IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 29, 2018, accepted May 14, 2018, date of publication May 22, 2018, date of current version June 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2839684

A Detection Method for Anomaly Flow in Software Defined Network

HUIJUN PENG[1], ZHE SUN[1], XUEJIAN ZHAO[1], SHUHUA TAN[2], AND ZHIXIN SUN[1]

[1]Key Laboratory of Broadband Wireless Communication and Sensor Network Technology, Nanjing University of Posts and Telecommunications, Nanjing 210003, China

[2]National Engineering Laboratory for Logistics Information Technology, YuanTong Express Co., Ltd, Shanghai 201705, China

Corresponding author: Zhixin Sun (sunzx@njupt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61373135, Grant 61672299, Grant 61702281, and Grant 61602259, and in part by the Natural Science Foundation of Jiangsu Province under Grant BK20160913 and Grant BK20140883.

ABSTRACT As a new type of network structure, the Software Defined Network (SDN) provides a new solution for network flow management and optimization, which has made the accurate detection of anomaly SDN flows a hot research topic. This paper presents an SDN-based flow detection method, builds structures for detecting anomaly SDN flows and performs classification detection on the flows using the double P-value of transductive confidence machines for K-nearest neighbors algorithm. The experimental results show that the algorithm proposed achieves a lower false positive rate, higher precision, and better adaptation to the SDN environment than do other algorithms of the same type.

INDEX TERMS Intrusion detection, detection algorithms, nearest neighbor searches, SDN.

I. INTRODUCTION

With the development of Internet technology, network flows have increased rapidly. In addition, DDoS attacks [1], which seriously threaten network security, have become more prevalent. DDoS attacks mainly occupy target system resources or link bandwidths by directly or indirectly sending a large number of data packets, thereby causing host paralysis or network congestion. Due to the diverse attack methods and wide variety of attacks, it is impossible to guard against DDoS attacks. In recent years, scholars have been committed to researching DDoS anomaly detection techniques. They have applied Bayesian algorithm, wavelet analysis and support vector machine (SVM) to detect anomaly flows. However, the existing prevention and detection techniques for DDoS attacks have many security risks. For example, the dependence on software and devices increases the network burden; it is difficult for a single attack index to realize an exact match; and status updates slow down when security policies change.

As a new type of network structure, SDN [2] has many advantages. It manages an entire network with controllers and separates the data plane from the control plane. SDN deprives switches of control functions and enables them only to forward data packets. In addition, it implements centralized and uniform control over data, thereby realizing the optimal management of network resources and greatly improving the network's flexibility and controllability. Since SDN adopts OpenFlow [3], which is a protocol that runs between controllers and switches, the communication between the two should follow OpenFlow standards. OpenFlow switches take flow tables as the core and match and forward data packets in accordance with flow table information while flow table information is issued and updated by upper controllers. This approach has achieved good results in resolving traditional DDoS attacks. However, SDN is characterized by strong openness, which indicates that users can customize networks freely by virtue of programming interfaces and are under the threat of DDoS attacks. Therefore, the detection of anomaly SDN flows has important research value and broad application prospects.

Based on the centralized control of an SDN, this paper applies its technology to DDoS anomaly detection and presents a detection method for anomaly SDN flows. The paper consists of six sections. Section II elaborates detection methods for anomaly flows in traditional networks and SDN separately. Section III introduces the basic structure and process flow for detecting anomaly SDN flows. Section IV proposes an optimized method for anomaly detection - the DPTCM-KNN algorithm. Section V performs simulation experiments on the algorithm proposed and compares it

with other algorithms, and Section VI summarizes the entire paper.

II. RELATED RESEARCHES

Currently, there are many detection methods for anomaly flows in traditional networks, and these methods are divided into three main types: methods based on statistical analysis, methods based on machine learning and methods based on data mining. PCA (Principal Component Analysis) is a data reduction technique based on statistical analysis [4]. Although this method can make full use of the correlation of global traffic in the event of an abnormality to improve the recognition efficiency, if the PCA method injects traffic with large fluctuations into the training phase, it can build a special attack to avoid detection by the PCA recognition system. Of the machine learning-based methods, Bayesian analysis [5] and correlation analysis [6] are the two most commonly used methods. The Bayesian analysis method uses a probability graph model to study the statistical correlation between anomalies and measured values. In addition to causality, the setting of various probabilities and the selection of models in this method have a large influence on the accuracy. If the probability or model parameters are not properly selected, then the recognition accuracy will be poor. Data mining-based analysis methods are divided into two main methods: classification and clustering. Support vector machine (SVM) [7] is a common data mining classification method. Using SVM for anomaly detection can guarantee that the prior knowledge is insufficient. There is good classification accuracy, but both SVM and neural networks are plagued with high false alarm rates to varying degrees. Cluster analysis [8] is an unsupervised learning process that requires no training process, but there is no significant difference in the distribution of the probability density between abnormal and normal samples, especially hidden worms and slow DDoS.

Among the data mining-based anomaly detection methods, the KNN algorithm (k-Nearest Neighbor) [9] is quite popular because it not only has mature theories, low retraining costs, high precision and strong real-timeliness, but it also supports instant incremental learning. Document [10] presents the TCM-KNN algorithm and optimizes the feature selection and feature weight mechanism. The original algorithm, which is based on the nearest neighbor concept of the KNN algorithm, explores the deviations between detection points and normal training sets by calculating p values to classify the detection points. However, the algorithm has some deficiencies: when the abnormal points serve as thresholds between normal and abnormal points, it is necessary to make a judgment on the basis of the relative deviation. Since the algorithm has defects in anomaly detection of the detection points, its precision must be improved further. Document [11] combines the KNN algorithm with the ACO (ant colony optimization) algorithm; the KNN-ACO algorithm is proposed on this basis and employs the ID3 (decision tree) algorithm for feature reduction. By performing experiments on the KDD CUP99 data sets, it is verified that the algorithm has a higher detection precision.

