Received August 29, 2019, accepted September 13, 2019, date of publication September 23, 2019, date of current version October 3, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2943146

An SDNFV-Based DDoS Defense Technology for Smart Cities

CHUANFENG XU[1,2,3], (Student Member, IEEE), HUI LIN[1,2,3], (Member, IEEE), YULEI WU[4], (Senior Member, IEEE), XUANCHENG GUO[1,2,3], (Student Member, IEEE), AND WENZHONG LIN[2], (Member, IEEE)

[1]School of Mathematics and Information, Fujian Normal University, Fuzhou 350117, China
[2]Fujian Provincial Key Laboratory of Information Processing and Intelligent Control, Minjiang University, Fuzhou 350121, China
[3]Fujian Provincial Key Laboratory of Network Security and Cryptology, Fujian Nomal University, Fuzhou 350117, China
[4]College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter EX4 4QF, U.K.

Corresponding authors: Hui Lin (linhui@fjnu.edu.cn) and Yulei Wu (y.l.wu@exeter.ac.uk)

This work was supported in part by the National Natural Science Foundation of China under Grant 61472008, in part by the Open Fund Project of Fujian Provincial Key Laboratory of Information Processing and Intelligent Control, Minjiang University, under Grant MJUKF-IPIC201902, and in part by the Fuzhou Science and Technology Bureau Project under Grant 2017-G-79.

ABSTRACT A software defined networking (SDN)-enabled smart city is a new paradigm that can effectively improve the cost efficiency and flexibility of data management through data-control separation. However, it faces significant security threats such as distributed denial of service (DDoS) attacks which jeopardize the security and availability of data and services by overloading the system with excessive traffic from distributed sources. To improve the DDoS defense capability and enhance the security of data management in SDN-enabled smart cities, this paper proposes a DDoS attack Defense strategy based on Traffic Classification (DDTC). We use software defined network function virtualization (SDNFV) architecture and traffic classification strategy, to improve the flexibility and reduce the load of SDN against DDoS attacks. Experimental results show that the proposed DDTC can not only launch DDoS attacks detection quickly, but also accurately track the sources of DDoS attacks. More importantly, it can reduce the risk of attack on the controller of SDN and improve the effectiveness of the system.

INDEX TERMS Distributed denial of service, flow classification, smart city, software defined networking, network function virtualization.

I. INTRODUCTION

The idea of the smart city has been widely discussed, due to its promising capability of enhancing the safety and quality of the life of urban citizens [1]. The smart city uses information and communication technologies, such as sensors and heterogeneous network infrastructure, to manage various types of data that require different kinds of processing techniques, such as cloud computing, network physical systems, and big data analytics [2]. In order to effectively utilize the collected big data and perform real-time data analysis for smart city services, software defined networking (SDN) paradigm is used to simplify resource management and provide a feasible solution for quality of service (QoS)-aware data transmission in smart cities [3].

As an emerging technology, SDN has unique features such as centralized control, programmability and flexibility. It separates the network into two planes: the control plane and the data plane. The former acts as a brain to control the entire network, and the latter forwards data through the network according to the instructions of the control plane [4]. SDN can manage data transmissions with diversified requirements through the service interface provided by the SDN controller.

Although effective solutions can be achieved through SDN, security vulnerabilities still exist [5]. The control plane acts as the core of an SDN architecture, and therefore once a security problem occurs on the control plane, it will affect, and may even destroy the entire network [6].

Distributed denial of service (DDoS) attacks provide an effective way for attackers to compromise the availability of an SDN system. SDN environment is favorable for such attacks since a DDoS attack only needs to destroy the controller in an SDN architecture to achieve system crashes [7].

The associate editor coordinating the review of this manuscript and approving it for publication was Hongxiang Li.

When a packet comes from an unmatched flow to a switch, it will be forwarded to the controller for processing. If the source addresses of the incoming packets at the switch are spoofed, which they usually are, the switch will not find a matched rule and will forward the packet to the controller. The massive DDoS spoofed packets can hold and exhaust the resources of a controller, which will make the controller unable to be accessed by the newly-arrived legitimate packets [8]. Even if there is a backup controller, it may face the same challenge [9].

