# A DDoS Attack Detection and Defense Scheme Using Time-series Analysis for SDN

Ramin Fadaei Fouladi[a,*], Orhan Ermiş[b], Emin Anarim[a]

[a]*Electrical and Electronics Engineering Boğaziçi University, İstanbul, Turkey*
[b]*EURECOM Sophia Antipolis, France*

**Abstract**

Software defined networking (SDN) has emerged as the integral part of cloud services since it provides flexible management capabilities to monitor and to analyze the network traffic with the help of programmable entities. Although, such functionalities play a significant role in terms of protecting the availability of cloud services against the security threats, SDN still has some vulnerabilities such as the distributed denial of service (DDoS) attacks. The DDoS attackers use spurious packets similar to normal ones and endanger the service continuity of SDN. Although conventional packet-based intrusion detection systems have broad databases to detect DDoS attacks, they are impotent of detection when the attack traffic is sheltered by the normal network traffic. The idea is therefore, to come up with a new countermeasure by observing and distinguishing the instant changes in network. In this work, we propose a DDoS attack detection and defense scheme using time-series analysis for SDN. The proposed scheme employs a model based on the upcoming traffic feature forecasting and the chaos theory together with the exponential filter and the dynamic threshold method to detect instant changes in the network. The experimental result shows that our algorithm has high detection rate and low false alarm.

*Keywords:* Distributed Denial of Service (DDoS), Software Defined Network (SDN), ARIMA, Chaos Theory, Exponential Smoothing

## 1. Introduction

Software-defined networking (SDN) separates the control and data planes to diminish the burden on conventional networks caused by lack of a central control mechanism [1]. This separation between the two planes increases the flexibility of network administration during troubleshooting and monitoring. In addition to data and control planes, SDN also comprises an application plane for software programs to provide efficient solutions for basic network functionalities such as load-balancing, intrusion detection, traffic monitoring, etc [2].

Although the centralization of the controller decreases the deployment cost, it makes the controller more vulnerable to various types of attacks such as distributed denial of service (DDoS) attack, network manipulation and data leakage [3]. The separation between the control and data planes also exposes the network to congestion, which potentially occurs after DDoS attacks. In addition, the attackers can target different elements of SDN, particularly southbound and northbound interfaces, switch hardware, and the controller, in order to attack the availability of the network [4].

Available DDoS attack detection mechanisms for

[*]Corresponding author
*Email address:* `ramin.fadaei@boun.edu.tr` (Ramin Fadaei Fouladi)

SDNs are generally based on statistical and machine learning approaches. Statistical methods employ a threshold to discriminate attacks from normal traffic. On the other hand, machine learning techniques use model training to generalize normal patterns [5]. Compared to machine learning, statistical algorithms are simpler; however, defining optimal thresholds is challenging. To address this problem, in this work, we propose a DDoS attack detection and defense scheme based on statistics and time-series analysis which is integrated into the SDN controller. The contributions of the paper are as follows:

(i) First, we extract key features from the flow table of OpenFlow switches to analyze their time-series representation. Therefore, the controller monitors each switch separately and applies the proposed algorithm to detect anomalies due to the DDoS attack event in particular switches. Such anomaly can be detected using the number of unique source IP addresses (USIP) feature, since during the attack phase this value increases significantly compared to the normal traffic.

(ii) In addition, when an attack occurs, it is possible to observe changes in the number of unique destination IP addresses (UDIP) feature; however, this change is not as significant as the change in the USIP feature. By considering the dramatic growth in the total number of packets (TPACK) in the flow table during the attack, the number of UDIP value substantially reduces when it is normalized by the TPACK values. Therefore, we employ normalized UDIP as the second feature to detect DDoS attacks.

(iii) The upcoming value of the USIP time-series is estimated by using an auto regressive integrated moving average (ARIMA) model and the error of estimation is examined for chaotic behaviour. Finally, according to the aforementioned method, a binary anomaly score is assigned to the traffic instance.

(iv) Another binary anomaly score is obtained by using the dynamic threshold method based on the exponential filter and the NUDIP time-series. The product of two mentioned binary scores is used to identify DDoS attack in each switch. After detecting the attack samples, the scheme activates the countermeasure mechanism.

(v) The chaotic behaviour and the dynamic threshold method resolve the problem of constant threshold in the previous works. Moreover, by monitoring each switch, the source of the attack can be identified and prompt countermeasure would be applied by modifying the packet forwarding policy at that switch. In order to evaluate system performance in successful detecting of DDoS, we implement our scheme on an example SDN network via using Mininet environment [6].

The rest of the paper is organized as follows. In the next section, we overview the existing studies in the literature. The background knowledge of related technologies is discussed in Section 3. Details of our DDoS attack detection and defense scheme are given in Section 4. Section 5 presents the performance evaluation of our scheme. Finally, Section 6 concludes the paper.

## 2. Related Work

In general, DDoS attack detection algorithms in the literature can be categorized as intrinsic and extrinsic [7]. While the former is related to the structural changes in SDN, the latter corresponds to the flow-based analysis. Since they provide better detection accuracy as far as the SDN environment is concerned, we focus on flow-based solutions in this work. Such solutions are also classified as either the statistical-based or the machine learning-based. Most of the works related to the statistical approach, are based on entropy concept. Entropy is the measure of randomness of an attribute within a specific period of time. For instance, a random variable with a high spread probability distribution has high entropy. On the other hand, in order to differentiate the attack traffic from the normal one, a machine learning method needs to be fed by a set of training samples. Each sample consists of a set of features obtained from the network traffic data to generalize the discriminating model.

Due to the promising results of entropy-based methods in detecting anomalies in traditional networks [8, 9, 10], they have also been adopted by detection algorithms for SDN. In [11], a threshold is obtained by using the entropy of the destination IP addresses. Later, this threshold is used to determine whether there is an attack or not. Particularly, if the entropy of the test sample is less than the defined threshold, it is labeled as an attack. In [7], an entropy-based method [12] is proposed for DDoS attack detection and mitigation. Different features from payload are extracted and used for the detection of the attack. The algorithm has three stages of nominal, preparatory and active mitigation. While the normal pattern baseline is obtained in an attack free scenario at nominal stage, the attack samples are identified at the preparatory stage. Later, the countermeasure against the attack traffic is carried out at the active mitigation stage. A combination of entropy and information distance is used in [13] to check the traffic for the DDoS attack detection. Although entropy-based methods are employed in DDoS attack detection, they suffer from some limitations. Entropy maps the probability distribution into a single number; therefore, the information of the distribution is discarded. Furthermore, two distributions with the same amount of uncertainty result in the same entropy value. The last but not the least, finding the optimal threshold for the generalized entropy is difficult. These limitations may yield high false positive rates during the detection process.

SDN-based DDoS attack detection methods also use machine learning algorithms for the detection procedure. Some features from the network flow are extracted and used by a machine learning model to differentiate malicious traffics from benign ones. In [14], six features including, average of packets per flow (APF), average of bytes per flow (ABF), average of duration per flow (ADF), percentage of pair-flow (PPF), grows of single-flow (GSF) and grows of different port (GDP) are extracted from the network traffic and introduced into a self-organizing map (SOM) [15] to create a selective model for separating attacks from normal traffic. In [16], a DDoS detection algorithm based on entropy and machine learning approaches is proposed. In the first phase, lightweight, a method based on entropy is used to detect attack traffics. In the second phase, heavyweight, different machine learning approaches are employed to classify abnormal traffics. In [17], authors use different machine learning methods to detect DDoS attack in SDN. According to the result, support vector machine (SVM) [18] algorithm has the best performance. In [19], a set of five features consisting of entropy of source IP (etsSrcIP), entropy of source port (etsSrcP), entropy of destination port (etsDstP), entropy of packet protocol (etsProtocol) and total number of packets (totalPacket) is fed into an SOM to classify the traffic as normal or attack. In [20], SVM and idle time out adjustment (IA) are used to detect DDoS attacks and to classify the traffic. The entropy of source and destination IP addresses are used as the vector for the SVM algorithm to generalize a specific DDoS attack detection model in [21]. Four different features including byte rate (BR), symmetric flows percentage (SFP), variation rate of asymmetric flow (VAFR) and flows percentage with small amount packets (FPSA) are extracted in [22]. The features are used as a four-tuple feature by a back propagation neural network (BPNN) to discriminate attack from normal traffics. In [23], different features are collected from the SDN and used in machine learning algorithm to detect the DDoS attack. Four different machine learning algorithms including SVM, K-nearest neighbor (KNN) [24], artificial neural network (ANN) [25] and naive Bayes (NB) [26], are employed. KNN and NB outperform with respect to success rates. In [27], a stack auto-encoder based on deep learning model is employed to classify the data obtained from the OpenFlow switch. A deep learning-based DDoS detection and defense architecture is proposed in [28]. The detection part consists of multiple deep learning layers such as CNN, LSTM and bidirectional RNN [29]. In [30], a deep learning model is proposed to detect DDoS attack in SDN. The statistics during the attack are analyzed and employed by the deep learning approach to classify the traffic into two classes, i.e., malicious and legitimate. In [31], different ensemble models including ensemble CNN, ensemble RNN, ensemble LSTM, and hybrid

RL are used for DDoS detection in SDNs. The result shows that the ensemble CNN outperforms.

