electronics

Article

A DDoS Attack Mitigation Scheme in ISP Networks Using Machine Learning Based on SDN †

Nguyen Ngoc Tuan [1], Pham Huy Hung [2], Nguyen Danh Nghia [1], Nguyen Van Tho [1], Trung Van Phan [3] and Nguyen Huu Thanh [1,*]

[1] School of Electronics and Telecommunications, Hanoi University of Science and Technology, Hanoi 10000, Vietnam; tuan.ncs16077@sis.hust.edu.vn (N.N.T.); nghia.nd152656@sis.hust.edu.vn (N.D.N.); tho.nv133799@sis.hust.edu.vn (N.V.T.)

[2] Viettel High Technology Industries Corporation, Hanoi 10000, Vietnam; hungph8@viettel.com.vn

[3] Chair of Communication Networks, Technische Universität Chemnitz, 09126 Chemnitz, Germany; trung.phan-van@etit.tu-chemnitz.de

* Correspondence: thanh.nguyenhuu@hust.edu.vn; Tel.: +84-912-523-624

† This paper is an extended version of our paper published in the 11th International Conference on ICT Convergence 2019, Jeju, Korea.

Received: 30 December 2019; Accepted: 25 February 2020; Published: 29 February 2020

Abstract: Keeping Internet users protected from cyberattacks and other threats is one of the most prominent security challenges for network operators nowadays. Among other critical threats, distributed denial-of-service (DDoS) becomes one of the most widespread attacks in the Internet, which is very challenging to mitigate appropriately as DDoS attacks cause the system to stop working by resource exhaustion. Software-defined networking (SDN) has recently emerged as a new networking technology offering unprecedented programmability that allows network operators to configure and manage their infrastructures dynamically. The flexible processing and centralized management of the SDN controller allow flexibly deploying complex security algorithms and mitigation methods. In this paper, we propose a novel DDoS attack mitigation in SDN-based Internet Service Provider (ISP) networks for TCP-SYN and ICMP flood attacks utilizing machine learning approach, i.e., *K*-Nearest-Neighbor (KNN) and XGBoost. By deploying a testbed, we implement the proposed algorithms, evaluate their accuracy, and address the trade-off between the accuracy and mitigation efficiency. Through extensive experiments, the results show that the algorithms can efficiently mitigate the attack by over 98.0% while benign traffic is not affected.

Keywords: distributed denial-of-service; software-defined networking; security; machine learning; KNN; TCP-SYN flood mitigation; ICMP flood mitigation.

1. Introduction

Internet cybercrime has been becoming a severe issue that governments and organizations should cope with nowadays. According to Cisco cybersecurity report, 30% organizations have experienced cyber attacks in 2019. Denial of Service (DoS) and Distributed DoS (DDoS) [1] are the most common attacks on the Internet recently. In DoS and DDoS, attackers take control of many hosts, known as botnets and use them to send extremely large numbers of request to victims to stop their services. Consequently, the destination victims with limited resources are overloaded and cannot provide service to the legitimate users. DDoS attack detection and mitigation techniques are discussed comprehensively in [1,2]. Software-defined networking (SDN) [3] is a new networking architecture in which the control plane and data plane are separated. While the forwarding engine is located in the switch, all network control functions, such as traffic monitoring, routing, etc., reside in a centralized software-based controller. This makes SDN architecture flexible as the network administrator can use programming languages such as Python, Java, or others to add new functionality in the controller, including security functions. The deployment of SDN in network security has gained special attention recently [4–16]. Therefore, in this work, we make use of the SDN technology in oder to mitigate DDoS attacks.

Regarding the categorization of DDoS attacks, there are several types of known DDoS attacks that are divided into three groups, as shown in Figure 1.

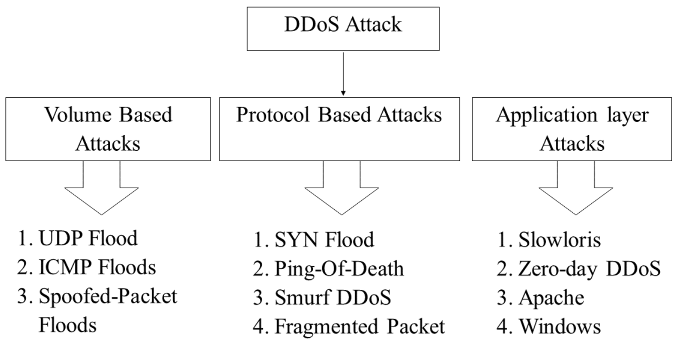


Figure 1. Distributed denial-of-service (DDoS) attacks classification [17].

- In volume-based attacks, the attacker attempts to create congestion by consuming all available bandwidth between the target and the Internet. A large amount of traffic is sent to a target by using a form of amplification or other means that create massive traffic, such as requests from a botnet [18]. A good example of this categorization is ICMP flood attack [1].
- Protocol-based attacks, also known as a state-exhaustion attack, cause a service disruption by consuming all the available state table capacity of servers or intermediate resources like firewalls and load balancers. Protocol attacks utilize weaknesses in layer 3 and layer 4 of the protocol stack to render the target inaccessible [18], e.g., TCP-SYN flood [1].
- On the other hand, the goal of application layer attacks is to exhaust the resources of the target. The attacker establishes a connection with the target and then exhausts the server resources by monopolizing processes and transactions. The attacker exploits a weakness in the layer 7 protocol stack [18].

It is noted that in volume-based and protocol-based attacks a large amount of traffic is sent, making them quite similar and easier to detect than application attack. Although there are many DDoS attack types, in this article we particularly study TCP-SYN and ICMP flood attacks, and then propose a novel model to mitigate these two attacks. Specifically, the TCP-SYN flood attack is based on a weakness of the TCP protocol stack, in which a TCP connection is initiated by the three-way handshake technique that leads to vulnerability. In the TCP-SYN flood, the attacker sends an extremely large number of unacknowledged, malicious TCP SYN messages, consequently a large number of half-open TCP connections leading to the resource exhaustion at the victim. Details of TCP-SYN Flood are presented in [19]. According to Kaspersky [20], TCP-SYN is the most common DDoS attack in the fourth quarter of 2018 that takes up over 50% of the total attacks. Meanwhile, the ICMP flood attack belongs to volume-based DDoS attacks, in which the attacker attempts to overwhelm the network bandwidth of a targeted device with ICMP echo-request packets, causing the target to become inaccessible to normal traffic. The details of ICMP flood are presented in [21].

Defeating against DDoS attacks has been extensively researched recently [4–16,21–23], but there exist some limitations of the current approaches that require further investigation, namely:

- *Detection* versus *mitigation*: most of research work focuses on DDoS detection. The number of mitigation research is quite limited as they are more difficult. The main methodology of current detection approaches is mainly based on the difference of network traffic characteristics and patterns during normal and attack states. However, once DDoS attack is detected, it is more difficult to differentiate attack traffic with innocent traffic. That is why DDoS attack is much more difficult to mitigate.

- *Complexity of detection and mitigation algorithms*: several approaches are based on simulation and offline analysis with complex operations. This makes it difficult to evaluate the performance of detection and mitigation algorithms under real network conditions and real-time.

Following our previous research [24] to address the aforementioned existing issues, in this paper we propose a new DDoS attack mitigation scheme. The contributions of this work are as the follows:

- *Characterizing attack traffic and innocent traffic in Internet Service Provider (ISP) networks*: by investigating real traffic traces, we find out important features that can be used to differentiate normal traffic and TCP-SYN, ICMP Flood attacks traffic in ISP network scenarios;
- *Proposing novel machine learning mitigation algorithms*: based on these features, a machine-learning algorithm integrated in the SDN controller has been developed to detect and drop attack traffic while innocent traffic is almost not affected;
- *An adaptive method to optimize the parameters of mitigation algorithms for accuracy improvement*: A *testbed* is deployed to evaluate the approach in real devices and real time. Based on experiments conducted in the testbed, an adaptive mechanism to improve the mitigation accuracy is proposed.

The rest of this paper is organized as follows. Section 2 analyzes related work concerning detection and mitigation solutions. Section 3 discusses our analysis of real traces with important features to differentiate normal and attack traffic. The details of our proposed approach are described in Sections 4 and 5, including mitigation algorithms and the trade-off between accuracy and system capacity. Section 6 presents the testbed setup and performance evaluation. Section 7 concludes the work and draws our future research direction.

