class imbalance problem should not be underestimated, especially with respect to the dynamically evolving nature of DDoS attacks. Future work can be focused on investigating an ensemble of intelligent schemes, strategically distributed across the network, using an appropriate feature selection model for an adaptive and efficient DDoS detection.

4.0 A Hybrid DDoS Detection Model

Machine learning techniques exhibit inevitable capability of offering high flexibility in the classification process, consequently improving the detection of DDoS attack traffic. These techniques are capable to intelligently learn the underlying data attributes without the need to explicitly describe normal and malicious activities. However, traditional machine learning techniques struggle to capture the evolving nature of DDoS attacks. We believe that a significant part of the problem stems from the failure of hand-crafted feature engineering to represent traffic behaviors. However, extracting desired features is costly, which is not affordable for an effective inline inspection for every single on-going network flows. To overcome these limitations, a light-weight Deep Learning approach is proposed in section 4.1, which is capable of distinguishing malicious traffic flow by inspecting only a small number of raw packets header information from each flow. As such, a truly real-time and affordable inline inspection is provided.

Although machine learning based techniques, such as the deep learning approach proposed in this chapter, hold great promises for accurately detecting DDoS attack traffic, most proposed DDoS detection solutions lost the connection to the problems that are required to address in the real world. Detailed challenges are discussed in section 4.2. Additionally, adapting a machine learning apporach to mitigate DDoS attack is not an easy job. Any assumptions or expectations for proposed machine learning approaches to be "ready-to-use" will lead to an ineffective mitigation system. To further address the challenges and fit the detection model to a distributed defense system, we introduce a network status awareness model, which plays a complementary role to the flow based detection model, and then extended the proposed detection model to facilitate a network-wide coordinate defense system.

4.1 A Long Short-Term Memory Enabled Framework for DDoS Detection

In this section, we describe a novel deep learning DDoS detection scheme, which only uses raw packet header information as input and does not require feature engineering. At the core of the scheme is Long Short-Term Memory, a Recurrent Neural Network(RNN) architecture, used to learn the network traffic behavior, and distinguish attack network flows from legitimate flows, by examining a relatively small number of packets from each flow. We start with the motivation behind this proposed scheme, then describe the details of the proposed detection model, and summarize this section with the detection performance discussion.

4.1.1 Motivation

A machine learning based traffic classification workflow usually requires a two-stepped process for feature engineering. In the first step, an appropriate set of features is extracted to characterize the signature of the collected data. In the second step, a feature selection algorithm is applied to eliminate irrelevant features [14]. This process is not only labour intensive, but also prone to errors. Firstly, generating a feature set that captures previously unseen attacks behavior is challenging, and often involves deep understanding of the network traffic behaviours and characteristics. Secondly, most feature selection algorithms are based upon the strong assumption that features are independent from each other [42, 46, 71], erroneously ignoring the intrinsic temporal and spatial correlations between the features. Lastly, training and feature selection, which are treated as two separate phases of the classification workflow, cannot be jointly optimized, thereby hindering the overall performance of the detection scheme. Deep learning holds promise for addressing these key limitations.

Deep learning methods stack multiple layers of non-linear transformation hierarchically, so that these methods can automatically extract complex representations hidden inside the raw input [40]. For the task of classification, with ascending the layers, representations that are important for class discrimination are amplified, and the irrelevant representations are suppressed [91]. This process inherently embeds feature selection. As a result, both the

transformations of the raw data into the distinctive representations and the classification of these data into legitimate and malicious traffic are optimized jointly within the training process. In addition to address these limitations, most importantly, the capability of deep learning methods to automatically extract features from raw input data offers a practical solution for a real-time flow by flow inspection.

To mitigate the DDoS attacks impact effectively without causing collateral damage to legitimate users, a desired detection model should be capable of providing network flow by flow based detection. A network flow is usually characterized as a sequence of packets that share the same $\langle$ *source IP*, *source Port*, *destination IP*, *destination Port*, *Protocol* $\rangle$. The ability of RNN to learn non-linear representations from sequential data makes it a natural fit to determine if a sequence of packets is or is not malicious. RNN extends the traditional feed-forward neural network by introducing a directional loop, so that the sequential dependence between the current packet and the historical information carried by previously observed packets is preserved. Thus, we propose to use a RNN model for DDoS attack traffic detection.

4.1.2 Proposed Detection Scheme

Formally, a network flow, which consists of N network packets, can be described as a sequence:

$$F = \{p^{(1)}, p^{(2)}, ..., p^{(i)}, ..., p^{(N)}\}, \quad p^{(i)} \in \mathbb{R}^m$$

where $p^{(i)}(1 \leq i \leq N)$ represents the ith packet of F. Each packet $p^{(i)} \in \mathbb{R}^m$ is an m-dimensional vector contains stored information $p^{(i)} = \{p_1^{(i)}, p_2^{(i)}, ..., p_m^{(i)}\}$. In our proposed detection scheme, the stored information is extracted from packets headers, including: ***source Port***, ***destination Port***, ***packet length***, ***Time To Live***, ***FIN***, ***SYN***, ***RST***, ***PSH***, ***ACK***, ***URG***, ***ECE***, and ***CWR***. To adequately explore the temporal information, we also include three temporal features, which are: ***time past since last packet***, ***time past since the first packet***, and ***average time interval between consecutive packets***.

The above defined sequence of packets can be viewed as the language of network. Inspired by the success of natural language processing, we employ LSTM as our detection model,

which is one of the most popular and efficient variants of RNN [62].

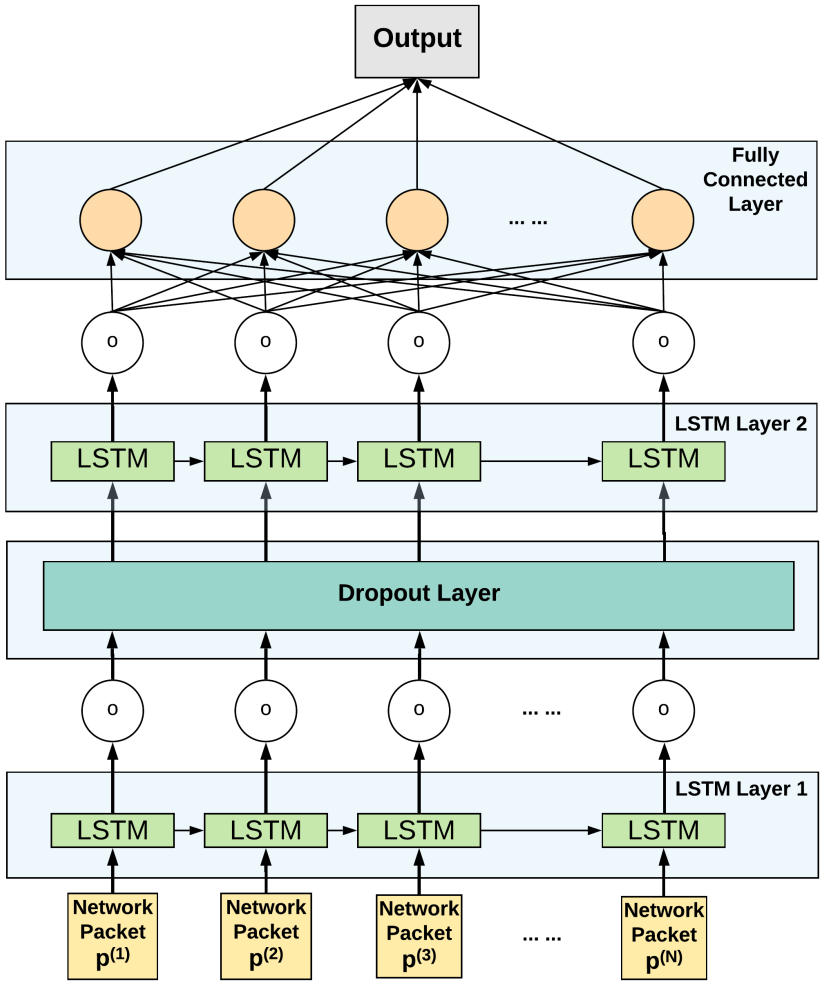

Figure 11: The Architecture of Proposed Deep Learning DDoS Detection Model

There is not a standard guidance for choosing the number of hidden layers for LSTM models. The rule of thumb is that two hidden layers should be enough for detecting complex features. To choose the proper number of layers and architecture for the LSTM model, we explored and evaluated multiple LSTM architectures, including one hidden layer, two hidden layers and with an embedded layer. The one hidden layer LSTM architecture and the LSTM model with an embedded layer are depicted in Appendices Figure 33 and A, respectively. The LSTM model with two hidden layers outperforms the other two architecture, hence, is

selected as the DDoS attack traffic flow-by-flow detector. Detailed description of the other LSTM models architecture and the performance evaluation can be found in Appendix A.

The selected two hidden layers LSTM model is depcited in Figure 11, which is a four-layered architecture, namely two LSTM layers, a dropout layer, and a fully connected layer. The LSTM layers learn both temporal and spacial representations from the input sequential data. Each unit in the LSTM layer contains three gates, namely *input*, *forget* and *output*, which work together to learn the transformation of input values, the relevant information from previously observed data, and the non-linear representation of the current state during training. The dropout layer masks a random fraction of the input units at each update step, while training the network. It adds noises to the LSTM layer, to avoid over-fitting and improve the robustness of the trained model. A fully connected layer is used for classification.

For each network flow, F, a subsequence of n packets, $S = \{p^{(j)}, ..., p^{(j+n-1)}\}, S \subset F$, is inspected. Using S, the model classifies the traffic flow as either legitimate or malicious. The value of n is pre-defined. If a flow does not have enough packets, S will be padded with fake packets. A fake packet is an m-dimensional vector with values of zeros. If a flow has more than n packets, only the first n packets are examined. The remaining packets are discarded. To reduce the detection delay, a time window threshold is applied. The network flow is examined when either n packets are observed or the time window threshold is reached. The sensitivity of the scheme to n is discussed in section IV.

4.1.3 Dataset & Data Preprocessing

The experimental evaluation framework uses a widely accepted benchmark datasets, CICIDS 2017 [67]. It was published by the Canadian Institute of Cybersecurity in 2017, and contains realistic background network traffic and a variety of attack traffic. The datasets cover five days of network traffic, two of which have DoS and DDoS attacks. We use these two days' traffic, denoted as Wednesday and Friday, as the evaluation benchmark. Table 4 presents the detailed attack types for each traffic collection.

Attack traffic in Wednesday was generated by four different tools. Three of them generated low-bandwidth application layer attack. These attacks require little bandwidth, and

Table 4: Attacks in the Experiment Dataset

Traffic Collection	**Attack Generated Tools**	**Brief Description of Attacks**
CICIDS 2017 Wednesday	HTTP Unbearable Load King (Hulk)	Volumetric Attack. Generate volumes of HTTP GET requests with randomly generated header values.
	slowloris	Low-Bandwidth Application Layer Attack. Open multiple HTTP connections. Continuously send partial HTTP requests
	slowHTTP	Low-Bandwidth Application Layer Attack. Send HTTP requests in pieces slowly, one at a time to a Web server.
	Golden Eye	Low-Bandwidth Application Layer Attack. Open multiple HTTP connections, and use "keep alive" packets.
CICIDS 2017 Friday	Low Orbit Ion Cannon (LOIC)	Volumetric Attack. Open multiple HTTP connections and continuously send HTTP request messages

very stealthy. They aim to keeping the HTTP connections as long as possible using similar but different strategies. Wednesday's traffic also contains a volumetric attack, which was generated by a tool named "Hulk". Hulk can flood the victim with huge HTTP GET requests from a single device. The packet header values of these requests are generated randomly to confuse the victim, and make it hard to be detected. Friday's attack was generated by Low Orbit Ion Cannon (LOIC). LOIC floods targeted server using junk TCP, UDP and HTTP GET requests through numerous attacking devices. Although classified as volumetric attack, the traffic behavior is more similar to low bandwidth application attack than traffic generated by Hulk, from the perspective of a single flow.

CICIDS 2017 datasets provide well formatted data files. In these files, each network flow is characterized by more than 80 statistical features, and associated with a label indicating whether it is a malicious flow. In addition to the well formatted network flow files, the raw trace files (in pcap format) are also provided. Since the raw data is not labeled, we need to reverse engineer the process to find the corresponding set of packets for each network flow. We carefully check the timestamp, the number of forward and backward packets, the

time duration, to make sure the mapping is correct and precise. However, the timestamp granularity is not fine enough to find all the mappings. We discard the network flows and packets that we did not find exact matches that satisfy our criteria.

4.1.4 Performance Evaluation

The goal of experiment 1 is to carry out a comparative analysis of the performance of the proposed scheme, which only use the raw packet header information, and the traditional machine learning methods, which rely on manually selected sophisticated features. In this experiment, we split the benchmark datasets into three parts: training, validation and testing, which contains 70%, 10% and 20% of the original data, respectively. We applied cross validation to optimize the hyperparameters for each model. The training and testing process is applied for Wednesday and Friday's traffic collection, separately. To decided the value of n for the proposed LSTM scheme, we analyze the number of packets associated with each flow in the training datasets. We choose 10 as the value of n, which is the round up value of the median. Hence, the LSTM model examines a sequence of 10 packets from each flow and then classifies the flow as either legitimate or malicious.

The proposed LSTM detection model is compared with the state-of-the-art traditional machine learning models, including Decision Tree (DT), Artificial Neural Networks (ANN) and Support Vector Machine (SVM). Results are represented in Table 5. It shows that both traditional machine learning methods and our proposed LSTM scheme are capable to achieve nearly perfect performance on the testing case. It is worth noting that without using flow-level statistical features, the proposed LSTM scheme can achieve not only competitive, but slightly better performance in terms of all evaluation metrics.

Experiment 1 – A Standard Evaluation Experiment

The goal of experiment 1 is to carry out a comparative analysis of the proposed scheme, which only use the raw packet header information, and the traditional machine learning methods, which rely on manually selected sophisticated features. In this experiment, we split the benchmark datasets into three parts: training, validation and testing, which contains 70%, 10% and 20% of the original data, respectively. We applied cross validation to optimize the

hyperparameters for each model. The training and testing process is applied for Wednesday and Friday's traffic collection, separately. To decided the value of n for the proposed LSTM scheme, we analyze the number of packets associated with each flow in the training datasets. We choose 10 as the value of n, which is the round up value of the median. Hence, the LSTM model examines a sequence of 10 packets from each flow and then classifies the flow as either legitimate or malicious.

The proposed LSTM detection model is compared with the state-of-the-art traditional machine learning models, including Decision Tree (DT), Artificial Neural Networks (ANN) and Support Vector Machine (SVM). Results are represented in Table 5. It shows that both traditional machine learning methods and our proposed LSTM scheme are capable to achieve nearly perfect performance on the testing case. It is worth noting that without using flow-level statistical features, the proposed LSTM scheme can achieve not only competitive, but slightly better performance in terms of all evaluation metrics.

Table 5: Experiment 1 Results

Models	Wednesday			Friday		
	P	R	F1	P	R	F1
DT	**0.9986**	0.9985	0.9985	**0.9998**	0.9991	0.9995
ANN	0.9971	0.9992	0.9982	0.9996	0.9998	0.9997
SVM	0.9505	0.5135	0.5925	0.8818	0.4543	0.5997
LSTM	0.9995	**0.9997**	**0.9991**	**0.9998**	**1**	**0.9999**

On one hand, the result confirms that the capability of machine learning techniques to recognize patterns is evident. Taking a closer look at the results, it finds that except for SVM, all other methods only mis-classified less than 20 flows for Friday's dataset. It is impressive, especially considering that tens of thousands of flows are contained in the test data. On the other hand, the results does not benefit our community any further to solve the practical problem by splitting the same dataset into training and testing. In this experiment, the training and testing set are generated by a same (set of) tools. We can assume all testing

samples are sufficiently represented in the training set, so the learning models can capture the inherited patterns in the testing set successfully. However, in the real-world detection scenario, the unseen attack is usually under-represented, if not un-represented at all, in the available training data. The challenge in DDoS detection is to capture the dynamic traffic behavior so that the similar attacking strategies can be identified in the future.

Experiment 2 – Testing on Unknown Dataset

In this experiment, we evaluate the proposed scheme's capability to capture the dynamic attack traffic behaviors. Specifically, we train the models on Wednesday and Friday's datasets separately, and test them on Friday and Wednesday's datasets, respectively. For convenience, we denote training on Wednesday's datasets and testing on Friday's datasets as "WeTrFrTest", and the other task as "WeTrFrTest". As discussed in section IV, attacks in these two days' datasets are generated by different tools but with similar attacking strategies. From the perspective of a single flow, the embedded attacking behavior should be similar. A successful detection scheme is expected to capture the commonalities.