While machine learning-based methods produce heavy computational load on system and require complex data demands, statistical-based methods seem more favorable. In this work, we propose a DDoS attack detection and defense scheme based on the time-series analysis. The USIP and NUDIP are extracted from the OpenFlow switches as the key features. In order to detect DDoS traffic, each feature is processed separately and simultaneously. While ARIMA model and chaos theory are applied on USIP to label the traffic samples as normal or abnormal, exponential filter and the dynamic threshold algorithm are carried out on NUDIP. The proposed scheme can identify the switches involved in the attack and sets a defense protocol against such attacks.

## 3. Background

In this section, we explain the theory part of different concepts utilized in this paper. We begin by discussing the OpenFlow[1] which is the key protocol of SDN. Having provided a brief introduction related to the protocol, we dive into the ARIMA model followed by chaos theory and exponential filter. These ideas play important roles in DDoS attack detection in this paper.

### *3.1. OpenFlow*

The OpenFlow protocol was first proposed in 2008. The main idea is to divide the network into two separate parts of control plane and data plane, which communicate with each other through a secure channel. Figure 1 displays the OpenFlow architecture.

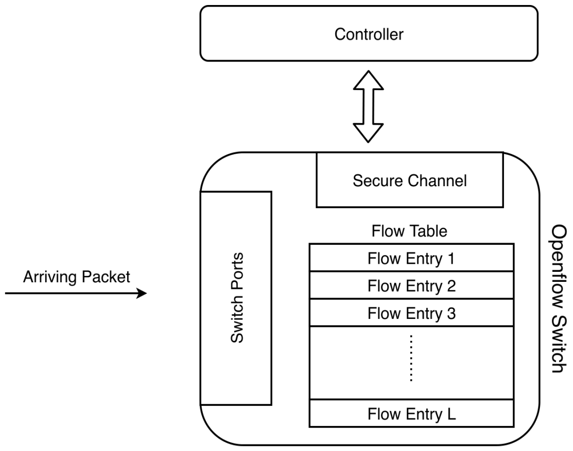

Figure 1: OpenFlow Architecture.

Packets are always handled as "flows" in an OpenFlow switch. When a new packet arrives at a switch, it is compared with the packets which are recorded as the entries inside the flow table of the switch. If it is matched with an instance in the flow table, the associated action is executed and the counter of the flow entry increments; otherwise, the new unknown packet is encapsulated into Packet-In massage and sent to the controller. A flow entry mainly includes three parts of header, counter and action. Header consists of different segments of a packet, such as IP addresses, ports, type of protocol, and etc. When the new packet arrives, its field is compared against the header of the flow entry to find a match. The counter stores the information of the packets matching this flow entry. The action part determines the type of action if a match is found.

[1]https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

### *3.2. Auto Regressive Integrated Moving Average (ARIMA)*

The Auto Regressive Integrated Moving Average (ARIMA) model is used to forecast a time-series which can be made stationary by differnecing if necessary. A stationary time-series data has the property that its statistical characteristics such as the mean and variance are constant over the time [32]. ARIMA is characterized by a three-tuple $\langle p, d, q \rangle$, where $p$ is the number of auto-regressive (AR) terms, $d$ is the number of differences required for stationarity, and $q$ is the number of moving average (MA) or lagged forecast errors terms. Let $\mathcal{Z} = \{z_1, z_2, \ldots, z_n\}$ be a stationary time-series, the general forecasting equation in terms of $z$ is represented as:

$$\hat{z_t} = \theta_0+\phi_1 z_{t-1}+\ldots+\phi_p z_{t-p}+\varepsilon_t-\theta_1\varepsilon_{t-1}-\ldots-\theta_q\varepsilon_{t-q},$$

where $\phi_i$ $(i = 1, ..., p)$ and $\theta_j$ $(j = 1, ..., q)$ are AR and MA items, respectively. $\varepsilon_k$ $(k = t, t-1, ..., t-q)$ are named as error terms which are generally assumed to be independent, identically distributed variables sampled from a normal distribution with zero mean. It is worth to note that, depending on the characteristic of the series, each $p$, $d$, and $q$ can be equal to zero; therefore, the model may be simplified. For instance, ARIMA$\langle p, 0, 0\rangle$, ARIMA $\langle 0, d, 0\rangle$, ARIMA $\langle 0, 0, q\rangle$ and ARIMA $\langle p, 0, q\rangle$ are pure AR $\langle p\rangle$, random walk, MA $\langle q\rangle$ and ARMA $\langle p, q\rangle$ respectively.

### *3.3. Chaos Theorem*

The error between the actual value and the estimated one is the key factor to detect DDoS attack samples. To assign anomaly score to the potential DDoS attack candidates in the network traffic, we analyze the chaotic behavior of the estimated error. Therefore, we employ the local Lyapunov exponent [33] as below:

$$\lambda_t = \frac{1}{t} ln(|\frac{e_t}{e_0}|), \tag{1}$$

where $e_0$, $e_t$ and $\lambda_t$ are the first prediction error, the $t^{th}$ prediction error and the lyapunov exponent at $t^{th}$ time instance, respectively. A Positive $\lambda_t$ may correspond to the prediction error which could be caused by DDoS attack or by some legitimate traffics [34]. On the other hand, the negative $\lambda_t$ refers to the normal behavior. We use these values to distinguish the DDoS attack candidates from the normal ones.

### *3.4. Exponential filter*

The exponential filter is a weighted combination of the previous estimate (output) with the newest input data, with the sum of the weights equal to 1, so that the output matches the input at a steady state. This weight assignment is accomplished by using a smoothing constant $\alpha$ $(0 \leq \alpha \leq 1)$ to calculate the smoothed value $v_t$ at time $t$ as follows:

$$v_t = \alpha \cdot v_{t-1} + (1 - \alpha) \cdot z_t. \tag{2}$$

where $\alpha$ controls the closeness of this new value using the recent observation. The function of $\alpha$ is defined as below:

$$f(n) = \begin{cases} v_t \cong z_t & \alpha \to 0 \\ v_t \cong v_{t-1} & \alpha \to 1 \end{cases}. \tag{3}$$

## 4. DDoS Attack Detection and Defense

In this section, we introduce the DDoS attack detection and defence scheme for SDN. The overall view of the scheme is as illustrated in Figure 2. The proposed method consists of four main modules, namely *Feature Extraction*, *Anomaly Detection for USIP*, *Anomaly Detection for NUDIP* and *DDoS Detection*. The proposed detection algorithm is illustrated in Algorithms 1 to 4. The controller uses the proposed scheme and monitors each switch separately to detect any anomalies related to DDoS attack traffic. First, *Feature Extraction* module extracts corresponding statistical features including USIP and NUDIP from the flow tables of the OpenFlow switch. while the USIP feature is used by the *Anomaly Detection for USIP* module to assign an anomaly score to the sample, the NUDIP feature is employed by *Anomaly Detection for NUDIP* module to release another anomaly score. The *Detection* module decides whether there is an anomaly or not with respect to anomaly scores. In the case of any abnormal conditions, this module raises anomaly alarm and activates the countermeasure part. The symbols used in this work are listed in Table 1

### *4.1. Feature Extraction*

Due to the spoofed IP addresses and random generation of source IP addresses, the number of flow entries with the unique source IP addresses increases during the DDoS attack. Although the number of unique destination IP addresses may not change significantly during the attack compared to the normal condition, the normalized value of this variable with respect to the total number of packets in the flow table decreases [35]. Therefore, in this work, we consider the number of unique source IP addresses and the normalized unique destination IP addresses as the key features for DDoS detection.