2. Related Work

There are a number of solutions for DDoS detection that can be classified into two types: statistic-based and machine learning-based. The statistic-based solution makes use of statistical methods to analyze traffic between normal and attack phase. In [4], a method relies on the deviation of the throughput from normal distribution when the server is in normal status. Attacks can be detected by a significant deviation in the mean throughput value from the normal status. In [7], Sufian Hameed et al. present a new collaborative DDoS mitigation with a new protocol that allows the SDN controller lying in different autonomous systems to securely communicate and transfer attack information with each other. In [13], the authors present an SDN-based DDoS attack detection framework which performs a two-stage granularity filtering procedure between coarse-grained detection data plane and fine-grained detection control plane for abnormal flows. It leverages the capacity of the SDN-enabled switch and controller. A lightweight flow monitoring algorithm is used in the SDN-enabled switches to capture the key features of DDoS attack traffics to analyze and detect the attack. As a solution to better utilize SDN for network measurement to detect DDoS attack, Ref. [14] proposes a method to detect DDoS attacks leveraging on SDN's flow monitoring capability. This method use measurement resources available in the whole SDN network to adaptively balance the coverage and granularity of attack detection.

The machine learning-based solution uses smart algorithms to find out hidden features inside network traffic in normal and attack states. In [25], a hybrid machine learning algorithm based on Seft-Organizing-Map (SOM) and K-Nearest-Neighbor (KNN) is used to detect ICMP attack. SOM is used to cluster traffic into attack and normal groups in training phase. KNN is used to assigned label for network status based on the label of k nearest neighbors. Other machine learning detection methods are based on determining the time of request between hosts [5], by getting request time, number of source host and number of destination host, and using different algorithms (Naive Bayes, KNN, K-means) to classify traffic into normal and attack. Using the entropy method to determine the randomness of the flow data, Ref. [15] presents a novel solution for the early detection and mitigation of TCP SYN flooding. The entropy information includes destination IP and few attributes of TCP flags. It is implemented as an extension module in Floodlight and evaluate it under different conditional scenarios. Hu et al. [16] proposes an efficient and lightweight framework to detect and mitigate DDoS attacks in SDN by using entropy of features and an SVM algorithm. Firstly, the network traffic information is collected through the SDN controller and sFlow agents. Then an entropy-based method is used to measure network features, and the SVM classifier is applied to identify network anomalies. Another approach using SVM algorithm is presented in [26]. The paper mostly focuses on anomaly traffic detection based on the entropy of IP source addresses and ports by integrating SVM into the Ryu controller. The performance of the system is relied on Mininet that can hardly be evaluated in realtine.

Although recent research work mostly involves in detecting DDoS attack, mitigating attack traffic is not easy to deploy as the system cannot process extremely large traffic in a very short time. Also, it is difficult to differentiate innocent traffic from attack traffic. A popular solution to mitigate DDoS attack is to drop all packets to protect the target. That is, normal traffic is blocked, too. Another solution is based on SDN and an Intrusion Detection System (IDS) to mitigate network attack as proposed by Manso et al. [11]. The work proposes a combination of the widely used Snort IDS [27] with Ryu controller [28] by deploying an Unix Domain Socket between them. Upon detecting attacks by using well-known, predefined signatures, Snort can then alert the SDN controller so that the controller can impose policies on SDN-enabled switches to block malicious blacklisted IPs. The system can detect various kinds of attacks but there are some disadvantages. Firstly, it deploys Snort's predefined rules rather than making use of machine learning algorithms to adapt the policies to the real situation of the traffic at the point of deployment in a specific network. Secondly, the test cases are performed based on Mininet, that cannot ensure the real-time performance in a real system.

As presented in the following sections, by finding important features of malicious TCP-SYN and ICMP flood attacks in an ISP network, the approach in this work makes used of machine learning algorithms being able to classify traffic into attack and normal, so that attack traffic can be mitigated. Moreover, the mitigation mechanism is deployed in a real testbed with physical network devices and servers, thus facilitating the performance evaluation of the newly developed system in realtime, under real network conditions.

3. Traffic Analysis

3.1. Traffic Dataset

In order to develop a new DDoS attack mitigation algorithm, there are two DDoS traffic datasets that have been used. We firstly study and analyze CAIDA 2007 dataset, which is a DDoS attack traffic traces on August 4, 2007 (20:50:08 UTC to 21:56:16 UTC) provided by the Center for Applied Internet Data Analysis [29]. CAIDA 2007 dataset is analyzed in detail in [30]. It is widely used until recently as the reliable dataset for many research work on network security, especially on DDoS modeling, detection and mitigation [6,22,31–33]. The total size of the original dataset is 21 GB. It includes normal traffic and attack traffic. The maximum and minimum packet sizes are 1500 and 46 bytes for malicious attack traffic and 1474 and 40 bytes for the innocent traffic, which are almost the same. However, in the detailed data analysis represented in the next subsection, we recognize some differences between the attack and normal traffic.

The second set of traffic traces have been created in our testbed by using Bonesi [34], a tool to emulate botnet traffic in a testbed environment on the wire. In the testbed the botnets are located in 31 different subnets representing 31 corporate networks. Traffic going from the corporates to the Internet are translated to 31 public IP addresses using NAT. The traffic is generated in 220 s, including 20 s of normal traffic and 200 s of TCP and ICMP attack traffic. We make use of this second dataset as the additional data for training and testing of the machine learning algorithms.

Table 1 presents the number of active source IPs, ports of TCP-SYN and number of source IPs, packets of ICMP in the CAIDA 2007 dataset, while Table 2 shows these corresponding data generated by our testbed.

Table 1. Number of active IPs and ports in CAIDA 2007 dataset.

Time (second)	# of IPs (TCP)	# of TCP Ports	# of IPs (ICMP)	# of ICMP Packets
0–20	2	149	9	7363
20–40	72	424	10	7191
40–60	84	946	9	6646
60–80	94	1291	65	9580
80–100	177	35702	739	370873
100–120	242	82685	1368	983884
120–140	332	134838	2126	1616179
140–160	421	180078	2714	2291028
160–180	504	233090	3365	2905617
180–200	585	186489	4020	2307083
200–220	663	100631	4583	1238029
220–240	747	265239	5218	2970610
240–260	817	262481	5817	3059760
260–280	890	254574	6447	3053683
280–300	1043	253794	7083	3013000

Table 2. Number of active IPs and ports in dataset generated by Bonesi.

Time (second)	# of IPs (TCP)	# of TCP Ports	# of IPs (ICMP)	# of ICMP Packets
0–20	30	3690	31	12540
20–40	31	214631	31	435002
40–60	30	21410	31	804340
60–80	30	23183	31	790350
80–100	30	25273	31	796565
100–120	30	11357	31	799529
120–140	30	26692	31	794226
140–160	30	27139	31	797614
160–180	31	26008	31	797680
180–200	31	25428	31	795097
200–220	31	13500	31	801166

3.2. Traffic Analysis

For TCP-SYN, a flow is defined as a TCP session with 4 specific parameters, namely [IP source addr., IP destination addr., source port number, dest. port number]. The analysis from CAIDA 2007 shows that in the first 20 s, the number of flows is low, which indicates that network is in the normal state. From the 20*th* second, the number of flows increases sharply. It is worthwhile to note that while the number of source IPs increases, the number of opened ports increases much more sharply. That is, the number of opened ports per IP source address escalates aggressively during the attack phase. Each attack source IP opens several ports in seconds before opening extremely high number of ports and sends very large number of packets to the victim. There are approximately 1000 attack IPs corresponding to the sources of malicious attack flows, each malicious flow attacks at different moment. In the top attack phase, the number of ports is over 240 times of the number of source IPs.

Also for ICMP flood attack, Table 1 shows that in the first 60 s, the number of source IPs is low. The number of ICMP packets and IPs begins increasing from the 60*th* second and reaches to over 370 thousands packets and more from 80*th* second, which indicates that network is flooded by ICMP packets from 80*th* second. From that time, the number of ICMP packets per IP is extremely high, reaching over 860 times at 160*th* second. This phenomenon can also be observed in the Bonesi datasets, as shown in Table 2.

From the both datasets it is observed that in the case of TCP-SYN flood the attacker creates an extremely high number of TCP flows by opening many ports per one IP address instead of just using a large number of faked IP addresses. Also, in the case of ICMP attack, an IP address is used to send many ICMP packets instead of using large number of IP addresses to send just few ICMP packets per address. One remarkable explanation of this observation is the follows: in an ISP network, where an ISP serves a few corporate networks, the attacker poisons hosts in some corporate LANs as botnets. When a botnet sends out a packet, the source IP address of the botnet from the local host is translated to public IP address by the NAT device (see Figure 2). Thus, due to the network translation mechanism the number of source IPs is limited, which leads to the above phenomenon. It is worthy to note that this observation always applies in ISP networks that serve several corporates, independent of the traffic patterns.