Although these detection methods for anomaly flows in traditional networks have a certain reference value for SDN security protection, they are characterized by poor scalability, low precision and low efficiency in identifying massive and rapid data flows. Therefore, the SDN-based detection of anomaly flows has gradually become a hot issue in recent years.

In terms of attack detection in an SDN, Documents [12], [13] analyze the relations between the data flow features and the threshold values and considers the algorithm precision and cost sensitivity, ultimately proposing an elephant flow detection system - a system with high detection precision. In response to new DDoS attacks with low-speed flows, Document [14] proposes a detection method for SDN controllers. This method classifies the interface flows and performs an SPRT (Sequential Probability Ratio Test) [15] on them, thereby accurately identifying the DDoS attacks and injured interfaces. Both methods have strong pertinence and fail to detect DDoS anomalies in real time.

With the convenient, efficient, rapid and comprehensive collection of sFlow, Document [16] collects data and realizes effective and scalable anomaly identification in the SDN control plane based on OpenFlow. Document [17] uses the program-controlled capability of OpenFlow to improve the features of remotely triggered black hole filters and mitigate flow anomalies caused by DDoS attacks. Document [18] presents a method for identifying and mitigating SYN flooding attacks. This method sends the flows captured by attackers from agents to sFlow collectors for analysis and regulates flow table rules with OpenFlow controllers, thereby preventing anomaly flows.

Document [19] proposes a lightweight and rapid DDoS detection method based on entropy. This method protects controllers by considering the abilities of SDN controllers and calculates the entropy in accordance with grouping the requests received by SDN controllers, thereby identifying anomalies quickly. To provide an SDN with better security in large data environments, Document [20] presents a fault-tolerant control structure (MCSSDN), in which each device is managed by multiple controllers. It resists Byzantine attacks on controllers and communication links between controllers.

Overall, although scholars have proposed many identification methods for anomaly SDN flows in recent years, the research is still in its infancy, and it still has many problems:

1) The SDN-based flow detection method puts a large amount of pressure on the controllers. With an increase in network flows, the controllers are required to monitor the entire network environment, issue flow tables to switches and detect flow anomalies, which greatly increases the burden on the controllers. Therefore, it is necessary to propose more perfect detection architectures for SDN anomalies.

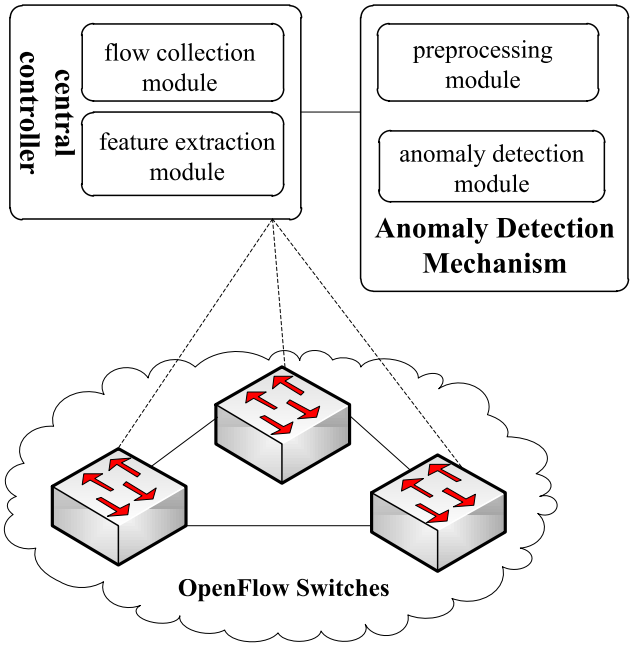

FIGURE 1. Detection Structure of Anomaly Flows in an SDN.

2) The existing anomaly detection algorithms for SDN are characterized by low precision and poor real-timeliness. At the same time, they do not support incremental learning. Therefore, it is crucial for SDN-based flow detection to improve and optimize the traditional KNN algorithm and establish high-precision and real-time identification models for anomaly flows so it can adapt to large-scale networks.

This paper proposes to construct detection architectures for anomaly SDN flows and regard the DPTCM-KNN algorithm as the core algorithm of anomaly flow detection mechanism, which improves the detection precision in the SDN environment and decreases the burden on controllers to a certain extent.

III. ANOMALY FLOW DETECTION BASED ON SDN

To solve the defects of SDN-based flow detection methods, this paper designs an anomaly flow detection method for SDN: the flow collection module of the controllers collects the flow table information and extracts the flow features, and the anomaly detection mechanism preprocesses the flow features and performs classification detection on the network flows with the DPTCM-KNN algorithm. Through division and coordination, the switches, controllers and the anomaly detection mechanism eventually realize anomaly detection in the SDN environment. The detection structure is shown in Fig. 1.

A. SWITCHES AND CONTROLLERS

In the SDN environment, OpenFlow switches serve as data forwarding devices and are responsible for collecting and forwarding data. By virtue of the OpenFlow protocol, switches send messages to controllers regularly to inform them of the current flow status and update the flow table information issued by the controllers in a timely manner when the anomaly detection mechanism finishes the flow detection. The central controller considers the RYU controller to be the core, and the RYU controller is the central execution unit of the OpenFlow networks for realizing programmable control. The RYU controller maintains basic information (topology and flow table rules) of the entire network, thereby managing and monitoring the entire network environment by controlling the switches.