Figures 12 and 13 represent the results for "WeTrFrTest" and "FrTrWeTest" respectively. The high variance presented by the performance of SVM shows that it is not a stable and trustworthy detection scheme in this evaluated scenario. Thus, we leave SVM out in the following discussions.

Training with Wednesday's dataset, the traditional machine learning methods intend to label more Friday's traffic as legitimate rather than malicious, shown as high precision scores and low recall scores. Differently, our proposed LSTM model intends to label more traffic as malicious rather than legitimate, causing a high recall score but a lower precision score. Balancing between the precision and recall, LSTM model slightly outperforms DT and ANN, shown by the highest F1 Score.

Training with Friday's dataset, the traditional machine learning methods fail to detect the attack traffic in Wednesday's dataset. The extremely low recall score presented by DT reveals that it classifies most traffic flows into the category of legitimate. ANN performs better than DT, but the similar conclusion can be drawn from the result. The proposed LSTM performs equally good in all measurements, and it shows the significantly improvements in both recall and F1 score.

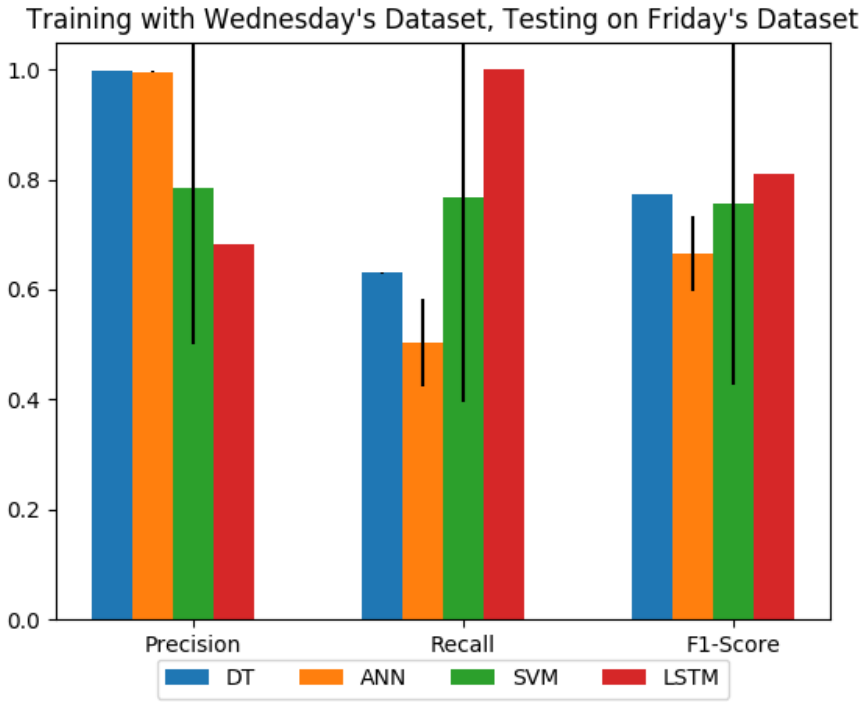

Figure 12: Training with Wednesday's Dataset, and Testing on Friday's Dataset

Experiment 3 – The Impact of Different Values of n

To classify network traffic, our proposed detection scheme needs to examine n packets of a network flow. In this experiment, we study the impact of different values of n on the performance. We train the proposed models with different values of $n \in \{3, 5, 10, 20, 30, 40, 50\}$.

Examining only the first 3 and 5 packets, the model lose the capability to distinguish attack and legitimate traffic. It simply label all traffic as attack traffic. Analyzing the training data, we observe that flows, which have large number of packets, are usually legitimate traffic flows. The model may have learned this specific feature, and simply make the decision accordingly.

Table 6 and 7 present the results for "WeTrFrTest" and "FrTrWeTest", respectively. From the table, it is observed that allowing the model to examine more packets for each flow, with increasing n values, does not necessarily improve the performance. In Table 7, examining 50 packets significantly degrade the scheme's performance. Additionally, examin-

ing larger number of packets increases the standard deviation of all measured metrics, which indicates the consistency of the performance is also degraded by a larger value of n.

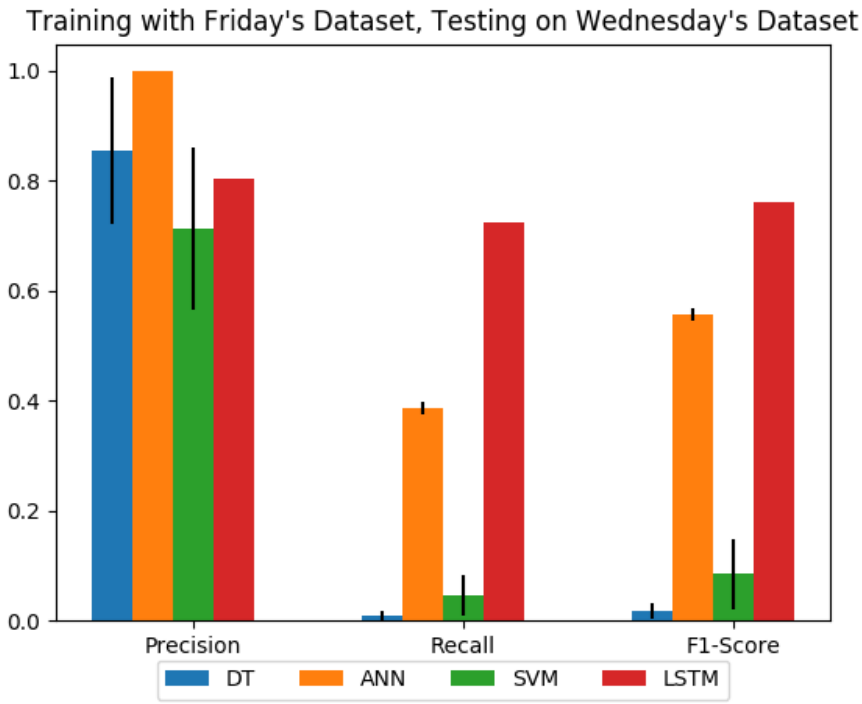

Figure 13: Training with Friday's Dataset, and Testing on Wednesday's Dataset

It seems to benefit the detection scheme by examining higher number of packets from each flow, since more data is provided for the model to learn traffic behaviors. However, if network flows are mostly short, such as in our benchmark, then the short flows will be padded with zeros. These padding values may confuse the system and cause the performance degradation.

4.2 A Hybrid DDoS Detection Model

Machine learning techniques, including deep learning approaches, hold great promises for accurately detecting DDoS attack traffic. As illustrated in the previous section, using only

Table 6: Evaluating the Impact of Different Values of n – WeTrFrTest

n	Ave* P	Ave R	Ave F1	STD* P	STD R	STD F1
10	0.6823	0.9999	0.8111	1.0584	1.74e-05	7.86e-6
20	0.7125	0.9999	0.8321	0.0005	1.46e-5	0.0003
30	0.7205	0.9981	0.8368	0.0143	0.0041	0.0081
40	0.6271	0.8752	0.7283	0.2241	0.3294	0.2729
50	0.7321	0.9949	0.8432	0.0259	0.0064	0.0146

the raw packet header information, the proposed LSTM model even presents the potential to capture the involving attack behaviors. However, most proposed DDoS detection solutions lost the connection to the problems that are required to be addressed in the real world.

Table 7: Evaluating the Impact of Different Values of n – FrTrWeTest

n	Ave P	Ave R	Ave F1	STD P	STD R	STD F1
10	0.8026	0.7235	0.7610	9.04e-6	2.87e-5	1.42e-5
20	0.8889	0.6010	0.7171	4.49e-6	1.01e-5	6.24e-6
30	0.8870	0.5957	0.7132	0.0007	0.0138	0.0103
40	0.8889	0.6010	0.7171	1.20e-5	7.87e-6	7.17e-6
50	0.8077	0.2821	0.2920	0.0127	0.3636	0.3756

Firstly, the detection capability of a proposed machine learning tool is usually evaluated by a well-formatted dataset. Although the evaluation metrics are well studied and truly represent the detection capability, it does not translate to real-world DDoS attack mitigation affects. Secondly, due to the special sensitive information that network traffic could carry on, most datasets are either simulated or emulated in a research lab. The simulation environment plays a critical role which are broadly neglected in the detection evaluation.

The collected datasets could contain biases, and be built into the trained model. As such, a poor performance will be observed, when the approach is applied in the real-world, or even under a different simulation environment. Lastly, adapting a machine learning approach to mitigate DDoS attack is not an easy job. Any assumptions or expectations for proposed machine learning approaches to be "ready to use" will lead to an ineffective mitigation system. In Chapter 6, we use experiments further discuss these issues.

To improve the performance of the proposed LSTM detection model in real-world detection, and reduce the effect, In this section, we propose to have an additional detector, which is aware of the current network status. When the LSTM model first introduce a network status detection approach, which plays a complementary role to the LSTM model. Furthermore, we extend the stand alone detection model to a distributed and hybrid scheme, so that it facilitates the coordinate detection mode and could also better serve in the proposed distributed defense system.

4.2.1 A Network Status Detection

Numerous works have been published to discuss how to statically characterize network status, indicating whether an anomaly or an intrusion event is taking place. In Chapter 2, we have reviewed and discussed the limitations of traditional approaches on identifying abnormal status of the network. One of the major challenges comes from the difficulty of defining a reliable normal profile of the network. Fortunately, leveraging machine learning techniques, the characteristics can be learned automatically. Additionally, the objective of this extra detector is to identify network status, rather than specifically distinguish between the benign and DDoS attack traffic. With this relaxed objective, challenges of feature engineering that are faced by machine learning techniques for developing an effective DDoS detection are actually relieved. Furthermore, with SDN, which offers the logically centralized view of the entire network, the severe impact caused by the observation location is also alleviated.

DDoS attacks are typically launched from a very large number of distributed, remotely controlled devices, organized into botnets and aimed at attacking the same target [5]. It is often the case that a significant number of these devices are unwitting members of botnets,

operating with spoofed network addresses. From the perspective of a network administrator, when such attack events are happening, you likely to experience dramatically increased number of source IP addresses that are targeting a small range of, if not the same, destination IP addresses. These compromised devices usually have been injected with malware, with the objective of rendering the target unresponsive to legitimate users. Different from people's response, the injected malware usually doesn't response properly to the slow responsive caused by connection congestion or overwhelmed computational resource exhaustion. By analyzing the traffic, increased number of asymmetric traffic flows could be observed.

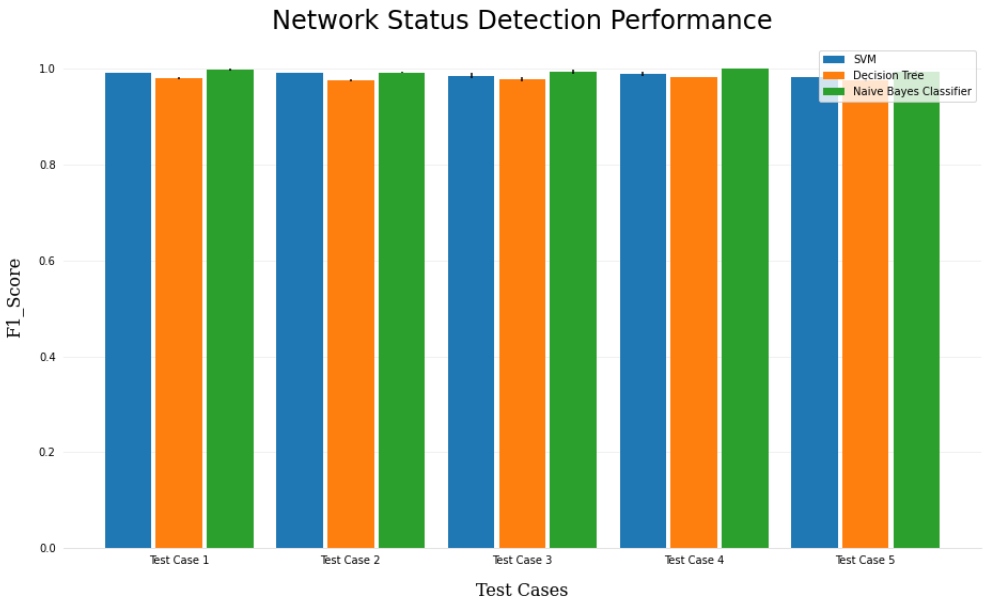

Figure 14: Network Status Detection Performance

According to the above observations and published works, we selected a set of features for the network status detector. Firstly, we use entropy, which measures the randomness, to represent the variance of the distribution of source IP addresses and source ports. Entropy is a popular information metric in capturing the changes of network status. In [82], the authors demonstrate that the entropy values of both source IP addresses and source port numbers are significantly increased under a DDoS attack, comparing with a normal status. Secondly, we measure the number of packets and the packet size carried by each flow within

the pre-defined time interval. With genuine traffic, number of packets or number of bytes for different flows could vary dramatically. However, traffic sent by a program may have junk packets with similar sizes. Lastly, we count the number of asymmetric traffic flows. All features selected are summarized as below:

(1) entropy of the source IP addresses

(2) entropy of the source Port

(3) standard deviation of the number of packets carried by each flow.

(4) standard deviation of the number of bytes carried by each flow.

(5) the proportion of interactive flow entries, which is calculated by the number of interactive flow entries divided by the total number of flow entries.

These five selected features are applied with three classifiers: Naive Bayes classifier, Support Vector Machine, and Decision Tree. The dataset used to evaluate the detector's performance is also from CICIDS 2017. The Friday's data, which contains DDoS attack traffic, is chosen to assess these three classifiers' performance. The dataset is split into three sets: training, evaluation and testing, the same as described in the previous section. The traffic trace of each dataset is divided into a series of 10 seconds intervals. If attack traffic is observed inside an interval, then the interval is labeled as an attack sample, otherwise, it is considered as a legitimate sample. The standard evaluation process is conducted to choose the best hyper-parameters for each classifier, and then compare their performance against the test dataset. The performance comparison results, F1 score, is presented in Figure 14. Appendix B shows the performance comparison results for Precision and Recall scores. Naive Bayes classifier outperforms the other two, and is chosen to form the hybrid detector with the LSTM model.

4.2.2 The Hybrid Detection Scheme

Due to the reasons that are discussed at the beginning of this section, features learned by the LSTM model could be biased, which will degrade the detection accuracy when it is deployed in a real-world defense system, or even in a different emulation environment. In the Chapter 6, we use experiments to further elaborate this issue. To reduce this undesired

effect, we propose having a detector, which is aware of whether the network is under an DDoS attack event or not, as a guidance that complements to the LSTM model, especially when it is not confident about the predicted results.

The workflow for the hybrid detection scheme is illustrated in Figure 15. The LSTM detection model is a flow by flow detector. The input data is from the raw packets, which are mirrored from switches. The mechanism of which packets and how the packets are mirrored from switches are described in Chapter 5. Based on given packets, the detector extracted header information, and store them for the corresponding flows. After receiving a pre-defined number of packets from a specific flow, the detector is triggered. It reports the probability of the given flow being malicious. As a binary classification model, the threshold is usually set as 0.5. If the probability score is higher than this threshold, the flow is classified as malicious, otherwise it is classified as legitimate. In our proposed hybrid detection scheme, we define three value ranges for the reported probability, which represent malicious, suspicious and legitimate. If the inspected flow is identified as an attack flow, then an alarm is raised, and the detection model immediately installs the dropping rule for this malicious flow. If the LSTM model does not have sufficient confidence to classify the flow as attack, but labels it as suspicious. Then collaboration mode will be invoked. The LSTM detector consults the current network status. If there is no DDoS attack event undergoing, the suspicious flow will be labeled as legitimate, otherwise, it will be labeled as malicious, and the corresponding dropping rule will be installed into switches.

The input for network status detector is the rule table statistics from switches. The sentinel, which hosts the detection scheme (details are described in Chapter 5), periodically requests flow stats from switches. After receiving the flow stats, the detector first extracts desired features, and then identifies the current network status. If an DDoS attack event is identified, the detector immediately updates the network status to be "under an attack event". If no DDoS attack event is detected, the detector does not update the network status instantly. It will check whether the normal network status has been observed globally. Two reasons are behind this design: first, even under a DDoS attack event, not all switches are experiencing attack traffic. To better protect the network wide hosts and servers, the network status should be updated in a more conservative fashion; second, the network status should

be protected from constantly swaying. If an attack event is observed, the network should keep cautious for a while until a reliable and stable normal state is confirmed. An unstable complementary consultant could interfere the LSTM detection and raise the number of false negative, which could cause the most unintended consequences.