Many solutions for DDoS attacks against the SDN architecture have been proposed, and they can be divided into two categories: intrinsic and extrinsic solutions [10]. Extrinsic solutions rely on the properties of traffic flows in the network [11]–[13], whereas intrinsic solutions mainly rely on the structural features of SDN environment [14], [15]. The extrinsic solution helps SDN to accurately defend against DDoS attacks by adopting high-precision methods, but the use of these methods will also bring greater computation burden to SDN [16]. However, the intrinsic solution can reduce the load on the SDN-based DDoS attack defense system compared to the extrinsic solutions. For example, when a part of the SDN network faces a large number of DDoS attacks, the network function virtualization (NFV) technology concentrates on other parts of the SDN network to help the attacked part defend against DDoS attacks by virtualizing the SDN application and deploying it in the attacked part. SDN realizes the dynamic allocation of resources through the combination of NFV technology, and avoids the threat that the attacked part of a network generates too much traffic load and cannot handle normal traffic [17]. Therefore, a new architecture, software defined network enabled network functions virtualization (SDNFV), is built in combination of NFV and SDN [18].

SDNFV relies on the centralized control of SDN to provide convenient network-wide management while SDNFV still allows network functions (NF) to perform local control for stateful packet forwarding [19]. SDNFV dynamically instantiates well-defined NF in the network to handle packet flow and reduces the burden on the SDN controller [20]. Since an attacker can initiate multiple types of attacks, and these attacks can be different in size for different switches, priority should be given to solving the severely affected part, so that the system can still maintain normal operation. In addition, SDNFV can virtualize the application in the severely affected part of a network to perform the corresponding function [20]. Therefore, SDNFV brings new opportunities for the system to achieve the defense of DDoS attacks. Based on the SDNFV architecture, this paper proposes an SDNFV-based traffic classification strategy (DDTC) to implement a recoverable DDoS attack defense system. It divides the risk of conviction and adopts appropriate strategies. The main contributions of this paper are summarized as follows:

1) We propose an SDNFV-based DDoS defense architecture that is able to virtualize network functions for reducing the load of an SDN controller and the system cost in the event of DDoS attacks.
2) An enhanced random forest (RF) algorithm based on mutual information and reinforcement learning is devised for reducing the complexity of training a model and improving the accuracy of detecting a DDoS attack.
3) Backtracking based on conditional entropy and traffic classification for each suspicious and dangerous switch is proposed, which reduces the time complexity of the current backtracking method for traversing all switches and improves the rate of traceability to the attack source.
4) We propose a flow classification strategy, named DDTC that can handle the affected parts of a network as quickly as possible to provide safer flow tables and accordingly higher security. In addition, DDTC also contributes to the resource layout of an SDNFV architecture.

The rest of the paper is organized as follows. Section II introduces related work. In Section III, we provide the background knowledge. We present the proposed model and its associated attack types in Section IV. The detailed introduction of our approach is presented in Section V, and in Section VI, the performance of our approach is confirmed with extensive experiments. Section VII concludes the paper.

II. RELATED WORK

The frequency and magnitude of DDoS attacks have been constantly increasing with detrimental impact on information and communication systems [21]. Accordingly, there is a booming body of research on this topic. In this section, we focus on SDN and SDNFV related works and discuss it. Firstly, the studies related to DDoS defense mechanisms in SDN are examined. Then the security policies based on the SDNFV architecture are reviewed to render the state-of-the-art in the field of research for DDoS attacks.

A. DDOS DEFENSE MECHANISMS IN SDN

Bu *et al.* [20] proposed a novel security mechanism that combines both the hybrid machine learning models and the enhanced history-based IP filtering scheme. In this work, detecting DDoS attack flow is through a combination of support vector machine (SVM) and self-organizing map (SOM), which increases the accuracy of detecting DDoS attacks. In addition, the history IP records can reduce the possibility of DDoS attacks masquerading IP. However, the combination of machine learning algorithms greatly increases the computational complexity of the system. Given that most DDoS attacks have the ability to disguise IP, if the attacker periodically modifies the IP, the history-based detection method will have a significant impact on the normal flow. The authors in [22] proposed a DDoS attack detection scheme based on the XGBoost algorithm to protect the SDN controller, which can handle rapidly growing network traffic. Castro-Ramos *et al.* [9] presented a DDoS attack detection method based on the entropy change of a target IP address. This work uses statistical methods to detect the presence of DDoS attacks when the entropy exceeds a pre-defined threshold. This method requires less computation, but the accuracy is not high; it is only effective for attacks without forgery IP. Giotis *et al.* [23] discussed a modular architecture for the separation of the data collection process from the SDN control plane with the employment of sFlow monitoring data. Wang *et al.* [24] proposed a safe-guard scheme (SGS) to reduce the impact of DDoS attacks on the controllers, by deploying multiple controllers in the control plane through a clustering algorithm. Software-defined-networking score (SDNScore) was proposed in [25] to make switches smarter for implementing DDoS attack detection schemes. However, the concept of "capable switch" is a controversial issue in the literature, because it has a major conflict with the concept of digital separation of SDN [26]. Piedrahita et al. presented FlowFence [13] in this work, the network effectively mitigates the DDoS attacks by limiting the bandwidth usage on the congested interface. However, this mechanism is a rate limiting mechanism that does not completely eliminate the effects of DDoS attacks.

