A Moving Window Principal Components Analysis Based Anomaly Detection and Mitigation Approach in SDN Network

Mingxin Wang[1], Huachun Zhou[1] and Jia Chen[1]
[1] National Laboratory of Next Generation Internet Interconnection Devices,
School of Electronic and Infomation Engineering, Beijing Jiaotong University
Beijing 100044, China
[e-mail: {14111007, hchzhou, chenjia}@bjtu.edu.cn]
*Corresponding author: Mingxin Wang

Received July 9, 2017; revised March 1, 2018; accepted March 19, 2018; published August 31, 2018

Abstract

Network anomaly detection in Software Defined Networking, especially the detection of DDoS attack, has been given great attention in recent years. It is convenient to build the Traffic Matrix from a global view in SDN. However, the monitoring and management of high-volume feature-rich traffic in large networks brings significant challenges. In this paper, we propose a moving window Principal Components Analysis based anomaly detection and mitigation approach to map data onto a low-dimensional subspace and keep monitoring the network state in real-time. Once the anomaly is detected, the controller will install the defense flow table rules onto the corresponding data plane switches to mitigate the attack. Furthermore, we evaluate our approach with experiments. The Receiver Operating Characteristic curves show that our approach performs well in both detection probability and false alarm probability compared with the entropy-based approach. In addition, the mitigation effect is impressive that our approach can prevent most of the attacking traffic. At last, we evaluate the overhead of the system, including the detection delay and utilization of CPU, which is not excessive. Our anomaly detection approach is lightweight and effective.

Keywords: Software Defined Networking, Principal Components Analysis, anomaly detection, mitigation

This paper is supported by NSAF under Grant No. U1530118, NSFC of China under Grant No. 61471029, 61271202 and 61232017 and Fundamental Research Funds for the Central Universities under Grant No. 2015YJS028 and W17JB00180.

http://doi.org/10.3837/tiis.2018.08.022 **ISSN : 1976-7277**

1. Introduction

Network security is always a critical issue along with the development of network. One of the major threats to cyber security is the Distributed Denial-of-Service (DDoS) attack. It is a kind of network anomaly behavior which aims to consume the resources of network devices. DDoS attacks are initiated and continued by large number of attackers grouped and distributed globally who start populating the unwanted traffic packets in order to acquire the memory, network resources and completely deplete them. Nevertheless, when the attack happens, the network traffic pattern must have changed. Some features of traffic, such as the traffic volume, the entropy of source and destination address or port will reflect the network state. We can detect the network abnormal behavior by analyzing the features of the network traffic flows. However, it requires a close coordination of many network components to defend against an attack. Conventional routers are difficult to modify their behavior. In addition, it is hard for us to obtain an overall view of the network state because of the limitation imposed by traditional IP network.

Software-Defined Networking (SDN) [1] is a novel network architecture, which decouples the control plane from the forwarding plane. The control logic is centralized into the control plane controller while the data plane devices simply forward the traffic according to the flow table rules. Furthermore, SDN provides a simple and robust way to access all layers of network traffic management via a simple interface. The feather of central controlling makes it easier to detect the DDoS attack and alleviate the impact of it. OpenFlow [2] is an implementation of SDN that can natively achieve basic flow features, such as packet and byte count of per flow from each switch port. OpenNetMon [3] is designed as a network parameter monitoring module in the POX [4] controller platform. FlowSense [5] is another SDN monitoring module, which can get the flow statistics information from the SDN switches. It is convenient for us to analyze the flow statistics features and build the Traffic Matrix. Moreover, we can take advantage of these features to classify different types of traffic flows from which we can detect the type of anomalous traffic with ease.

Considering the limitation of traditional IP network as well as the new features of SDN network, we try to solve the anomaly detection problem in SDN. However, the monitoring and management of high-volume feature-rich traffic in large networks brings significant challenges in either storage, transmission or computational costs to both the switch and the controller. Motivated by these issues, we introduce Principal Components Analysis (PCA) into SDN network.

In this paper, we propose a moving window Principal Components Analysis based DDoS attack detection and mitigation approach in SDN network. In SDN network, the controller can acquire the traffic statistics information from data plane. With the statistics information, we can build the Traffic Matrix as a measurement of origin-destination (OD) traffic intensity between nodes. In addition, considering the correlation of different traffic features such as source address, destination address, source port, and destination port, we can build the Traffic Matrix (TM) of entropy for each of these traffic features. Considering both of these factors, we further use PCA to reduce the data dimensionality and transpose the network traffic into normal subspace and anomalous subspace. In this case, the controller can detect the anomalous traffic in real time. After detecting the anomaly traffic, we can immediately trigger the mitigation mechanism to install defense flow table rules onto data plane switches to prevent the further attack. At last, we simulate an experiment in Mininet [6] to evaluate the

detecting effect from two aspects, detection probability (D_p) and false alarm probability (F_p). We compare our approach with the entropy-based approach by means of the Receiver Operating Characteristic (ROC) curves. Additionally, we simulate an experiment to evaluate the effect of mitigation mechanism. We also estimate the detecting delay and system overhead of our approach.

The contributions of this paper are: (i) we develop a SDN network state monitoring and anomaly traffic detecting and mitigating component in the POX SDN controller. (ii) we build the Traffic Matrix by combining traffic volume with entropy of traffic features and propose a moving window PCA based approach to detect the anomalous traffic. (iii) we evaluate our proposed approach by the experiment run on Mininet.

The remaining of this paper is organized as follows. In Section 2, we present background and review related work. In Section 3, we introduce our anomaly detection approach in detail and present the system design. In Section 4, we simulate an experiment with both benign traffic and DDoS attack to evaluate our approach and give some associated results, including a performance analysis of the system. At last, in Section 5, we make a conclusion of this paper and present our further work.

2. Related Work and Background

2.1 Related Work

Research in network anomaly detection has been greatly concerned. Many researches focus on detecting anomalous traffic by measuring the traffic volume offline (such as flow numbers, packet counts and bytes) [7] [8]. However, not all anomaly will cause the traffic volume changing dramatically.

As the limitation of traditional IP network, many researches try to detect the anomaly or the DDoS attack based on the traffic on single link or single point [9] which take advantage of the characteristic of time to discover the anomaly. However, many anomalous behaviors will cause minute change on single link or single point. There are some methods to detect DDoS attacks and protect the networks in conventional networks. However, it does not offer very reliable and flexible defense solutions. Related study [7] indicates that almost all the anomaly behavior will lead to the distribution of IP address and port change. Information entropy theory has also been used in traditional networks for anomaly traffic detection [10]. Entropy of various flow features can be used to model a high-level view of the flows observed in the network. It is possible to detect deviations that indicate an anomaly by analyzing the entropy information within a time interval.