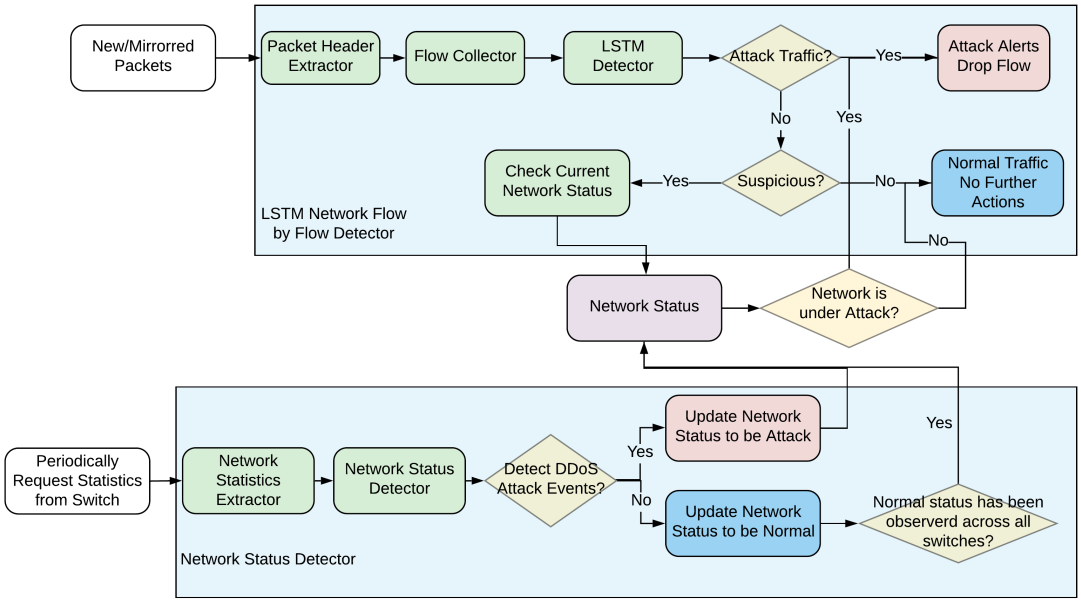

Figure 15: The Hybrid Detection Scheme

4.3 Summary

In this chapter, we introduce a hybrid DDoS detection approach, which combines a LSTM-based networ traffic flow-by-flow DDoS detector and a network status detector.

The LSTM detection model avoids manual feature engineering, thereby addressing a significant shortcoming of classic machine learning methods. The ability of the scheme to automatically learn complex representations to successfully classify legitimate and attack traffic flows is empirically confirmed. Using unknown traffic, the results also show the potential to capture the involving DDoS attack traffic behaviours. Most importantly, the proposed LSTM model offers a practical solution for real-time inline inspection.

The network status detector uses a classic machine learning model. In stead of identifying specific attack network flow, it leverages network statistics to be aware of the network status. It plays a role in improving the LSTM detection results, especially when the LSTM detector is not confident about the decision.

Finally, we introduce a novel mechanism that unites both the deep learning and the network status detector to construct a hybrid DDoS attack detection scheme. Two models are complement each other. Their empirical performance is evaluated in Chapter 6.

5.0 A SDN-Centric Infrastructure Design

SDN physically separates the control plane and the data plane of the networking protocol stack [38]. The disaggregation of these two planes adds flexibility to the network architecture, enabling the controller to develop a global view of the network and share information about the entire state of the network with the applications. Thus, SDN offers a new opportunity to effectively defend against DDoS attacks ascribing to its flexibility, programmability and maintainability.

Harnessing the availability of network state information, a number of SDN-based approaches have been proposed to detect and defend against DDoS attacks [13, 20, 78, 86]. Although improvement has been reported, a number of fundamental challenges remain to be addressed. First, SDN centralized controller does not only provide a global view of the network, but also facilitates a variety of dynamic responses. By leveraging this property, a defense system is, naturally, designed as an application attached to a centralized controller. Via the centralized controller, these defense systems are capable of collecting real-time network traffic statistics from switches, and integrating mitigation rules into switches. However, naively utilizing the centralized controller could cause the defense system itself become a bottleneck. The tremendous and heterogenous data required from switches could cause the communication channel between switches and controllers under congestion, and further exhaust resources of both the controller and switches. To this end, an effective defense system must be lightweight to avoid excessive communication costs and processing power usage, especially, at the peak of an attack. Second, a SDN switch is only able to report basic flow counter information [1]. Detection approaches that rely on sophisticated flow statistics need to deploy extra appliances or implement flow information pre-processing [92], which incurs extra deployment costs and makes inefficient use of the resources. The potential of high speed transmission provided by SDN is, thereby, wasted. Thus, how to make full use of the SDN characteristics and smartly mitigate the DDoS attack using basic flow information is still a challenge problem.

In this chapter, we introduce a lightweight distributed and collaborative DDoS defense

system, which circumvents the above challenges. By leveraging a distributed architecture, both the computational and communication burdens are relieved from the controller. This defense system monitors ongoing network traffic, distinguishes malicious traffic from legitimate traffic, and blocks attack traffic in an efficient and effective manner.

5.1 Backgrounds

Before introducing the defense system in detail, this section describes the basic working mechanism of the SDN. Figure 16 depicts the layered architecture of a distributed SDN. The infrastructure layer contains network elements and devices, such as routers, switches, virtual switches, etc., which provide network packets forwarding in accordance with the instructions from the control plane. The middle layer of the architecture is the control layer, which is regarded as the network operating system. It supervises the network forwarding behavior through south bound open interface. To improve the scalability and reliability of SDN, the control layer is usually a distributed system [8]. The communications among controllers are enabled by the East-West bound APIs. The top layer is the application layer, which comprises end-user applications. These applications are designed for the network control logic and management. Routing, traffic engineering, intrusion detection, and load balancing are typical examples. The boundary between the application layer and the control layer is traversed by the northbound open interface.

Having these separate layers, how does a single network packet get across the network? As the first standard communication protocol defined between the control and forwarding layers, OpenFlow has been adopted the most in SDNs. In Figure 17, we use OpenFlow as an example protocol to demonstrate how the controller guides the switches to forward packets. When a new packet comes (step 1), which does not match any rules in the switch's flow tables, the switch sends a *_PacketIn* message to the controller (step 2). The controller could send the switch a *_PacketOut* message to indicate the forwarding action directly or a *_flow_mod* message (step 3) to add a new entry into flow tables (step 4). In both manners, the correct forwarding port is included inside the action list, and the switch forwards the

packet via the required port (step 5). If a new entry, also known as a forwarding rule, is installed in the switch, then future packets from the same network flow do not need to be sent to the controller. The switch is capable of applying the matched action automatically. With this programmatic framework, SDN enables security applications to tailor the network behaviors for mitigating DDoS attack impact and further protect connected hosts.

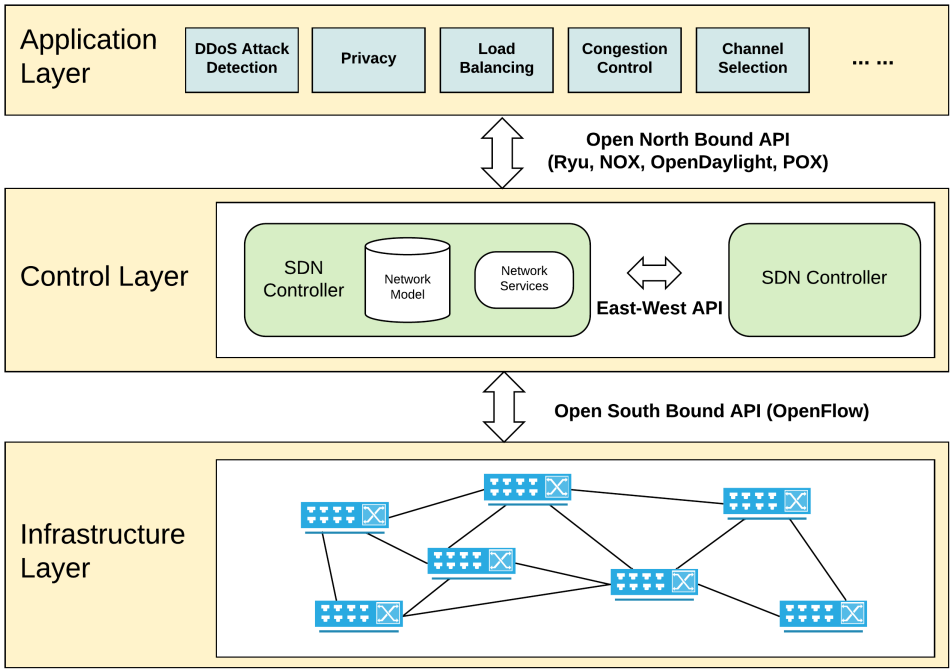

Figure 16: A Three Layer Distributed SDN Architecture

5.2 Traffic Flow Setup in SDN with DDoS Detection

Taking advantage of SDN's programmability, the proposed DDoS detection model could be implemented as an application, which is attached with the SDN controller. The workflow of how the controller guides the switches to forward packets is then updated and shown in Figure 18 and 19. Figure 18 illustrates when a new packet comes, which does not match any rules in the switch's flow tables, how the controller guides the switch to forward the packet. First, the switch forwards the packet to the controller via a *_PacketIn* message (step

1), same as without the DDoS detection model. After receiving the *_PacketIn* message, the controller sends out the forwarding rules to the switch via the *_flow_mod* message (step 3). This forwarding rule does not only include the forwarding port to the destination, but also include the forwarding port to the controller. The switch installs the rule as a new entry in its flow tables (step 4). The packet is forwarded via the required port to its target destination (step 5.2). Additionally, the packet header information is forwarded to the DDoS detection model by the controller (step 5.1).

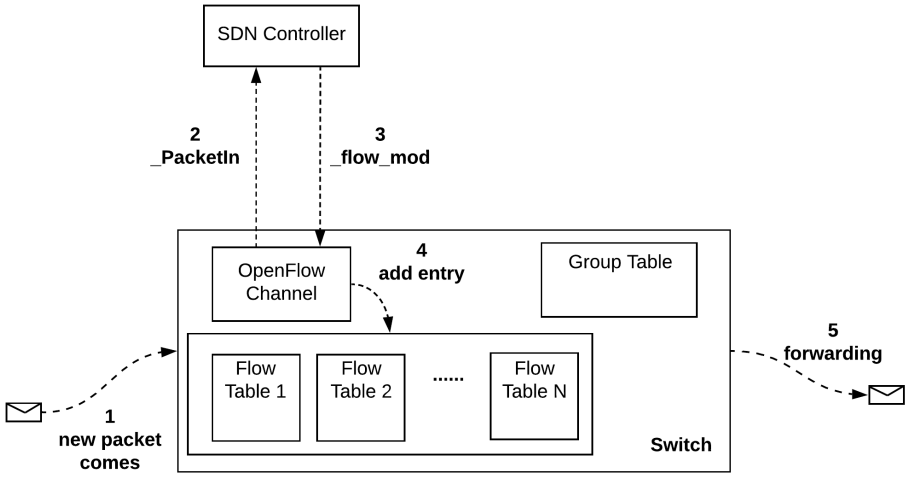

Figure 17: Network Flow Set-up in SDN

Installed with the forwarding rule, how does switches deal with packets from the same network flow is depicted in Figure 19. When a known packet comes to the switch, the switch forwards the packet to its destination through the desired port, which is indicated by the entry in the flow table (step 2.1). At the same time, the packet is also forwarded to the controller (step 2.2), since the port to the controller is also part of the flow entry's action list. The controller forwards the packet header information to the DDoS detection model (step 3). The LSTM detector is triggered after receiving N packets from the same network flow, meanwhile it sends out a message to the controllers for stopping forwarding future packets from the same network flow (step 4). To eliminate the controller's port from the action list, another *_flow_mod* message is sent to the switch (step 5), so that packets will

not be mirrored to the controller any more. Additionally, the DDoS detection application periodically sends out the flow statistics request via the controller (step 6), which requires the switch to send flow statistics based on its flow tables (step 9). With both flow statistics information and the packet header information, the DDoS detection model classifies the assessed network flow with either the label of legitimate or attack. If an attack network flow is detected, an attack alert will be sent to the controller immediately (step 8). As soon as the controller receives the attack alert, a new *_flow_mod* message is broadcast to all associated switches (step 9) to drop malicious network flows via adding a flow entry, which guides the switch to drop packets from the malicious network flow (step 10). If the examined flow is legitimate, the detection model does not communicate with controllers, and then no actions are sent to switches. Readers may notice that with the introduced workflow, the controller has to handle a number of extra messages for each single network flow. Considering the huge number of network flows a switch deals with, the tremendous data sent to the controller could cause the communication channel between switches and controllers under congestion, and further exhaust resources of the controller. In the next section, we describe our design of a distributed and collaborative SDN centric defense architecture, which relieves both the computational and communication burdens from the controller.

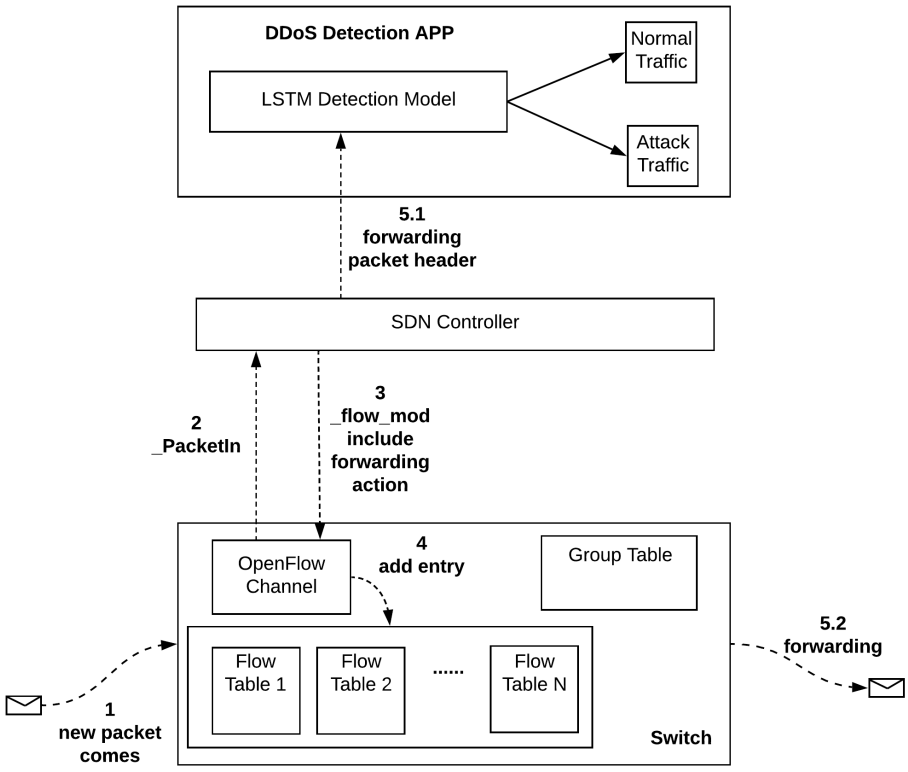

Figure 18: Network Flow Set-up in SDN – with DDoS Detection Model

5.3 System Design Architecture

SDN offers a great opportunity to effectively defend against DDoS attacks ascribing to its logically centralized control, programmatic framework, and vendor agnostic implementation interfaces. However, simply assuming that these characteristics can provide any desired flow statistics and easily gather tremendous data from switches may cause the defense system itself become a bottleneck. In this section, we leverage SDN features further and propose a distributed infrastructure, which coordinately defenses against DDoS attacks and obviates the burden from both the switch-to-controller communication channels and controller's computational resources.