B. SDNFV STRATEGIES FOR SECURITY

SDNFV proposes a new technology which takes advantage of SDN and NFV to strike the right balance of efficiency, flexibility, and ease of control [19]. It relies on the centralized control of SDN to provide convenient, network-wide management, while it still permits local control by NF when forwarding depends on packet state. SDNFV provides the ability to dynamically instantiate well-defined functions in the network to handle packet flows, while SDNFV reduces the burden on the SDN controller. Several solutions have been developed to overcome SDNFV security issues [17], [18], [27]. Machado *et al.* [18] proposed a solution called ANSwer which combines NFV and SDN architectures to quickly identify and handle different anomalies in different scenarios, by monitoring and analyzing the behavior of the network infrastructure. Sampaio *et al.* [17] used NFV and reinforcement learning to build an SDN detection and mitigation system. In this work, the system collects the evidence of abnormal behaviors from network metrics and coordinates the detection and the mitigation to maintain network operation by taking appropriate policies based on the reward of reinforcement learning. The authors in [27] presented a decoy chain deployment method against penetration attacks. This work considers the security status of networks and deploys the decoy chains with the resource constraints.

Aiming at the shortcomings in the above-mentioned studies, this paper proposes a scheme based on SDNFV architecture and uses traffic classification for DDoS defense through traffic classification technology, which can solve both the overload problem of DDoS attacks and defense attacks faced by SDN controllers.

III. PRELIMINARY

Before we elaborate our approach, some preliminaries need to be presented to facilitate the understanding.

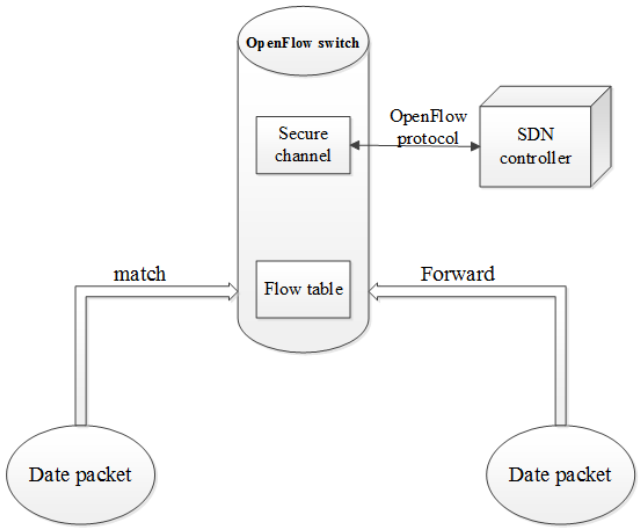


FIGURE 1. **The OpenFlow forwarding process.**

A. SDN PACKET FORWARDING MECHANISM

The OpenFlow protocol is one of the mainstream SDN southbound interface protocols, which mainly regulates the communication standard between the control plane and the data plane. Its forwarding process is shown in Figure 1. The OpenFlow switches use a flow table to match and forward packets. When a packet arrives at a switch, if the flow table in the switch has a flow entry that can match the packet, the switch performs the corresponding action. If there is no flow entry matching the packet, the switch will pass the data packet to the controller over the secure channel. The controller then determines the operation for the unmatched packet, for example, generating a new flow entry to the switch [10].