Table 1: Definitions of Symbols

| **Symbol** | **Definition** | **Symbol** | **Definition** |
|---|---|---|---|
| $H_S$, $H_D$ | Hash tables for extracting USIP and UDIP | $counter$ | The hash table counter variable |
| $r$ | Number of flow entries in the flow table | $k$ | Number of samples for training the normal models |
| $Model\_X$, $Model\_Y$ | Training flags for USIP and NUDIP | $\mathcal{X} = \{x_1, \ldots, x_k\}$ | Time-series of the number of unique source IP addresses |
| $\mathcal{Y} = \{y_1, \ldots, y_k\}$ | Time series of the number of normalized unique destination IP addresses | $\langle x_t, y_t \rangle$ | USIP and UDIP instances at time interval $t$ |
| $\langle p, d, q \rangle$ | The order of ARIMA, Number of AR ,Differencing and MA terms | $e_t$ | The prediction error of ARIMA at time interval $t$ |
| $\lambda_t$ | Lyapunov exponent at time interval $t$ | $0 \leq \alpha \leq 1$ | Smoothing constant of the exponential filter |
| $f(\alpha)$ | Function of $\alpha$ | $EXP\_f1(\alpha_1), EXP\_f2(\alpha_2)$ | Exponential filters, $\alpha_1$ close to 1, $\alpha_2$ much less than 1 |
| $\mathcal{D}_f$ | Time-series of the distance between the outputs of two exponential filters | $\mathcal{M}$ | Time-series of the Rolling median for $\mathcal{D}_f$ |
| $w$ | Window size of the rolling median | $min\_dist$ | Minimum distance between each element and other elements in $\mathcal{M}$ |
| $\mu_m$ | Mean value of $min\_dist$ | $\sigma_m$ | Standard deviation of $min\_dist$ |
| $q$ | The number of standard deviation, $\sigma_m$ to define the threshold | $th$ | The threshold value |
| $score_{1,t}$, $score_{2,t}$ | Anomaly score for $x_t$ and $y_t$ | $DestIP_{max}$ | The destination IP adddress with the maximum occurrence |

In the feature extraction module, for each specific time interval $t$, we obtain statistics from the flow tables of each switch to extract aforementioned features. The value of time interval $t$ may be chosen according to the corresponding network infrastructure and the traffic volume in the network. As shown in the Algorithm 1, two hash tables[2] namely $H_S$ and $H_D$ are used to find USIP and UDIP, respectively. If a source IP address is not listed in $H_S$, it is added to the table and the corresponding counter is set to 1. On the other hand, if the IP already exists in the table, the counter is incremented by one. The same procedure is applied for the destination IP addresses and the corresponding hash table $H_D$. Once all entries of the flow table are processed, the USIP feature of the sample, $x_t$, is obtained by counting the total number of non-empty elements of the $H_S$; Also, the total number of non-empty elements of $H_D$ are counted as UDIP and then it is divided by the total number of packets in the flow table to obtain normalized UDIP (NUDIP), denoted as $y_t$. These two features are treated as time-series and employed in DDoS detection process independently to inspect the possible instances of DDoS attack.

[2]A data structure that uniquely maps keys for each values.

## *4.2. Anomaly Detection for USIP*

As described in the Algorithm 2, by applying ARIMA and chaos theory on the $x_t$ feature, a binary anomaly score ($score_{1,t}$) is obtained for each time interval $t$.

ARIMA $\langle p, d, q \rangle$ model is employed to predict $x_t$ as $\hat{x_t}$ [36]. In order to estimate the value of $x_t$, the ARIMA model is generated in advance; therefore, the Model_X flag is initially set to 'True' in Algorithm 2. Then, $k$ samples of the USIP are stored in $\mathcal{X}$ to train the model. If $\mathcal{X}$ is a non-stationary time-series, differencing with $d \geq 1$ is applied. If necessary, to stabilize the variance of $\mathcal{X}$, Box-Cox transformation [37] can be used. Akaike's information criterion (AIC),

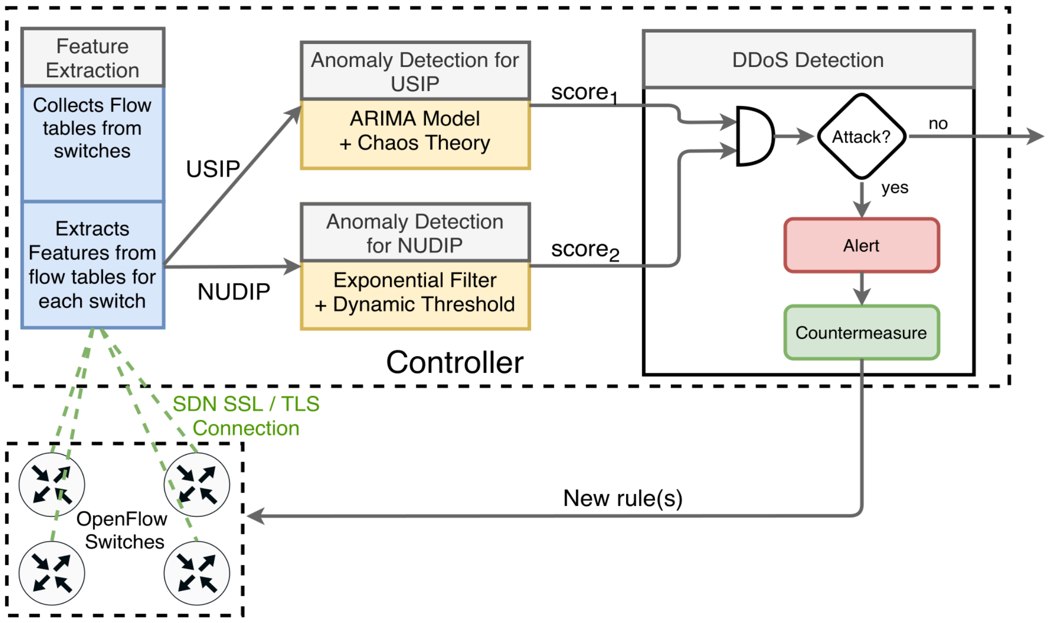
Feature Extraction
Collects Flow tables from switches
Extracts Features from flow tables for each switch
USIP
NUDIP
Anomaly Detection for USIP
ARIMA Model + Chaos Theory
Anomaly Detection for NUDIP
Exponential Filter + Dynamic Threshold
score1
score2
DDoS Detection
Attack?
no
yes
Alert
Countermeasure
Controller
SDN SSL / TLS Connection
OpenFlow Switches
New rule(s)

Figure 2: Proposed DDoS Detection and Defense Model for SDN.

corrected AIC(AICc) and Bayesian information criterion (BIC) may be used for choosing the order of the model. Minimizing such criteria results in obtaining optimal model [38]. We use ARIMA model to generate the normal pattern of USIP; therefore, during the model generation $\mathcal{X}$ should not contain any attack instances [39].

Once the model is generated, the flag, Model_X, is set to 'False' and the training phase is completed. For each upcoming traffic sample, $x_t$, the estimated value, $\hat{x_t}$, is obtained by using the normal model. Then, an anomaly score is given by analyzing the chaotic behaviour of the estimation error. The absolute value of the prediction error, $e_t$ is calculated by:

$$e_t = |x_t - \hat{x_t}|. \quad (4)$$

In some time intervals, the prediction errors differ. Such heterogeneous outcomes might be raised by the anomaly of the USIP instance, $x_t$. Thus, in order to assign anomaly score, Lyapunov exponent, $\lambda_t$ of the error, $e_t$, is calculated and the positive value of the $\lambda_t$ may correspond to anomaly. If the $\lambda_t$ is negative, the denominator error term in Lyapunov equation (Equation 1), $e_0$, is replaced by $e_t$ and the instance is labeled as normal ($score_{1,t} = 0$); otherwise, it is labeled as anomaly ($score_{1,t} = 1$). However, due to the similarity of USIP pattern, some normal traffic samples (i.e., flash crowd events) may be classified as attack candidates that also yields high false positive rate. Therefore, as introduced in the next section, we use NUDIP feature to increase the accuracy.