An instance of the proposed infrastructure is depicted in Figure 20. The network ele-

ments in the proposed infrastructure can be roughly classified into three categories, namely the *controller*, *sentinels*, and *switches*. This proposed system does not need to modify the current packet forwarding diagram. We use OpenFlow [49], which is one of the most widely adopted open standards for SDN architecture, as a representative of network control APIs of SDN. Switches in this proposed architecture could be any off-the-shelf OpenFlow compliant switches. This enhances the potential of deploying the proposed defense system in a practical setting. A controller refers to an SDN controller, which acts as the network operating system. It interacts with OpenFlow switches to instruct switches with the forwarding rules, and request network statistics from switches' flow tables, as discussed in previous sections. Sentinels are simplified controllers, which are dedicated to make all decisions driving network-wide security, including DDoS detection, dynamic installation of throttling rules and

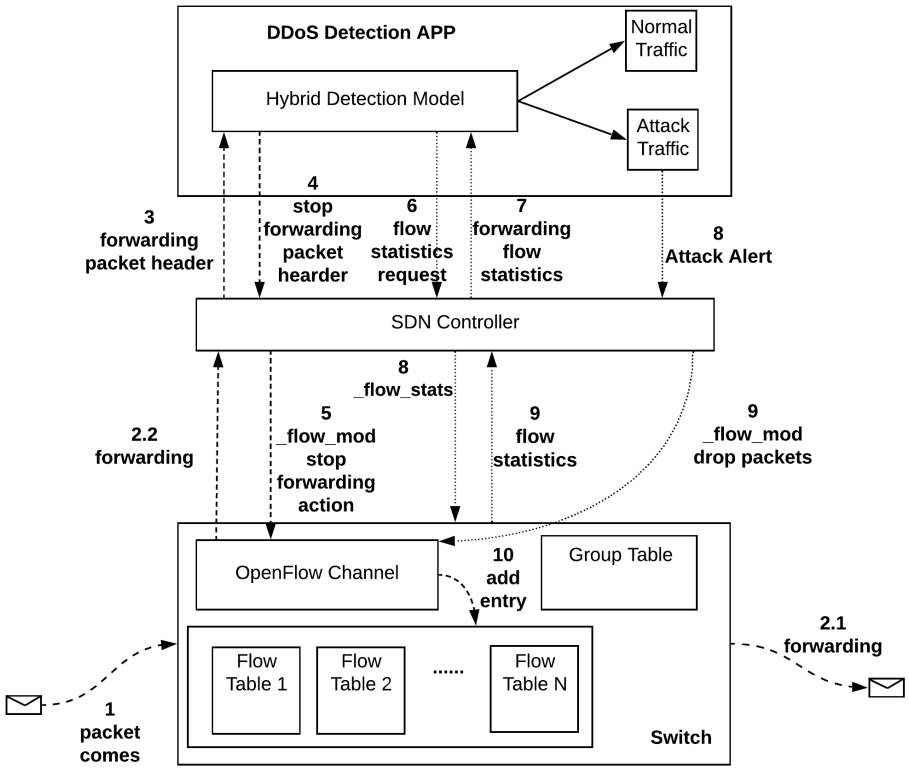

Figure 19: Network Packets Forwarding – with DDoS Detection Model

communication of throttling signals across different sub-networks. The lightweight DDoS detection model is equipped as an application for each sentinel. As such, sentinels are capable to make independent decisions regarding its monitored network traffic. As shown in the graph, each sentinel guards a subset of the switches, so that the high demand of the computational resources and communication tasks can be distributed into several sentinels. The number of sentinels can be decided by multiple factors, such as the size of the network, the cost budget, the expected traffic volume, and etc.

Communication channels among sentinels are built with the "publish and subscribe" messaging paradigm to facilitate information exchange among sentinels. Each sentinel is subscribed to its neighboring sentinels for receiving messages, at the meantime, itself is a publisher for sending out messages. With this communication channel, sentinels are easily to obtain a global view of the network without interacting with the controller. Additionally, we also build a communication channel between sentinels and the controller, so that sentinels are capable of installing a network-wide throttling policy via the controller.

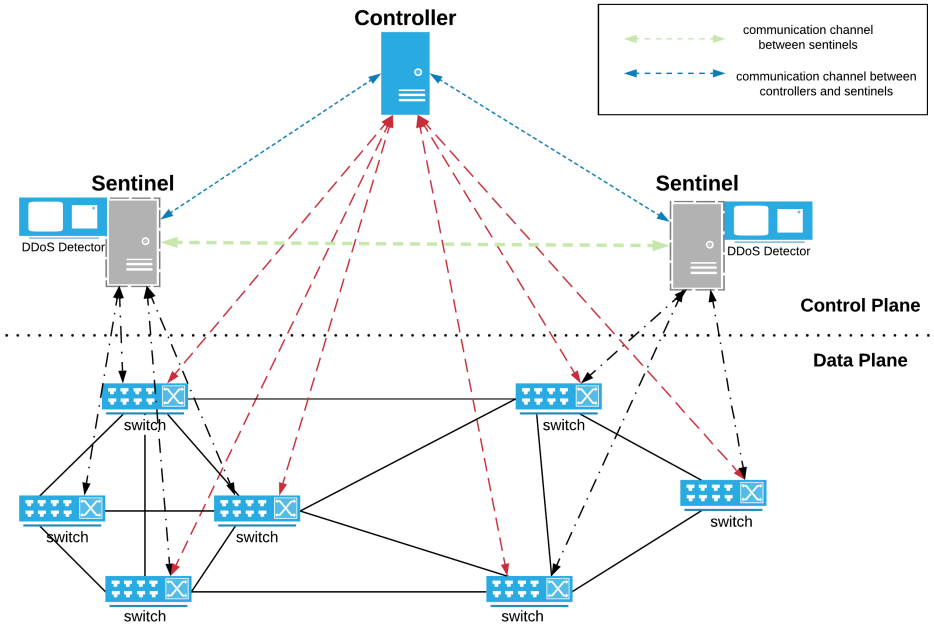

Figure 20: The Architecture of the Proposed Defense System

The workflow diagram for a particular network flow is shown in Figure 21. When a new packet arrives at the switch and does not match any rules in the switch's flow tables, the switch will consult the controller about the action for this packet via OpenFlow Packet_In message. The controller, relies on the network topology, sends back a Flow_Mod message to inject a forwarding rule into the switch's flow tables. This forwarding rule does not only indicate the desired forwarding port to the destination for this coming packet, but also instruct the switch to mirror future packets from the same network flow to the sentinel. Upon receiving the forwarding policy from the controller, the switch forwards the packet to the desired destination and also mirrors the packet to the sentinel. After N packets from the same network flow have been observed, or a pre-defined time window has reached, the DDoS detection application, which is attached with the sentinel, is triggered. Then, the sentinel informs the switch to stop mirroring future packets. Due to the space limitation, Figure 21 only depicts one situation that N packets have been observed. As soon as the N packets have been forwarded to the sentinel, sentinel will remove the forwarding rule from the switch, so that future packets from the same flow won't be forwarded to the sentinels any more. If the detection model identifies the examined network flow as attack traffic, the sentinel will immediately inform the controller, and the controller installs a drop action for this identified malicious network flow globally. If the detection model identifies the network flow as legitimate traffic, no further actions will be applied. The installed rules are kept valid for guiding future packets with correct forwarding path to its destination. Given this proposed defense system, the burdens of continuous traffic monitoring and extra communication between switches and controllers are carried out by sentinels.

5.4 Extended Collaborative Detection Models

Since DDoS attack is distributed across the whole network by nature, a coordinate monitoring and detection system is required to efficiently defend against DDoS attacks. To this end, we extend the detection model described in Chapter 4 to facilitate collaborations among detection nodes. This collaboration, which overlays different areas of the network, is capable

of providing more comprehensive view of the network traffic status, hence lead to more accurate detection results [43]. In this section, we present two collaborative detection modes, one is only extending LSTM model alone without the involvement from network status detector, and the other is extending the network status detector to be aware of the global network.

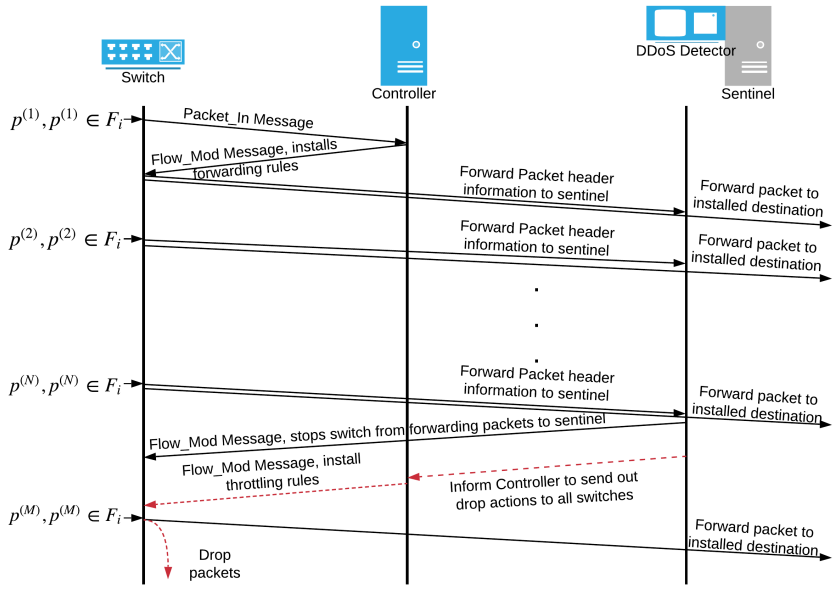

Figure 21: The Workflow Diagram for a Sequence of Packets

In our proposed defense system architecture, the detection model is implemented as an application attached to a specialized controller device, which is called sentinels (as described in the previous section). In a large network, each sentinel is associated with multiple switches. In each sentinel, information observed by these multiple guarded switches can easily be shared. The collaboration should not only be facilitated inside each sentinel, but also should be conceived among sentinels. Hence, the network traffic monitoring can be realized in a truly globally fashion. To achieve this goal, we implement a communication channel among sentinels, so that the collaboration does not only happen inside each sentinel, but also among different sentinels across network areas.

When the detection model is triggered, based on given packets' header information, it reports the probability of the given network flow being malicious. As a binary classification

model, the threshold is usually set as 0.5. If the probability score is higher than this threshold, the flow is classified as malicious, otherwise it is classified as legitimate. In our proposed detection model, we define three value ranges for the reported probability, which represent malicious, suspicious and legitimate. The two collaborative modes react differently when a suspicious network flow is detected.

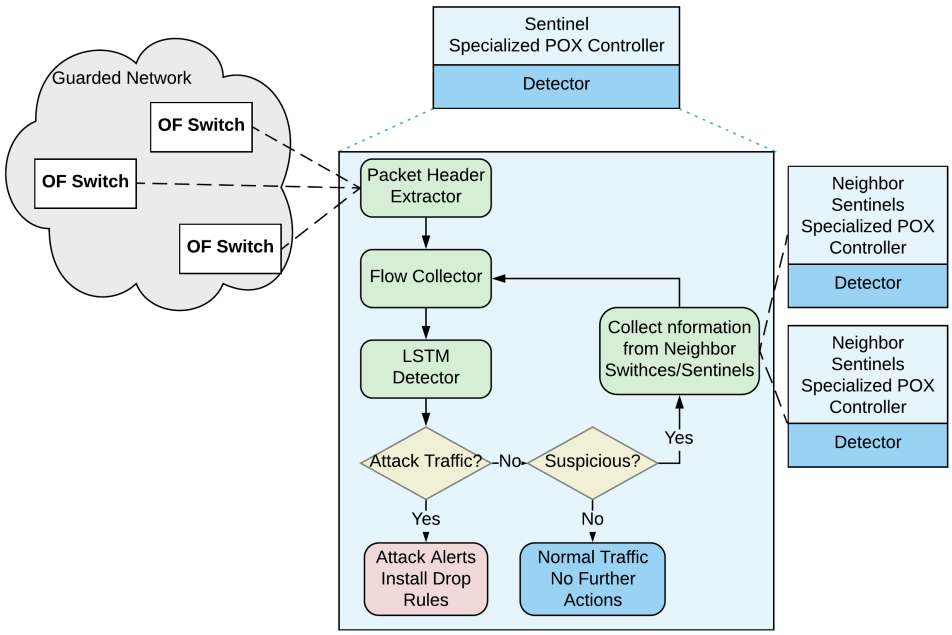

Figure 22: Collaborative Detection Workflow – LSTM Model Alone

In the LSTM alone mode, the collaboration mode will be invoked. The sentinel gathers information from its neighbored sentinels, and the LSTM model concatenates these additional data and applies the classification algorithm again. With the obtained centralized information, if LSTM model is still not confident with the classification result, in other words, the probability of examined network flow being malicious still falls in the range of suspicious, 0.5 will be used as the threshold for making the final decision. Figure 22 illustrates the collaborative detection loop with the LSTM model alone.

With the hybrid detection model, if the LSTM detector is not confident about the classification results, it consults the network status. The workflow has been thoroughly described

in Chapter 4. The network status detector periodically collects network statistics from attached switches. Based on the extracted features, the detector keeps the network status profile updated, which indicates whether the network is under a DDoS attack or not. Same as the LSTM model, three value ranges are designed for the network status detector. If the network status detector cannot make the assured decisions, it seeks help from neighbored sentinels. Depends on other neighbor networks status, the detector adjusts its uncertain decisions. The abstracted detection loop is illustrated in Figure 23.

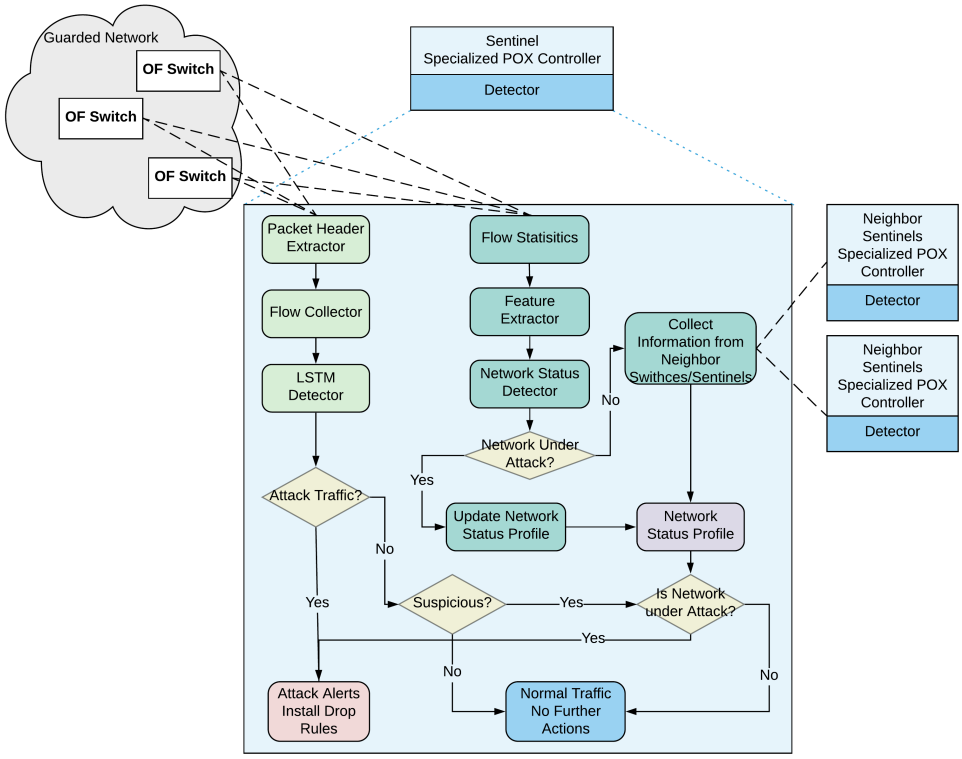

Figure 23: Collaborative Detection Workflow – Hybrid Model

5.5 Summary

In this chapter, we demonstrate the system architecture design for our proposed DDoS defense system in SDN networks environment, which leverages the OpenFlow protocol capabilities to maintain a global view of the network, without causing a bottleneck. Specifically, we illustrate in detail the workflow of a single controller communicates with both switches and detection models to accomplish the detection and mitigation tasks. To alleviate the communication and computational burdens from the controllers, we introduce a specialized controller, sentinel, and present how sentinels work in a large-scale networks. To maintain the logically global view and fully exploit SDN's potential for DDoS defense, we extend the detection models to allow them to be operated under a distributed and collaborative manner.

6.0 Evaluation of the Proposed DDoS Defense System

DDoS defense system is a complex system, the effectiveness of its mitigation capability is subject to multiple factors, such as attack detection, percentage of legitimate traffic delivered during an attack, overhead introduced to controllers, and etc. It is difficult for an analytical framework to precisely capture all these measuring aspects of a DDoS defense system in a real environment. Therefore, a functional prototype is necessary to analyze its validity as well as measure its actual performance. In this chapter, we provide a prototype implementation of the proposed defense system in a SDN environment, and evaluate its effectiveness of mitigating the impact of DDoS attacks on target hosts.

6.1 Experimental Setup

The proposed DDoS defense system is implemented using Mininet [50], an emulator for deploying large networks on limited resources. Mininet provides convenience and realism at very low cost, and have been widely used to evaluate the performance and demonstrate the functionalities in the research field of SDN. With Mininet, we construct the whole SDN architecture, including all three layers: the application layer, the control layer and the data layer. We choose OpenFlow protocol [49] as a representative of network control APIs for SDN, which governs the communication between controller/sentinels and the network forwarding devices (switches). OpenFlow is one of the most widely adopted open standards for SDN architecture, and has already become the de facto standard. The OpenFlow switches created by Mininet provide the same packet delivery semantic that would be provided by a hardware switch. Both user-space and kernel-space switches are available. Switches in this proposed architecture could be any off-the-shelf OpenFlow compliant switches. This enhances the potential of deploying the proposed defense system in a practical setting. Using Mininet, we implement the proposed detection model, or called security module, upon POX controller, which is a well-known and widely adopted SDN controller platform. The

detection model is equipped as an application to each sentinel. Communication channels are also built among sentinels, and between sentinels and the controller.