To finish the anomaly flow detection, the controllers are divided into the flow collection module and feature extraction module. Here, the flow collection module sends requests (p_flow_stats_request) to switches every 10 seconds to obtain the flow table information and submits the information to the feature extraction module. The feature extraction module extracts the flow feature information (including data bytes from the source host to the target host, and lost packets, errors, port rates and data bytes from the target host to the source host), classifies the information, forms flow feature vectors and sends them to the anomaly detection mechanism. The anomaly detection mechanism preprocesses the flow feature vectors, performs anomaly detection and provides feedback on the detection results to the central controller via secure channels, thereby finishing the modification of the flow tables.

B. ANOMALY DETECTION MECHANISM

The anomaly detection mechanism centrally analyzes the data flow information and detects anomalies. It obtains a large number of flow feature vectors by virtue of the SDN controllers; it preprocess these vectors and classifies by the anomaly detection methods to identify the anomalies. The anomaly detection mechanism consists of two parts: the preprocessing module and the anomaly detection module. Here, the preprocessing module is mainly responsible for standardizing and normalizing the flow feature vectors sent by the controllers. It is assumed that there are n flow feature vectors and each vector has t features($X_{ij}(1 \leqslant i \leqslant n, 1 \leqslant j \leqslant t)$). The preprocessing processes of X_{ij} are described below:

- Standardization

$$X'_{ij} = \frac{X_{ij} - Mean_j}{AvgDev_j} \tag{1}$$

where,

$$Mean_j = \frac{X_{1j} + X_{2j} + \cdots + X_{nj}}{n} \tag{2}$$

$$AvgDev_j = \frac{\left|X_{1j} - Mean_j\right| + \cdots + \left|X_{nj} - Mean_j\right|}{n} \tag{3}$$

In Eq. (1), $Mean_j$ is the mean value, and $AvgDev_j$ is the mean absolute deviation value. During standardization, the following points should also be considered:

1) If $Mean_j = 0, X_{ij} = 0$
2) If $AvgDev_j = 0, X_{ij} = 0$

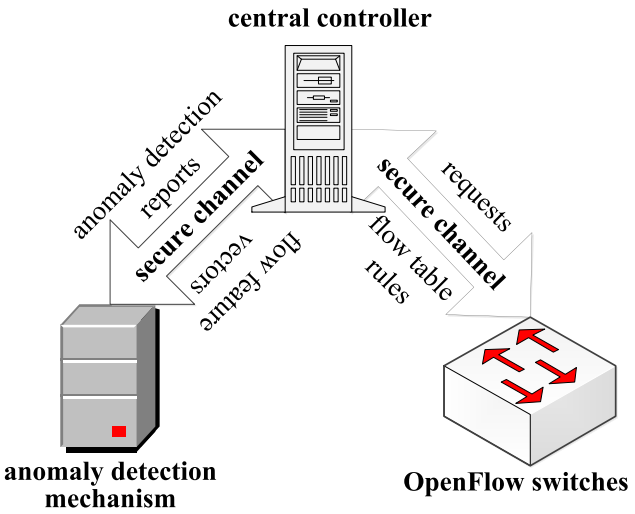

FIGURE 2. A Process Flow Chart of Anomaly Flows.

- Normalization
  Normalization means normalizing the standardized data into [0, 1]. It is assumed that X''_{ij} is the normalized value of X'_{ij}.

$$X''_{ij} = \frac{X'_{ij} - X_{min}}{X_{max} - X_{min}} \tag{4}$$

where,

$$X_{min} = min\{X'_{ij}\} \tag{5}$$
$$X_{max} = max\{X'_{ij}\} \tag{6}$$

After the preprocessing module of the anomaly detection module finishes the preprocessing process, it passes the anomaly detection module to perform anomaly detection on the preprocessed stream feature vector. The anomaly detection module uses the DPTCM-KNN algorithm to detect the traffic. The concept of TCM (Transductive Confidence Machines) is to obtain the confidence level of a detection point by virtue of stochastic algorithm theory, and it uses the probability p to measure whether the detection point belongs to the category. The larger the p value is, the more likely it is that the detection point belongs to the category. The DPTCM-KNN algorithm explores the absolute deviation and relative deviation between the detection point and other points, and it introduces strangeness and independence, thereby obtaining two p values (p_1 and p_2) and judging the abnormality of the detection point on this basis. The anomaly detection module generates an anomaly detection report at regular intervals according to the detection result of the DPTCM-KNN algorithm.

C. SDN FLOW PROCESSING

Under the SDN architecture, when receiving a large number of data flows, the switches enforce the following processes, as shown in Fig. 2.

1) The OpenFlow switches submit information to the central controller to inform it of the current flow status. Meanwhile, they send requests to the controller for the latest flow table forwarding rules.
2) The central controller extracts the flow features from the flow table information that is sent by the switches and summarizes by the feature extraction module, forms multi-dimensional flow feature vectors, and sends them to the anomaly detection mechanism.
3) The anomaly detection mechanism preprocesses (standardizes and normalizes) the flow feature vectors that were received and employs the DPTCM-KNN algorithm to calculate the strangeness and independence of the detection points, thereby finishing the anomaly detection.
4) The anomaly detection mechanism generates anomaly detection reports after receiving the detection results and sends the reports to the central controller regularly by virtue of secure channels. The central controller can control the parameters related to the anomaly detection mechanism in accordance with the network conditions.
5) The central controller modifies the flow table forwarding rules according to the reports received and issues them to OpenFlow switches for anomaly treatment.