## *4.3. Anomaly Detection for NUDIP*

The adaptive thresholding method is used to assign another anomaly score ($score_{2,t}$) for each sample. As described in the Algorithm 3, the method uses the $y_t$ feature of the sample. In order to generate the model, $k$ samples of NUDIP are stored in $\mathcal{Y}$ as a time-series. $\mathcal{Y}$ is processed by two exponential filters namely EXP_f1 and EXP_f2. The corresponding smoothing constant of EXP_f1 filter, $\alpha_1$, is chosen as close as possible to 1; hence, the output of this filter follows the trend in the $\mathcal{Y}$. On the other hand, EXP_f2 filter has $\alpha_2$ much less than 1; thus, the out-

**Algorithm 1** Extract Statistic Features (USIP and NUDIP) from the Flow Table

```
1: function FEATURE_EXTRACTION($t$)
2:     $H_D(Des_{IP}, counter) \leftarrow 0$
3:     $H_S(Src_{IP}, counter) \leftarrow 0$
4:     for $\forall(Src_{IP}, Des_{IP}) \in Flow\_T$ do
5:         if $Src_{IP} \notin H_S$ then
6:             $H_S(Src_{IP}, counter) \leftarrow 1$
7:         else
8:             $H_S(Src_{IP}, counter) \leftarrow counter + 1$
9:         end if
10:        if $Des_{IP} \notin H_D$ then
11:            $H_D(Des_{IP}, counter) \leftarrow 1$
12:        else
13:            $H_D(Des_{IP}, counter) \leftarrow counter + 1$
14:        end if
15:    end for
16:    $x_t \leftarrow$ Number of non-empty elements in $H_S$ /*USIP sample*/
17:    $y_t \leftarrow$ Number of non-empty elements in $H_D$ /*UDIP sample*/
18:    $p_t$ /*# of packets */
19:    $y_t \leftarrow y_t/p_t$ /* NUDIP sample*/
20:    return $H_D(Des_{IP}, occurrence)$, $x_t$, $y_t$
21: end function
```

**Algorithm 2** Anomaly Score Computation for Unique Source IP Addresses

```
1: $Model\_X \leftarrow True$
2: $x\_count \leftarrow 0$
3: $\mathcal{X} \leftarrow []$
4: for each time interval $t$ do
5:     if $Model\_X$ then
6:         $\mathcal{X}||x_t$, where $||$ stands for concatenation
7:         $x\_count++$
8:         if $x\_count > k$ then
9:             Estimate $ARIMA_{\langle p,d,q \rangle}$ using $\mathcal{X}$
10:            $Model\_X \leftarrow False$
11:        end if
12:    else
13:        $\hat{x_t} \leftarrow ARIMA_{\langle p,d,q \rangle}(\{x_{t-p-d}, \dots, x_{t-1}\})$
14:        $e_t \leftarrow |x_t - \hat{x}_t|$ /*Prediction error*/
15:        $\lambda_t \leftarrow \frac{1}{t} ln(|\frac{e_t}{e_0}|)$ /*Lyapunov Exponent*/
16:        if $\lambda_t < 0$ then
17:            $e_t$ is normal /*Normal traffic*/
18:            $score_{1,t} \leftarrow 0$
19:            $e_0 \leftarrow e_t$ /*Update $e_0$*/
20:        else
21:            $e_t$ is chaotic /*DDoS candidate*/
22:            $score_{1,t} \leftarrow 1$
23:        end if
24:    end if
25: end for
```

put of the filter follows the change in $\mathcal{Y}$. The absolute difference between outputs of these two filters, $\mathcal{D}_f$ is used for extracting the discriminative feature. The difference is independent from the trend change; therefore, it almost fluctuates around zero, which reflects the dynamic change of the $\mathcal{Y}$. By applying rolling median with the window size of $w$ on the $\mathcal{D}_f$, a median time-series, $\mathcal{M}$, is generated. The minimum distance of each instance in $\mathcal{M}$ from the rest of samples in $\mathcal{M}$ are calculated and recorded in a new set, denoted as $min\_dist$. The mean value, $\mu_m$, and standard deviation, $\sigma_m$, of $min\_dist$ are calculated. Once the aforementioned parameters are calculated, the Model_Y is set to 'False' and anomaly scoring part is activated. For each $y_t$ feature of upcoming traffic sample, using EXP_f1 and EXP_f2, the corresponding absolute difference, $D_{f_t}$ is calculated. The median of $\{D_{f_{t-w}}, \dots, D_{f_t}\}$ is calculated and stored in $m_t$. The distances between $m_t$ and each element in $\mathcal{M}$ is calculated and the minimum distance is obtained as $dist_t$. If $dist_t$ is less than the threshold value $\mu_m + q \times \sigma_m$, the instance is labeled as normal ($score_{2,t} = 0$), otherwise, it is labeled as the abnormal point ($score_{2,t} = 1$). Moreover, if the sample is normal, the first elements of $\mathcal{M}$ and $min\_dist$ are discarded and then $m_t$ and $dist_t$ are appended respectively. As the result, $\mu_m, \sigma_m$ and $th$ are updated.

### *4.4. DDoS Detection, Alert and Countermeasure*

After binary scores for USIP, $score_{1,t}$ and NUDIP, $score_{2,t}$ are computed, these values are used by the DDoS attack detection module. For each traffic sample, if $score_{1,t}$ $AND$ $score_{2,t}$ is equal to 0, then there is no anomaly; Otherwise, the sample is abnormal and there is a DDoS attack to the system. Then the controller raises an attack alarm.

The countermeasure module uses the hash table $H_D$ to identify the destination IP address with the maximum occurrence, so called $DestIP_{max}$. This IP and the corresponding switch ID are added to a discard list. Then controller updates the flow table(s) of the corresponding switch by setting the drop action for each flow entry with $DestIP_{max}$ as the destination IP address.

Although the defense mechanism discards the flow from the DDoS attacker and alleviates the burden on both controller and the switch, it also filters out all packets to the victim's IP address routed through that switch. Therefore, the controller should update the action of the flow entries as soon as the attack is over.

**Algorithm 3** Anomaly Score Computation for Normalized Unique Destination IP Addresses

```
 1: $Model\_Y \leftarrow True$
 2: $y\_count \leftarrow 0$
 3: $\mathcal{Y} \leftarrow []$
 4: for each time interval $t$ do
 5:     if $Model\_Y$ then
 6:         $\mathcal{Y} || y_t$
 7:         $y\_count++$
 8:         if $y\_count > k$ then
 9:             $\mathcal{D}_f \leftarrow [0]$
10:             $\mathcal{M} \leftarrow []$
11:             $min\_dist \leftarrow []$
12:             $l_1 \leftarrow \mathcal{Y}[1]$
13:             $h_1 \leftarrow \mathcal{Y}[1]$
14:             for $\forall y_i \in \mathcal{Y}[2:]$ do
15:                 $l_i \leftarrow \alpha_1 \cdot l_{i-1} + (1-\alpha_1) \cdot y_i$
16:                 $h_i \leftarrow \alpha_2 \cdot h_{i-1} + (1-\alpha_2) \cdot y_i$
17:                 $\mathcal{D}_f || |h_i - l_i|$
18:             end for
19:             for $\forall i \in 1, 2, \ldots, k-w$ do
20:                 $\mathcal{M} || Med(D_{f_i}, \ldots, D_{f_{i+w}})$
21:             end for
22:             for $\forall m_i \in \mathcal{M}$ do
23:                 $min\_dist || min(|m_i - m_j|$ /* where $j \in 1, \ldots, k-w)$ and $j \neq i$*/
24:             end for
25:             $\mu_m \leftarrow Mean(min\_dist)$
26:             $\sigma_m \leftarrow StdDev(min\_dist)$
27:             $th \leftarrow \mu_m + q \times \sigma_m$
28:             $Model\_Y \leftarrow False$
29:         end if
30:     else
31:         $l_t \leftarrow \alpha_1 \cdot l_{t-1} + (1-\alpha_1) \cdot y_t$
32:         $h_t \leftarrow \alpha_2 \cdot h_{i-t} + (1-\alpha_2) \cdot y_t$
33:         $D_{f_t} \leftarrow |h_t - l_t|$
34:         $m_t \leftarrow Med(\{D_{f_{(t-w)}}, \ldots, D_{f_t\}})$
35:         $dist_t \leftarrow min(|m_t - m_i|)$, for all $m_j \in \mathcal{M}$
36:         if $dist_t < th$ then
37:             $score_{2,t} \leftarrow 0$ /*Normal*/ /*Discard the first element of $\mathcal{M}$ and $min\_dist$, then add the corresponding new values*/
38:             $\mathcal{M} \leftarrow \mathcal{M}_2^{k-w} || m_t$
39:             $min\_dist \leftarrow min\_dist_2^{k-w} || dist_t$
40:             $\mu_m \leftarrow Mean(min\_dist)$
41:             $\sigma_m \leftarrow StdDev(min\_dist)$
42:         else
43:             $score_{2,t} \leftarrow 1$ /*Attack*/
44:         end if
45:     end if
46: end for
```