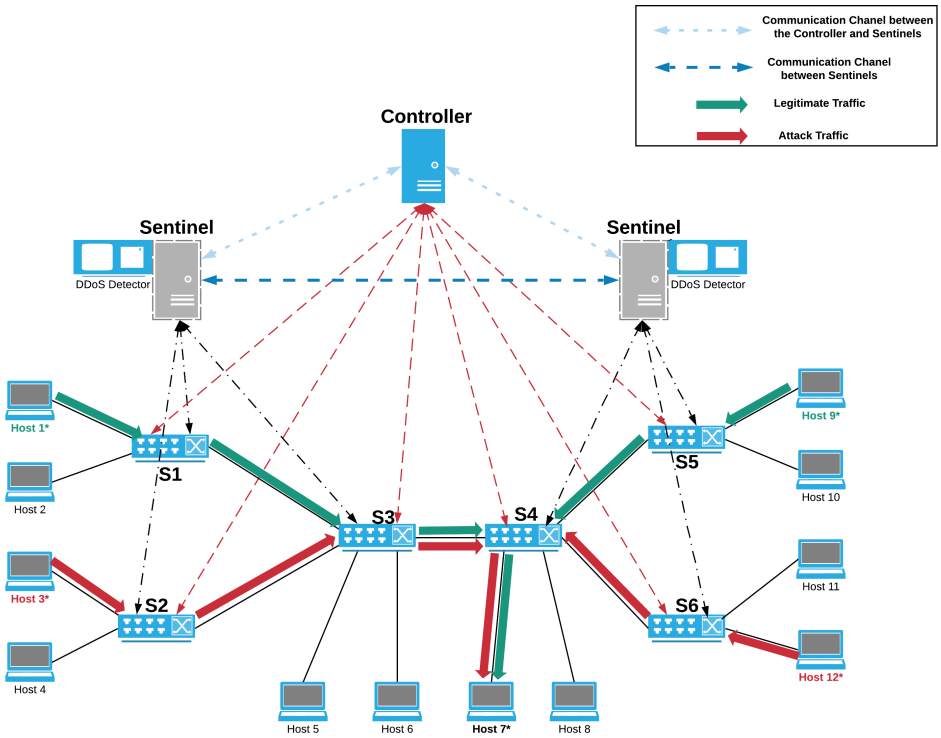

Figure 24: Topology of the Testing Scenario

Mininet does not only provide a handful default topologies, but also allows developers to customize topologies as needed. Figure 24 depicts our testing scenario's topology, which consists of a SDN controller, two sentinels, six OpenFlow switches and 12 hosts. The detection module is implemented as an application attached with each sentinel, so that one sentinel has the capability of making independent security decisions. The six switches are connected to a single controller, and guarded by two sentinels. Host 7 in the topology is the destination for all network traffic, including both legitimate and attack traffic. Hosts 1, 3, 9 and 12 are the sources to send out network traffic. Green colored path indicates the legitimate traffic, and the red colored paths are the attack traffic traversing paths. Sentinels in different positions are observing different types of traffic, some of them, such as s1 and s2,

only observe one type of the traffic, while others, such as s3 and s4, observe mixed types of traffic. Actually, s4 is the last switch before the traffic is delivered to the destination host, which experiences the aggregated traffic. We believe this reflects the realistic situations.

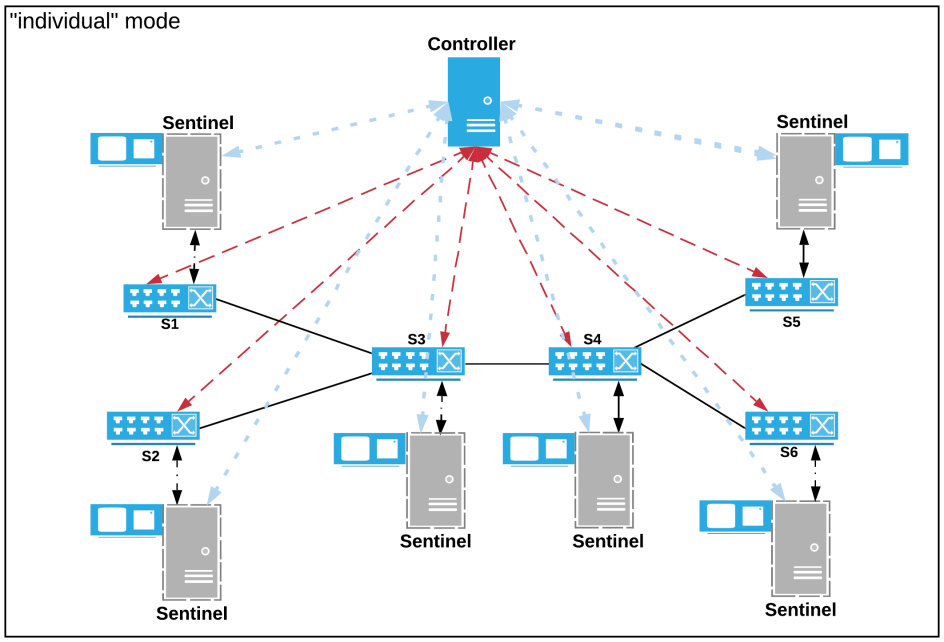

Figure 25: Individual Defense Scheme

To emulate the realistic network environment with this simple topology, we preserve all IP addresses from the original datasets, so that switches are dealing with significant diverse network traffic flows. To replay the traffic among hosts, we modify the MAC address. When switches forward the unseen packets to the controller, the controller determines the correct forwarding path based on packets' MAC address. At the same time, a forwarding rule, which is matching the IP address, is installed in switches.

6.2 Comparative Schemes

In order to compare our proposed collaborative defense systems with other non-collaborative and "ideal" systems, we implement three different defense schemes. In scheme 1, which is called "individual" mode, we attach the detection sentinel to each switch. Sentinels apply their detection role individually. Although sentinels follow the same work flow to inform the controller any identified attack flows, they do not collaborate with their peers to gather network traffic/status information. Additionally, the detection model implemented within sentinels does not maintain the suspicious range. If the probability of the inspected flow being malicious is higher than 0.5, then the flow is identified as an attack flow. Otherwise, it is labeled as a legitimate flow. Figure 25 depicts this scheme. This is a very expensive scheme, due to the resources allocated for each sentinel. Although there are no communication costs among sentinels, it requires that one sentinel fully guards only one switch.

Scheme 2 is a "semi-collaborative" mode, in which each sentinel guards a couple of switches. Within the same sentinel, information observed among the guarded switches is shared in the detection model. However, the collaboration among peer sentinels is not provided. Figure 26 illustrates this scheme.

In Scheme 3, we implement a "centralized" scheme, in which one sentinel is able to monitor the entire network, and the detection model is attached as an application to the centralized sentinel. This actually represents an "ideal" scheme, disregarding the intensive communication and computational burdens and the potential risk of introducing a bottleneck to the network. In this case, we assume the system has infinite buffering and computational resources. In this scheme, the global status is easily to be obtains, without the need of building any communication channels. Figure 27 depicts scheme 3 applied in our testing topology.

6.3 Dataset and Evaluation Metrics

The dataset, CICIDS 2017, we used in this evaluation framework is the same dataset we used for assessing the detection performance. This dataset was published by the Canadian Institute of Cybersecurity in 2017, and is a widely accepted benchmark datasets. It contains realistic background network traffic and a variety of attack traffic. The datasets cover five days of network traffic, one of which has DDoS attacks. We use this one days' traffic as the evaluation benchmark. The recorded DDoS attacks were generated by Low Orbit Ion Cannon (LOIC). LOIC floods targeted server using junk TCP, UDP and HTTP GET requests through numerous attacking devices.

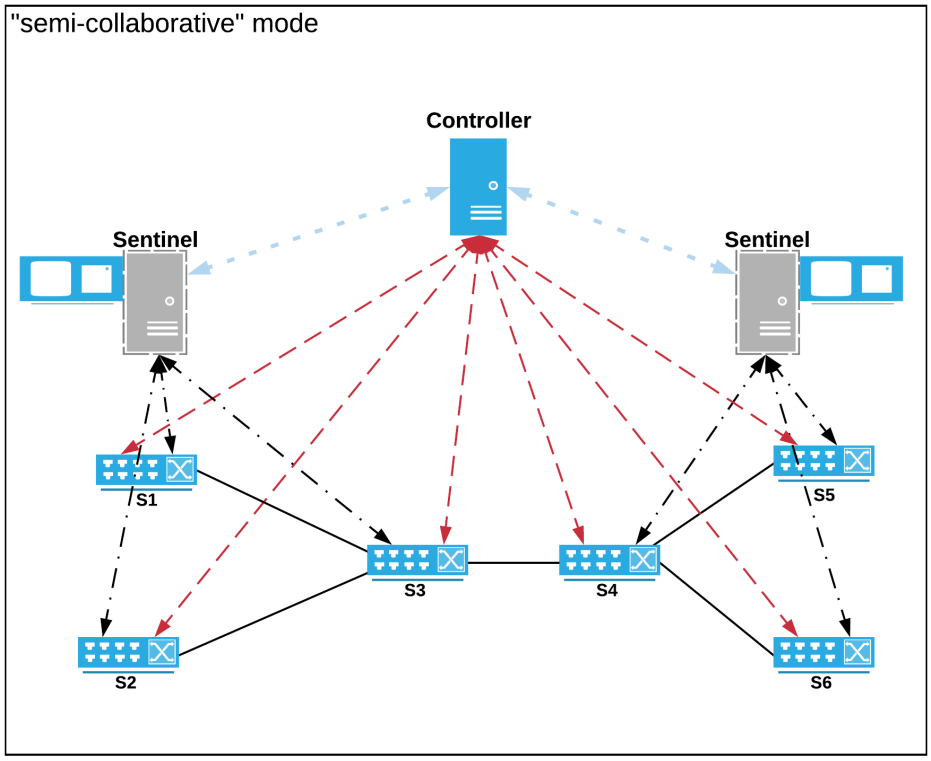

Figure 26: 'Semi-Collaborative' Defense Scheme

CICIDS 2017 datasets provide well formatted data files. In these files, each network flow is characterized by more than 80 statistical features, and associated with a label, indicating

whether it is or not a malicious flow. In addition to the well formatted network flow files, the raw trace files (in pcap format) are also provided. Since the raw data is not labeled, we need to reverse engineer the process to find the corresponding set of packets for each network flow. To ensure the correctness of the mapping, we carefully check the timestamp, the number of forward and backward packets, and the time duration for each flow. However, the granularity of the timestamp provided in the well formatted data is not fine enough to find all the mappings. We discard the network flows and packets that we could not find exact matches that satisfy our mapping criteria. The number of packets for each teat cases is summarized in table 8.

The evaluation for detection models has been discussed in Chapter 4, and is omitted from this chapter. Here, we focus on evaluating the effectiveness of the impact mitigation

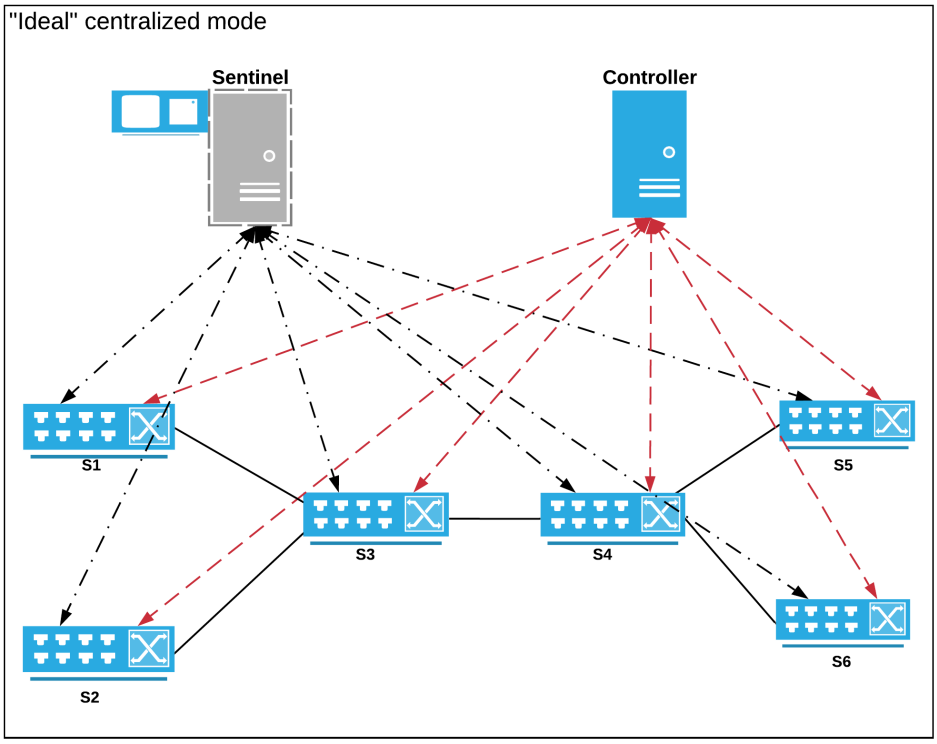

Figure 27: 'Ideal' Centralized Defense Scheme

Table 8: Number of Packets in Each Test Case

Test Cases	Total Packets	Legitimate Packets	Attack Packets
1	395,757	278,129	117,628
2	243,026	41,699	201,327
3	250,873	43,518	207,355
4	245,139	66,339	178,800
5	395,344	335,648	59,696

provided by the overall defense system. Ideally, from the perspective of the targeted host, an efficient and effective defense system should be able to protect the host from receiving attack traffic as much as possible, at the meantime, it should also deliver the legitimate traffic to the desired destination as much as possible. To evaluate how effective the capability of mitigating the DDoS attack impact, we use the following performance metrics:

- Mitigation Rate (MR): to measure the percentage of blocked attack traffic in all attack traffic.
- Delivery Rate (DR): to measure the percentage of legitimate traffic has been successfully delivered in all legitimate traffic.
- Overheads (OH): to measure the number of extra communication messages received and have to be dealt by the controller.

6.4 Evaluation Results

In this section, we describe and discuss the attack traffic mitigation effectiveness in terms of the three performance evaluation metrics: the attack traffic mitigation effectiveness, the legitimate traffic deliver rate and the overhead for controllers. We first examine the performance of using the LSTM model alone as the detector, then examine the performance

of the hybrid model as the detection module. Both detection models are evaluated based on their extend modes, which facilitate the collaboration among different detection nodes across the whole network.

6.4.1 Attack Traffic Mitigation Effectiveness

We capture the replayed traffic at the destination host, which is host 7 (refer to Figure 24) in our testing topology. We measure the number of delivered attack packets, and calculate the mitigation rate by using the number of dropped attack packets divided by the number of total attack packets. The higher the mitigation rate the better the defense system performed in reducing the impact of the DDoS attack traffic.

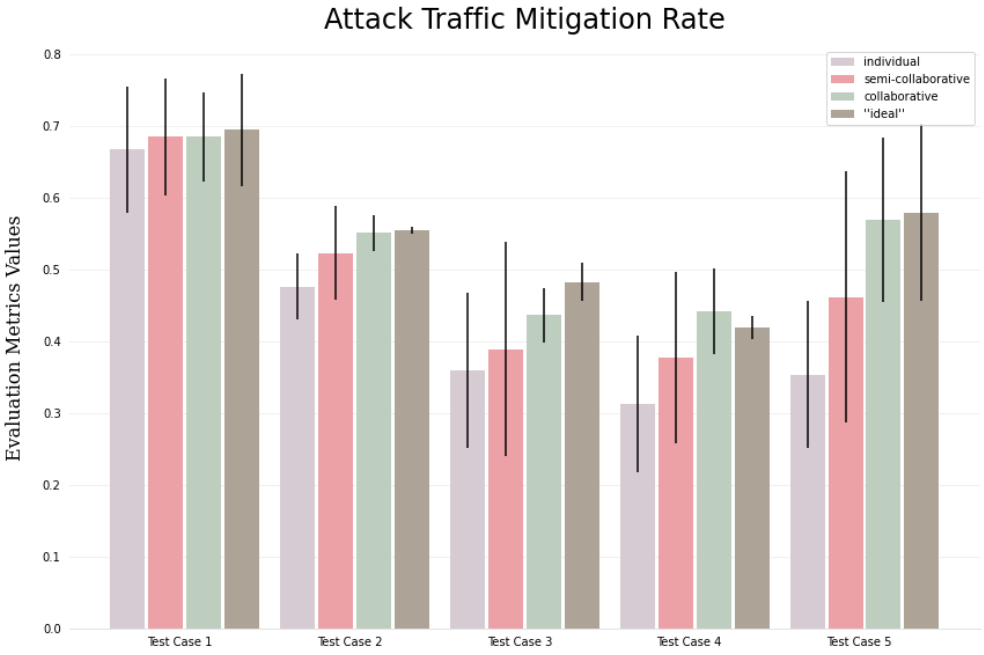

Figure 28: Mitigation Effectiveness Evaluation 1

Let's first discuss the LSTM model alone as the detection module. Figure 28 presents the attack traffic mitigation rate of four evaluated schemes. Focusing on the LSTM collaborative scheme first, which is labeled "collaborative" in the figure. Among the five test cases, the highest mitigation rate mean value is around 70%, while the lowest mitigation

rate mean value is around 40%. Specifically, using the proposed defense system, at least 40% of the attack packets are throttled for all test cases. Comparing with other schemes, our proposed scheme outperforms both individual and semi-collaborative modes in all test cases, and slightly worse than the centralized mode. For test case 4, the proposed defense systems performs even better than the centralized mode, although it has higher variance. Additionally, the individual mode performs the worst in all test cases. This result proves that collaboration indeed helps the detection model better understand the network traffic and make more accurate decisions.