Recent researches have indicated that SDN is a promising way to implement network monitoring and anomaly detection schemes. Due to the programmable features and reconfigurable nature of SDN, flexible and robust approaches can be deployed to detect and prevent DDoS attacks. [10] combines OpenFlow and sFlow to implement a network-wide anomaly detection and mitigation mechanism. However, as the implementation bases on sampling via sFlow, false alarm probability was quite high in attack detection. [11] proposes an entropy-based early and fast DDoS attack detecting approach in SDN. However, the setting of the threshold is empirical which is based on the experiment. [12] proposes a lightweight DDoS flooding attack detection method using neural networks technique and classifies anomaly traffic by means of self-organizing map (SOM). In [20], Lim et al. proposed a DDoS blocking application (DBA) using SDN to block legitimately looking DDoS attacks efficiently. The system works in collaboration with the targeted servers for attack detection. The prototype

is demonstrated to detect HyperText Transfer Protocol (HTTP) flooding attack. In [21], Wang et al. proposed an entropy based light-weight DDoS detection system by exporting the flow statistics process to switches. Although the approach reduces the overhead of flow statistics collection in the controller, it attempts to bring back the intelligence in network devices. [22] proposes a deep learning based multi-vector DDoS detection system in SDN environment. It use deep learning for feature reduction of a large set of features derived from network traffic headers. However, the system has some limitations in terms of processing capabilities. A graphical approach is proposed in [23] that relies on OpenFlow based switches to trace-back the origin of attacks where all paths that are vulnerable to anomaly attacks can be determined. In [24], a Remote Triggered Black Hole (RBTH) approach is proposed for the prevention of DDoS attacks in SDN. The controller plays a major role in detecting the malicious traffic between OpenFlow switches and discarding them to prevent further damage to the network.

SDN technology gives us a privilege to configure the devices to serve our need in anomaly detection. However, in large-scale SDN network, such as large-scale ISP, there usually exists many Border Gateway Protocol (BGP) routers. The monitoring and management of high-volume feature-rich traffic in large networks brings significant challenges in either storage, transmission or computational costs to both the switch and the controller. We treat the traffic measurement between every two BGP routers as an input vector $\boldsymbol{X}_i$. $\boldsymbol{X}_i$ is the ith sample belong to higher dimensional space $\boldsymbol{R}^p, \left(p = n^2 \right)$, where n is the number of BGP routers. Thus, we must try to reduce dimensions of the input vectors.

PCA can reduce the amount of dimensions required to classify the given set of data points and produces a set of principal components, which are orthogonal eigenvectors [13]. PCA can be used to transpose the network traffic into normal subspace and anomalous subspace [14]. [15] introduces an architecture to collect, extend, and select flow features for traffic classification in OpenFlow-based networks by PCA which can be used to classify different types of traffic flows. However, the work focus on traffic classification but not anomaly detection. Our goal is to adopt PCA approach in SDN network to exploit advantages from both sides. As far as we known, we are the first one using PCA to detect and mitigate anomaly in SDN network.

In this section, we will firstly briefly introduce the definition of Traffic Matrix and the concept of sample entropy. We will also introduce the background knowledge of Principal Components Analysis Method. Then in section 3.2, we will present the method of building Traffic Matrix based on SDN in detail.

2.2 Traffic Matrix and Sample Entropy

In order to describe the overview network state, we need to build the Traffic Matrix based on the global traffic information.

Firstly, we give the definition of Traffic Matrix. Traffic Matrix describes the origin-destination traffic intensity between nodes in the network. In terms of the type and scale of the network nodes, we can classify the Traffic Matrix into link level Traffic Matrix, routing level Traffic Matrix and Points of Presence (PoPs) level Traffic Matrix. In this paper, we focus on routing level Traffic Matrix and PoPs level Traffic Matrix. Abilene [19] is the Internet2 backbone network, connecting over 200 US universities and peering with research networks in Europe and Asia. It consists of 11 PoPs, spanning the continental US.

Assume that there are n BGP routers in a backbone network. We can measure the traffic of each OD pair at a certain interval for a period. Afterwards, we array these measured value into a Traffic Matrix with $t \times p$, where $p = n^2$ and t is the number of the samples. Each sample

consists of p observations. x_{ij} indicates the traffic features of jth OD pair in time interval i. The traffic features can include source address, destination address, source port, destination port and so on. In this paper, we consider both the traffic volume and the entropy of these traffic features.

OpenFlow's ability of per-flow statistics has already been applied by some researches for building Traffic Matrix. OpenTM [18] estimates a Traffic Matrix by keeping track of statistics for each flow. The application in controller queries switches on regular intervals and stores statistics in order to derive the Traffic Matrix. OpenNetMon [3] continuously monitors all flows between predefined link-destination pairs on throughput, packet loss and delay. It is essential for traffic engineering purposes and is beneficial for building Traffic Matrix.

Sample entropy can capture the degree of dispersal or concentration of a distribution. Assuming that S is the total number of samples in an observation cycle where $\mathrm{S} = \sum_{i=1}^{N} n_i$. N is the occurrence number of traffic feature Y where Y_i occurs n_i times. Then the entropy (H) of this traffic feature will be calculated as:

$$\mathrm{H}(\mathrm{Y}) = -\sum_{i=1}^{N}\left(\frac{n_i}{S}\right)\log_2\left(\frac{n_i}{S}\right) \tag{1}$$

The value of sample entropy lies in the range (0: $\log_2 N$). The metric takes on the value 0 when the distribution is maximally concentrated, *i.e.*, all observations are the same. Sample entropy takes on the value $\log_2 N$ when the distribution is maximally dispersed (*i.e.*, $n_1 = n_2 = \cdots = n_N$).

A notable characteristic of entropy variation during a DDoS attack is that the entropy of source address increases while the one of destination address decreases.

According to the concept of sample entropy, we can build the entropy matrix, which is a specific form of Traffic Matrix. In an entropy matrix, each element represent the entropy of certain traffic feature from source to destination. The detail of construction process will be introduced in 3.2. Taking the Traffic Matrix as the input, we measure it by means of PCA to detect the anomaly traffic, which will be introduced in section 3.3.

2.3 Principal Components Analysis Method

Principal Components Analysis is a descriptive statistical method belonging to the factorial category, which is a linear transformation that transforms the multi-dimensional dataset into new coordinates. The new coordinates are the principal components of the original data set. The first principal component characterizes the largest variance of the input data while the second principal component represents the largest variance of the residual data after removing the first principal component. The other principal components obtain the maximum variance within the remaining data while all these principal components are orthogonal. As a result, all the principal components are sorted by the amount of data variance in descending order.

Given a Traffic Matrix $\boldsymbol{X}$ that consists of T zero-mean observations. Each row of $\boldsymbol{X}$ is a p dimensional variables $\boldsymbol{X}_i \in \boldsymbol{R}^p$ (i = 1, 2, ···, T). The PCA explains the variance-covariance structure of the set of variables through a few new variables. The new variables are called as principal components, which are foundations of the original variables. The principal components of the dataset are the eigenvectors v_i of the input matrix's covariance matrix **C**.

These eigenvectors are aggregated into the matrix $\boldsymbol{U} \in \boldsymbol{R}^{n \times n}$ in the order of increasing component variance. We use the calculated principal components to build the normal subspace and anomalous subspace.