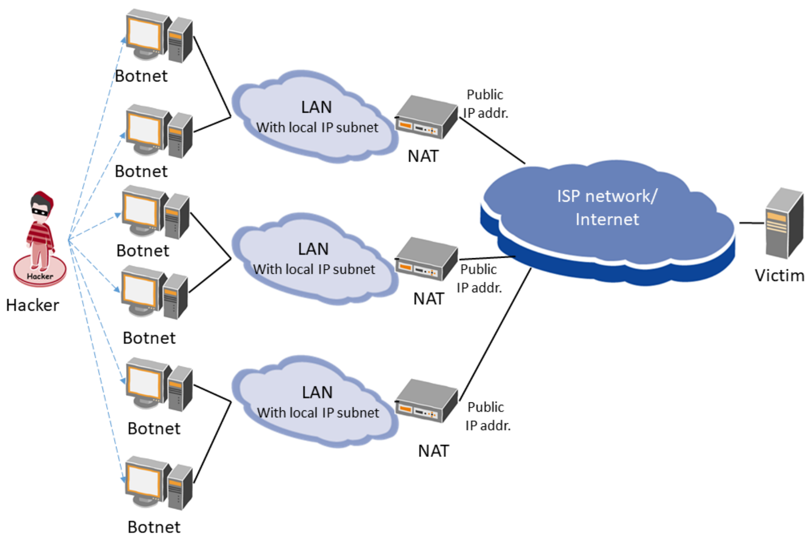
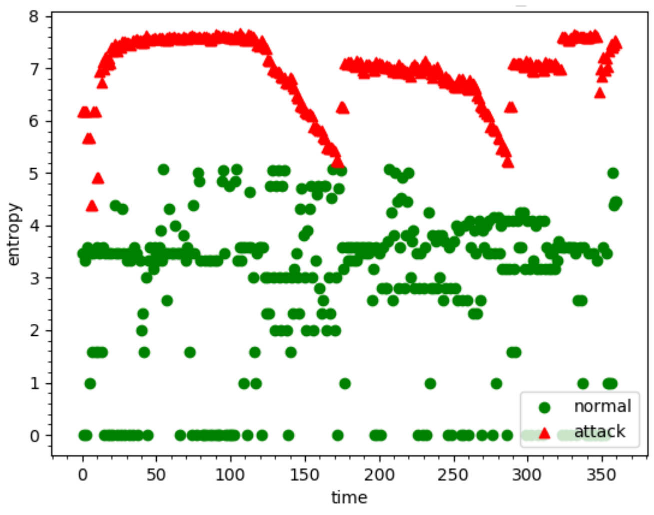
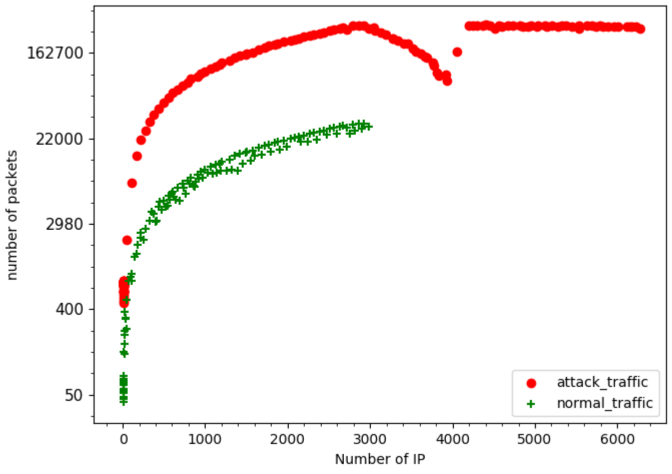


Figure 2. DDoS attack in an Internet Service Provider (ISP) network.

It is also noted that attacks inside the cloud might be different, where attackers can use many IP source addresses. Figure 3a presents the average entropy of port per each IP address in case of normal and attack state. The entropy of port per IP in normal state is around 5 while the entropy in attack state is considerably higher. Figure 3b presents logarithm of number ICMP packets per IP. It shows the different between normal and attack traffic. Based on the results, it is observed that the number of ports per IP address, the entropy of ports per each IP address and number of ICMP packets per IP are important features for developing DDoS detection and mitigation algorithms. The difference of the pattern between attack and normal status lets us use machine learning for this mission.



(a) TCP-SYN



(b) ICMP

Figure 3. (**a**) Average entropy of source ports per each IP address; (**b**) Logarithm of number of ICMP packets per each IP address.

4. KNN-Based Approach for DDoS Attack Mitigation

In this section, we propose an approach to mitigate DDoS flood attack. Figure 4 represents a system architecture of an SDN-based security gateway. The gateway resides at the border of a corporate network that receives traffic coming in and out of the corporate network. The system consists of two components, namely the SDN-enabled switch and the SDN controller. All intelligent, programmable functionalities of the system locates at the controller, which imposes different policies on the SDN-enabled switch. In our architecture, we extend the controller to support DDoS detection and mitigation mechanisms. The Mitigation Block is based on machine learning algorithms developed inside the controller. When a new incoming flow arriving at the switch does not match in the flow table, the switch sends a query to the controller in the `packets-in`. At this stage the Mitigation Block performs an ML-based detection and mitigation algorithm to decide if the flow is innocent or attack based on port's entropy for TCP traffic or logarithm of the number of packets for ICMP traffic as mentioned above. For a malicious flow, the controller then imposes a rule on the switch by creating a `packets-out` with 2 features: "*IP_address = attack IP*" and "*action = drop*" to block the attack flow, while normal traffic is treated as usual in the controller and forwarded to the destination.

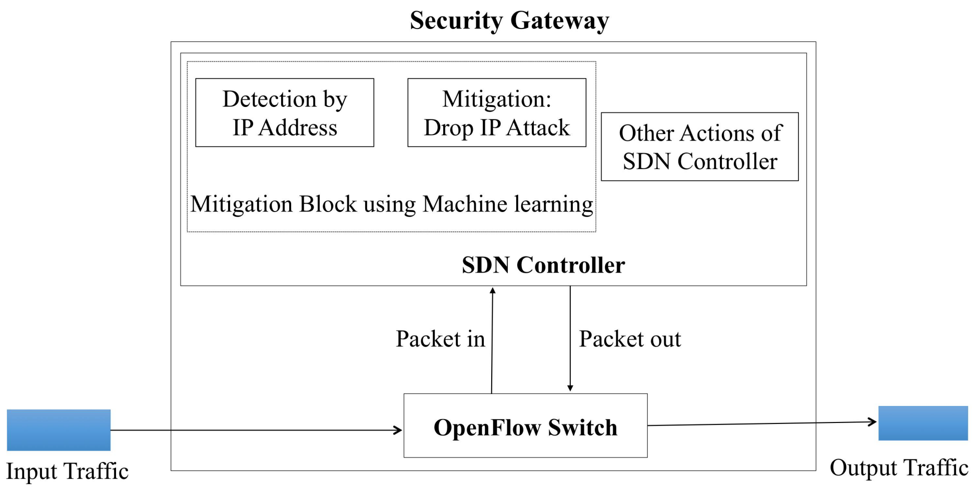


Figure 4. System architecture.

In the mitigation functional block, each IP source is monitored and processed following the diagram shown in Figure 5. The information of each IP source is collected by the OpenFlow switch (OvS) and then is sent to the controller. Let $X = \{X_1, ..., X_n\}$ be the set of source IP addresses. In the case of attack, a source IP X_i may open many ports denoted as $x_i = \{x_{i_1}, ..., x_{i_m}\}$ or have number of ICMP packets n_i. The controller calculates the entropy of the number of TCP ports as in Equation 1 or logarithm of the number of ICMP packets per IP address X_i:

$$H(X_i) = -\sum_{j=1}^{m} p(x_{i_j}) \log[p(x_{i_j})], \tag{1}$$

where $p(x_{i_j})$ is defined in the Equation (2); $p(x_{i_j})$ reflects the number of flows carried in a number of packets sent during a particular time.

$$p(x_{i_j}) = \frac{\text{Number of packets in flow}[\text{IP}(X_i), \text{port}(x_{i_j})]}{\sum \text{all packets of all ports opening in IP}(X_i)}. \tag{2}$$

Let $H(X) = \{H(X_1), ..., H(X_n)\}$ be the entropy vector of n IP addresses; $H(X_i)$ is the entropy of the ith IP that needs to be normalized in the range $[-1, 1]$ for the KNN algorithm. The normalized $y_i = \hat{H}(X_i)$ can be calculated as:

$$y_i = \left\{ \tanh \left[0.1 \left(\frac{H(X_i) - \mu_i}{\sigma_i} \right) \right] \right\}, \tag{3}$$

where μ_i and σ_i are the mean and the standard deviation of $H(X_i)$. Both values are used in real-time DDoS attack mitigation.