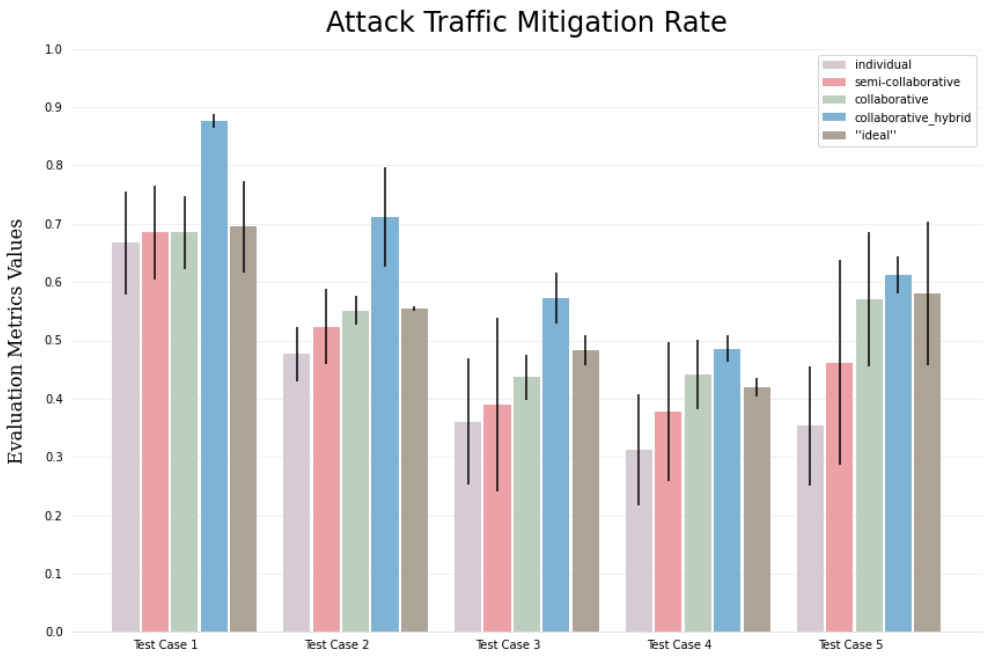

Figure 29: Mitigation Effectiveness Evaluation 2

Including the hybrid detection model into the comparison, the result is represented in Figure 29. Obviously, being aware of the network status significantly improves the attack traffic mitigation rate, even surpasses the "ideal" scheme with LSTM detector alone. It raises the lowest mitigation rate across all test cases to be around 50%, which means at least 50% of attack traffic has been filtered out before reaching the target destination. It achieves almost 90% mitigation rate in test case 1, in which case the victim is well protected.

Although at least 50% of the attack packets have been filtered out, higher attack mitigation rate is expected from the employed detection model, especially observing the near-faultless detection results. Recall that the detection model uses the first N packets to trigger the detection. Although the network status detector periodically actively investigating whether a DDoS attack event is happening, it could not set up flow drop rules to mitigate the attack impact. Hence, for the simplified topology that we used in this evaluation work, these first N packets could already be delivered when the flow is recognized as an attack flow, no matter whether the classification of the inspecting flow is correct or not. If attack flows are short, then the mitigation would not be efficient enough, even though it achieves high accuracy when testing alone.

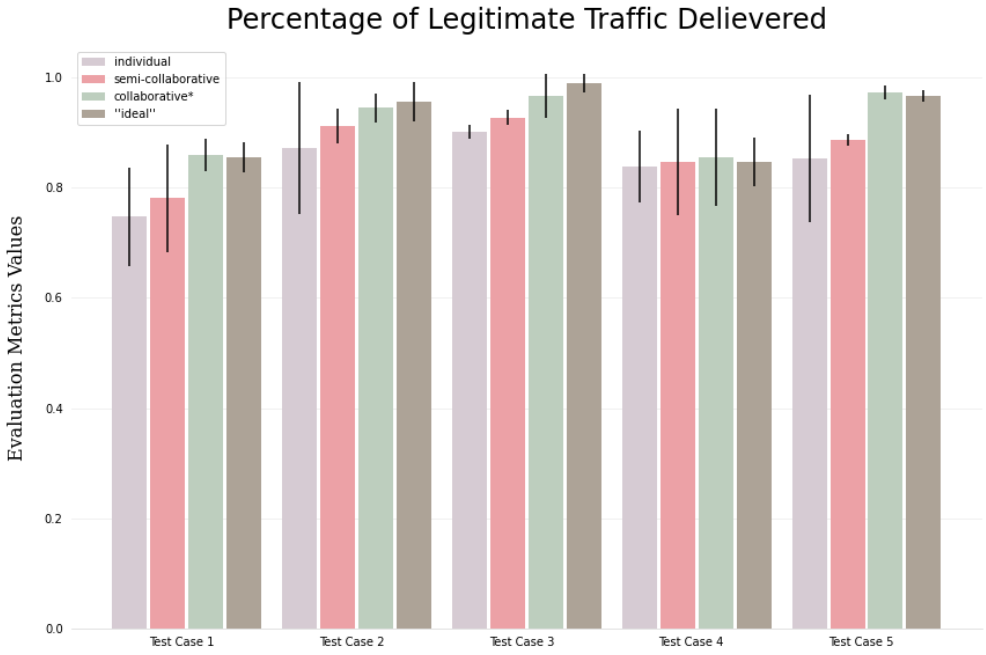

Figure 30: Mitigation Effectiveness Evaluation 3

6.4.2 Legitimate Traffic Delivery Rate

The calculation of legitimate traffic delivery rate is also based on the captured replayed traffic at the destination end, using the number of delivered legitimate packets divided by the

number of total legitimate packets. Using LSTM model alone, results of four evaluated modes are represented in Figure 30. With the collaborative LSTM detector, the defense system has successfully delivered at least 80% of the legitimate packets to the desired destination for all test cases. For some cases, such as test cases 3 and 5, the delivery rate is even close to 100%. Comparing with other modes, the proposed collaborative scheme clearly outperforms individual and semi-collaborative modes, and even slightly surpasses the centralized mode in certain test cases. This results prove that the proposed defense system is capable of maintaining a fine balance between providing security and not causing collateral damage to legitimate services.

Including the hybrid detection model into the comparison, Figure 31 represents the legitimate delivery rate for all schemes. Except for test case 4, in which the defense system achieves almost 100% legitimate traffic delivery rate, being aware of the network status actually degrade the performance of the proposed defense system in terms of legitimate delivery rate. Specifically, in test case 1 and 5, with network status detector, the legitimate traffic delivery rate is slightly worse than the collaborative and 'ideal' modes, but still better than individual and semi-collaborative modes. However, in test case 2 and 3, with hybrid detector, more damages are caused to legitimate users than any other modes.

It worth noting that, again, better performance is expected according to the detection evaluation result. This raises an open challenge for machine learning based DDoS detection models, which is widely neglected. Most published DDoS machine learning based detection models present the promising results for certain given test datasets. In these datasets, flow features for the entire flow are summarized and provided. However, to achieve a real-time detection and mitigation, decisions have to be made in fly. It is impossible to know any summarized flow features or statistics in advance. Great performance in an off-line stand alone testing does not guarantee the efficiency in reality.

6.4.3 Overhead

In our proposed defense system, we leverage the specially designed "sentinels" to alleviate the burden of traffic monitoring and malicious traffic detection from the controller. In other

words, controllers neither need to collect packet information nor request flow statistics from switches to support the DDoS detection. The only extra event that is introduced by the DDoS defense system and needs to be handled by the controller is messages sent from sentinels. When an attack flow is detected, sentinels inform the controller to install throttling rules for the identified attack flow globally. Hence, from the perspective of a controller, the overhead brought by the proposed defense system is the number of messages received from sentinels.

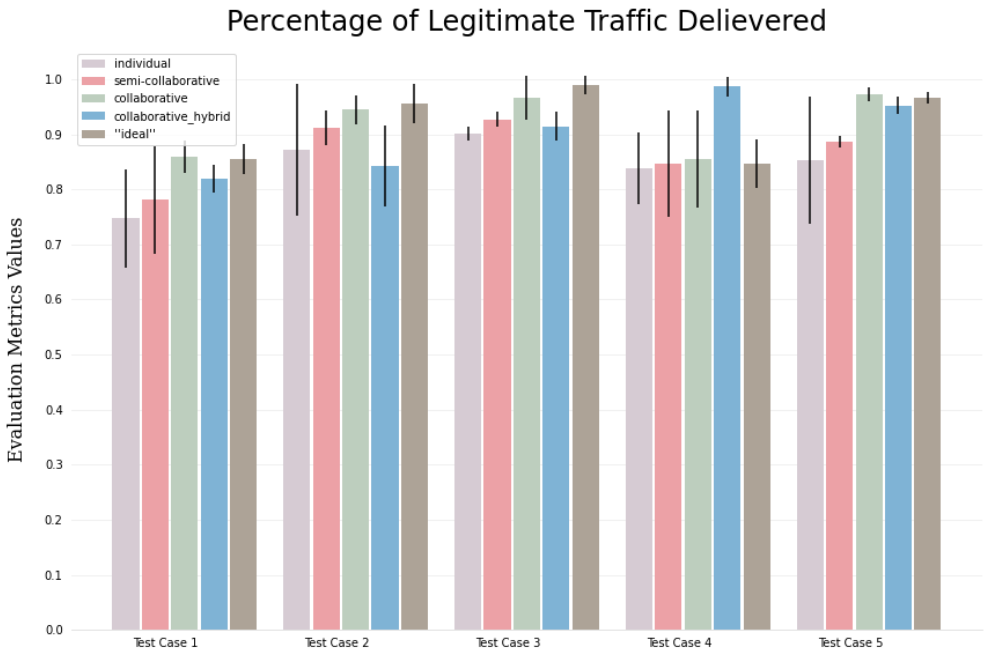

Figure 31: Mitigation Effectiveness Evaluation 4

Under the OpenFlow protocol, for packets that cannot be found matched entry in any flow tables in switches, the default action is to forward these packets to the controller, which creates Packet_In events. The controller instructs the switch to install new flow entries in the rule table. Each entry has two associated time-out fields: idle time-out and hard time-out. The number of Packet_In messages sent to the controller is obviously affected by the values of these two parameters. If a reliable security application is equipped, for any confirmed malicious flows, the idle and hard time-out values could be enlarged, which leads to reducing the Packet_In events. With an effective defense system, although extra

messages are introduced by the deployed security application, the overall messages dealt by the controller could be actually decreased.

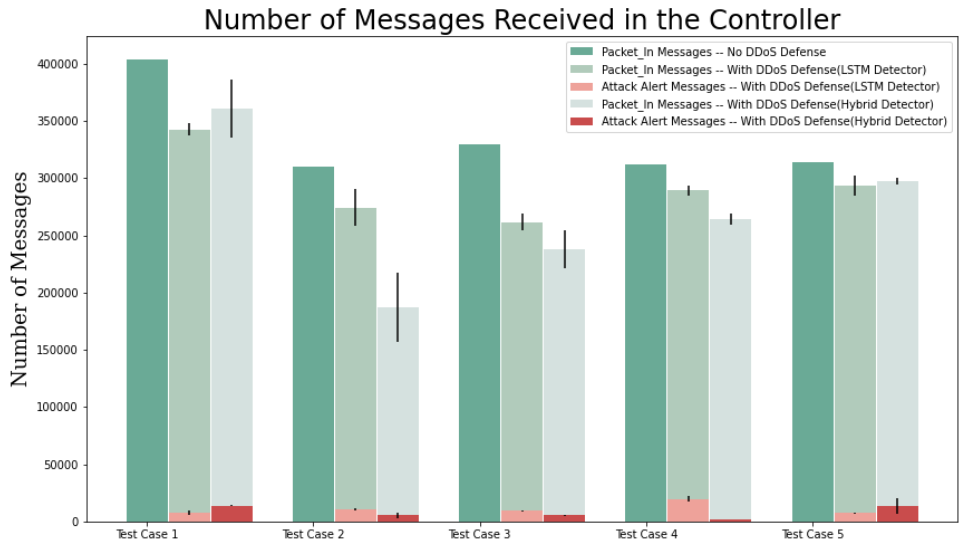

Figure 32: Number of Messages Received in the Controller

We measure the number of messages received by the controller, including messages created by both Packet_In events and attack alert events. In our experiment, we follow the POX controller's default settings, in which the idle time-out and hard time-out are set as 10 and 30 seconds, distinctively. If flows have been confirmed as an attack or legitimate flow, the controller installs the flow entry with idle time-out and hard time-out set as 100 and 300 seconds.

Figure 32 compares the number of messages received in the controller between with and without the proposed DDoS defense system. The bars in the color of green series represent the number of Packet_In messages, which is sent from switches due to unmatched entry. The bars in the color of red series represent the number of attack alert messages sent by sentinels to the controller. We stack the number of Packet_In messages and attack alert messages, so that it is easy to compare the total number of messages received in the controller side. The performances of the proposed defense system using the LSTM detection model and the

hybrid detection model are represented separately. As shown in the graph, for all test cases, with deploying the proposed DDoS defense system, the total number of messages is reduced, no matter which detection model is equipped. If only considering the number of attack alert messages sent to the controller, it occupies a very small percentage of the total messages. The result confirms that the proposed defense system is indeed a lightweight scheme.

6.5 Summary

In this chapter, we present a prototype implementation of the proposed defense system in a SDN environment, without any modifications to the OpenFlow protocol features or OpenFlow switches. This enhances the potential of deploying the proposed defense system in a practical setting. Additionally, we illustrate the design of an evaluation framework to assess the performance of the proposed scheme and carry out a comparative analysis of its performance with other schemes. The experimental results demonstrate that the proposed scheme can effectively detect and throttle DDoS attack traffic without degrading the quality of service for legitimate users. The comparative analysis also shows that collaboration among sentinels across the network indeed improves the detection accuracy, and leads to a more effective mitigation of DDoS impacts than stand alone systems. Furthermore, the overheads caused by the proposed defense system is limited.

7.0 Conclusion and Future Directions

A DDoS attack is an attempt to disrupt legitimate access to targeted computing resources, including computer systems, network devices, servers and web applications. It is known as one of the most destructive attacks on the Internet. With the advent and the emergence of Cloud Computing and Internet of Things, on one hand, the revolutionized technology enables the availability of services and applications to everyone. On the other hand, these techniques also benefit attackers to exploit the vulnerabilities and deploy attacks in more efficient ways. A subscription-based business model, DDoS-as-a-service, also known as "booters" or "stressers", even provides DDoS attacks as a low-cost service, and causes DDoS attacks becoming accessible to the general public. Despite significant advances in the state-of-the-art of system and network security, defending against DDoS attacks remains a challenging problem.

This thesis presents an intelligent, distributed and collaborative DDoS defense system to address the challenge. The proposed defense system unifies SDN and machine learning techniques to provide a practical, yet efficient and effective solution to mitigate DDoS impact. Detection plays a critical role in a DDoS defense system. Any mitigation strategies, which are applied without successfully distinguish malicious traffic from legitimate traffic, will lead to a service degradation from the perspective of legitimate users, and could further cause sever collateral damages. The detection model that is integrated into the proposed defense system takes advantage of machine learning techniques, and combines a network flow by flow detector and a network status detector. Its ability to distinguish between attack and legitimate flows by only examining a relatively small number of a network flow packets enables the defense system to achieve a truly real-time inspection.

Leveraging the flexibility and the programmability of SDN, this thesis proposes a novel defense system architecture comprising a network of peers, referred to as Sentinels, that dynamically and collaboratively defend against DDoS attacks. Each sentinel, which is a specialized controller, monitors a sub-area of the network, and is capable of making decisions driving network-wide security control and policies individually. The communication and

collaboration functionalities among sentinels are also provided by the proposed architecture to achieve more effective DDoS mitigation.

This thesis systematically studies the effectiveness of and the overheads brought by the proposed defense system. As a first step, a prototype implementation is provided in the context of SDN. To compare with non-collaborative and "ideal" models, different modes are also implemented. Then, the empirical evaluation with various measure metrics confirms that the proposed DDoS defense system can effectively detect and throttle DDoS attack traffic without degrading the quality of service to legitimate users. Additionally, the overheads caused by the proposed defense system is minimal.