We firstly make spectral decomposition for the covariance matrix **C**:

$$\boldsymbol{C} = \frac{1}{T}\sum_{i=1}^{T}\left(\boldsymbol{X}_i - \mu\right)\left(\boldsymbol{X}_i - \mu\right)^{\mathrm{T}} \tag{2}$$

$$\boldsymbol{C}\boldsymbol{U} = \boldsymbol{U}\boldsymbol{\Lambda} \tag{3}$$

where $\mu = \frac{1}{T}\sum_{i=1}^{T}\boldsymbol{X}_i$ represents the mean vector of input vector $\boldsymbol{X}_i$. $\boldsymbol{U}$ represents the eigenvector matrix that is constituted with eigenvector $\boldsymbol{u}_i$. $\boldsymbol{\Lambda}$ represents the eigenvalues matrix where the eigenvalues are $\{\lambda_j\}$, (j = 1, 2, …, *p*). We select the first *k* dominant PCs constructing the normal subspace $\hat{S}$ and the remaining *p-k* PCs construct the anomalous subspace $\tilde{S}$, which is assumed as the anomalous subspace.

To project the raw input data $\boldsymbol{X}_i$ onto the normal subspace $\hat{S}$, we can acquire the normal subspace projection vector $\hat{\boldsymbol{a}}_i$ where:

$$\hat{\boldsymbol{a}}_i = \left(\boldsymbol{X}_i - \mu\right)\boldsymbol{U}_k \tag{4}$$

Vice versa, we can get the anomalous subspace projection vector $\tilde{\boldsymbol{a}}_i$ where:

$$\tilde{\boldsymbol{a}}_i = \left(\boldsymbol{X}_i - \mu\right)\boldsymbol{U}_{n-k} \tag{5}$$

$\boldsymbol{U}_k$ represents the eigenvectors which is constructed by the first k eigenvalues while $\boldsymbol{U}_{n-k}$ is the eigenvectors constructed by the last n-k eigenvalues.

Let *s* represent the square of residual vector's 2-norm where:

$$s = \left\|\tilde{\boldsymbol{a}}_i\right\|_2^2 \tag{6}$$

We call *s* as Square Prediction Error (SPE), which can measure the errors between the raw input vectors and the reconstruction input vectors.

3. Moving Window PCA based Anomaly Detection and Mitigation Approach

In this section, we will introduce our moving window PCA based anomaly detection and mitigation approach. We develop an anomaly detection and mitigation component as an application runs on the SDN controller. At first, we will give the system module design in detail. After have acquired the flow feature statistics from the data plane switches, we will then build the Traffic Matrix as the input of our anomaly detection algorithm. In addition, we will introduce the workflow of moving window PCA based anomaly detection and mitigation algorithm. The output of the algorithm reflects whether there is anomaly traffic in the network. As soon as the anomaly is detected, the mitigation procedure will be triggered to alleviate the impact of the attack.

3.1 System Module Design

In this part, we propose an anomaly detection and mitigation component as an application runs

on the SDN controller. The component is responsible for monitoring the network state, collecting and analyzing flow features, further detecting the abnormal network behavior and mitigating the anomaly in real time. We take advantage of the native statistic ability of OpenFlow protocol to acquire flow table entries and port statistics information from data plane switches. The component comprises 3 main modules: (1) Flow Feature Requesting module, (2) Flow Statistic Analysis module, (3) Anomaly Detection and Mitigation module. The architecture of the system is shown in **Fig. 1**.

Flow Feature Requesting module is responsible for sending flow feature request messages to all the data plane switches periodically. The module can control the rate of flow feature request messages. The time interval of the feature request messages has a bearing on the accuracy and timeliness of anomaly detection. The longer the interval of feature request messages is, the longer the anomaly detection time will be. However, if the interval is too short, the overhead of the controller and switch will increase. Also, the accuracy will decrease.

Flow Statistic Analysis module is responsible for collecting native flow features and port statistics information, such as flow byte and packet counters of each flow entries, from data plane switches and further storing them in the data structure. The counters of per flow contains three fields: packet numbers, byte and duration of the flow. Flow Statistic Analysis module can parse the event messages collected in the certain cycle and build the Traffic Matrix of traffic volume or entropy according to the statistic information and topology.

Anomaly Detection and Mitigation module runs the anomaly detection algorithm, which is responsible for handling the Traffic Matrix and judging whether there is anomaly traffic in the network according to the Q statistic value threshold [17]. In our design, we use Principal Component Analysis technique to reduce dimensions of input vectors and divide them into normal subspace and abnormal subspace which will be introduced explicitly in Section IV. Once the anomaly traffic is detected, we can locate the source-destination pair of the anomaly flows with the aid of OpenFlow's flow based forwarding mechanism. Furthermore, we can install the defense flow table rules on the corresponding switches to drop the anomaly flows.

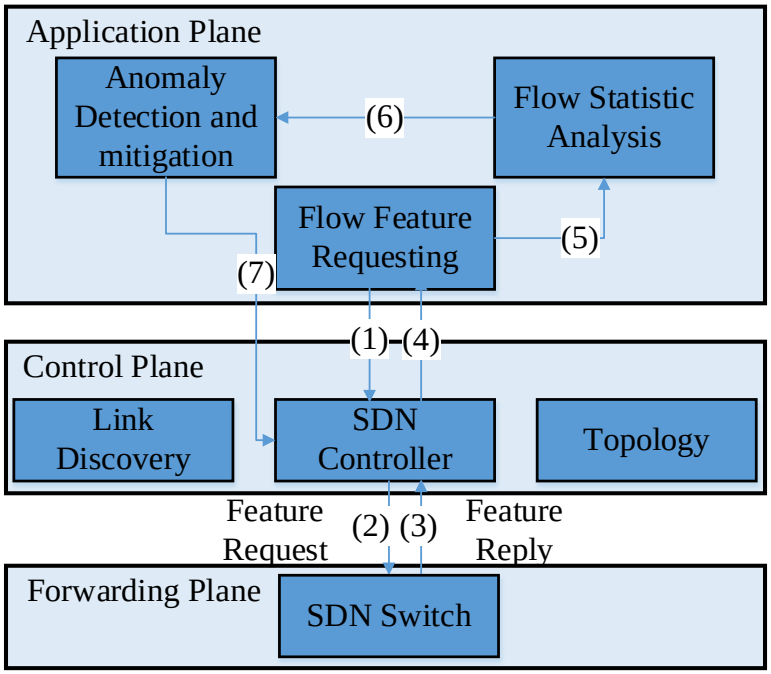

Fig. 1. Architecture Design of Anomaly Detection System

3.2 The Construction of Traffic Volume Matrix and Entropy Matrix

In our design, we need to build the Traffic Matrix as the input of anomaly detection and mitigation algorithm. Flow Feature Requesting module will periodically send flow feature request messages to all the switches on regular intervals. Then we can store the replay messages in the certain cycle and analyze the statistics to build the Traffic Matrix.

The Flow Statistic Analysis module can collect the response messages from the data plane switches. As POX is an event triggered based SDN controller, each switch's response message will trigger the *flow_stats_handle* function to count the flow table entries on the certain switch and store them in the dictionary data structure. After aggregating all the response messages from all the switches, the module will calculate the traffic volume and entropy from one switch to another and further build the Traffic Matrix.

The traffic volume matrix reflects the amount of traffic from one switch's subnetwork to another switch's subnetwork. The entropy matrix reflects the information of the certain flow feature from one switch's subnetwork to another switch's subnetwork.