IV. ANOMALY DETECTION ALGORITHM

A. RELATED DEFINITIONS

In the second section, a TCM-KNN algorithm proposed in reference [10] is introduced. The algorithm combines the TCM idea with the KNN classification algorithm. The algorithm combines the TCM idea with the KNN classification algorithm to calculate the KNN distance D_i^y from the unknown point i to class y. The distance from this point to another class is D_i^{-y}. The ratio $\frac{D_i^y}{D_i^{-y}}$ is used to estimate the relative suitability of the unknown point i and the class, and the p-value is used to assess the extent to which the unknown point belongs to the class. However, when the algorithm is applied to anomaly detection, only the relative deviation between the unknown point and the class is considered, and the absolute deviation is ignored; thus, the invasion cannot be fully detected, which affects the accuracy of the anomaly detection. Therefore, this paper proposes the DPTCM-KNN algorithm, which introduces the concept of independence of estimation of absolute deviation, and uses the strangeness of the relative deviations as a criterion for making a joint judgment. By calculating the double p-values that are used for anomaly detection, the detection loopholes in the TCM-KNN algorithm are avoided as much as possible, and the accuracy of the anomaly detection is improved.

For a training set that contains n elements $\{(x_1, y_1), \cdots, (x_n, y_n)\}$, $X_i(x_i^1, x_i^2, \cdots, x_i^t)$ is the value set of the t features that are extracted from the detection point i, while y_i is the category that the detection point i belongs to. Usually, it ranges from 1 to c. Elements of the anomaly detection systems are divided into two parts, namely, normal elements and abnor-

mal elements. To classify the detection points, the following definitions should be made.

Definition 1 (Euclidean Distance Between Two Points): The Euclidean distance is usually used to measure the proximity between two points and is expressed as D_{ij}^{y} (Euclidean distance between point i and point j in category y). The calculation formula is as follows:

$$D_{ij}^{y} = \sqrt{\sum_{a=1}^{t}(X_{ia} - X_{ja})^2} \tag{7}$$

where, X_i and X_j are the value sets of the extracted features of i and j, respectively; X_{ia} is the value of the a^{th} attribute of X_i; and t is the length of the vector X_i.

The Euclidean distance measures the absolute spatial distance between the various points. The larger the distance is, the greater the difference between two points will be. The calculation of the Euclidean distance is a fundamental step of the KNN algorithm.

Definition 2 (Strangeness): It is assumed that the normal elements and abnormal elements of the anomaly detection systems belong to y and -y, respectively. When the Euclidean distances between i and the other points in category y are sorted in accordance with the KNN algorithm, D_{ij}^{y} represents the Euclidean distance between i and the j^{th} nearest neighbor in y. The ratio is defined as strangeness in the DPTCM-KNN algorithm.

$$\alpha_{iy} = \frac{\sum_{j=1}^{k} D_{ij}^{y}}{\sum_{j=1}^{k} D_{ij}^{-y}} \tag{8}$$

where k is number of nearest neighbors in the KNN algorithm and iy is the strangeness of i relative to y. Strangeness describes the belonging degree of a point to a category. The greater the strangeness is, the less likely it is that the point belongs to the category.

Definition 3 (Independence): Independence is the sum of the Euclidean distances between a detection point and its k nearest neighbors, namely, the absolute deviation between the two. It is expressed as follows:

$$\theta_{iy} = \sum_{j=1}^{k} D_{ij}^{y} \tag{9}$$

where, θ_{iy} is the independence of i relative to y; k is the number of nearest neighbors; and D_{ij}^{y} is the distance between i and the j^{th} nearest point. At the same time, θ_{iy} represents the belonging degree of a detection point to a category. The smaller the independence is, the more likely it is that the point belongs to the category.

In contrast to the strangeness, which describes the relative distance between a detection point and a category, the independence measures the absolute distance between the two and explores the abnormal points that are far from the normal points. The two metric values are complementary and integral, and they influence the precision of the TCM-KNN algorithm.

Definition 4 (Double p Value): The probability p is used to demonstrate the anomaly probability of a detection point relative to other points, namely, the belonging degree of a detection point to a category. The greater the p value is, the more likely it is that the point belongs to y (normal points). The calculation formula is as follows:

$$p_1(\alpha_i) = \frac{\#\{j = (1 \ldots n) : \alpha_j \geq \alpha_i\}}{n+1} \tag{10}$$

where, α_i is the strangeness of i in y; α_j is the strangeness of the j^{th} train point relative to y; and # is the element number of the finite set, namely, the number of training points whose strangeness is greater than that of i in y. The calculation formula of p is as follows:

$$p_2(\theta_i) = \frac{\#\{j = (1 \ldots n) : \theta_j \geq \theta_i\}}{n+1} \tag{11}$$

$p_1(\alpha_i)$ and $p_2(\theta_i)$ constitute double p values and serve as anomaly detection standards of the DPTCM-KNN algorithm. The detection point is deemed to be normal only when both requirements are met.

B. ALGORITHM DESIGN

As an anomaly detection algorithm based on classification, DPTCM-KNN has strong real-timeliness and supports incremental learning. It takes strangeness and independence as the judgment methods, thereby obtaining two p values and improving the detection precision of the TCM-KNN algorithm. The main steps of the algorithm are described below:

Step 1 (Calculate the Euclidean Distances Between the Training Set Samples): The DPTCM-KNN algorithm classifies the detection points into normal points and abnormal points by virtue of training, employing Eq. (7) to calculate the KNN distances between the various points and other points of the same type, namely, the Euclidian distances between the various normal and abnormal points (D_{ij}^{y} and D_{ij}^{-y}). It saves the calculation results.

The calculation of the Euclidean distance serves as the first step and the preparation phase of the TCM-KNN algorithm. It provides the basis for the subsequent strangeness and independence calculations.