K-Nearest Neighbors (KNN) [35,36] algorithm is used to find K nearest neighbors for evaluating the entropy of ports or logarithm of number of ICMP packets. It then defines *thresholds* to differentiate two states of the data set, namely *normal* and *attack* with an equivalent ratio. A data point is marked as attack or normal based on the labels of k nearest Euclidean distance points. In the mitigation phase, the algorithm in the controller marks the traffic as attack or normal and sets label for each flow.

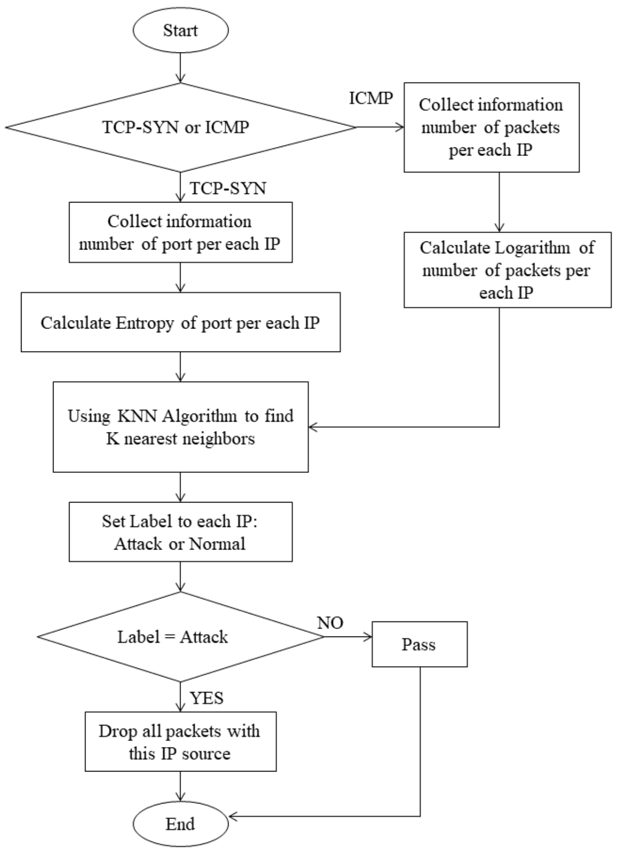

Start
TCP-SYN or ICMP
ICMP
Collect information number of packets per each IP
TCP-SYN
Collect information number of port per each IP
Calculate Logarithm of number of packets per each IP
Calculate Entropy of port per each IP
Using KNN Algorithm to find K nearest neighbors
Set Label to each IP: Attack or Normal
Label = Attack
NO
Pass
YES
Drop all packets with this IP source
End

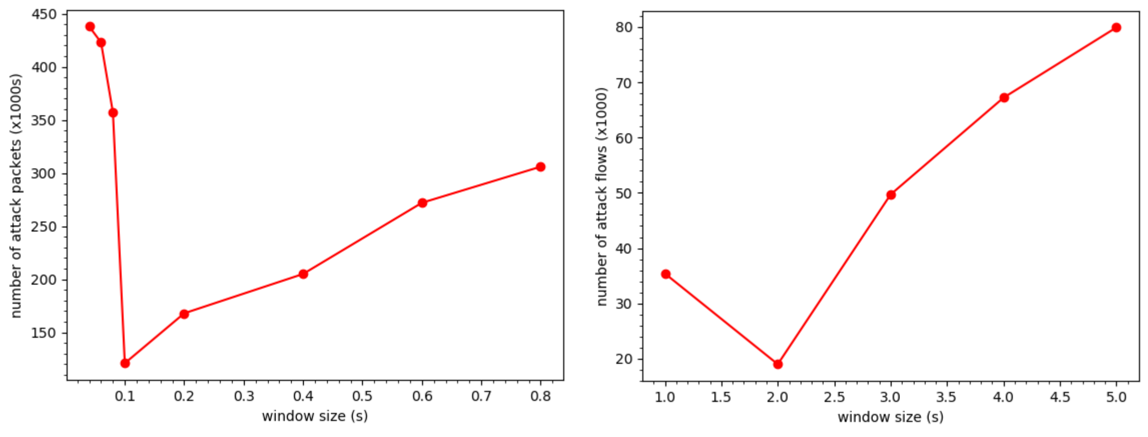
Figure 5. Mitigation process diagram.

5. Adaptice Monitoring Time Window

Real-time is an important factor in a normal network operation as traffic should be monitored and processed online. In general, many features of a real-time series dataset cannot be extracted instantly, as they are calculated based on an accumulative collection of data points over a time duration. In this work this time duration is referred to as the monitoring time window (MTW). For instance, as for the TCP-SYN mitigation algorithms proposed in Section 4, in an MTW period the number of source IP addresses as well as their corresponding opened ports are collected, so that entropy of ports belonging to various source IP addresses can be calculated. In the training phase, the KNN algorithm decides a *threshold* of the entropy of the opened ports. During the testing phase, entropy of ports collected over the monitoring time window is compared with the threshold, thus incoming traffic with the corresponding source IPs can be classified as normal or attack. The same process also applies for ICMP flood attack.

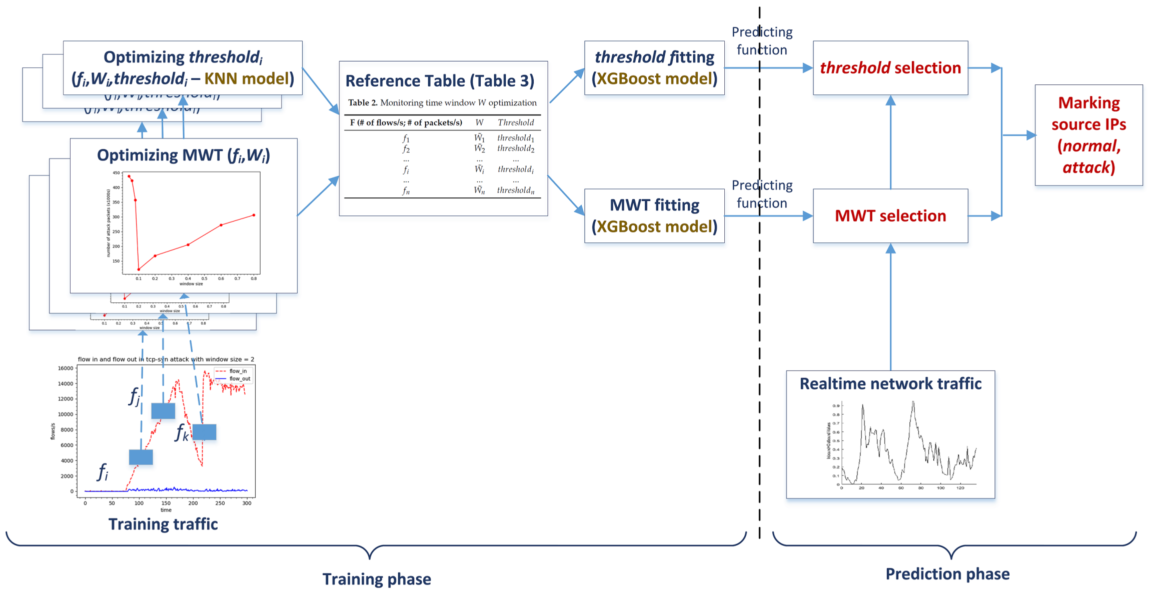
Choosing the size of the monitoring time window is of important as it has an impact on the performance of the system as well as the accuracy of the mitigation algorithms. If the monitoring time window is too short, collected information (e.g., the number of opened ports per IP addresses) would not be sufficient for an accurate attack detection (e.g., entropy of ports and the corresponding threshold). Consequently, the system cannot block malicious incoming traffic efficiently. On the other hand, longer monitoring time window would improve the accuracy of the detection and mitigation algorithms. However, during the data collection process in a MTW period, malicious attack flows have not been detected and blocked yet. Thus in both cases if the MTW is too long or too short, attack flows are not blocked efficiently, leading to a high number of attack flows coming to the victim. This phenomenon can be seen in Figure 6a,b and will be discussed later in this section. Based on this observation, we propose a method to optimize the monitoring window time, so that the efficiency of the mitigation algorithms proposed in Section 4 can be improved.