For instance, when building the source address entropy matrix, for each switch pair *ij*, we will look up every flow table entry on switch *i* to find out the entries which source address is behind this switch *i* and their destination address is behind switch *j*. We will classify these flow table entries. Then, we will analyze each entry in these entries to calculate the entropy of their source address as an element of current sample of the source address entropy matrix.

In our design, Traffic Matrix as the input of the anomaly detection algorithm is not the key point of the paper. We assume that the subnetwork of each switch is already known by us. So that we can easily build the Traffic Matrix from one switch to another switch by means of the source and destination address field. Furthermore, the Traffic Matrix is constructed by 10 continues samples in our design. When the window (number of continue samples) is too long, large fluctuations caused by the dynamics of network traffic is regarded as abnormal. However, if the window is too short, the accuracy will be affected.

3.3 Moving Window PCA based Anomaly Detection and Mitigation Algorithm

To design a real-time anomaly detection and mitigation algorithm, we need to decrease the interval of the feature request messages. In addition, after one operation, we cannot wait until next continues samples have been generated. So we propose a moving window mechanism. In our design, the window size of the Traffic Matrix is fixed in length. When the new sample comes, we will keep moving the window to load the new sample and pop the oldest sample simultaneously to build the new fixed length Traffic Matrix. The diagram of moving window based Traffic Matrix is shown in **Fig. 2** where $p = n^2$ and n is the number of switches. $\boldsymbol{X}_i \in \boldsymbol{R}^p$ is the sample at cycle i which consists of p elements ($x_{i,1}, x_{i,2}, \ldots, x_{i,p}$). Each element x_{ij} indicates the traffic features of jth OD pair at cycle i.

Every new cycle, the Traffic Matrix will be updated as the input of anomaly detection and mitigation algorithm. The algorithm is shown in Algorithm I. The Anomaly Detection and Mitigation module will execute PCA method to calculate the SPE value, which is defined in Equations (6).

Afterwards, we can detect whether the network traffic is normal according to the threshold calculated by Q statistic method [17]. Q statistic method is a procedure for testing the residual vector, which is obtained according to PCA method, associated with a single observation vector and for an overall test for a group of observations. We can use Q statistic method to measure whether the residual vector has a sensible alteration. The threshold δ_α^2 is defined as

Equations (7).

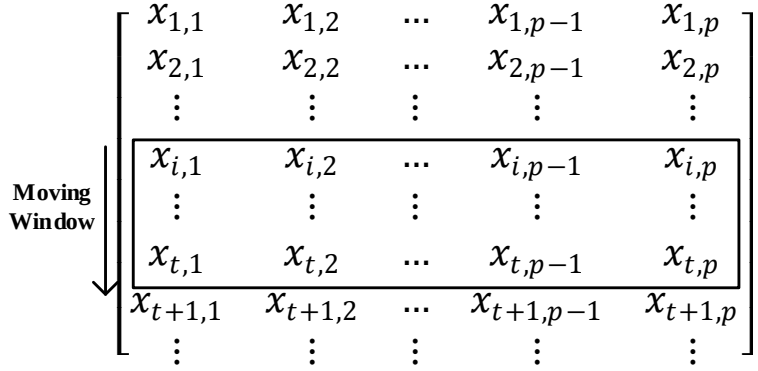

Fig. 2. Diagram of Moving Window based Traffic Matrix

$$\delta_\alpha^2 = \phi_1 \left[\frac{c_a \left(2\phi_2 h_0^2\right)^{1/2}}{\phi_1} + 1 + \frac{\phi_2 h_0 \left(h_0 - 1\right)}{\phi_1^2} \right]^{1/h_0} \tag{7}$$

of which, $h_0 = 1 - \frac{2\phi_1\phi_3}{3\phi_2^2}$, which is relative weight of historical data. $\phi_i = \sum_{j=k+1}^{p} \lambda_j^i, i = 1,2,3.$ λ_j is the jth eigenvalue calculated according to covariance matrix **C** which is introduced in section 2.2.and c_a is $1-a$ quantile of standard normal distribution where a usually set as 0.001. When $a = 0.001$, c_a can be calculated as 2.326. If $s \geq \delta_\alpha^2$, we usually think the network traffic is abnormal.

Nevertheless, in our design, the window size of input Traffic Matrix is fixed in length. The SPE value and Q statistic value will vary every new cycle. So we should not use a permanent Q statistic value as the threshold to evaluate whether the anomaly happens. Instead, we need to compare the SPE value in current cycle with Q statistic value in last cycle.

If the variation of traffic features is smooth and steady, the Q statistic value will not fluctuate widely. When the anomaly traffic emerges, on the basis of the percentage of anomaly traffic occupies, the SPE and Q statistic value will both vary to a certain degree. In our approach, at every new cycle we compare the SPE value and the Q statistic value both of this cycle and last cycle. In our design, we consider the anomaly happens at time interval t when SPE(t) > Q(t-1). We consider one or several measurements including traffic volume, source address entropy, destination address entropy, source port entropy and destination port entropy simultaneously when we run our anomaly detection algorithm.

Above all, we propose the moving window PCA based anomaly detection algorithm as shown in Algorithm 1.

The parameters appeared in our algorithm is shown in **Table 1**. The input of the algorithm includes the data sample of current cycle, size of the moving window, percentage of principal components and the interval of feature request messages, which are crucial to the effect of anomaly detection.

Algorithm 1: Moving Window PCA based Anomaly Detection Algorithm

Input:
$\boldsymbol{d}_t$ #new data sample, t is the sequence of cycles
m #size of moving window
p #percentage of principal components
in #interval between feature request messages
Initialization:
window_size = 0; s(0), q(0) = 0
Process:
for t = 2, 3, …, **do:**
acquire the new traffic feature sample $\boldsymbol{d}_t$ to build Traffic Matrix ***X(t)*** with moving window size as window_size
if window_size < ***m*:**
window_size += 1
else:
window_size = ***m***
calculate the dimension of principal components according to ***p***
calculate the anomalous subspace projection vector of current cycle with Equations (5)
calculate the SPE value **s(t)** and Q statistic value **q(t)** with Equations (7) and record them
if s(t) > q(t-1) **and** window_size >= ***m*:**
generate an **alert**
trigger the mitigation process
endif
end for
Output:
s(t) #SPE value of cycle t
q(t) #Q statistic value of cycle t
alert

The window size in our approach is set as 10, which means the Traffic Matrix contains 10 samples. When the moving window is too long, large fluctuations caused by the dynamics of network traffic will be regarded as anomaly. Besides, large amount of data will increase the time for computing. However, if the window size is too small, the accuracy will decrease.

Table 1. Parameters in Algorithm 1

Parameter	Description
$\boldsymbol{d}_t$	New data sample at *t*th cycle
m	Size of the moving window
p	Percentage of principal components
in	Interval of feature request messages

The percentage is used to calculate the dimension k of normal principal components. It means how much information will be preserved during dimensionality reduction. Generally, we use eigenvalues to determine the value of k. $\lambda_i \left(i = 1, 2, \ldots, n\right)$ represents the eigenvalues of the Traffic Matrix and we arrange them in descending order. We reserve the first k eigenvalues which occupies the percentage as we input.

$$percentage = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \tag{8}$$

Generally, as the percentage increases, the detection probability will increase and the false alarm probability will decrease.