Step 2 (Calculate the Strangeness and Independence of the Training set Samples): To calculate the strangeness of the various points, the DPTCM-KNN algorithm obtains the k nearest neighbors of each point in accordance with the Euclidean distances calculated in Step 1; it summarizes the k nearest neighbors to obtain $\sum_{j=1}^{k} D_{ij}^{y}$ and $\sum_{j=1}^{k} D_{ij}^{-y}$ and calculates strangeness of the various points with Eq. (8). Similarly, it obtains the independence of the various points by virtue of Eq. (9).

Step 3 (Calculate the Strangeness and Independence of the Detection Points): The algorithm calculates the Euclidean distances between i and the various points in the training set, sorts the Euclidean distances after the calculation and saves the distances between i and the k nearest neighbors. Based on the distance information, Eq. (8) and Eq. (9),

Algorithm 1 The DPTCM-KNN Algorithm

Parameters:k(number of nearest neighbors); m_1(number of normal subsets y); m_2(number of abnormal subsets -y); τ_1 and τ_2 (confidence thresholds)

Input:

i (points to be detected)

Output:

normal or abnormal

1: **for** $i = 1$ to m_1 **do**
2: Calculate the Euclidean distances D_{ij}^{y} of the points in the normal subset by virtue of Eq. (7)
3: save the results
4: **end for**
5: **for** $i = 1$ to m_2 **do**
6: Calculate the Euclidean distances D_{ij}^{-y} of the points in the abnormal subset by virtue of Eq. (7)
7: save the results
8: **end for**
9: Calculate α for each point in the normal subset by Eq. (8)
10: Calculate θ for each point in the normal subset by Eq. (9)
11: Calculate α_i of the detection point i by Eq. (8)
12: Calculate θ_i of the detection point i by Eq. (9)
13: Sort the strangeness and independence of the points in the normal subset
14: calculate p_1 and p_2 of the detection point i by Eq. (10) and Eq. (11)
15: **if** $p_1 \geq \tau_1 \&\& p_2 \geq \tau_2$ **then**
16: **return** normal
17: **else**
18: **return** abnormal
19: **end if**

it obtains the strangeness and independence of the detection point i.

Step 4 (Obtain the Double p Values of the Detection Points): The algorithm sorts the strangeness of n normal points in descending order to obtain $\{\alpha_1, \alpha_2, \cdots, \alpha_n\}$ and obtains $p_1(\alpha_i)$ by virtue of α_i and Eq. (10). Similarly, it is feasible to obtain $\{\theta_1, \theta_2, \cdots, \theta_n\}$ by sorting the independence of the normal training sets and obtain $p_2(\theta_i)$ by virtue of Eq. (11). Here, p_1 and p_2 constitute the double p values and jointly serve as the anomaly detection standards.

Step 5 (Identify the Anomalies): Anomalies of the detection points are determined by the confidence level δ and are influenced by $\tau(\delta = 1 - \tau)$. Here, τ is the significance level (0.01, 0.05 or 0.1) of the hypothesis testing. It is controlled by the central controller.

To prove that the detection point i is normal, the following requirements should be met:$p_1(\alpha_i) \geq \tau_1$ and $p_2(\theta_i) \geq \tau_2$. In other words, the detection point is normal when $p_1(\alpha_i) \geq \tau_1 \&\& p_2(\theta_i) \geq \tau_2$; otherwise, it is abnormal.

To summarize, the process of the anomaly detection algorithm is described as follows.

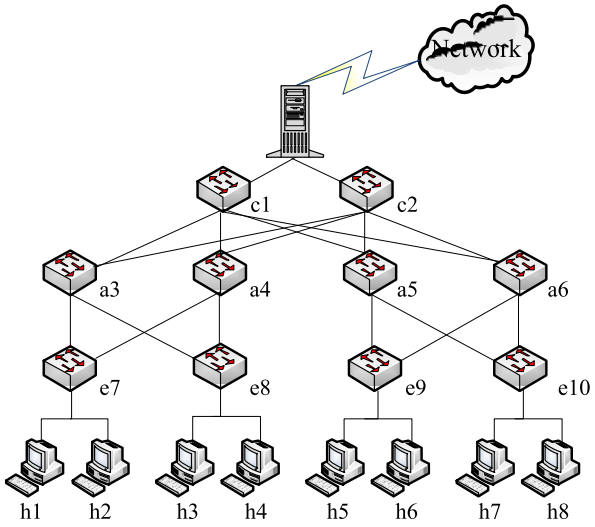

FIGURE 3. The Topology Diagram.

C. ANALYSIS OF THE ALGORITHM

Assume that the number of normal subsets is m_1 and the number of abnormal subsets is m_2. When the DPTCM-KNN algorithm is training a subset, the distance between all the sample points in the two subsets must be obtained first, and then, the strangeness and independence of all the points in the normal set are calculated. Thus, the time complexity of the entire training process is $O(m_1 * (m_1 + m_2))$. When the algorithm performs anomaly detection on the detection point i, only the distance operation between detection point and the two subset sample points can be performed to obtain its strangeness and independence. At the same time, the sorting algorithm is used to determine the double p value. The average time complexity of the DPTCM-KNN algorithm for anomaly detection is $O(m_1 log_2 m_1)$.

In summary, the DPTCM-KNN algorithm presented in this paper can achieve lower complexity in the anomaly detection phase, and it is suitable for a real-time online SDN-based anomaly detection architecture.

V. EXPERIMENTS AND DISCUSSIONS

To test the performances of the DPTCM-KNN algorithm, the paper simulates SDN environments with Mininet and an ryu controller. The experiment designs a network topology based on multiple data centers to collect flow data, as shown in Fig. 3.

In Fig. 3, c1 and c2 are core switches; a3 to a6 are aggregation switches and convergent points of edge switches; e7 to e10 are edge switches; and h1 to h8 are hosts. Since DDoS attacks are attacks of multiple computers on one target host, this experiment selects h1 as the object of attack and all of the remaining hosts to ping it, to cause a problem of resource insufficiency.