The study of defending against DDoS attack and mitigating its impact in this thesis is not meant to be complete. Using the current detection model, although only a small number of packets is inspected for flow level detection, a faster detection is favored. For example, if the path from the detection point to the victim is really short, the delay caused by triggering the detection module could lead to install a useless throttling rule, even though attack flow is correctly identified. This observation points to the future direction of improving the detection model with examining even less number of packets.

In current design, each detector was trained with the same training data. This is ignorant of the influence of the locations where the sentinels are deployed at. Based on the analysis conducted in [43], locations indeed affect the performance of detection approaches. To fully explore the capability of machine learning models, and further utilize the SDN control plane, customized training procedure for detection model according to its designed deployment location is desired. Additionally, the sentinel should be aware of the difference between in-bound and out-bound traffic. As such, not only the decision threshold for the detection model could be dynamically adjusted, but also adaptive mitigation strategies could be applied.

Lastly, the idea of collaboration among sentinels can be applied to security policies installation as well. Instead of having throttling rules installed in swithces only by the controller, sentinels can utilize the collaboration channels and propagate the attack flow information, and install security policies globally without sending any extra messages to the controller. Hence, sentinels form an autonomous DDoS defense system.

Appendix A Different LSTM Models

To select the proper number of hidden layers for the proposed LSTM detection model, we have assessed multiple options. In addition to the described model in chapter 4, we also evaluate LSTM models with one hidden layer and with an embedded layer. Considering the complexity of our detection problem, we start with one hidden layer. Figure 33 depicts the architecture of the LSTM model with one hidden layer. Furthermore, inspired by the word embedding from natural language processing, we add one embedded layer for the FLAGS fefatures, which is depicted in Figure A. We applied the K-fold evaluation to select the optimal hyper-parameters of learning rate and the dropping rate. The learning rate is set in range $[0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001]$, and the dropping rate is set in range $[0, 0.01, 0.02, 0.03, 0.04, 0.05]$. When the dropping rate is 0, the drop out layer is actually ignored. After obtaining the best hyper-parameters for each model, we test and compare them.

LSTM model with two hidden layers outperforms both the one hidden layer model, and the model with an embedded layer. Due to the similarity between the one hidden layer and two hidden layers structures, we draw the ROC curve to see whether a proper threshold selection could achieve better results using one hidden layer LSTM. Among all 20 test cases we analyzed, most test cases show the same pattern as test case 5 that is illustrated as an example in Figure 36. It is obvious that the two hidden layers LSTM performs better, no matter what threshold is chosen. Only one exception is observed, which is shown in Figure A. In this exceptional case, when threshold is chosen around 0.15 and 0.2, which is a rarely chosen and narrowed range, the one hidden layer LSTM slightly outperforms the two hidden layers LSTM model. Hence, LSTM model described in Chapter 4 is chosen as the flow-by-flow DDoS attack traffic detector in the proposed defense system.

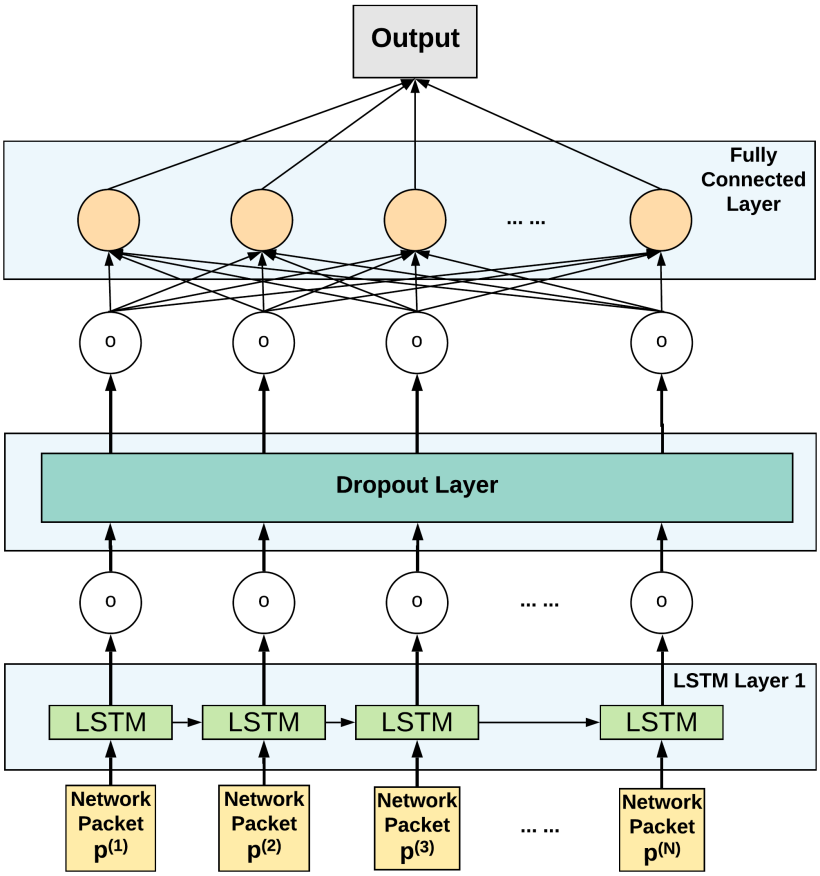

Figure 33: LSTM Model with Single Hidden Layer

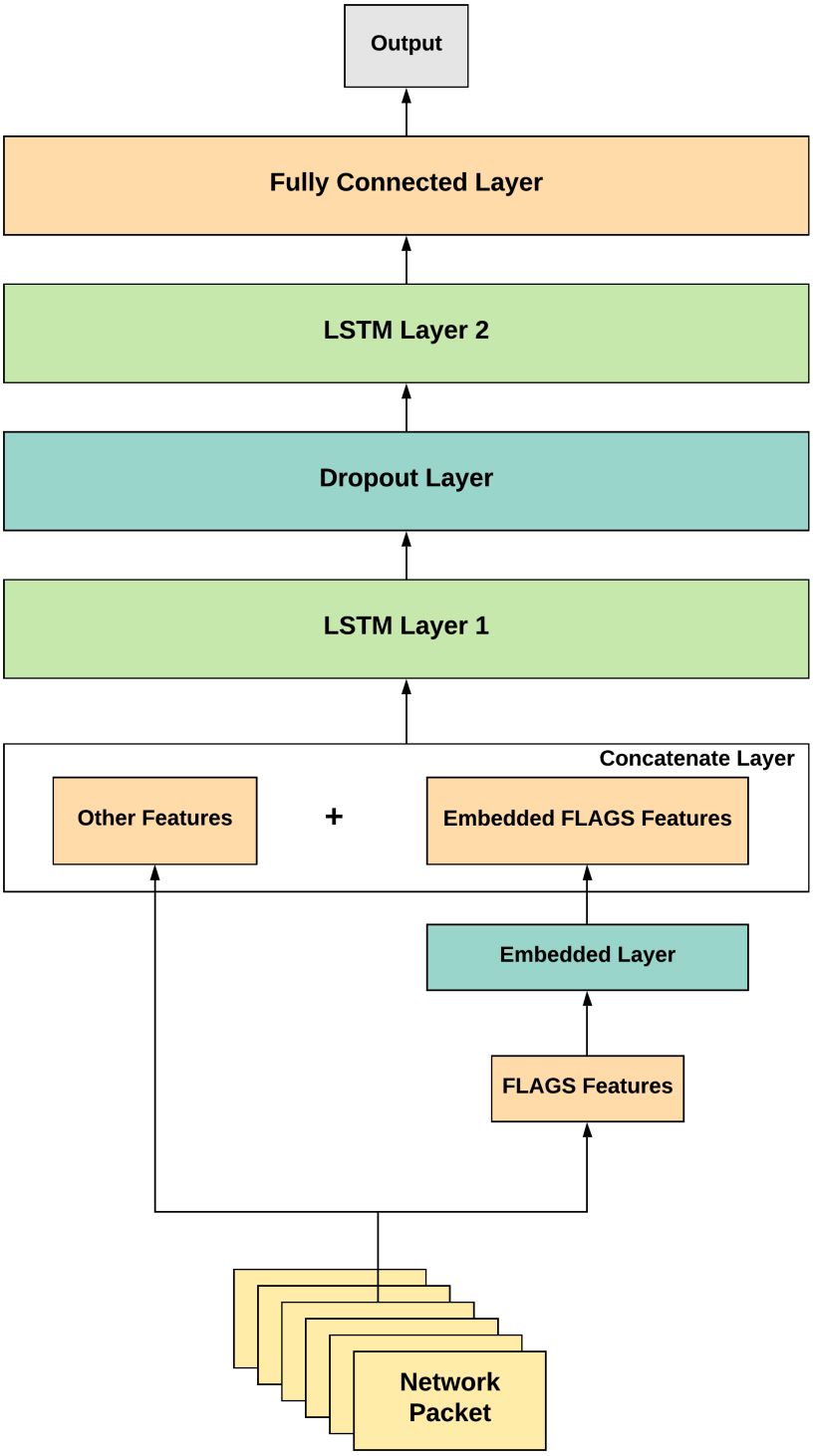

Figure 34: LSTM Model with Embedded Layer

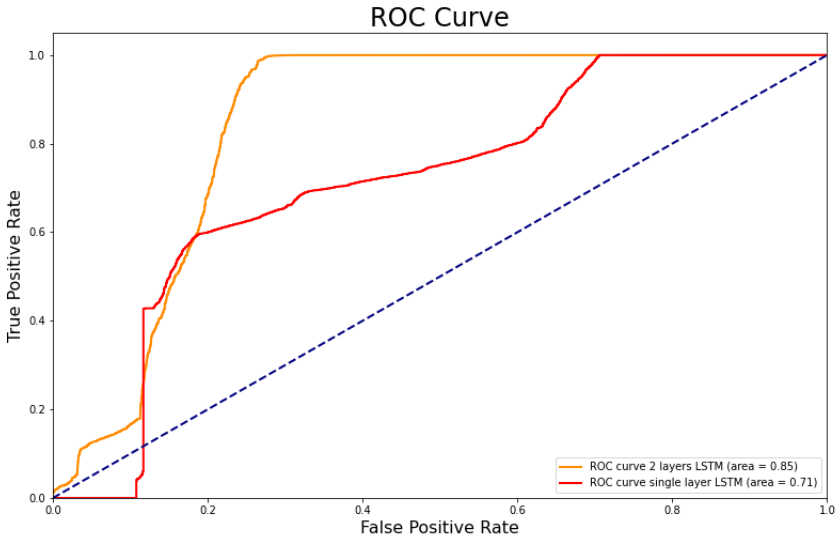

Figure 35: ROC Curve of One Hidden Layer

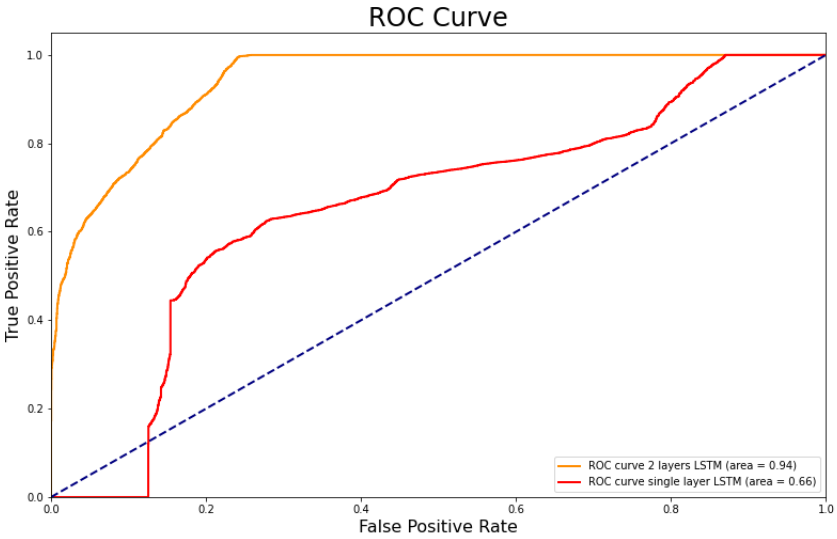

Figure 36: ROC Curve of Two Hidden Layers

Appendix B Network Status Detection Performance

We applied three classifiers, Naive Bayes Classifier, Decision Tree and Support Vector Machine, for network status detection. In Section 4.1, the F1-score of each classifier's performance is presented. In this appendix session, the precision and recall scores are shown in Figure 37 and 38. These three classifiers all perform well in detecting whether DDoS attack event is happening in the network. Naive Bayes classifier slightly outperforms the other two, and, hence, become the choice for our network status detector.

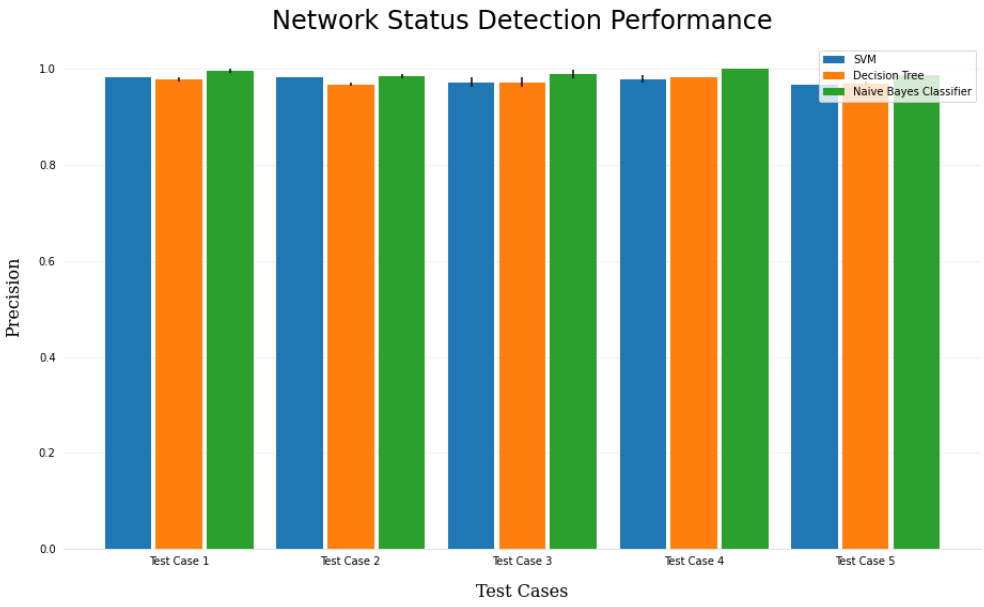

Figure 37: Network Status Detection Results – Precision

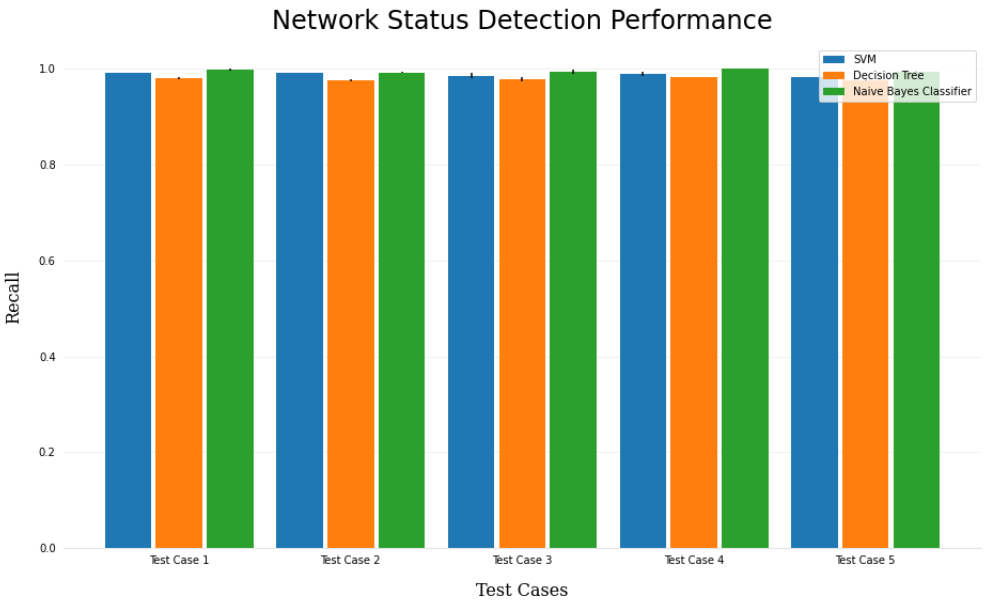

Figure 38: Network Status Detection Results – Recall

Bibliography

[1] OpenFlow switch specification: Version 1.3.0.

[2] Sharad Agarwal, Travis Dawson, and Christos Tryfonas. Ddos mitigation via regional cleaning centers. Technical report, Sprint ATL Research Report RR04-ATL-013177, 2003.