The interval of feature request messages has an effect on both the accuracy and timeliness. If the interval is too short, the traffic information in one sample will not be sufficient. On the contrary, if the interval is too long, the timeliness will decrease. In our design, we set the interval as 20 seconds so that the whole moving window can cover the Traffic Matrix as long as 200 seconds. These parameters can be adjusted according to the specific experiment.
Above all, the parameter "percentage" is a variable that has an impact on both the detection probability and the false alarm probability. In the experiment, we will focus on this parameter to evaluate its impact.

3.4 Anomaly Mitigation Mechanism

Once the anomaly is detected according to the anomaly detection algorithm, the anomaly mitigation procedure will be triggered by the alert. In our design, we take advantage of OpenFlow protocol's openness to switch and per-flow statistics ability to locate the attacker and mitigate the anomaly traffic. We propose an anomaly mitigation algorithm as shown in Algorithm 2.

At first, we will look up the record of the traffic volume matrix and entropy matrix to locate several switch pairs, which contribute the largest percentage of traffic volume or information entropy during the detection cycle. Then we will mark the flows matching these switch pairs as suspicious flows. Furthermore, we will acquire the flow table entries of these suspicious switches to locate the flows that occupy the largest proportion of packet or byte counts and their destination is the victim. As is known to all that, it's hard to find every zombie hosts during a DDoS attack. To alleviate the impact of the attack, we will arbitrarily mark these flows as attack flows and treat the hosts that match these flows as zombie hosts.

After the attackers and the suspicious switches have been identified, anomaly mitigation module will firstly send messages to the switches to delete the malicious flow table entries and release the occupied storage space. Furthermore, the anomaly mitigation module will install the defense flow table rules with the highest priority (65,535) to block subsequent packets from these zombie hosts for a period of time. The header field of the defense flow table rule is composed of the source address and the ingress port of the attackers while the action field is set as "Drop".

Algorithm 2: Anomaly Mitigation Algorithm

Process:
look up the last sample of the Traffic Matrix
for each switch pair **in** last sample d_t , **do:**
select the top *n* switch pairs { $d_{s1}, d_{s2}, \ldots, d_{sn}$ } which contribute the largest traffic volume or information entropy
end for
mark these switches as suspicious switches
for each flow **in** suspicious switches, **do:**
select the top several flows which occupy the largest proportion of packet or byte counts
end for

mark these flows as attack flows
mark the hosts which match the attack flows as attackers
install defense flows to the corresponding switches

Output:

defense flows

4. Experiment Result and Evaluation

4.1 Prototype Implementation and Experiment Topology

We develop a component of anomaly detection and mitigation in POX controller, a python based OpenFlow controller platform. The system environment of the controller is Ubuntu 15.04 and the CPU is adopted Intel Xeon E3-1245 with 8G in memory. The component is composed of three modules: (1) Flow Feature Requesting module, (2) Flow Statistic Analysis module and (3) Anomaly Detection and Mitigation module, which are all run on POX controller.

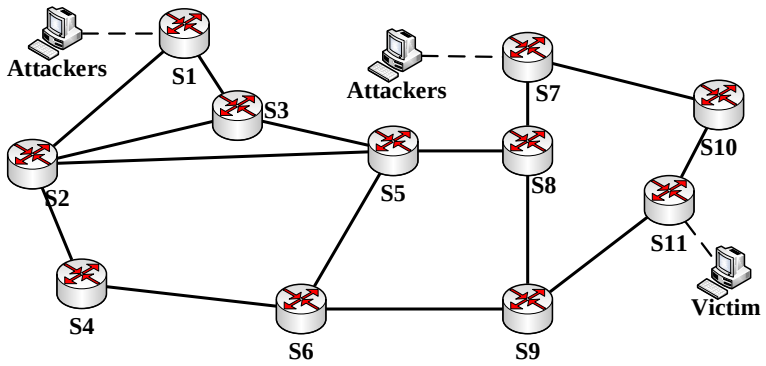

Fig. 3. Experiment Topology in Mininet

Furthermore, we imitate the topology of Abilene network to build an experiment topology deploying 11 OpenFlow switches in Mininet environment. There are 10 attackers and 10 benign users behind either subnetwork 1 or subnetwork 7. The rest nine subnetworks represent the benign users, each of which comprises 10 benign hosts. One of the hosts under subnetwork 11 represents the victim. The connection topology of the network devices is shown in **Fig. 3**.

4.2 Experiment Setting

We use two common metrics to estimate the detecting effort, detection probability and false alarm probability, computed with Equations (9) and (10).

$$D_p = \frac{TP}{TP + FN} \tag{9}$$

where True Positives (TP) are attack traffic classified as attacks, and False Negatives (FN) are attack traffic classified as legitimate.

$$F_p = \frac{FP}{TN + FP} \tag{10}$$

where False Positives (FP) are legitimate traffic classified as attack, and True Negatives (TN) are legitimate traffic classified as legitimate.

The traffic in our experiment consists of three parts: background traffic, normal traffic and anomaly traffic, which are all generated by using Scapy [16], a python based powerful

interactive packet manipulation program.

The experiment consists of two stages. The first stage is the false alarm testing stage while the second stage is the attack detection testing stage. During the first stage, we keep adjusting the parameters to compare the false alarm probability and finally ascertain them. While in the second stage, we compare the detection probability by adjusting the percentage of anomaly traffic injected in the normal traffic.

Firstly, the background traffic is generated to guarantee the building of entropy matrix. Every one second, one host of each subnetwork will be selected randomly to send one ICMP packet to where the destination host is selected randomly among all the hosts.

Secondly, the normal traffic is generated periodically. We set the interval as 10 seconds which is the same as that of feature request messages. Every 1 second, one or several normal users behind each switch will start a regular conversation to the victim host. The request rate of the conversation obeys Poisson Distribution where the expectation is set as 1. The duration of the conversation obeys Negative Exponential Distribution where the expectation is set as 10 seconds. We set the moving window as 10 cycle while each cycle lasts for 10 seconds.

At last, we inject attacking traffic into the normal and background ones every 30 cycles where each attack lasts for 30 seconds. In our experiment, we simulate the DDoS attack as the anomaly and evaluate the detection probability of our approach when the anomaly traffic occupies 5%, 10%, 15% of the total amount of OD pairs' traffic on average.

4.3 False Alarm Probability Testing Stage

False alarm stage is used to evaluate the false alarm probability of our approach. There is only background traffic and benign traffic in the network during this stage.

Firstly, we compare the false alarm probability when the parameter percentage varies. We adjust the parameter "*percentage*" in **Table 2** to measure its influence on False alarm probability. For each "percentage", we simulate 1000 cycles to evaluate the false alarm probability.

Table 2. False alarm probability of the Anomaly Detection Approach

Input/Percentage	***0.8***	***0.85***	***0.9***	***0.95***
Traffic Volume Matrix	18.8%	12.7%	10.5%	8.1%
Source Address Entropy Matrix	14.9%	8.4%	7.8%	6.5%
Destination Address Entropy Matrix	15.4%	9.2%	9.1%	6.8%
Combination Matrix	12.2%	7.8%	7.5%	5.3%

From the table we can find that as the growth of "*percentage*", the false alarm probability will decrease no matter the input Traffic Matrix is traffic volume of entropy. Obviously, the combination matrix of traffic volume and entropy has the best performance.