**Algorithm 4** DDoS Detection

```
1: for each time interval $t$ do
2:     if $\neg Model\_X$ & $\neg Model\_Y$ then
3:         if $score_{1,t} \times score_{2,t} == 1$ then
4:             DDoS attack is detected
5:         else
6:             Normal traffic is detected
7:         end if
8:     end if
9: end for
```

Although the countermeasure module takes the discard action for particular flow rules during the attack, it does not delete those flows from the flow table. Therefore, the statistic features remains unchanged. Since the detection module continues to monitor the switch, if $score_{1,t}$ $AND$ $score_{2,t}$ returns from 1 to 0, the attack alarm is cleared. The $DestIP_{max}$ and the corresponding switch ID are removed from the discard list and the action section of the flow entries returns back to the normal state. Furthermore, there would be always some normal samples that are labeled as attack incorrectly. Because the detection module continuously monitors both USIP and NUDIP features, the state of the network will return to its normal in a few intervals.

## 5. Performance Evaluation

In this section, first, we evaluate the performance of the proposed scheme using the complexity analysis and then, we evaluate the detection and countermeasure performance using simulations.

### *5.1. Algorithm Complexity Analysis*

The computational complexity analysis of the proposed algorithms can be divided into two phases, namely normal model generation phase and DDoS attack detection phase.

The feature extraction, Algorithm 1, is applied prior to both the model generation and attack detection phases. For a given flow table with $r$ entries, the statistics features USIP ($x_t$) and NUDIP ($y_t$) are obtained using hash table indexing. Therefore, the complexity of this process is $O(r)$.

The normal model generation phase consists of the training phase for ARIMA, Algorithm 2 and adaptive thresholding method, Algorithm 3, where the former uses $\mathcal{X}$ and the latter uses $\mathcal{Y}$ as the training data that corresponds $r$ flow entries. The complexity of generating each time-series with $k$ elements is $O(r \times k)$.

The training phase of ARIMA contains $\mathcal{X}$ time-series generation and ARIMA modeling. The complexity of ARIMA which deals with the identification of its parameters and model estimation is $O(k^2)$ [40]. As a result, the overall complexity is $O(k^2 + r \times k)$.

Different time-series including $\mathcal{Y}$, $\mathcal{D}_f$, $\mathcal{M}$ and $min\_dist$ are employed in the training phase of the adaptive thresholding method. The $\mathcal{D}_f$ has the complexity of $O(k)$. In order to calculate $\mathcal{M}$ and $min\_dist$, the method of "quicksort" proposed by Tony Hoare is employed [41]. While the $\mathcal{M}$ computation has the average complexity of $O((k-w) \times w)$, with the worst case of $O((k-w) \times w^2)$, the $min\_dist$ has the average complexity of $O((k-w)^2)$ with the worst case of $O((k-w)^3)$. Moreover, both the mean value and standard deviation computation parts have the complexity of $O(k-w)$. The total complexity of the phase is $O(k + (k-w) \times w + (k-w)^2)$ in the best case and $O(k + (k-w) \times w^2 + (k-w)^3)$ in the worst case. The final complexity of the training part is $O(k^2 + r \times k + (k-w) \times w + (k-w)^2)$ in the best case and $O(k^2 + r \times k + (k-w) \times w^2 + (k-w)^3)$ in the worst case. Let assume $w << k$ then, we can consider $(k-w) \approx k$; therefore, the worst complexity is simplified to $O(r \times k + k^3)$.

Finally, the $score_1$ computation has the complexity of $O(k+r)$. Since computation of $score_2$ includes the median, minimum, mean and standard deviation calculations, the worst case computational complexity is $O(w^2 + (k-w)^2)$. Let assume $w << k$, the worst complexity is $O(k^2)$. The complexity of the DDoS detection, Algorithm 4, is $O((k+r) + k^2) \approx O((r + k^2)$.

### *5.2. Dataset*

In order to feed the network with real data, we use real network traffic dataset from MAWI Working Group Traffic Archive [42]. Since 2002, MAWILab has been collecting traffic measurement analysis on the Internet. We use this dataset to generate the normal traffic in our simulation network. We use the Jan 12, 2012 part of the MAWI dataset with the traffic data size of 1.1 $GB$. The MAWI traffic is regenerated using TcpReplay tool [43] and fed to the network. The IP addresses in MAWI dataset are also refashioned according to the host IPs in the experimental network. Therefore, each host in the network contributes to the traffic. In total, almost twelve hours normal and background traffic is regenerated by using TcpReplay. 255-minute (more than four hours) length attack traffic is injected into the normal traffic. The attack starts after 320 minutes that the normal traffic is started.

### *5.3. Simulation Environment*

In order to simulate the network, Mininet emulation environment [6] is used to simulate the network. Due to its design and capabilities, it is appropriate for

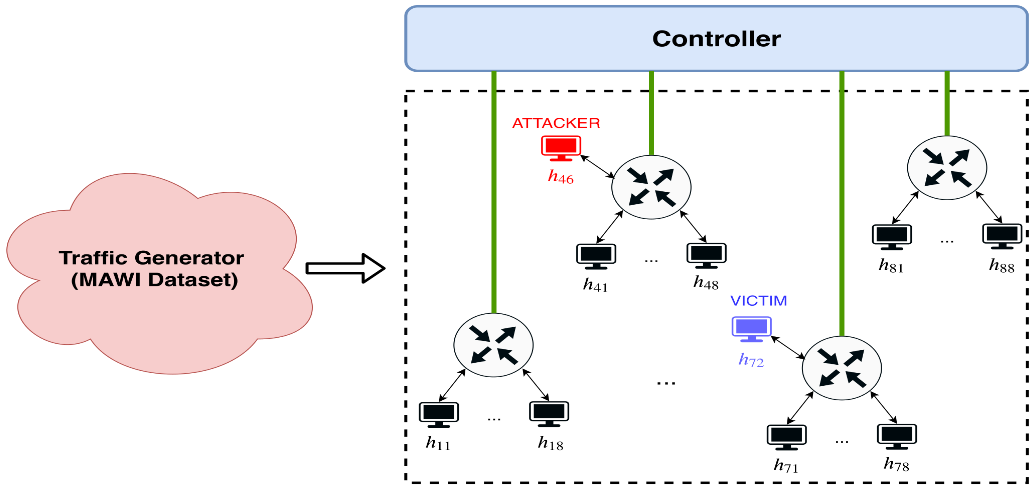

Figure 3: Experimental Topology.

experimenting with SDN. POX controller [44] is run on a PC with 8 GB RAM and Intel Core i7 processor as the SDN controller .

The network topology for the experiment is as shown in Figure 3. There are eight OpenFlow enabled switches, namely $S1, S2, \dots, S8$, in the network. Each switches $Sk$ is connected to eight hosts, $h_{kj}$. We assume that, one of the host of $S4$, $h_{46}$, is infected by a malicious user who is generating attack on a victim which is the host, $h_{72}$ connected to the switch, $S7$. We assume that the attacker uses different compromised hosts on the Internet and then tries to enter to the network through $S4$ to attack the victim [7, 11, 13].

### 5.4. Performance Metrics

In this work, we deal with the problem of the separation attack traffics from normal ones. There are some parameters that should be taken into account while evaluating the performance of detection including true positive ($TP$), true negative ($TN$), false positive ($FP$) and false negative ($FN$). $TP$ and $TN$ are two metrics that reflect the success rate of true detection. While $TP$ shows the number of attack samples detected by the algorithm, $TN$ on the other hand, represents the number of benign samples which are correctly labeled as normal instances. $FP$ and $FN$, in contrast, are used to show the failure of the method in correct detection. $FP$ is related to the number of normal samples incorrectly classified as malicious samples and $FN$ is the number of attack samples that are misplaced as normal.