[3] Majjed Al-Qatf, Yu Lasheng, Mohammed Al-Habib, and Kamal Al-Sabahi. Deep Learning Approach Combining Sparse Autoencoder with SVM for Network Intrusion Detection. *IEEE Access*, 6:52843–52856, 2018.

[4] Sardar Ali, Irfan Ul Haq, Sajjad Rizvi, Naurin Rasheed, Unum Sarfraz, Syed Ali Khayam, and Fauzan Mirza. On mitigating sampling-induced accuracy loss in traffic anomaly detection systems. *ACM SIGCOMM Computer Communication Review*, 40(3):4–16, 2010.

[5] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. Understanding the mirai botnet. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1093–1110, 2017.

[6] Katerina Argyraki and David R Cheriton. Scalable network-layer defense against internet bandwidth-flooding attacks. *IEEE/ACM Transactions on Networking (ToN)*, 17(4):1284–1297, 2009.

[7] Katerina J Argyraki and David R Cheriton. Active internet traffic filtering: Real-time response to denial-of-service attacks. In *USENIX annual technical conference, general track*, volume 38, 2005.

[8] Fetia Bannour, Sami Souihi, and Abdelhamid Mellouk. Distributed sdn control: Survey, taxonomy, and challenges. *IEEE Communications Surveys & Tutorials*, 20(1):333–354, 2018.

[9] Sunny Behal and Krishan Kumar. Detection of ddos attacks and flash events using novel information theory metrics. *Computer Networks*, 116:96–110, 2017.

[10] Steven Michael Bellovin, Marcus Leech, and Tom Taylor. Icmp traceback messages. 2003.

[11] Sajal Bhatia, Desmond Schmidt, and George Mohay. Ensemble-based ddos detection and mitigation model. In *Proceedings of the Fifth International Conference on Security of Information and Networks*, pages 79–86. ACM, 2012.

[12] Monowar H Bhuyan, Hirak Jyoti Kashyap, Dhruba Kumar Bhattacharyya, et al. Detecting distributed denial of service attacks: methods, tools and future directions. *The Computer Journal*, 57(4):537–556, 2013.

[13] Rodrigo Braga, Edjard de Souza Mota, and Alexandre Passito. Lightweight ddos flooding attack detection using nox/openflow. In *LCN*, volume 10, pages 408–415, 2010.

[14] Anna L Buczak and Erhan Guven. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys & Tutorials*, 18(2):1153–1176, 2016.

[15] CAIDA. The CAIDA ucsd 'DDoS Attack 2007' Dataset. `https://www.caida.org/data/passive/ddos-20070804_dataset.xml`, 2007. Accessed: Sep.2019.

[16] Yuan Cao, Yuan Gao, Rongjun Tan, Qingbang Han, and Zhuotao Liu. Understanding internet ddos mitigation from academic and industrial perspectives. *IEEE Access*, 6:66641–66648, 2018.

[17] Ruiliang Chen, Jung-Min Park, and Randolph Marchany. Nisp1-05: Rim: Router interface marking for ip traceback. In *IEEE Globecom 2006*, pages 1–5. IEEE, 2006.

[18] Ashley Chonka, Jaipal Singh, and Wanlei Zhou. Chaos theory based detection against network mimicking ddos attacks. *IEEE Communications Letters*, 13(9):717–719, 2009.

[19] Cisco. Cisco visual networking index: Forecast and trands, 2017-2022 white paper. `https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html`, 2019. Accessed: Sep.2019.

[20] Yunhe Cui, Lianshan Yan, Saifei Li, Huanlai Xing, Wei Pan, Jian Zhu, and Xiaoyang Zheng. SD-Anti-DDoS: Fast and efficient ddos defense in software-defined networks. *Journal of Network and Computer Applications*, 68:65–79, 2016.

[21] DARPA. 1999 DARPA Intrusion Detection Evaluation Data Set. https://www.ll.mit.edu/ideval/data/1999data.html. Accessed: Sep.2019.

[22] Darrell Etherington and Kate Conger. Large ddos attacks cause outages at twitter, spotify, and other sites. https://techcrunch.com/2016/10/21/many-sites-including-twitter-and-spotify-suffering-outage/, 2018. Accessed: Sep. 2019.

[23] Thomer M Gil and Massimiliano Poletto. Multops: A data-structure for bandwidth attack detection. In *USENIX Security Symposium*, pages 23–38, 2001.

[24] Shahzeb Haider, Adnan Akhunzada, et al. A deep CNN ensemble framework for efficient DDoS attack detection in software defined networks. *Ieee Access*, 8:53972–53983, 2020.

[25] Alex Henthorn-Iwane. Analyzing the Wikipedia DDoS attack. https://blog.thousandeyes.com/analyzing-the-wikipedia-ddos-attack/, 2019. Accessed: Sep.2019.

[26] Félix Iglesias and Tanja Zseby. Analysis of network traffic features for anomaly detection. *Machine Learning*, 101(1-3):59–84, 2015.

[27] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. A Deep Learning Approach for Network Intrusion Detection System. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies*, pages 21–26. ICST, 2016.

[28] Hossein Hadian Jazi, Hugo Gonzalez, Natalia Stakhanova, and Ali A Ghorbani. Detecting http-based application layer dos attacks on web servers in the presence of sampling. *Computer Networks*, 121:25–36, 2017.

[29] Zhu Jian-Qi, Fu Feng, Yin Ke-Xin, and Liu Yan-Heng. Dynamic entropy based dos attack detection method. *Computers & Electrical Engineering*, 39(7):2243–2251, 2013.

[30] Lei Jiao, Ruiting Zhou, Xiaojun Lin, and Xu Chen. Online scheduling of traffic diversion and cloud scrubbing with uncertainty in current inputs. In *ACM MOBIHOC*, 2019.

[31] Mattijs Jonker, Anna Sperotto, Roland van Rijswijk-Deij, Ramin Sadre, and Aiko Pras. Measuring the adoption of ddos protection services. In *Proceedings of the 2016 Internet Measurement Conference*, pages 279–285. ACM, 2016.

[32] Kübra Kalkan, Levent Altay, Gürkan Gür, and Fatih Alagöz. JESS: Joint entropy-based DDoS defense scheme in SDN. *IEEE Journal on Selected Areas in Communications*, 36(10):2358–2372, 2018.

[33] Mohammad Karami and Damon McCoy. Understanding the emerging threat of ddos-as-a-service. In *Presented as part of the 6th USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2013.

[34] Dongho Kim, Jerry T Chiang, Yih-Chun Hu, Adrian Perrig, and PR Kumar. Craft: A new secure congestion control architecture. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 705–707. ACM, 2010.

[35] Jihyun Kim, Jaehyun Kim, Huong Le Thi Thu, and Howon Kim. Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection. In *2016 International Conference on Platform Technology and Service (PlatCon)*, pages 1–5. IEEE, 2016.

[36] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. Ddos in the iot: Mirai and other botnets. *Computer*, 50(7):80–84, 2017.

[37] Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

[38] Diego Kreutz, Fernando Ramos, Paulo Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *arXiv preprint arXiv:1406.0440*, 2014.

[39] Mohit Kumar. Biggest-ever ddos attack hits github website. `https://thehackernews.com/2018/03/biggest-ddos-attack-github.html`, 2018. Accessed: Sep.2019.

[40] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep Learning. *Nature*, 521(7553):436, 2015.

[41] Chuanhuang Li, Yan Wu, Xiaoyong Yuan, Zhengjun Sun, Weiming Wang, Xiaolin Li, and Liang Gong. Detection and defense of ddos attack–based on deep learning in

openflow-based sdn. *International Journal of Communication Systems*, 31(5):e3497, 2018.

[42] Jundong Li and Huan Liu. Challenges of Feature Selection for Big Data Analytics. *IEEE Intelligent Systems*, 32(2):9–15, 2017.

[43] Xiaoyu Liang and Taieb Znati. An Empirical Study of Intelligent Approaches to DDoS Detection in Large Scale Networks. In *2019 International Conference on Computing, Networking and Communications (ICNC)*, pages 821–827. IEEE, 2019.

[44] Xiaoyu Liang and Taieb Znati. On the performance of intelligent techniques for intensive and stealthy ddos detection. *Computer Networks*, page 106906, 2019.

[45] Francisco Sales de Lima Filho, Frederico AF Silveira, Agostinho de Medeiros Brito Junior, Genoveva Vargas-Solar, and Luiz F Silveira. Smart detection: an online approach for DoS/DDoS attack detection using machine learning. *Security and Communication Networks*, 2019, 2019.

[46] Huan Liu and Lei Yu. Toward Integrating Feature Selection Algorithms for Classification and Clustering. *IEEE Transactions on Knowledge & Data Engineering*, (4):491–502, 2005.

[47] Xin Liu, Xiaowei Yang, and Yanbin Lu. To filter or to authorize: Network-layer dos defense against multimillion-node botnets. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 195–206. ACM, 2008.

[48] Xin Liu, Xiaowei Yang, and Yong Xia. Netfence: preventing internet denial of service from inside out. *ACM SIGCOMM Computer Communication Review*, 41(4):255–266, 2011.

[49] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[50] Team Mininet. Mininet: An instant virtual network on your laptop (or other pc)-mininet. `http://mininet.org/`, 2017. Accessed: Sep.2019.

[51] Jelena Mirkovic, Gregory Prier, and Peter Reiher. Attacking ddos at the source. In *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*, pages 312–321. IEEE, 2002.

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

[53] Prateek Mittal, Dongho Kim, Yih-Chun Hu, and Matthew Caesar. Mirage: Towards deployable ddos defense for web applications. *arXiv preprint arXiv:1110.1060*, 2011.

[54] Quamar Niyaz, Weiqing Sun, and Ahmad Y Javaid. A Deep Learning based DDoS Detection System in Software-Defined Networking (SDN). *arXiv preprint arXiv:1611.07400*, 2016.

[55] Matheus P Novaes, Luiz F Carvalho, Jaime Lloret, and Mario Lemes Proença. Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment. *IEEE Access*, 8:83765–83781, 2020.

[56] Opeyemi Osanaiye, Haibin Cai, Kim-Kwang Raymond Choo, Ali Dehghantanha, Zheng Xu, and Mqhele Dlodlo. Ensemble-based multi-filter feature selection method for ddos detection in cloud computing. *EURASIP Journal on Wireless Communications and Networking*, 2016(1):130, 2016.

[57] Opeyemi Osanaiye et al. Distributed denial of service resilience in cloud: review and conceptual cloud DDoS mitigation framework. *Journal of Network and Computer Applications*, 67:147–165, 2016.

[58] Eric Osterweil, Angelos Stavrou, and Lixia Zhang. 20 years of ddos: a call to action. *arXiv preprint arXiv:1904.02739*, 2019.

[59] Bryan Parno, Dan Wendlandt, Elaine Shi, Adrian Perrig, Bruce Maggs, and Yih-Chun Hu. Portcullis: Protecting connection setup from denial-of-capability attacks. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 289–300. ACM, 2007.

[60] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Computing Surveys (CSUR)*, 39(1):3, 2007.

[61] Trung V Phan and Minho Park. Efficient distributed denial-of-service attack defense in SDN-based cloud. *IEEE Access*, 7:18701–18714, 2019.

[62] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent Advances in Recurrent Neural Networks. *arXiv preprint arXiv:1801.01078*, 2017.

[63] José Jair Santanna, Roland van Rijswijk-Deij, Rick Hofstede, Anna Sperotto, Mark Wierbosch, Lisandro Zambenedetti Granville, and Aiko Pras. Booters - an analysis of ddos-as-a-service attacks. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 243–251. IEEE, 2015.

[64] CJTM Schutijser. Comparing ddos mitigation techniques. In *24th Twente Student Conference on IT*, page 105, 2016.

[65] Dongwon Seo, Heejo Lee, and Adrian Perrig. Pfs: Probabilistic filter scheduling against distributed denial-of-service attacks. In *2011 IEEE 36th Conference on Local Computer Networks*, pages 9–17. IEEE, 2011.

[66] Gao Shang, Peng Zhe, Xiao Bin, Hu Aiqun, and Ren Kui. FloodDefender: Protecting data and control plane resources under SDN-aimed DoS attacks. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[67] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *ICISSP*, pages 108–116, 2018.

[68] Nathan Shone, Tran Nguyen Ngoc, Vu Dinh Phai, and Qi Shi. A Deep Learning Approach to Network Intrusion Detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1):41–50, 2018.

[69] Gaurav Somani, Manoj Singh Gaur, Dheeraj Sanghi, Mauro Conti, and Rajkumar Buyya. DDoS attacks in cloud computing: Issues, taxonomy, and future directions. *Computer Communications*, 107:30–48, 2017.

[70] Ralf C Staudemeyer. Applying Long Short-Term Memory Recurrent Neural Networks to Intrusion Detection. *South African Computer Journal*, 56(1):136–154, 2015.

[71] Jiliang Tang, Salem Alelyani, and Huan Liu. Feature Selection for Classification: A Review. *Data classification: algorithms and applications*, page 37, 2014.

[72] Tuan A Tang, Lotfi Mhamdi, Des McLernon, Syed Ali Raza Zaidi, and Mounir Ghogho. Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 202–206. IEEE, 2018.

[73] Yuan Tao and Shui Yu. Ddos attack detection at local area networks using information theoretical metrics. In *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 233–240. IEEE, 2013.

[74] Tcpreplay. Tcpreplay. `http://tcpreplay.appneta.com/`. Accessed: Sep.2019.

[75] Theerasak Thapngam, Shui Yu, Wanlei Zhou, and S. Kami Makki. Distributed Denial of Service (DDoS) detection by traffic pattern analysis. *Peer-to-Peer Networking and Applications*, 7(4):346–358, 2014.

[76] Thomas Vissers, Tom Van Goethem, Wouter Joosen, and Nick Nikiforakis. Maneuvering around clouds: Bypassing cloud-based security providers. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1530–1541. ACM, 2015.

[77] Luis Von Ahn, Manuel Blum, and John Langford. Telling humans and computers apart automatically. *Communications of the ACM*, 47(2):56–60, 2004.

[78] Bing Wang, Yao Zheng, Wenjing Lou, and Y Thomas Hou. Ddos attack protection in the era of cloud computing and software-defined networking. *Computer Networks*, 81:308–319, 2015.

[79] Haining Wang, Cheng Jin, and Kang G Shin. Defense against spoofed ip traffic using hop-count filtering. *IEEE/ACM Transactions on Networking (ToN)*, 15(1):40–53, 2007.

[80] Haining Wang, Danlu Zhang, and Kang G Shin. Detecting syn flooding attacks. In *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1530–1539. IEEE, 2002.

[81] Wei Wei, Feng Chen, Yingjie Xia, and Guang Jin. A rank correlation based detection against distributed reflection dos attacks. *IEEE Communications Letters*, 17(1):173–175, 2013.

[82] Yang Xiang, Ke Li, and Wanlei Zhou. Low-rate ddos attacks detection and traceback by using new information metrics. *IEEE transactions on information forensics and security*, 6(2):426–437, 2011.

[83] Abraham Yaar, Adrian Perrig, and Dawn Song. Siff: A stateless internet flow filter to mitigate ddos flooding attacks. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pages 130–143. IEEE, 2004.

[84] Xiaowei Yang, David Wetherall, and Thomas Anderson. Tva: a dos-limiting network architecture. *IEEE/ACM Transactions on Networking (ToN)*, 16(6):1267–1280, 2008.

[85] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. A Deep Learning Approach for Intrusion Detection using Recurrent Neural Networks. *Ieee Access*, 5:21954–21961, 2017.

[86] Xiang You, Yaokai Feng, and Kouichi Sakurai. Packet in message based ddos attack detection in sdn network using openflow. In *2017 Fifth International Symposium on Computing and Networking (CANDAR)*, pages 522–528. IEEE, 2017.

[87] Shui Yu, Song Guo, and Ivan Stojmenovic. Can we beat legitimate cyber behavior mimicking attacks from botnets? In *2012 Proceedings IEEE INFOCOM*, pages 2851–2855. IEEE, 2012.

[88] Shui Yu, Song Guo, and Ivan Stojmenovic. Fool me if you can: Mimicking attacks and anti-attacks in cyberspace. *IEEE Transactions on Computers*, 64(1):139–151, 2013.

[89] Xiaoyong Yuan, Chuanhuang Li, and Xiaolin Li. DeepDefense: Identifying DDoS Attack via Deep Learning. In *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–8. IEEE, 2017.

[90] Saman Taghavi Zargar, James Joshi, and David Tipper. A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. *IEEE communications surveys & tutorials*, 15(4):2046–2069, 2013.

[91] Matthew D Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[92] Jing Zheng, Qi Li, Guofei Gu, Jiahao Cao, David KY Yau, and Jianping Wu. Realtime DDoS defense using COTS SDN switches via adaptive correlation analysis. *IEEE Transactions on Information Forensics and Security*, 13(7):1838–1853, 2018.