B. NFV

NFV is an emerging technology that uses software to implement network functions, which are implemented as virtual machines running on commodity servers. NFV not only provides the benefit of flexibility, but also reduces the cost by running on commodity platforms like x86- or ARM-based servers instead of specialized hardware [28]. Currently, the NFV architecture includes three key elements: network function virtualization infrastructure (NFVI), virtualized network function (VNF) and NFV management and orchestration (NFV MANO) [29]. NFVI is a combination of hardware and software resources, which constitutes the environment for deploying VNF [30]. A VNF is an implementation of NF deployed on a virtual resource such as a VM [31]. NFV MANO provides the functionality required to configure a VNF, as well as related operations such as the configuration of VNF and the infrastructure to run these functions. It includes the orchestration and lifecycle management of physical and/or software resources that support infrastructure virtualization, as well as VNF lifecycle management. It also includes a database for storing information and data models [32].

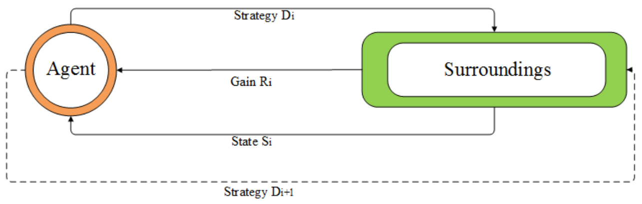


FIGURE 2. The Enhanced learning schematic.

C. REINFORCEMENT LEARNING AND RANDOM FOREST CLASSIFIER

1) REINFORCEMENT LEARNING

Reinforcement learning is a weakly supervised machine learning method [33]. It can be trained autonomously without giving annotated data in various states, and it has broad application prospects for solving complex optimization decision problems. Figure 2 depicts a typical reinforcement learning process: at each moment, the agent observes the environment and makes a decision D_i based on state S_i, while the agent gains a reward R_i and takes the next decision D_{i+1} based on the reward. After multiple iterative steps, the agent gains decision-making experience and adjusts its decision-making strategy based on the final cumulative reward after multi-step decision makings.

2) RANDOM FOREST CLASSIFIER

Random forest is a combined classifier [34]. It uses boostrap resampling method [35] to extract multiple samples from the original sample and construct a sub-data set, and then uses the sub-data set to form the basic decision tree and train it. The classification result is eventually obtained by a voting method. The training and classification processes of the random forest algorithm [36] are shown in Figure 3. According to the bootstrap idea, the original training sample set is subjected to random sampling with reentry, and k training sample subsets are obtained. The training sample subset has the same number of samples as the original training sample set. Then, a classification and regression trees (CART) is generated for each training sample subset, and a total of k CART decision trees are constructed. Next, test sample set is imported into k CART decision trees for classification. Finally, through the voting mechanism, the classification result of the largest number of votes is selected as classification results of the entire random forest. The detailed steps for random forest classification are as follows.

Let the number of samples in the training sample set S be N, and each sample has M attributes. The procedure of establishing a random forest algorithm contains K decision trees as follows [36]–[39]:

1) According to the boostrap resampling method, a new training data subset d_i with a sample number n is randomly selected from the original data set D. Then, the process is repeated k times to obtain different training data subsets $d_0, d_1, d_2, \ldots d_{k-1}$ which are different from each other.
2) A CART decision tree is established on each training data subset d_i, and a random feature vector is introduced in the training process of the decision tree. The most common method is to randomly select the F features in the original feature set to split the decision tree nodes.
3) Predictive samples are used to build classification models, that is, to construct k CART decision trees. The test sample set is then predicted by the predictive model and the classification result is obtained. Finally, all decision trees adopt a voting mechanism to vote for the best classification results as the final classification result.