For the flows generated by the data center networks in mininet, the ryu controller collects the flow table information

TABLE 1. Summaries of the data.

Type	Size	Piece	Number of feature
Normal	2704K	42930	11
Abnormal	737K	10244	11

TABLE 2. Details of the data feature.

No.	Feature name	No.	Feature name
1	duration	7	srv_serror_rate
2	protocol type	8	dst_host_count
3	src_bytes	9	dst_host_srv_count
4	dst_bytes	10	dst_host_serror_rate
5	count	11	dst_host_srv_rerror_rate
6	srv_serror_rate		

every 10 seconds and generates samples for testing. The received flow table information is extracted, preprocessed and classified in accordance with the DPTCM-KNN algorithm. If the flows are normal, then the SDN switches do not suffer DDoS attacks; otherwise, they are attacked. Overall, 53,174 data flows were collected during the experiment. The following distribution samples were obtained through preprocessing and training.

Among them, the 11 characteristic information items of the sample are as follows.

A. PARAMETER SELECTION

In this section, the detection rate TPR is employed to evaluate the effects of the proposed DPTCM-KNN algorithm.

$$TPR = \frac{TP}{TP + FN} \tag{12}$$

where TP represents the number of anomaly flows that are identified as anomaly flows and FN represents the number of anomaly flows that are identified as normal flows. In other words, TPR is the percentage ratio of the number of correctly classified anomaly flows to the total number of anomaly flows.

K and τ are especially important to the precision of the DPTCM-KNN algorithm. If the K value is too large, then many bad positions will be introduced into the algorithm, thereby increasing the errors. If the K value is too small, then the contingency of the algorithm detection will be increased. Similarly, the values of τ_1 and τ_2 also affects the detection rate of the algorithm.

To simulate the optimal parameter combination of the DPTCM-KNN algorithm, 2000 samples are selected from the training sets, and matlab2014a is introduced for simulation under different τ_1, τ_2 and K values. The experimental results are as follows.

According to Table 3, the TPR of the DPTCM-KNN algorithm increases with a decrease in τ_1 and reaches its maximum (95.1%) when $\tau_1 = 0.01$ and $K = 20$. In Table 4, the detection rate reaches a maximum of 94.6% at $\tau_2 = 0.05$ and K=20; thus, the optimal K value for DPTCM-KNN is 20. Therefore, (K, τ_1, τ_2)=(20, 0.01, 0.05) is selected as the optimal parameter combination for the DPTCM-KNN algorithm in this paper.

TABLE 3. Relation between TPR and K, τ_1.

K	τ_1	TPR(%)
10	0.01	93.4
10	0.05	92.7
10	0.1	91.6
20	0.01	95.1
20	0.05	94.3
20	0.1	93.1
30	0.01	94.2
30	0.05	92.8
30	0.1	91.5
50	0.01	92.9
50	0.05	91.4
50	0.1	90.1
80	0.01	91.6
80	0.05	90.2
80	0.1	88.9

TABLE 4. Relation between TPR and K, τ_2.

K	τ_2	TPR(%)
10	0.01	92.7
10	0.05	93.1
10	0.1	92.3
20	0.01	94.1
20	0.05	94.6
20	0.1	93.2
30	0.01	92.8
30	0.05	93.5
30	0.1	91.2
50	0.01	91.1
50	0.05	92.8
50	0.1	89.8
80	0.01	89.9
80	0.05	91.2
80	0.1	88.4

TABLE 5. Relations between TPR and R.

R	0.25	0.5	0.75	1	1.25	1.5	1.75
TPR(%)	95.18	95.86	96.98	96.23	95.43	95.58	94.89

In addition, the detection rate of the DPTCM-KNN algorithm is also related to the number of the normal subset y and the abnormal subset -y in the training set. Let $R = \frac{m_2}{m_1}$, m_1 and m_2 are the number of normal and abnormal subsets. We randomly take 4,000 samples and use the above optimal parameters to examine the detection rate of the DPTCM-KNN algorithm with different R values.

It can be seen that the DPTCM algorithm achieves the highest TPR when R takes 0.75, in other words, when the number of normal and abnormal subsets reaches 4 to 3, the algorithm has the highest detection rate. Therefore, we use an R of 0.75 as the best R value of the DPTCM-KNN algorithm.

B. PERFORMANCE COMPARISON

This section not only employs TPR, but also adopts FPR and ACC to evaluate the algorithm performances. The formula is as follows:

$$FPR = \frac{FP}{TN + FP} \tag{13}$$

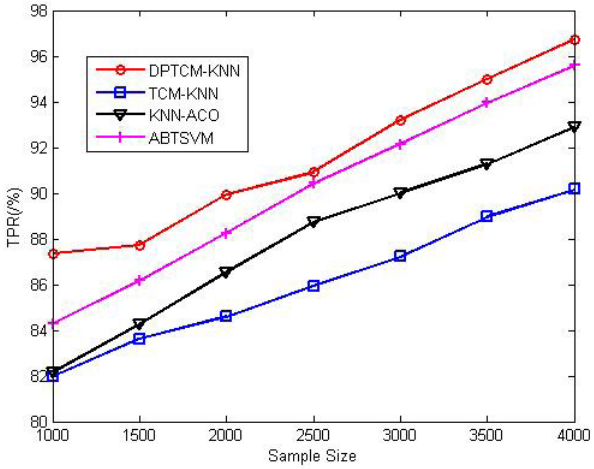

FIGURE 4. Relation between TPR and the sample size.