We capture the statistic of first 100 cycles at "*percentage*" = 0.95. The SPE value and Q statistic value of traffic volume matrix, source address entropy matrix, destination address entropy matrix and combination matrix are measured simultaneously and shown respectively in **Fig. 4** (a), (b), (c) and (d). From the figures, we can also find out that the combination matrix has the best performance in false alarm probability test.

In our design, we set the moving window size as 10. Under normal circumstances, when the percentage exceed 95%, the dimension of normal principal components will reach to 9. When the anomaly emerges, dimension of normal principal components will decrease according to the proportion of anomaly. However, if the percentage of principal components is higher, although the false alarm probability is lower, the anomaly will be more difficult to detect. The proportion of anomaly must be higher enough to be detected.

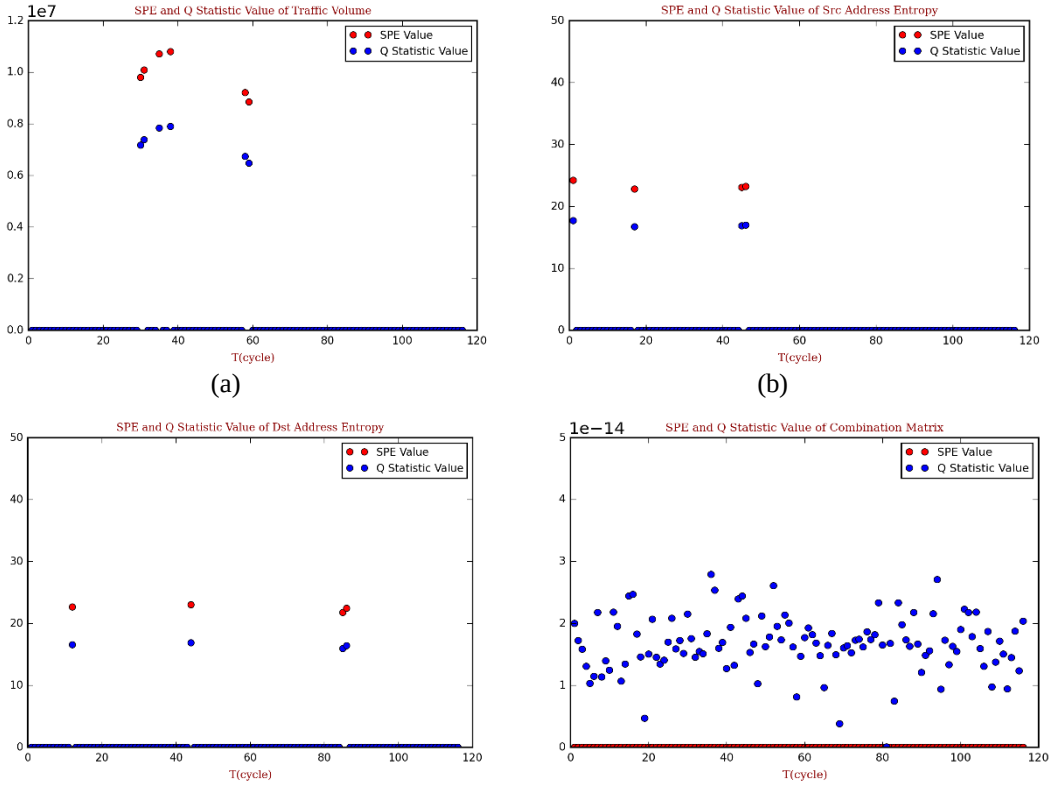

Fig. 4. SPE and Q statistic value of traffic volume matrix, source address entropy matrix, destination address entropy matrix and combination matrix without attack

4.4 Detection Probability Testing Stage

The second stage of the experiment is the attack detection testing stage. In this stage, we simulate the DDoS attack for several times with the parameter "percentage" is set as 0.9. For each time, we vary the proportion of DDoS attack traffic, which is injected into normal traffic. We evaluate the detection probability of our approach when the anomaly traffic occupies 5%, 10%, 15% of the total amount of OD pairs' traffic on average. For each proportion, we do the experiment for 11 times. We identify an attack is successfully detected only when it is detected in the certain cycle where the attack happens.

Table 3. Detection probability of the Anomaly Detection Approach with "percentage" set as 0.9

Input/Anomaly Proportion	*5%*	*10%*	*15%*
Traffic Volume Matrix	18.2%	72.7%	90.9%
Source Address Entropy Matrix	72.7%	90.9%	100%
Destination Address Entropy Matrix	63.6%	81.8%	90.9%
Combination Matrix	81.8%	90.9%	100%

The detection probability of different anomaly proportion is shown in **Table 3**. From **Table 3** we can find that when the anomaly traffic occupies over about 10% ~ 15% of total OD pairs' traffic, the combination Traffic Matrix can detect the anomaly every time.

4.5 Comparison

To compare with the entropy-based anomaly detection techniques in Ref [10], [11], we simulate an experiment with the anomaly traffic occupies 5% and 15% respectively. Both our approach and entropy-based approach take advantage of the native statistics collecting capability of OpenFlow protocol.

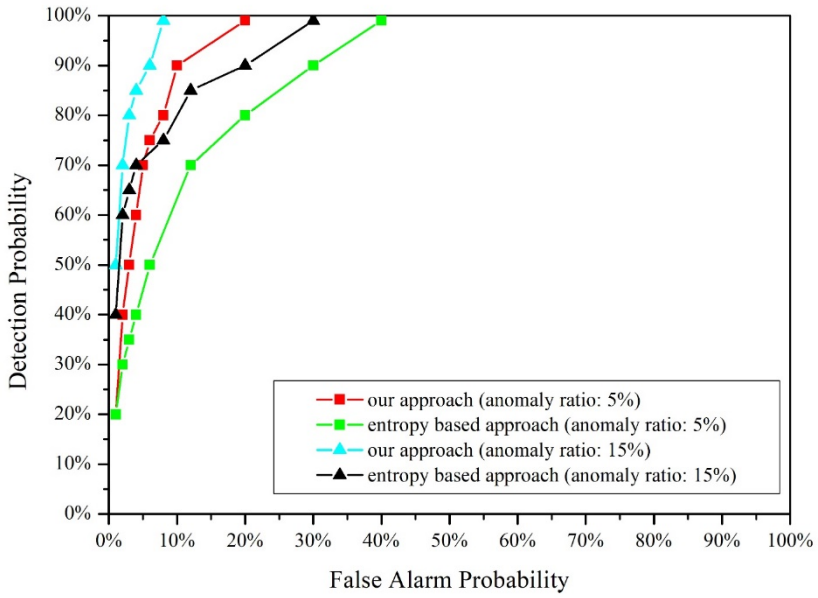

Fig. 5. ROC curves for anomaly ratio 5% and 15% cases between our approach and entropy based approach

In the following, ROC curves are presented in **Fig. 5**, with respect to detection probability and false alarm probability. From the figure, we can find that our approach has better performance than entropy-based approach when the anomaly ratio is either 5% or 15%. It is because that the threshold value of entropy-based approach will vary in different scenarios. In our approach, the threshold is calculated every new cycle according to the eigenvalues in last cycle.