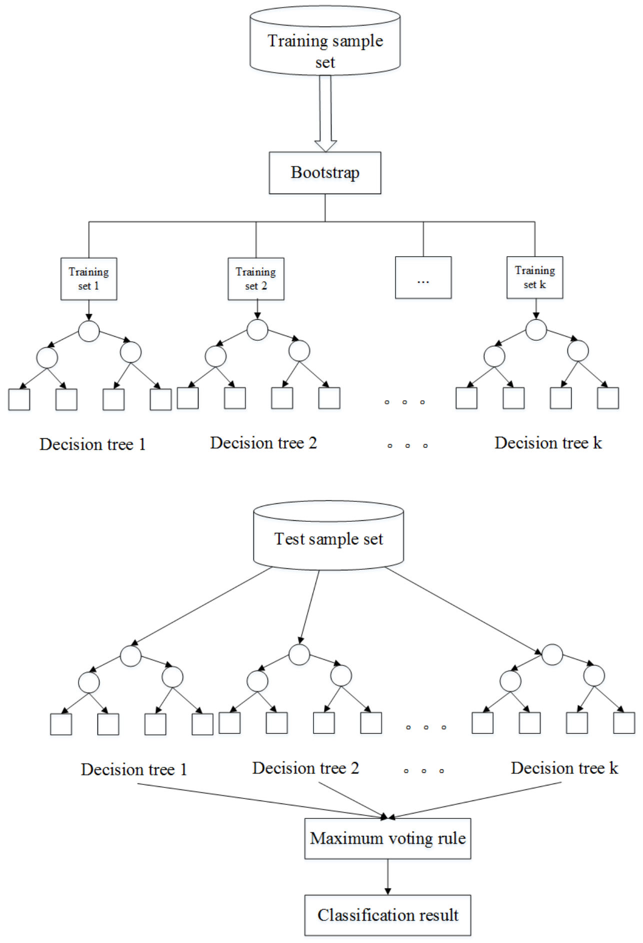


FIGURE 3. Random forest flow chart.

IV. SYSTEM MODEL AND ATTACK MODEL

In this section, we will outline the system model and present the proposed threat model.

A. SYSTEM MODEL

The proposed modules of the SDNFV architecture are illustrated as follows:

1) Control plane: The control plane includes an SDN controller and an NFV-O controller coordinated by the SDNFV application.The SDNFV application includes flow table orchestration, service chain management, and placement engine; NFV-O includes NFV orchestrator,

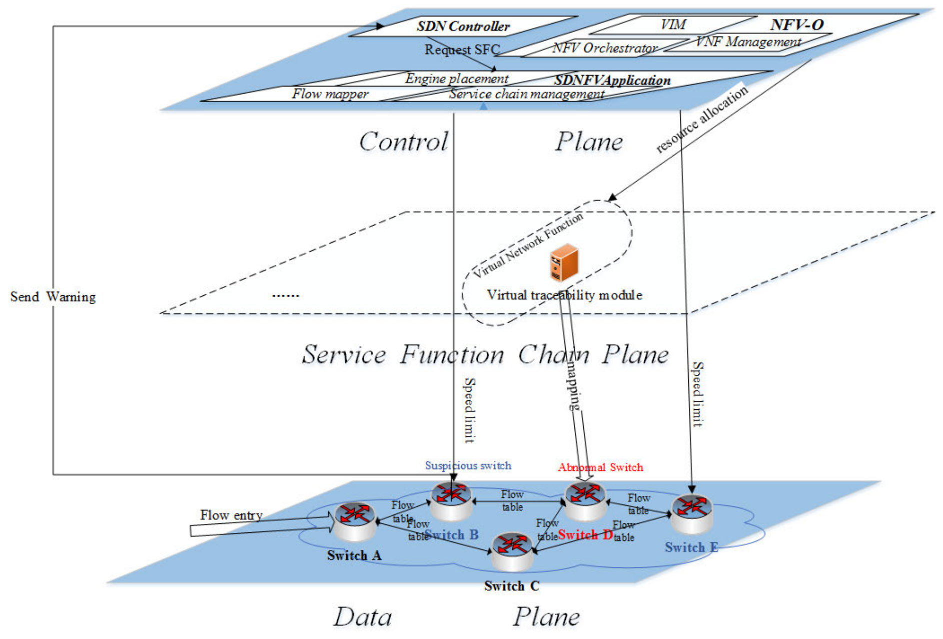


FIGURE 4. **An example of defense technology based on the SDNFV architecture.**

VNF management, and virtualized-infrastructure-manager (VIM).

2) Data plane: The data plane includes multiple switches that support the OpenFlow protocol.
3) Service function chain plane: The service function chain plane is a virtual plane composed of multiple chains.Each chain includes multiple VNF. In our DDoS defense solution, we consider only one chain used to implement the attack backtracking function.

In order to describe the working principle of the system model, an example is shown in Figure 4.In this example, there are five switches, namely A, B, C, D, and E, in the *data plane*. The *service function chain plane* has a chain of virtualized attack backtracking functions. The *control plane* has an SDN controller, an NFV-O controller,and an SDNFV application. First, the SDN controller continuously monitors the rate of packet arrivals from the switch to determine if there is any abnormal. Once the controller observes an anomaly, the corresponding suspicious switch is identified, for example the switch D. Then, the controller advertises the entire network and announces that the switch D is a suspicious switch [40], [41]. The other switches reduce collaboration with suspicious switches for packet forwarding, thereby indirectly reducing the number of packets going through the suspicious switch. After an SDN controller detects a dangerous switch, i.e., the switch D in this example, it requests the SDNFV application to build a service function chain. Then, the flow table orchestration, service chain management, and placement engine in the SDNFV application work together to build the attack backtracking service chain and map it to switch D. Finally, the switch D and the final attack source are cleared. In this way, the SDNFV architecture not only effectively removes the hazard of the suspect switch to help defend against DDoS attacks, but also provides higher security and self-processing capabilities.