Using the four previously mentioned parameters, we utilize four metrics including true positive rate ($TPr$), false positive rate ($FPr$), $F1\_score$ and total accuracy ($ACC$). While $F1\_score$ gives a better measure of the incorrectly classified cases, $ACC$ represents the ability of the system in true detection of both normal and DDoS attack. $TPr$, $FPr$, $F1\_score$ and $ACC$ are defined as:

$$TPr = \frac{TP}{TP + FN}, \tag{5}$$

$$FPr = \frac{FP}{FP + TN}, \tag{6}$$

$$F1\_score = \frac{2TP}{2TP + FP + TN}, \tag{7}$$

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}. \tag{8}$$

Furthermore, we employ receiver operating characteristic, ROC, curve to observe the performance of the scheme in separating attack from the normal traffic. The area under the curve of ROC, AUC, represents the degree of separability. The ROC curve is plotted with $TPr$ against the $FPr$ where $TPr$ is on y-axis and $FPr$ is on the x-axis.

### *5.5. Experimental Results*

In this section, we evaluate the performance of the proposed detection and countermeasure scheme via simulation. We present our experimental results with respect to detection and countermeasure performances.

#### *5.5.1. Detection Performance*

In this section, we discuss the detection performance of the proposed scheme, by applying the method on the simulation environment. The controller organizes the network traffic passing through switches. At that point, the feature extraction module collects statistics from flow tables for each switch in every minute and extracts their features. Then, the USIP feature is processed by the first anomaly detection algorithm (Algorithm 2), which employs a time-series approach based on ARIMA and chaos theory to assign an anomaly score to the sample. On the other hand, the NUDIP feature is processed by another anomaly detection algorithm (Algorithm 3) which assigns another anomaly score to the same sample by applying the proposed adaptive thresholding method. Finally, our detection algorithm (Algorithm 4) decides whether the sample is an attack instance or a normal traffic instance. The time interval $t = 1$ minute is selected empirically and by considering the infrastructure of the experimental setup network.

Our scenario is illustrated in Figure 3. The attack is originated from the switch, $S4$; therefore, the following result are based on the analysis applied on the data extracted from $S4$. Totally, we obtain 729 traffic samples in our simulations. First 218 points are used in initializing the normal behavior of the network and the remaining 511 samples are left for test purposes.

Training and test phases of USIP are shown in Figure 4. The trained model is employed to predict the upcoming value for $x_t$. Figure 5 displays the original and the predicted value for the USIP. Because the

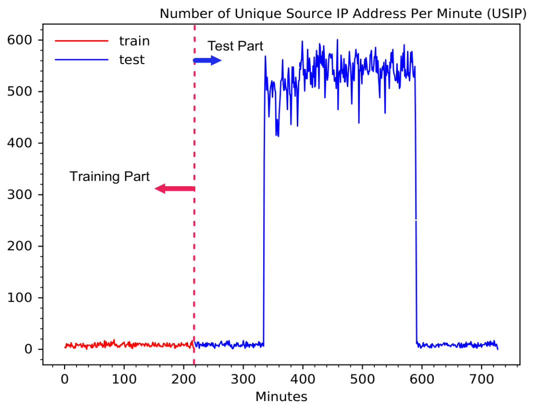

Figure 4: The Number of Unique Source IPs.

ARIMA model is trained based on the normal traffic data, the estimated value diverges from the actual value during the attack. Thus, the prediction error which is shown in Figure 6, is different from the normal condition. The chaotic behaviour of the error is used to differ attack samples from normal traffic. Therefore, the Lyapunov exponent of the error is estimated. The positive value of the Lyapunov value may correspond to attack samples. In order to convert the negative and positive Lyapunov values to scores, we map negative and positive values as 0 and 1, respectively. The Lyapunov values and the corresponding scores are depicted in the Figure 7. The actual DDoS attack instances are labeled by the red small triangles in the figure.

Figure 8 shows the NUDIP time-series, the output of two exponential filters, EXP_f1 and EXP_f2, and their difference. By applying rolling median with the window size of $w = 10$ on the difference, $\mathcal{M}$ is generated and used as the new feature to distinguish between normal and attack traffic. The threshold value, $4 \times \sigma_m$, is empirically selected from the data and by considering Chebyshev's theorem which describes the minimum proportion of the measurements that must lie within one, two, or more standard deviations of the mean [45].

Figure 9 displays the median of difference and the

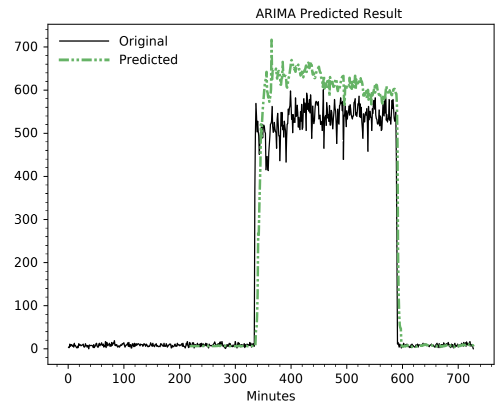

Figure 5: The Original USIP and The Predicted Output By using ARIMA model.

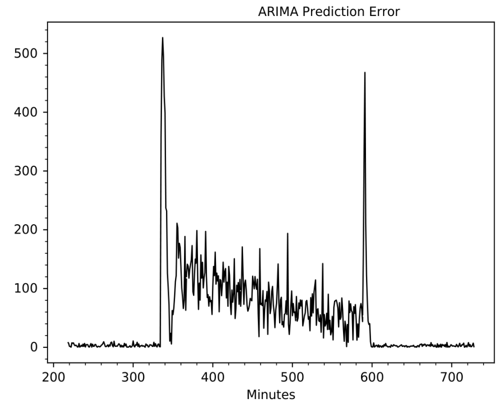

Figure 6: ARIMA Prediction Error.

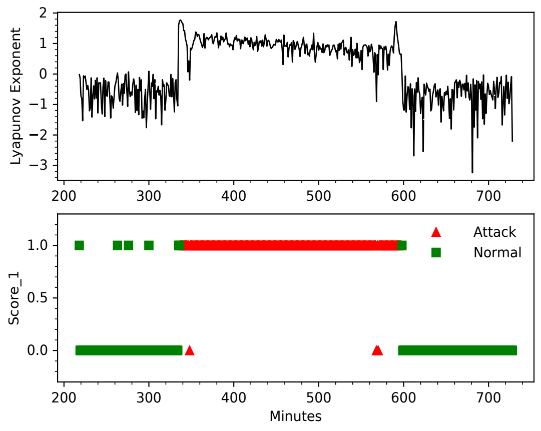

Figure 7: Local Lyapunov Exponent For ARIMA Prediction Error and Corresponding Anomaly Score.

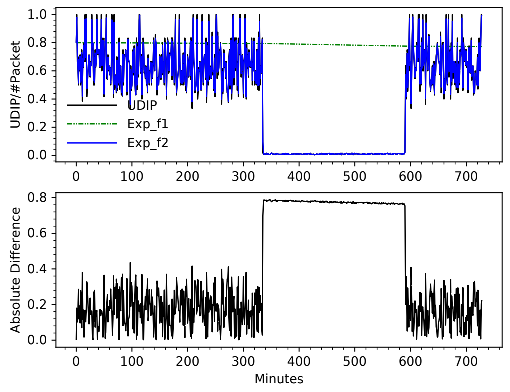

Figure 8: NUDIP Time-series and The Outputs of Filters.

corresponding score for the test part of the NUDIP. During the attack, the median is higher than the normal one. The detection algorithm decides whether the sample is anomaly or not. The final score of the detection algorithm is shown in Figure 10.

The proposed algorithm is applied on each switch $Si$ in the network and corresponding scores, $score_1$ and $score_2$ are obtained. Figure 11 displays the final anomaly score for all switches in the network, obtained by multiplying the corresponding $score_1$ and $score_2$. As shown in the Figure 10 and Figure 11, the controller immediately identifies two engaged switches in the DDoS attack scenario once the attack starts.