Furthermore, we compare our work with the Machine Learning (ML) based approach in [12], [22], [26]. Both our proposed PCA based approach and entropy based approach do not need the training process. The anomaly is detected according to the variation of traffic volume or entropy. The ML based approach need a large size of training set to make better performance. However, slow training process is not suitable for real-time detection. Over-fitting may happen during neural network training. Above all, our PCA based approach has a better performance on detecting anomaly in real-time. ML based approaches are more applicable for anomaly classification which is the direction of our future research.

4.6 The Effect of Mitigation

When the anomaly traffic is detected with our approach, we can locate the anomaly hosts according to the suspicious flow features which are stored temporarily in the controller. Furthermore, the controller can install defense flow table rules on the corresponding switches to drop the packets from the attackers. We emulate an experiment of DDoS attack to test the mitigation effort.

We simulate two times of DDoS attack at 960s and 1160s respectively. The packet count of OD pairs is shown in **Fig. 6**. The first attack happens at 960s, which lasts for 30 seconds. The Anomaly Detection and Mitigation module detects the anomaly and further install the defense flow table rules on the corresponding switches to drop the attack flows. (In the experiment, we set the lifetime of defense flow table rules as 65535 seconds to make sure the mitigation effect.) We simulate the second attack with the same attackers at 1160s. Although the attack's volume is higher and the duration is longer, the attack flows are almost dropped by the entrance switches. As a result, the victim does not receive most of the attack traffic.

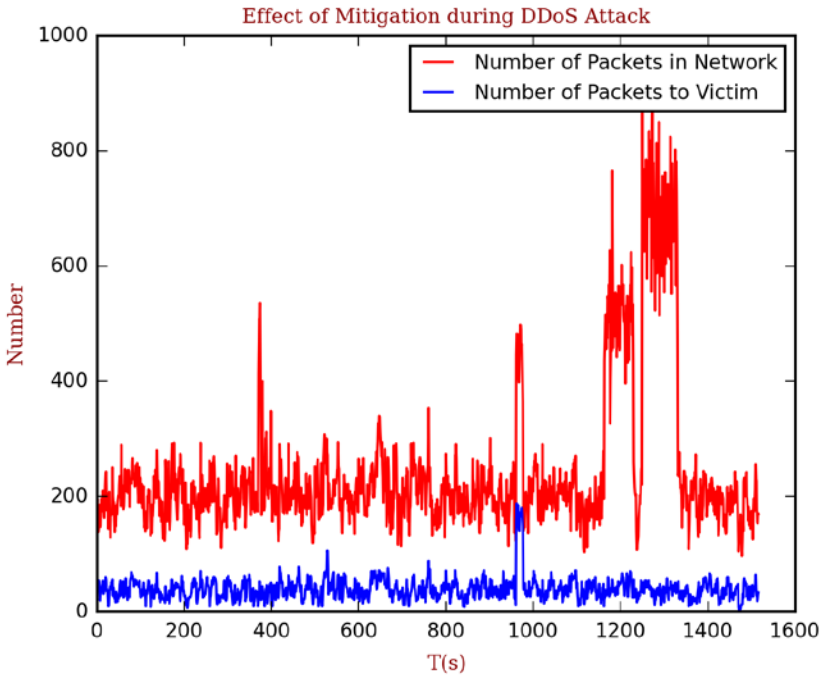

Fig. 6. The effect of mitigation during the DDoS attack

4.7 Detection Time and Overhead Analysis

At last, we analyze the detection time and the overhead of the anomaly detection approach. In our approach, we compare the SPE value in current cycle with Q statistic value in last cycle to confirm whether an attack happens. When the attack happens at cycle t, the alert will be generated immediately. We add the timer at the start and the end of the anomaly detection component. In our experiment, as we adopt the fixed length window size as the input vector, so that the processing delay is only in relation to the dimension of the input Traffic Matrix. As there are 11 switches in the topology, the dimension of the matrix is 121. The average of the processing delay is about 40 *ms*. Obviously, the duration of one cycle is much more longer than the processing delay.

Furthermore, we also write a program to monitor the CPU usage of the controller with the anomaly detection and mitigation component Running and Sleeping respectively. We acquire the CPU utilization every one second and record the values of 500 seconds. The result is shown in **Fig. 7**. We can find that the red points represent the POX controller is not running, while the blue points and green points represent the POX controller is startup and the anomaly detection and mitigation component in the controller is on running. We also can notice the green points that every 10 seconds the CPU utilization surges one time. It is because the detection algorithm runs at 10 seconds period. However, as we have known that the processing delay occupies a little part of each cycle. Therefore, the anomaly detection component will have little influence on the performance of the system.

Above all, our anomaly detection approach is a lightweight one which can detect the network anomaly in real time.

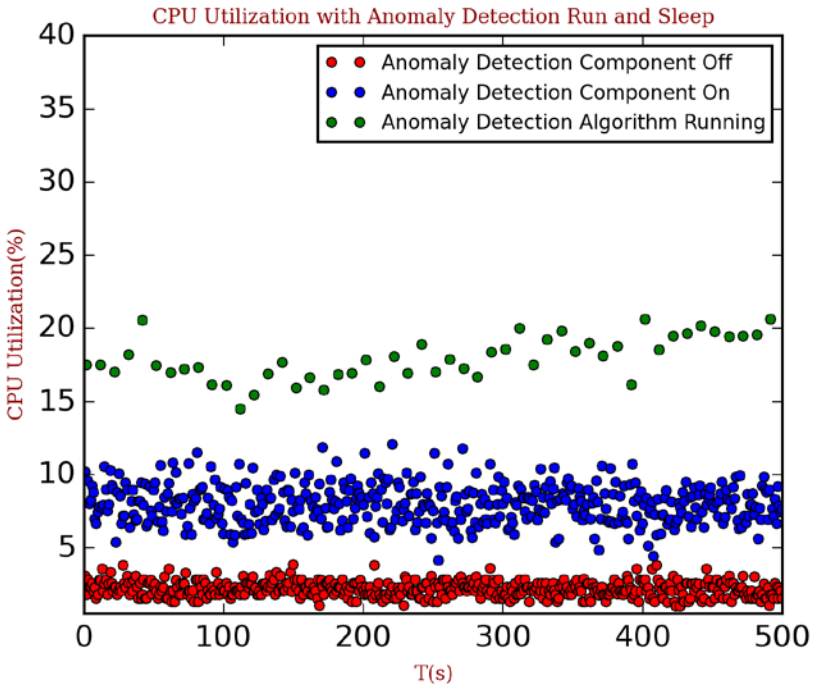

Fig. 7. CPU utilization with the anomaly detection run and sleep

5. Conclusion and Further Work

In this paper, we propose an anomaly detection and mitigation approach in SDN network and develop a prototype for the system. We take advantage of the native property of SDN by getting the global view of network and acquiring native flow features statistic from the data plane switches. By analyzing the volume and entropy of the flow features, we further build the Traffic Matrix as the input of the anomaly detection algorithm. Furthermore, we propose a moving window PCA based anomaly detection and mitigation approach to detect the abnormal behavior in real-time, which can prevent the subsequence attacking flows. In addition, we simulate experiments to evaluate our approach. In the experiments, we evaluate the detecting effect from two aspects, detection probability and false alarm probability. We compare the ROC curves of our approach with the entropy-based approach, showing that our approach has better performance. Besides, we simulate a DDoS attack to evaluate the effect of mitigation approach. At last, we evaluate the detection delay and the overhead of our approach, which is timely and has little influence on the performance of the controller.