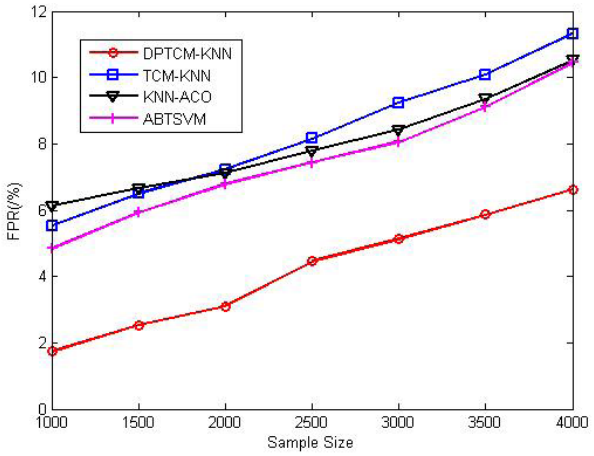

FIGURE 5. Relation between FPR and the sample size.

where FP represents the number of normal flows that are identified as anomaly flows and FN represents the number of normal flows that are identified as normal flows. In other words, FPR is the percentage ratio of the number of correctly classified normal flows to the total number of normal flows.

$$ACC = \frac{TP + TN}{TP + FP + TN + FN} \tag{14}$$

ACC, which is the percentage ratio of the number of correctly classified samples to the total number of samples, reflects the discrimination capability of the classifier.

In this paper, (K, τ_1, τ_2, R)=(20, 0.01, 0.05, 0.75) is selected as the optimal parameter combination of the DPTCM-KNN algorithm. The TCM-KNN algorithm in Document [10], the KNN-ACO algorithm in Document [11] and the ABTSVM algorithm in Document [7] are considered as control groups. The present paper performs simulation experiments on the four algorithms under different sample numbers and demonstrates the simulation results in fig. 4 to fig. 6.

The experimental results show the following: Although the detection rate TPR is higher in the ABTSVM algorithm, its false positive rate FPR has no obvious advantage. With an

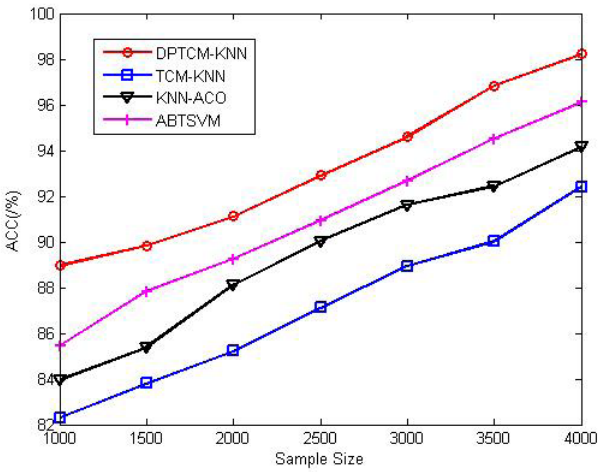

FIGURE 6. Relation between ACC and the sample size.

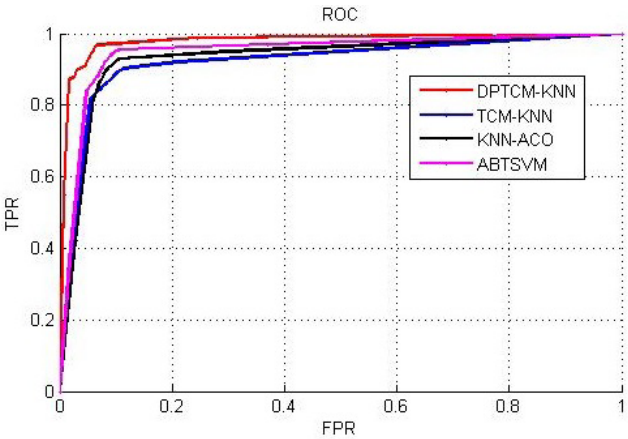

FIGURE 7. ROC plots of the algorithms.

TABLE 6. AUC of algorithms.

Algorithm	DPTCM-KNN	TCM-KNN	KNN-ACO	ABTSVM
AUC	0.9788	0.9207	0.9311	0.9487

increase in the number of samples, the TPR of the TCM-KNN algorithm has an increase rate that is relatively sluggish, and the false positive rate increases significantly, exceeding that of the KNN-ACO algorithm. The DPTCM-KNN algorithm presented in this paper has the highest detection rate TPR and accuracy ACC. When the number of samples is 4000, these can reach 96.7% and 98.2%, respectively; at the same time, it has the lowest FPR compared to the other comparison algorithms. The fig.7 shows the ROC curve of the four algorithms and calculates the AUC (Area Under Curve) to more objectively evaluate the classification performance of the algorithm.

Fig.7 and Table 6 show that the DPTCM-KNN algorithm has the largest AUC under the ROC graph. Compared with the other algorithms, the DPTCM-KNN algorithm significantly improves the classification performance of the anomaly flow. It can be seen that the DPTCM-KNN algorithm has a higher detection rate, accuracy, and low false positive rate, and it

can realize high-precision anomaly detection in the SDN environment.

VI. CONCLUSION

To solve the defects of the SDN-based flow detection methods, this paper builds an architecture for detecting anomaly flows under an SDN environment and proposes an anomaly flow detection algorithm DPTCM-KNN as an anomaly detection mechanism. The algorithm takes strangeness and independence as its dual inspection standard, which are the loopholes of the TCM-KNN algorithm in its detection, and improves the accuracy of the anomaly flow detection. Finally, through platforms such as mininet and matlab2014a, this paper conducts simulations of the algorithm. The experimental results show that the DPTCM-KNN algorithm improved the detection rate and accuracy rate of the anomaly flow detection and, at the same time, reduced the false positive rate in the process of detection, which indicates that the algorithm has good performance under an SDN environment.