Figure 7 shows the DDoS mitigation process for incoming traffic based on machine learning deployed in this work. There are two operational phases in the model, namely the training phase and the prediction phase. During the training phase, the monitoring window size as well as the corresponding threshold are defined based on the information extracted from the training traffic. We then develop prediction functions that are used in the prediction phase for mitigating malicious real incoming traffic. The process is descried in detail below.



(a) ICMP **(b)** TCP-SYN

Figure 6. Performance of the system for each monitoring time window value in (**a**) ICMP flood attack—45,000 $pkts/s$ and (**b**) TCP-SYN attack—12,000 $flows/s$ (CAIDA 2007).



Optimizing *threshold*ᵢ (fᵢ,Wᵢ,*threshold*ᵢ – KNN model)
Optimizing MWT (fᵢ,Wᵢ)
Reference Table (Table 3)
Table 2. Monitoring time window W optimization
threshold fitting (XGBoost model)
MWT fitting (XGBoost model)
Predicting function
Predicting function
threshold selection
MWT selection
Marking source IPs (*normal*, *attack*)
Realtime network traffic
Training traffic
Training phase
Prediction phase

Figure 7. DDoS mitigation process based on machine learning.

Let P be the percentage of attack traffic that is blocked at the security gateway (see Figure 4). Ideally, the system should block all attack traffic, that is $P_{ideal} = 100\%$. Thus, P can be considered as the mitigation efficiency of the security gateway. Thus, the objective of the system is to maximize this mitigation efficiency.

Let W be the monitoring time window that needs to be determined and F be the features of attack traffic, i.e., the number of TCP flow per second or the number of ICMP packet per second. Then, P is defined as a function g of F and W:

$$P = g(W, F) \tag{4}$$

For each attack traffic scenario with a corresponding value f_i of number of TCP flow per second or the number of ICMP packet per second, we will have an optimized $\tilde{W}_i$ with the best mitigation efficiency p_i. For each value of f_i, the optimal window size can be chosen by trying many values of W_i and finding the value that maximizes p_i. After that, the KNN algorithm is deployed for each monitoring time window $\tilde{W}_i$ to find a threshold T_i corresponding to p_i.

Figure 6a shows the number of malicious ICMP packets that get through the system in an ICMP flood attack with 45,000 $pkts/s$, depending on the monitoring time window. As presented in the figure, $MTW = 0.1s$ is the optimal window size. Similarly, the optimal monitoring window size in case of TCP-SYN flood with 12,000 $flows/s$ is 2.0s, as shown in Figure 6b. Thus, for each traffic scenario f_i, an optimal monitoring window size $\tilde{W}_i$ and the corresponding threshold T_i can be found, so that a Reference Table can be predefined as shown in Tables 3–5 are the aforementioned reference table with optimized MTW and threshold under various traffic scenarios for TCP-SYN and ICMP flood, respectively. The reference table, however, represents only a limited number of traffic scenarios with their corresponding parameters. In order to predict an arbitrary feature variable value f_c based on known data points in the reference table, the XGBoost [37] algorithm has been used. XGBoost is a machine learning-based algorithm belonging to gradient boosting [38] that is widely used recently. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solves many data science problems in a fast and accurate way.

Table 3. Reference table with optimized monitoring time window (MTW) and threshold for given traffic scenarios.

F (# of Flows/s; # of Packets/s)	$\tilde{W}$	*Threshold*
f_1	$\tilde{W}_1$	T_1
f_2	$\tilde{W}_2$	T_2
...	...	...
f_i	$\tilde{W}_i$	T_i
...	...	...
f_n	$\tilde{W}_n$	T_n

Table 4. Optimized monitoring time window $\tilde{W}_{TCP}$ for TCP-SYN attack under different traffic scenarios.

F (# of TCP Flows/s)	$\tilde{W}_{TCP}$	*Threshold*
500	1.6	5.606
3000	1.7	5.723
6000	1.75	5.781
9000	1.8	5.835
12,000	2.0	6.087
15,000	2.1	6.223

Table 5. Optimized monitoring time window $\tilde{W}_{ICMP}$ for ICMP attack under different traffic scenarios.

F (# of ICMP Packets/s)	$\tilde{W}_{ICMP}$	*Threshold*
2500	0.07	0.372
4000	0.075	0.454
15,000	0.085	0.715
35,000	0.095	0.824
45,000	0.10	0.922
80,000	0.15	1.367

As shown in the figure, two XGBoost instances have been used in the training phase. The first instance is used to predict the monitoring time window $\tilde{W}_i$ based on the parameters F_i and W_i in the training data as described in Table 3. The second instance predicts threshold T_i with training data including F_i, W_i and threshold T_i. During the prediction phase, based on data gathered in the current MTW $\tilde{W}_i^t$, the next monitoring window time $\tilde{W}_i^{(t+1)}$ and its corresponding threshold T_{i+1} are predicted based on the prediction functions (see also Figure 7).

6. Performance Evaluation

6.1. Tesbed Setup and Parameter Settings

In order to evaluate the performance of our approach, a testbed is deployed as shown in Figure 8, which includes POX [39] as the SDN controller, OpenVSwitch (OvS) [40] as the SDN-enabled switch, a normal user, attack traffic replay machine, and the victim. We use TCP replay to replay the CAIDA attack traffic while using Bonesi [34] to generate both innocent and attack NATed IP traffic with opened ports in different experimental scenarios. The configurations of the testbed are shown in Table 6. MWT is adaptive by using the method discussed above. For the sake of simplicity, only results from the CAIDA 2007 dataset are mostly shown and discussed as they are more complex, and the results from the Bonesi dataset generated in our testbed also support the same findings of this work.

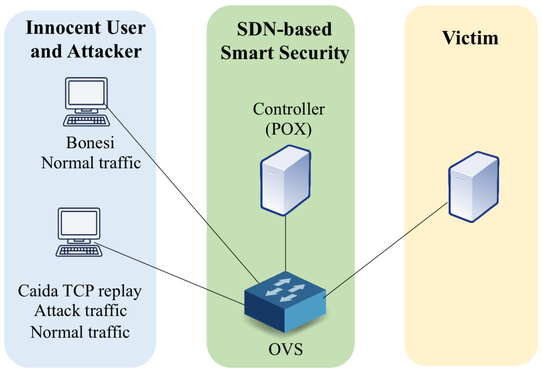

Innocent User and Attacker
Bonesi Normal traffic
Caida TCP replay Attack traffic Normal traffic
SDN-based Smart Security
Controller (POX)
OVS
Victim

Figure 8. Testbed setup.

Table 6. Testbed configurations.

	OvS	Controller	Traffic Generator	Victim
Version	2.5.5	POX with OpenFlow 1.0	Bonesi; CAIDA 2007, 5 pcap files, 25 min attack	
CPU	6 Xeon(R) 2.67 GHz	6 Xeon(R) 2.53 GHz	Xeon(R) 3.70 GHz	Xeon(R) 2.67 GHz
RAM	64 GB	32 GB	16 GB	64 GB
Cores	24	16	12	24
OS	Ubuntu 14.04	Ubuntu 14.04	Ubuntu 14.04	Ubuntu 14.04

In the training phase of the aforementioned machine learning algorithms (see Figure 7), 80% of the CAIDA dataset and normal traffic generated by Bonesi are sent to testbed for training. In the testing phase, the remaining 20% of the CAIDA dataset and normal traffic are used. The following aspects are to be addressed in the evaluation:

- We firstly argue the deployment of the KNN algorithm in the proposed approach by comparing the accuracy and complexity of some machine learning algorithms, since they are important trade-off in realtime operations.
- The performance of the KNN algorithm based on accuracy, precision, recall and F1 score is then evaluated.
- The next step is to discuss the mitigation efficiency of the proposed adaptive MTW in both TCP-SYN and ICMP flood cases.
- Finally, behaviour of innocent traffic under DDoS attack is investigated.