In our design, the PCA based anomaly detection and mitigation approach is not limited to detecting DDoS attack, but also can be used to detect other kinds of anomaly, including worm, port scanning, etc. We also attempt to study better anomaly classification approach with the help of machine learning technique in the future. Furthermore, the mitigation mechanism in our design is just an initiative scheme. In further work, we are planning to study the anomaly mitigation mechanism to make it more intelligent in identifying and preventing the attackers.

References

[1] ONF, “Software-Defined Networking: The New Norm for Networks,” 2012. Article (CrossRef Link)

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “Openflow: Enabling innovation in campus networks,” SIGCOMM CCR, vol. 38, no. 2. Article (CrossRef Link)

[3] Adrichem, N. L. M. Van, C. Doerr, and F. A. Kuipers. "OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks," Network Operations and Management Symposium IEEE, 2014:1-8. Article (CrossRef Link)

[4] POX. At Article (CrossRef Link)

[5] Yu, Curtis, et al. "FlowSense: monitoring network utilization with zero measurement cost," in *Proc. of International Conference on Passive and Active Measurement Springer-Verlag*, pp. 31-41, 2013. Article (CrossRef Link)

[6] Lantz, Bob, Brandon Heller, and Nick McKeown. "A network in a laptop: rapid prototyping for software-defined networks," in *Proc. of Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ACM, 2010. Article (CrossRef Link)

[7] Lakhina, Anukool, M. Crovella, and C. Diot. "Mining anomalies using traffic feature distributions," in *Proc. of ACM SIGCOMM 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Philadelphia, Pennsylvania, Usa, August DBLP, pp. 217-228, 2005. Article (CrossRef Link)

[8] Soule, Augustin, and N. Taft. "Combining filtering and statistical methods for anomaly detection," in *Proc. of Conference on Internet Measurement 2005*, Berkeley, California, Usa, October DBLP, pp. 31-31, 2005. Article (CrossRef Link)

[9] Li, Ming. "Change trend of averaged Hurst parameter of traffic under DDOS flood attacks," *Computers & Security*, vol. 25, no. 3, pp. 213-220, 2006. Article (CrossRef Link)

[10] Giotis, K, et al. "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," *Computer Networks the International Journal of Computer & Telecommunications Networking*, vol. 62, no. 5, pp. 122-136, 2014. Article (CrossRef Link)
[11] Mousavi, Seyed Mohammad, and M. St-Hilaire. "Early detection of DDoS attacks against SDN controllers," in *Proc. of International Conference on Computing, NETWORKING and Communications IEEE*, pp. 77-81, 2015. Article (CrossRef Link)
[12] Braga, Rodrigo, E. Mota, and A. Passito. "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *Proc. of IEEE Conference on Local Computer Networks*, LCN 2010, 10-14 October 2010, Denver, Colorado, Usa, Proceedings DBLP, 408-415, 2010. Article (CrossRef Link)
[13] Wold, Svante, K. Esbensen, and P. Geladi. "Principal component analysis," *Chemometrics & Intelligent Laboratory Systems*, vol. 2, no. 1, pp. 37-52, 1987. Article (CrossRef Link)
[14] Lakhina, Anukool, M. Crovella, and C. Diot. "Diagnosing network-wide traffic anomalies," *Acm Sigcomm Computer Communication Review*, vol. 34, no. 4, pp. 219-230, 2004. Article (CrossRef Link)
[15] Silva, Anderson Santos Da, et al. "Identification and Selection of Flow Features for Accurate Traffic Classification in SDN," in *Proc. of IEEE, International Symposium on Network Computing and Applications IEEE*, 134-141, 2015. Article (CrossRef Link)
[16] Scapy. At Article (CrossRef Link)
[17] Jackson, J. Edward, and G. S. Mudholkar. "Control Procedures for Residuals Associated with Principal Component Analysis," *Technometrics*, vol. 21, no.3, pp. 341-349, 1979. Article (CrossRef Link)
[18] Tootoonchian, Amin, M. Ghobadi, and Y. Ganjali. "OpenTM: Traffic Matrix Estimator for OpenFlow Networks," in *Proc. of International Conference on Passive and Active Measurement Springer-Verlag*, 201-210, 2010. Article (CrossRef Link)
[19] Historical Abilene Connection Traffic Statistics. At Article (CrossRef Link)
[20] Lim, S., et al. "A SDN-oriented DDoS blocking scheme for botnet-based attacks," in *Proc. of Sixth International Conf on Ubiquitous and Future Networks IEEE*, pp. 63-68, 2014. Article (CrossRef Link)
[21] Wang, Rui, Z. Jia, and L. Ju. "An Entropy-Based Distributed DDoS Detection Mechanism in Software-Defined Networking," *Trustcom/bigdatase/ispa IEEE*, pp. 310-317, 2015. Article (CrossRef Link)
[22] Niyaz, Quamar, W. Sun, and A. Y. Javaid. "A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN)," 2016. Article (CrossRef Link)
[23] Francois, J, and O. Festor. "Anomaly traceback using software defined networking," in *Proc. of IEEE International Workshop on Information Forensics and Security IEEE*, pp. 203-208, 2014. Article (CrossRef Link)
[24] Giotis, Kostas, G. Androulidakis, and V. Maglaris. "Leveraging SDN for Efficient Anomaly Detection and Mitigation on Legacy Networks," in *Proc. of Third European Workshop on Software Defined Networks IEEE Computer Society*, 85-90, 2014. Article (CrossRef Link)
[25] Li, Chuanhuang, et al. "Detection and defense of DDoS attack–based on deep learning in OpenFlow based SDN," *International Journal of Communication Systems*, 2018. Article (CrossRef Link)

Mingxin Wang received the B.S. degree from Beijing Jiaotong University in 2013, where he is currently pursuing the Ph.D. degree in telecommunications and information system. His research interests include next generation Internet, communication system security, network management technologies and software defined networking.

Huachun Zhou received the B.S. degree from the People's Police Officer University of China in 1986 and the M.S. and Ph.D. degrees from Beijing Jiaotong University in 1989 and 2009, respectively. His recently research projects include Research on Models and Algorithms of Information-Centric Mobile Internet, supported by the National Natural Science Foundation of China, and Research on the Future Space-Ground Integrated Network, supported by the National High Technology of China. His research interests include mobility management, mobile and secure computing, routing protocols, and network management technologies.

Jia Chen is an Associate Professor with the National Laboratory of Next Generation Internet Interconnection Device, Beijing Jiaotong University. She received BEng degree in Communication Engineering in 2005 from Beijing University of Posts and Telecommunications. She received Master and PhD degree in 2006 and 2010 respectively from Department of Electrical and Electronic Engineering, University College London. Her current research interests include architecture and protocol design for the future Internet.