REFERENCES

[1] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[2] K. Kalkan and S. Zeadally, "Securing Internet of Things (IoT) with software defined networking (SDN)," *IEEE Commun. Mag.*, to be published.

[3] J. Suárez-Varela and P. Barlet-Ros, "Towards a netflow implementation for openflow software-defined networks," in *Proc. 29th Int. Teletraffic Congr. (ITC)*, vol. 1, Sep. 2017, pp. 187–195.

[4] X. Zhao, Y. Lin, and J. Heikkilä, "Dynamic texture recognition using multiscale PCA-learned filters," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 4152–4156.

[5] L. J. Wang, Q. P. Hu, and M. Xie, "Bayesian analysis for NHPP-based software fault detection and correction processes," in *Proc. IEEE Int. Conf. Ind. Eng. Manage. (IEEM)*, Dec. 2015, pp. 1046–1050.

[6] F. Cheng and X. Qiu, "Network anomaly detection based on frequent subgraph mining approach and association analysis," in *Proc. IEEE Int. Conf. Netw. Infrastruct. Digit. Content (IC-NIDC)*, Sep. 2016, pp. 12–16.

[7] P. Burai, L. Beko, C. Lenart, and T. Tomor, "Classification of energy tree species using support vector machines," in *Proc. 6th Workshop Hyperspectral Image Signal Process., Evol. Remote Sens. (WHISPERS)*, Jun. 2014, pp. 1–4.

[8] S. T. Rutgers and M. A. Vasarhelyi, "Cluster analysis for anomaly detection in accounting data: An audit approach," *Int. J. Digit. Accounting Res.*, vol. 11, no. 17, pp. 69–84, 2011.

[9] D. Li and A. Wang, "Improved KNN algorithm for scattered point cloud," in *Proc. IEEE 2nd Adv. Inf. Technol., Electron. Automat. Control Conf. (IAEAC)*, Mar. 2017, pp. 1865–1869.

[10] Y. Li and L. Guo, "TCM-KNN scheme for network anomaly detection using feature-based optimizations," in *Proc. ACM Symp. Appl. Comput. (SAC)*, New York, NY, USA: ACM, 2008, pp. 2103–2109. [Online]. Available: http://doi.acm.org/10.1145/1363686.1364194

[11] S. Jaiswal, K. Saxena, A. Mishra, and S. K. Sahu, "A KNN-ACO approach for intrusion detection using KDDCUP'99 dataset," in *Proc. 3rd Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, Mar. 2016, pp. 628–633.

[12] P. Xiao, W. Qu, H. Qi, Y. Xu, and Z. Li, "An efficient elephant flow detection with cost-sensitive in SDN," in *Proc. 1st Int. Conf. Ind. Netw. Intell. Syst. (INISCom)*, Mar. 2015, pp. 24–28.

[13] J. Ros-Giralt, A. Commike, R. Lethin, S. Maji, and M. Veeraraghavan, "High-performance algorithms and data structures to catch elephant flows," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2016, pp. 1–7.

[14] P. Dong, X. Du, H. Zhang, and T. Xu, "A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[15] Y. Yao, "Group-ordered SPRT for decentralized detection," *IEEE Trans. Inf. Theory*, vol. 58, no. 6, pp. 3564–3574, Jun. 2012.

[16] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," *Comput. Netw.*, vol. 62, no. 5, pp. 122–136, 2014.

[17] K. Giotis, G. Androulidakis, and V. Maglaris, "Leveraging SDN for efficient anomaly detection and mitigation on legacy networks," in *Proc. 3D Eur. Workshop Softw. Defined Netw. (EWSDN)*, Sep. 2014, pp. 85–90.

[18] M. Nugraha, I. Paramita, A. Musa, D. Choi, and B. Cho, "Utilizing openflow and sflow to detect and mitigate SYN flooding attack," *J. Korea Multimedia Soc.*, vol. 17, no. 8, pp. 988–994, 2014.

[19] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2015, pp. 77–81.

[20] H. Li, P. Li, S. Guo, and A. Nayak, "Byzantine-resilient secure software-defined networks with multiple controllers in cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 4, pp. 436–447, Oct. 2014.

HUIJUN PENG was born in Nanjing, Jiangsu. She received the B.S. degree from the Nanjing University of Posts and Telecommunications in 2016, where she is currently pursuing the Ph.D. degree with the College of Computer Science. Her current research interests include software defined network, big data network, and wireless sensor network.

ZHE SUN was born in Shandong, China, in 1982. He received the B.S. degree from the Shenyang University of Technology in 2003, and the M.S. degree from Jiangsu University in 2010, and the Ph.D. degree from Zhejiang University in 2015. He is currently with the Nanjing University of Posts and Telecommunications. His research interests include evolution computation, intelligent control, and neural network.

XUEJIAN ZHAO received the M.Sc. and Ph.D. degrees in computer application technology from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2007 and 2011, respectively. He is currently an Associate Professor with the School of Modern Posts, Nanjing University of Posts and Telecommunications, Nanjing, China. His current research interests include wireless sensor networks, ad hoc networks, and big data.

SHUHUA TAN served as the Senior Director of the Innovation Research Center, YuanTong Express Co., Ltd. He has been engaged in logistics informationization planning, architecture design, and innovation and RD management. He is a member of the Shanghai Chapter of the China Computer Society Youth Computer Technology Forum and an AC member.

ZHIXIN SUN was born in Xuancheng, China, in 1964. He received the Ph.D. degree from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 1998. From 2001 to 2002, he held a post-doctoral position with the School of Engineering, Seoul National University, South Korea. He is currently a Professor and the Dean with the School of Modern Posts, Nanjing University of Posts and Telecommunications. His research interests are cloud computing, cryptography, and traffic identification.