Table 2 shows the TPr, FPr, F1_score and ACC of the proposed method for each switch in the network. We just concentrate on the accuracy of $S4$ and $S7$. The detection algorithm performance on both switch $S4$ and $S7$ is the same and equals to 98.82%. Other switches are almost insensitive to the attack. Figure

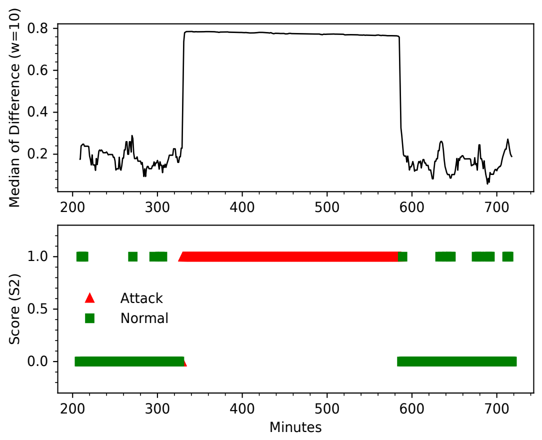

Figure 9: Median of Difference and The $\text{Score}_2$.

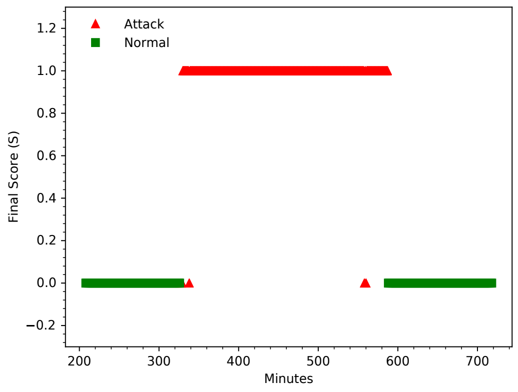

Figure 10: Final Score of the Proposed Method.

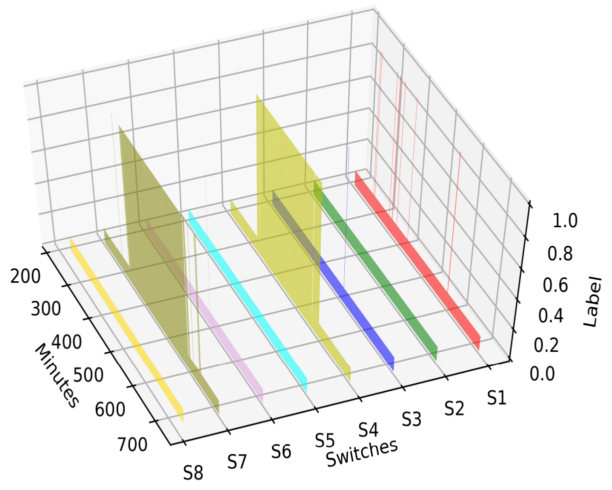

Figure 11: Final Anomaly Scores for All Switches.

Table 2: The Result of DDoS Attack Detection for Different switches

| **Switch** | **TPr(%)** | **FPr(%)** | **F1_score** | **ACC(%)** |
|---|---|---|---|---|
| **S1** | 1.92 | 1.60 | — | — |
| **S2** | 0 | 0 | — | — |
| **S3** | 0.38 | 0 | — | — |
| **S4** | 97.70 | 0 | 0.988 | 98.82 |
| **S5** | 0 | 0 | — | — |
| **S6** | 0.38 | 0 | — | — |
| **S7** | 98.46 | 0.8 | 0.988 | 98.82 |
| **S8** | 0 | 0 | — | — |

12 displays the ROC curves for both switches $S4$ and $S7$. The AUC values for both switches are equal to 0.99.

Because both thresholds are chosen dynamically, the detection performance is better compared to entropy-based detection algorithms which operates with a constant threshold. Furthermore, most of the referenced methods (see Section 2) have been carried out by topologies with just one switch, but in this work, the capability of DDoS detection and countermeasure for multiple switches is also discussed.

### *5.5.2. Countermeasure Performance*

In order to observe the effectiveness of the defense algorithm, we run another simulation on the same experiment. The countermeasure part of the detection module is activated after a while that the attack starts. The activation delay is intentionally added to see the effect of the DDoS attack on the USIP and NUDIP features. The attack is originated from the Switch, $S4$, and the victim resides in the Switch, $S7$. As the attack begins, the anomaly alarms for both mentioned switches are raised. The rules in the flow

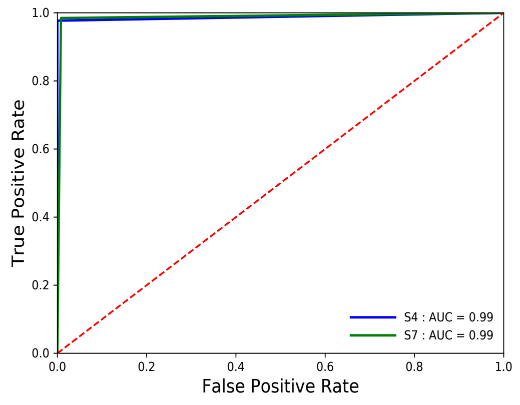

Figure 12: The Receiver Operating Characteristic (ROC).

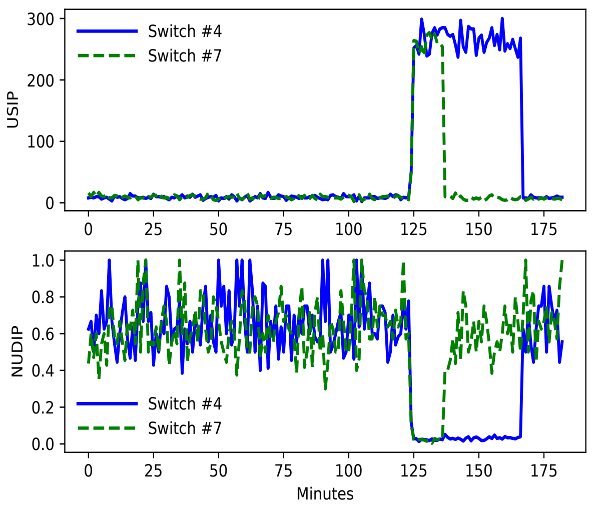

Figure 13: USIP and NUDIP for Switch #4 and Switch #7

tables of both switches are updated to drop all packets with the destination IP addresses as the same as the most occurrence IP address in the corresponding hash tables.

Figure 13 shows the USIP and NUDIP features for both switches. As the attack starts, both features are changed accordingly. Because the attack is originated from $S4$, when the countermeasure is activated, the pattern of both features revert to their normal condition in Switch $S7$. As a result, the anomaly alarm of the switch, $S7$ is cleared. The anomaly patterns in $S4$ does not change until the end of the attack. Once the attack is finished, the controller clears the alarm for the $S4$ and routing process of the switch is returned to its normal state.

The result shows that the proposed countermeasure scheme can be considered as an effective mechanism to mitigate DDoS attacks. Therefore, when an attack occurs, as soon as the involved switches are detected by the controller, the rules of the flow table entries are updated. As a result, the attack traffic that threatens involved switches in the network is isolated to protect the availability of the network.

## 6. Conclusion

In this paper, we propose a DDoS attack detection and countermeasure scheme based on time-series analysis for SDN. The proposed scheme monitors each OpenFlow switch individually to find any anomalies caused by DDoS attacks. For this purpose, four modules including *Feature Extraction*, *Anomaly Detection for USIP*, *Anomaly Detection for NUDIP* and *DDoS Detection*, are added to the SDN controller. In this work, our scheme monitors the traffic features to distinguish DDoS attack instances. For this purpose, we consider both the number of unique source IP addresses and the normalized unique destination IP addresses. In Anomaly Score Computation for USIP algorithm, we employ the ARIMA forecasting error and chaos theory. For Anomaly Score Computation for NUDIP algorithm, we adopt the dynamic threshold method. Then, the results obtained from these two algorithms are used for labeling each traffic sample as normal or DDoS traffic. Moreover, if any DDoS attack instances are detected in a switch, the DDoS attack alarm is raised for that switch and the countermeasure module updates the flow table of the switch accordingly to prevent the DDoS attack. In order to assess the performance of DDoS detection, we implement our scheme on an example SDN network by using Mininet environment [6]. The simulation results show that our scheme detects anomalies and affected switches in the network with a high accuracy (98.82%).

## References

[1] R. Sahay, W. Meng, C. D. Jensen, The application of software defined networking on securing computer networks: A survey, Journal of Network and Computer Applications 131 (2019) 89–108.

[2] J. C. C. Chica, J. C. Imbachi, J. F. Botero, Security in sdn: A comprehensive survey, Journal of Network and Computer Applications (2020) 102595.