6.2. Experimental Results

In Table 7, we firstly compare some common machine-learning algorithm to evaluate their performance, each algorithm processes 100 records in testing data. As can be seen, KNN can trade off the processing time with low complexity while guaranteeing high performance and high accuracy compared to others.

Table 7. Performance of different ML algorithms.

Parameters	KNN	Decision Tree	NeuralNet
Calculation time (ms)	0.411	0.405	14.325
Accuracy	0.982143	0.981522	0.988425
Precision	0.990291	0.976744	0.966942
Recall	0.971429	0.984375	0.983193
F1-score	0.980769	0.980544	0.975000

Table 8 shows the accuracy, precision, recall and F1-score of KNN with different numbers of neighbor-k (3, 5, 7, 9, 11). If the k value is too high, the algorithm will be overfitting. Overfitting is a modeling error that occurs when a function is too closely fit to a limited set of data points. For this reason, k equals to 9 is the best choice according to our calculation, in which:

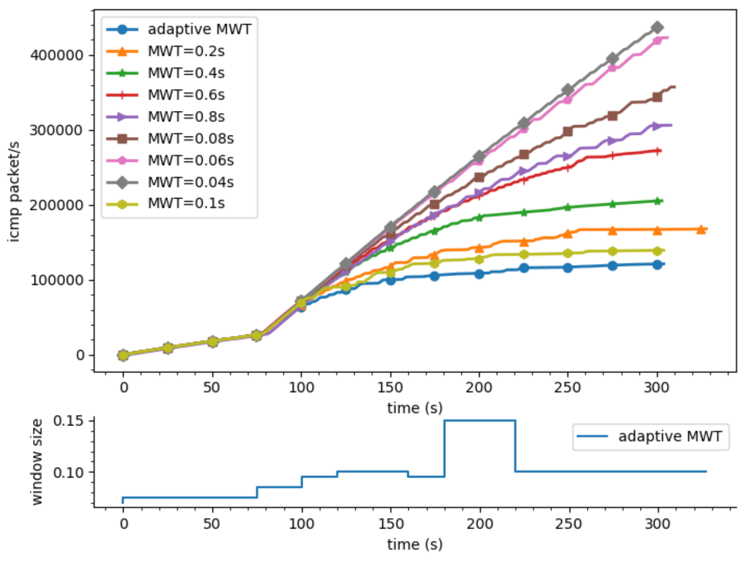
- Accuracy is the number of correctly detected cases in all tests.
- Precision is how accurate the model is out of those predicted positive, how many of them are actually positive.
$$\text{precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}. \quad (5)$$
- Recall calculates how many of the Actual Positives that the model captures through labeling it as Positive.
$$\text{recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}. \quad (6)$$
- F1-score is a function of Precision and Recall. It balances between Precision and Recall; therefore, F1 might be a better measurement to use.
$$\text{F1-score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \quad (7)$$

Table 8. Performance of k-Nearest-Neighbor (KNN).

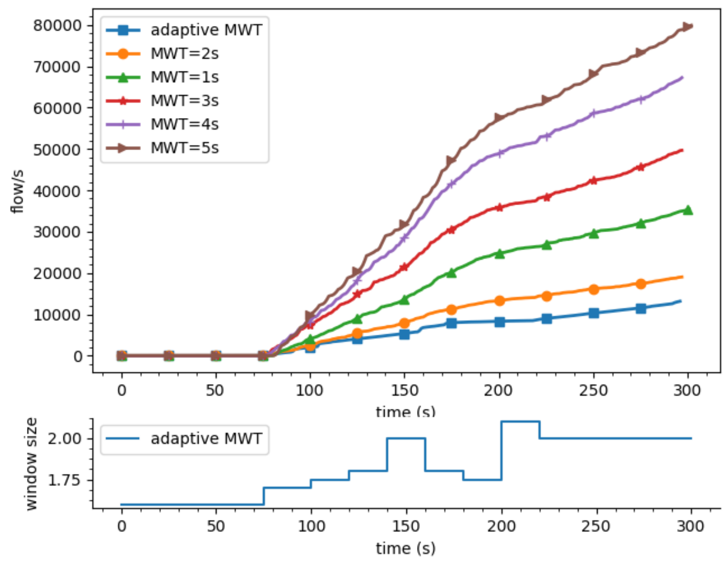
K-Neighbors	Accuracy	Precision	Recall	F1-Score
3	0.964286	0.940171	0.990991	0.964912
5	0.968750	0.947368	0.990826	0.968610
7	0.973214	0.956897	0.991071	0.973684
9	0.991071	0.991453	0.991453	0.991453
11	0.982143	0.990291	0.971429	0.980769

As shown in Table 8, the accuracy of the algorithm is over 96% and up to 99% in case $k = 9$, while the system performance can be adapted to its capacity.

As can be seen in Figure 9a,b, the number of malicious ICMP packets and the number of TCP flows that get through the gateway to the destination vary depending on the monitoring window time. If adaptive window size is applied as described previously in Section 5, which is visible in the lower part of the figures, then the system can block malicious traffic most efficiently. It is also worthwhile to note that too large or too small window sizes, e.g., $MWT_{TCP} = 1$ s or $MWT_{TCP} = 5$ s as shown in Figure 9b, are not the optimal values for the best mitigation efficiency.



(a) ICMP



b

(b) TCP-SYN

Figure 9. Mitigation efficiency depending on monitoring window time in two cases: (**a**) ICMP Flood attack and (**b**) TCP-SYN attack.

Moreover, it is also observed that in an SDN network environment, TCP-SYN flood attack has a negative impact not only on the destination victim, but also on the SDN controller. Since each TCP-SYN message is considered as a new flow that should be processed by the controller, a large number of TCP-SYN may cause the controller to be overloaded. Thus, selecting the right MWT as in the adaptive selection method can also minimize the number of requests arriving in the controller and protect the controller from collapse.

Figure 10 shows the number of TCP flows entering (`flows_in` as red dashed line) and flows exiting (`flows_out` as blue solid line) the system in a second, where `flows_in` include normal and attack traffic. As can be seen in normal condition, from the 0th second to 75th second, the system lets all flows through, no flow is dropped. In the attack phase, after the 75th second, the number of `flows_in` increases to 15,000 flows per second but `flows_out` are approximately 700 flows. That means the attack is restricted by the proposed mitigation algorithm, most of the attack flows are dropped.

Figure 11a presents the number of TCP-SYN attack flows per second to victim. It shows that the victim still receives attack traffic from the 80th second. However, the number of flows is less than 600 flows per second while the number of TCP-SYN flows coming to the network reaches 15,000 flows per second as shown in Figure 10. That is, most of attack traffic is dropped. There is less than 4% attack traffic coming to the victim, which is acceptable. The reason is that the information of the traffic is collected during the monitoring window time. At the beginning of the attack, the number of ports per IP of an attack flow does not sufficiently reach the threshold, the algorithm labels it as the normal traffic and forwards it to the victim. In the next window time, the number of ports per IP reaches the threshold and the controller decides to label the flow as attack and drop.

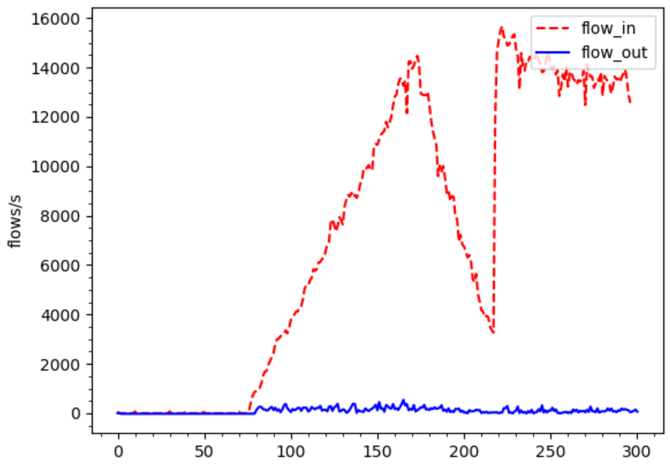
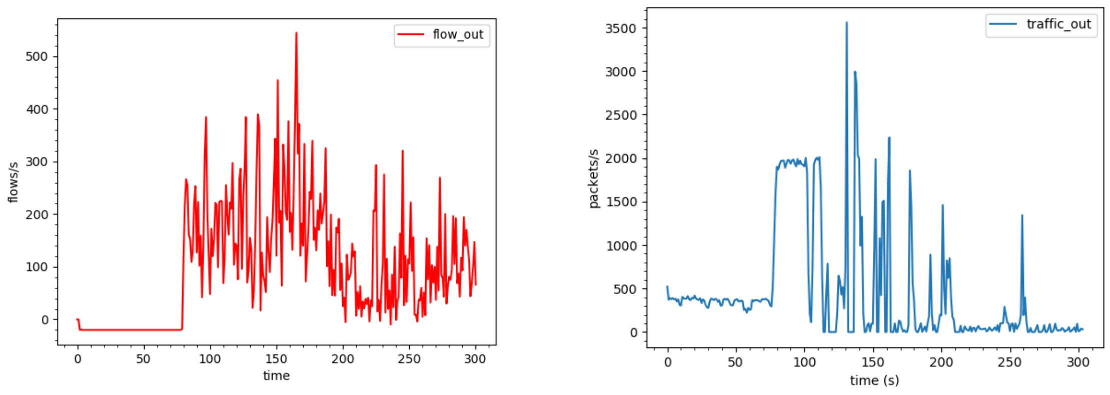