[3] T. Ubale, A. K. Jain, Survey on ddos attack techniques and solutions in software-defined network, in: Handbook of Computer Networks and Cyber Security, Springer, 2020, pp. 389–419.

[4] S. Gupta, B. B. Gupta, Detection, avoidance, and attack pattern mechanisms in modern web application vulnerabilities: present and future challenges, International Journal of Cloud Applications and Computing (IJCAC) 7 (3) (2017) 1–43.

[5] N. Z. Bawany, J. A. Shamsi, K. Salah, DDoS attack detection and mitigation using sdn: methods, practices, and solutions, Arabian Journal for Science and Engineering 42 (2) (2017) 425–441.

[6] J. Yan, D. Jin, Vt-mininet: Virtual-time-enabled mininet for scalable and accurate software-define network emulation, in: Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research, 2015, pp. 27:1–27:7.

[7] K. Kalkan, L. Altay, G. Gür, F. Alagöz, Jess: Joint entropy-based DDoS defense scheme in SDN, IEEE Journal on Selected Areas in Communications 36 (10) (2018) 2358–2372.

[8] W. Lee, D. Xiang, Information-theoretic measures for anomaly detection, in: Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001, IEEE, 2000, pp. 130–143.

[9] P. Bereziński, M. Szpyrka, B. Jasiul, M. Mazur, Network anomaly detection using parameterized entropy, in: IFIP International Conference on Computer Information Systems and Industrial Management, Springer, 2015, pp. 465–478.

[10] S. Yu, W. Zhou, R. Doss, W. Jia, Traceback of ddos attacks using entropy variations, IEEE Transactions on Parallel and Distributed Systems 22 (3) (2010) 412–425.

[11] S. M. Mousavi, M. St-Hilaire, Early detection of ddos attacks against sdn controllers, in: 2015 International Conference on Computing, Networking and Communications (ICNC), IEEE, 2015, pp. 77–81.

[12] H. Bozdogan, Akaike's information criterion and recent developments in information complexity, Journal of mathematical psychology 44 (1) (2000) 62–91.

[13] K. S. Sahoo, D. Puthal, M. Tiwary, J. J. Rodrigues, B. Sahoo, R. Dash, An early detection of low rate ddos attack to sdn based data center networks using information distance metrics, Future Generation Computer Systems 89 (2018) 685–697.

[14] R. Braga, E. de Souza Mota, A. Passito, Lightweight ddos flooding attack detection using nox/openflow., in: LCN, Vol. 10, 2010, pp. 408–415.

[15] T. Kohonen, The self-organizing map, Proceedings of the IEEE 78 (9) (1990) 1464–1480.

[16] A. S. da Silva, J. A. Wickboldt, L. Z. Granville, A. Schaeffer-Filho, Atlantic: A framework for anomaly traffic detection, classification, and mitigation in sdn, in: NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium, IEEE, 2016, pp. 27–35.

[17] N. Meti, D. Narayan, V. Baligar, Detection of distributed denial of service attacks using machine learning algorithms in software defined

networks, in: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, 2017, pp. 1366–1371.

[18] B. Scholkopf, A. J. Smola, Learning with kernels: support vector machines, regularization, optimization, and beyond, MIT press, 2001.

[19] T. M. Nam, P. H. Phong, T. D. Khoa, T. T. Huong, P. N. Nam, N. H. Thanh, L. X. Thang, P. A. Tuan, V. D. Loi, et al., Self-organizing map-based approaches in ddos flooding detection using sdn, in: 2018 International Conference on Information Networking (ICOIN), IEEE, 2018, pp. 249–254.

[20] T. V. Phan, T. Van Toan, D. Van Tuyen, T. T. Huong, N. H. Thanh, Openflowsia: An optimized protection scheme for software-defined networks from flooding attacks, in: 2016 IEEE Sixth International Conference on Communications and Electronics (ICCE), IEEE, 2016, pp. 13–18.

[21] J. Cui, M. Wang, Y. Luo, H. Zhong, Ddos detection and defense mechanism based on cognitive-inspired computing in sdn, Future Generation Computer Systems 97 (2019) 275–283.

[22] Y. Wang, T. Hu, G. Tang, J. Xie, J. Lu, Sgs: Safe-guard scheme for protecting control plane against ddos attacks in software-defined networking, IEEE Access 7 (2019) 34699–34710.

[23] H. Polat, O. Polat, A. Cetin, Detecting ddos attacks in software-defined networks through feature selection methods and machine learning models, Sustainability 12 (3) (2020) 1035.

[24] N. Bhatia, et al., Survey of nearest neighbor techniques, arXiv preprint arXiv:1007.0085.

[25] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, H. Arshad, State-of-the-art in artificial neural network applications: A survey, Heliyon 4 (11) (2018) e00938.

[26] S. Umadevi, K. J. Marseline, A survey on data mining classification algorithms, in: 2017 International Conference on Signal Processing and Communication (ICSPC), IEEE, 2017, pp. 264–268.

[27] Q. Niyaz, W. Sun, A. Y. Javaid, A deep learning based ddos detection system in software-defined networking (sdn), arXiv preprint arXiv:1611.07400.

[28] C. Li, Y. Wu, X. Yuan, Z. Sun, W. Wang, X. Li, L. Gong, Detection and defense of ddos attack–based on deep learning in openflow-based sdn, International Journal of Communication Systems 31 (5) (2018) e3497.

[29] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, R. X. Gao, Deep learning and its applications to machine health monitoring, Mechanical Systems and Signal Processing 115 (2019) 213–237.

[30] R. U. Rasool, U. Ashraf, K. Ahmed, H. Wang, W. Rafique, Z. Anwar, Cyberpulse: A machine learning based link flooding attack mitigation system for software defined networks, IEEE Access 7 (2019) 34885–34899.

[31] S. Haider, A. Akhunzada, I. Mustafa, T. B. Patel, A. Fernandez, K.-K. R. Choo, J. Iqbal, A deep cnn ensemble framework for efficient ddos attack detection in software defined networks, Ieee Access 8 (2020) 53972–53983.

[32] N. Perraudin, P. Vandergheynst, Stationary signal processing on graphs, IEEE Transactions on Signal Processing 65 (13) (2017) 3462–3477.

[33] M. T. Rosenstein, J. J. Collins, C. J. De Luca, A practical method for calculating largest Lyapunov exponents from small data sets, Physica D: Nonlinear Phenomena 65 (1-2) (1993) 117–134.

[34] S. M. T. Nezhad, M. Nazari, E. A. Gharavol, A novel DoS and DDoS attacks detection algorithm using ARIMA time series model and chaotic system in computer networks., IEEE Communications Letters 20 (4) (2016) 700–703.

[35] R. C. Baishya, N. Hoque, D. K. Bhattacharyya, DDoS attack detection using unique source IP deviation., IJ Network Security 19 (6) (2017) 929–939.

[36] D. C. Montgomery, C. L. Jennings, M. Kulahci, Introduction to time series analysis and forecasting, Wiley series in probability and statistics, 2015.

[37] P. Legendre, D. Borcard, Box–cox-chord transformations for community composition data prior to beta diversity analysis, Ecography 41 (11) (2018) 1820–1824.

[38] G. Jain, B. Mallick, A study of time series models arima and ets, Available at SSRN 2898968.

[39] M. Thottan, C. Ji, Anomaly detection in ip networks, IEEE Transactions on signal processing 51 (8) (2003) 2191–2204.

[40] E. H. Pena, L. F. Carvalho, S. Barbon Jr, J. J. Rodrigues, M. L. Proença Jr, Anomaly detection using the correlational paraconsistent machine with digital signatures of network segment, Information Sciences 420 (2017) 313–328.

[41] Y. Yao, Using nonlinear difference equations to study quicksort algorithms, Journal of Difference Equations and Applications 26 (2) (2020) 275–294.

[42] M. W. Group, et al., Mawi working group traffic archive (2018).

[43] P. E. TCPReplay, Replay tools for* nix (2016).

[44] S. Kaur, J. Singh, N. S. Ghumman, Network programmability using POX controller, in: ICCCS International Conference on Communication, Computing & Systems, IEEE, Vol. 138, 2014.

[45] B. G. Amidan, T. A. Ferryman, S. K. Cooley, Data outlier detection using the chebyshev theorem, in: 2005 IEEE Aerospace Conference, 2005, pp. 3814–3819.