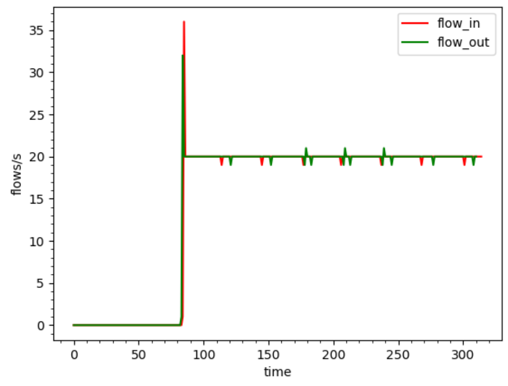
Figure 10. TCP flows entering and leaving the controller.



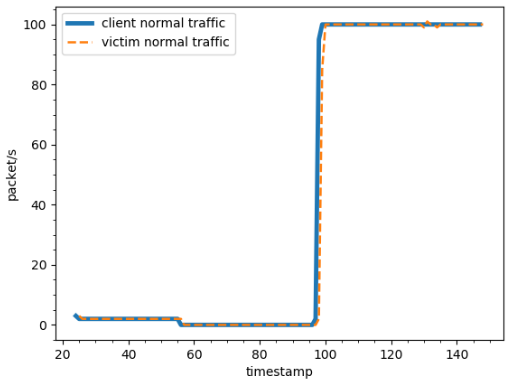
(a) TCP-SYN **(b)** ICMP

Figure 11. Attack traffic volume arriving at the victim: (**a**) TCP-SYN (flows per second) and (**b**) ICMP (packets per second).

Figure 12a presents normal arrival and departure TCP traffic at the gateway, where the red line is the volume of TCP traffic that enters the system in flows/s, and the green line is the TCP traffic that is delivered to the victim. The result shows that most of normal traffic flow is delivered to the victim. The same is also applied for ICMP traffic as presented in Figure 11b and Figure 12b. Overall, the experimental results in Table 9 confirm that our proposed approach can efficiently mitigate TCP-SYN Flood and ICMP Flood attacks up to more than to 98% and 99% respectively while normal traffic is almost not affected. In addition, the SDN-based Smart Security gateway (Figure 8) is robust against TCP-SYN Flood attack, ICMP Flood attack as the time window size is adapted automatically. The SDN controller is also protected from attack traffic and stays operational when the new mitigation model is applied.



(a) TCP-SYN



(b) ICMP

Figure 12. Normal traffic volume arriving at the victim: (**a**) TCP-SYN (flows/s); and (**b**) ICMP (packets/s).

Table 9. Efficiently Mitigate TCP-SYN Flood and ICMP Flood Attacks.

% Blocked ICMP Flood Attack Packets	% Blocked TCP-SYN Flood Attack Flows
99.4%	98.9%

Finally, Figure 13 shows five boxplots of the latency of innocent TCP users under TCP-SYN attack in 5 cases: (1) $MWT = 1s$; (2) $MWT = 2s$; (3) $MWT = 3s$; (4) no mitigation method is applied; and (5) adaptive window size. Each bar represents the maximum and minimum, the average (orange line) and the 50-percentile (the box) of latency. When the system does not use any DDoS mitigation method, only a small part of innocent traffic can come through with very high latency (average from 7.5s to 9.5s). As TCP-SYN mitigation is deployed with fixed window size larger than the average of the adaptive window size (3s—see Figure 9b), most innocent traffic can get to the destination with the latency around 0.18s. This implies that our mitigation approach can greatly improve the system performance. However, larger window size causes larger latency, as the controller should receive more inquiries within a window that leads to over-utilization. When the window size is too large, the controller is overloaded by too many malicious incoming requests. On the other hand, a window size smaller than the average of the adaptive window size (e.g., 1s in our experiment) decreases the accuracy of the mitigation algorithm, as the controller does not collect sufficient information to decide if a source IP is a malicious one. Consequently, some attacks can get through that increases the latency of innocent traffic, which is as high as 1.8s. On the contrary, in case adaptive window size is used, normal traffic can come through with the smallest latency, around 0.1 s.

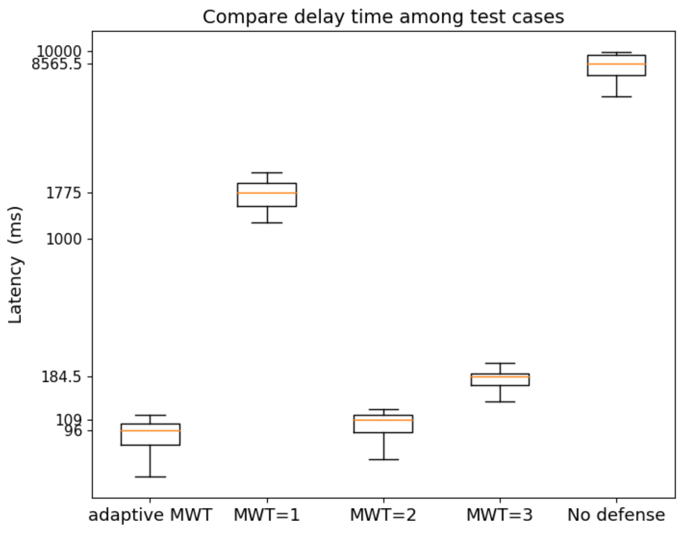


Figure 13. Latency of normal traffic.

7. Conclusions

This article presents a solution for mitigating DDoS attack, especially TCP-SYN flood attack and ICMP flood attack using machine learning in SDN-based ISP networks. A light-weight and fast machine-learning algorithm based on KNN that facilitates realtime operations is used to detect and mitigate attack traffic by tracing back IP sources of attack while normal traffic is almost not affected. In addition, we also propose a machine learning method to automatically adapt the monitoring window time based on traffic input for better performance in terms of mitigation efficiency. The concept has been realized in a real testbed with traffic generators, a POX controller, an SDN-enabled OpenVSwitch and the victim, so that realtime operations of the proposed algorithms can be investigated. Experimental results with the CAIDA traffic traces as well as the botnet traffic from the testbed show that over 98.0% of attack traffic are detected and dropped, and innocent traffic is almost not affected while realtime operation of the system can be maintained.

There are several research directions that would be worthy for more in-depth investigation in the future. As mentioned earlier in this work, DDoS mitigation is much more difficult than DDoS detection, as innocent traffic should be differentiated from attack traffic on-the-flight. In this paper we show some features that support this differentiation for ISP networks, where botnets reside in corporate networks with local IP subnets. With the increasing deployment of IoT and cloud computing paradigms, this picture might greatly be changed. In the cloud paradigm, as attacks may come inside the cloud, features for detection and mitigation of malicious traffic will be more complex. Moreover, as cloud computing allows virtualizing physical resources and facilitates a flexible and scalable *pay-as-you-go* provisioning of services, DDoS in the cloud can come from compromised VMs in an on-demand, flexible and more sophisticated manner. Thus, mitigating DDoS attack in these scenarios would be interesting and require more thorough research in the forthcoming few years.

Author Contributions: Conceptualization, N.H.T.; methodology, N.H.T. and N.N.T.; software, P.H.H., N.D.N. and T.V.P.; validation, N.H.T., N.N.T., P.H.H. and T.V.P.; formal analysis, P.H.H.; investigation, N.N.T., N.H.T. and P.H.H.; resources, N.D.N., N.V.T. and T.V.P.; data curation, N.D.N.; writing—original draft preparation, N.N.T. and T.V.P.; writing—review and editing, N.H.T.; visualization, P.H.H. and N.D.N.; supervision, N.H.T. All authors have read and agreed to the published version of the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mahjabin, T.; Xiao, Y.; Sun, G.; Jiang, W.D. A Survey of distributed denial-of-service Attack, Prevention, and Mitigation Techniques. *SAGE J.* **2017**, *13*, 12. [CrossRef]

2. Zargar, S.T.; Joshi, J.; Tipper, D. A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 2046–2069. [CrossRef]
3. Software-Define Networking: The New Norm for Networking. Available online: www.opennetworking.org (accessed on 26 February 2020).
4. Sangodoyin, A.; Modu, B.; Awan, I.; Disso, J.P. An Approach to Detecting Distributed Denial of Service Attacks in Software Defined Networks. In Proceedings of the IEEE 6th International Conference on Future Internet of Things and Cloud, Barcelona, Spain, 6–8 August 2018.
5. Barki, L.; Shidling, A.; Meti, N.; Narayan, D.G.; Mulla, M.M. Detection of Distributed Denial of Service Attacks in Software Defined Networks. In Proceedings of the 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, India, 21–24 September 2016.
6. Conti, M.; Lal, C.; Mohammadi, R.; Rawat, U. Lightweight solutions to counter DDoS attacks in Software Defined Networking. *Wirel. Netw.* **2019**, *25*, 2751–2768. [CrossRef]
7. Hameed, S.; Khan, H.A. SDN Based Collaborative Scheme for Mitigation of DDoS Attacks. *Future Internet* **2018**, *10*, 23. [CrossRef]
8. Lin, H.C.; Wang, P. Implementation of an SDN-based Security Defense Mechanism Against DDoS Attacks. In Proceedings of the 2016 Joint International Conference on Economics and Management Engineering and International Conference on Economics and Business Management, Wuhan, China, 18–19 June 2016; ISBN 978-1-60595-365-6.
9. Swami, R.; Dave, M.; Ranga, V. Software-defined networking-based DDoS Defense Mechanisms. *ACM Comput. Surv.* **2019**, *52*, 36. [CrossRef]
10. Alshamrani, A.; Chowdhary, A.; Pisharody, S.; Lu, D.; Huang, D. A Defense System for Defeating DDoS Attacks in SDN based Networks. In Proceedings of the 15th ACM International Symposium on Mobility Management and Wireless Access (MobiWac '17), New York, NY, USA, 21–25 November 2017; pp. 83–92. [CrossRef]
11. Manso, P.; Moura, J.; Serrão, C. SDN-Based Intrusion Detection System for Early Detection and Mitigation of DDoS Attacks. *Information* **2019**, *10*, 106. [CrossRef]
12. Chin, T.; Mountrouidou, X.; Li, X.Y.; Xiong, K.Q. An SDN-Supported Collaborative Approach for DDoS Flooding Detection and Containment. In Proceedings of the 2015 IEEE Military Communications Conference, Tampa, FL, USA, 26–28 October 2015.
13. Yang, X.; Han, B.; Sun, Z.; Huang, J. SDN-based DDoS Attack Detection with Cross-plane Collaboration and Lightweight Low Monitoring. In Proceedings of the GLOBECOM 2017—2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017; pp. 1–6.
14. Xu, Y.; Liu, Y. DDoS Attack Detection Under SDN Context. In Proceedings of the IEEE INFOCOM 2016—The 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, USA, 10–14 April 2016; pp. 1–9.
15. Kumar, P.; Tripathi, M.; Nehra, A.; Conti, M.; Lal, C. SAFETY: Early Detection and Mitigation of TCP SYN Flood Utilizing Entropy in SDN. *IEEE Trans. Netw. Service Manag.* **2018**, *15*, 1545–1559. [CrossRef]
16. Hu, D.; Hong, P.; Chen, Y. FADM: DDoS Flooding Attack Detection and Mitigation System in software-defined networking. In Proceedings of the GLOBECOM 2017—2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017.
17. Malik, M.; Singh, Y. A Review: DoS and DDoS Attacks. *Int. J. Comput. Sci. Mob. Comput.* **2015**, *4*, 260–265.
18. What is a DDoS Attack? Available online: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/ (accessed on 22 December 2019).
19. Bogdanoski, M.; Shuminoski, T.; Risteski, A. Analysis of the SYN Flood DoS Attack. *I. J. Comput. Netw. Inf. Secur.* **2013**, *8*, 1–11. [CrossRef]
20. Kupreev, O.; Badovskaya, E.; Gutnikov, A. *DDoS Attacks in Q4 2018*; Kaspersky Lab Report; Kaspersky Lab: Moscow, Russia, 7 February 2019.
21. Harshita, R.N. Detection and Prevention of ICMP Flood DDOS Attack. *Int. J. New Technol. Res.* **2017**, *3*, 63–69.
22. Bouyeddou, B.; Harrou, F.; Sun, Y.; Kadri, B. Detection of Smurf Flooding Attacks Using Kullback-Leibler-based Scheme. In Proceedings of the 2018 4th International Conference on Computer and Technology Applications (ICCTA), Istanbul, Turkey, 3–5 May 2018. [CrossRef]
23. Yusof, M.A.M.; Ali, F.H.M.; Darus, M.Y. Detection and Defense Algorithms of Different Types of DDoS Attacks. *Int. J. Eng. Technol.* **2017**, *9*, 410–414. [CrossRef]

24. Tuan, N.N.; Hung, P.V.; Nghia, N.D.; Tho, N.V.; Thanh, N.H. A Robust TCP-SYN Flood Mitigation Scheme Using Machine Learning Based on SDN. In Proceedings of the 10th International Conference on ICT Convergence (ICTC 2019), Jeju Island, Korea, 16–18 October 2019.
25. Nam, T.M.; Phong, P.H.; Khoa, T.D.; Huong, T.T.; Nam, P.N.; Thanh, N.H. Self-Organizing Map-Based Approaches in DDOS Flooding Detection Using SDN. In Proceedings of the 2018 International Conference on Information Networking, Chiang Mai, Thailand, 10–12 January 2018.
26. Yang, L.; Zhao, H. DDos attack identification and defense using SDN based on machine learning method. In Proceedings of the International Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN), Yichang, China, 16–18 October 2018.
27. Snort. Available online: https://www.snort.org/ (accessed on 26 February 2020).
28. Ryu. Available online: https://osrg.github.io/ryu/ (accessed on 26 February 2020).
29. CAIDA 2007 Dataset. Available online: https://www.caida.org/data/passive/ddos-20070804_dataset.xml (accessed on 30 July 2019).
30. Kato, K.; Klyuev, V. An Intelligent DDoS Attack Detection System Using Packet Analysis and Support Vector Machine. *Int. J. Intell. Comput. Res.* **2014**, *5*, 464–471. [CrossRef]
31. Singh, K.J.; De, T. Mathematical modelling of DDoS attack and detection using correlation. *J. Cyber Secur. Technol.* **2017**, *1*, 175–186. [CrossRef]
32. Hoque, N.; Kashyap, H.; Bhattacharyya, D.K. Real-time DDoS attack detection using FPGA. *Comput. Commun.* **2017**, *110*, 48–58. [CrossRef]
33. Abusitta, A.; Bellaiche, M.; Dagenais, M. An SVM-based framework for detecting DoS attacks in virtualized clouds under changing environment. *J Cloud Comp.* **2018**, *7*, 9. [CrossRef]
34. BoNeSi—The DDoS Botnet Simulator. Available online: https://github.com/Markus-Go/bonesi (accessed on 26 February 2020).
35. Cover, T.; Hart, P. Nearest Neighbor Pattern Classification. *IEEE Trans. Inf. Theory* **1967**, *13*, 21–27. [CrossRef]
36. Cheng, D.; Zhang, S.C.; Deng, Z.Y.; Zhu, Y.H.; Zong, M. kNN Algorithm with Data-Driven k Value. In Proceedings of the Advanced Data Mining and Applications, ADMA 2014, Guilin, China, 19–21 December 2014; pp. 499–512.
37. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16), New York, NY, USA, 13 August 2016; pp. 785–794. [CrossRef]
38. Friedman, J.H. Greedy Function Approximation: A Gradient Boosting Machine. *Annals Stat.* **2001**, *29*, 1189–1232. [CrossRef]
39. POX. Available online: https://noxrepo.github.io/pox-doc/html (accessed on 26 February 2020).
40. OpenVSwitch. Available online: https://www.openvswitch.org/ (accessed on 26 February 2020).