B. ATTACK MODEL

The SDN environment is vertically divided into two main functional layers: the control layer and the data layer, and thus they are usually the main targets of malicious DDoS attacks [11]. The DDoS attacks of SDN can be classified into two categories: control layer DDoS attack and data layer DDoS attack, where our work focuses on the former one. The details of the two attacks are described in Figure 5.

- Date layer DDoS attack:
  1) The attacker constantly sends a large amount of new traffic to the switch.
  2) The switches detect the new packets and encapsulate them into a large number of packet_in messages. These messages are sent to the SDN controller, consuming the resources of the SDN controller.
- Control layer DDoS attack:
  3) The attacker sends new traffic with more data to the switch.
  4) While the switch requests processing from the control plane, the corresponding data packets are stored in the local database, but it is easy to cause the database to be overloaded and cannot handle normal traffic.

V. DDTC: A DDOS ATTACK DEFENCE STRATEGY BASED ON TRAFFIC CLASSIFICATION

In this section, we propose a traffic classification solution for the above-mentioned SDNFV architecture to defense DDoS attacks, namely DDTC.

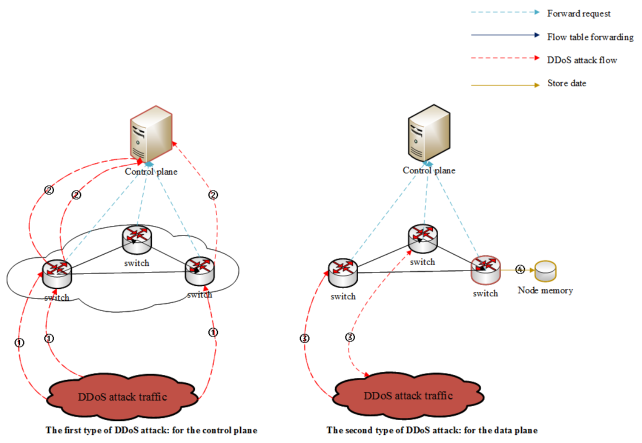


FIGURE 5. The two types of DDoS attacks in SDN.

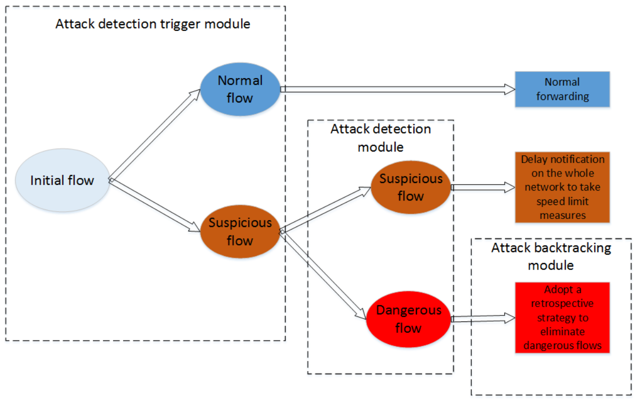

Attack detection trigger module
Normal flow
Normal forwarding
Initial flow
Suspicious flow
Attack detection module
Suspicious flow
Delay notification on the whole network to take speed limit measures
Attack backtracking module
Dangerous flow
Adopt a retrospective strategy to eliminate dangerous flows

FIGURE 6. Flow classification process and the corresponding modules.

The proposed DDTC consists of three modules: an attack detection trigger module, an attack detection module, and an attack backtracking module. Figure 6 shows the flow classification process and the corresponding modules. The *attack detection trigger module* is responsible for real-time detection of the system status. The *attack detection module* is responsible for detecting the DDoS attack flow, and the *attack backtracking module* is responsible for the path backtracking to locate the attack source. When the attack detection trigger module detects an abnormal condition, the SDN controller immediately uses the attack detection module to search the existing DDoS attack. When a DDoS attack is found, the attack backtracking module starts to work immediately. The attack backtracking module completes the path backtracking to clean up the attack source.

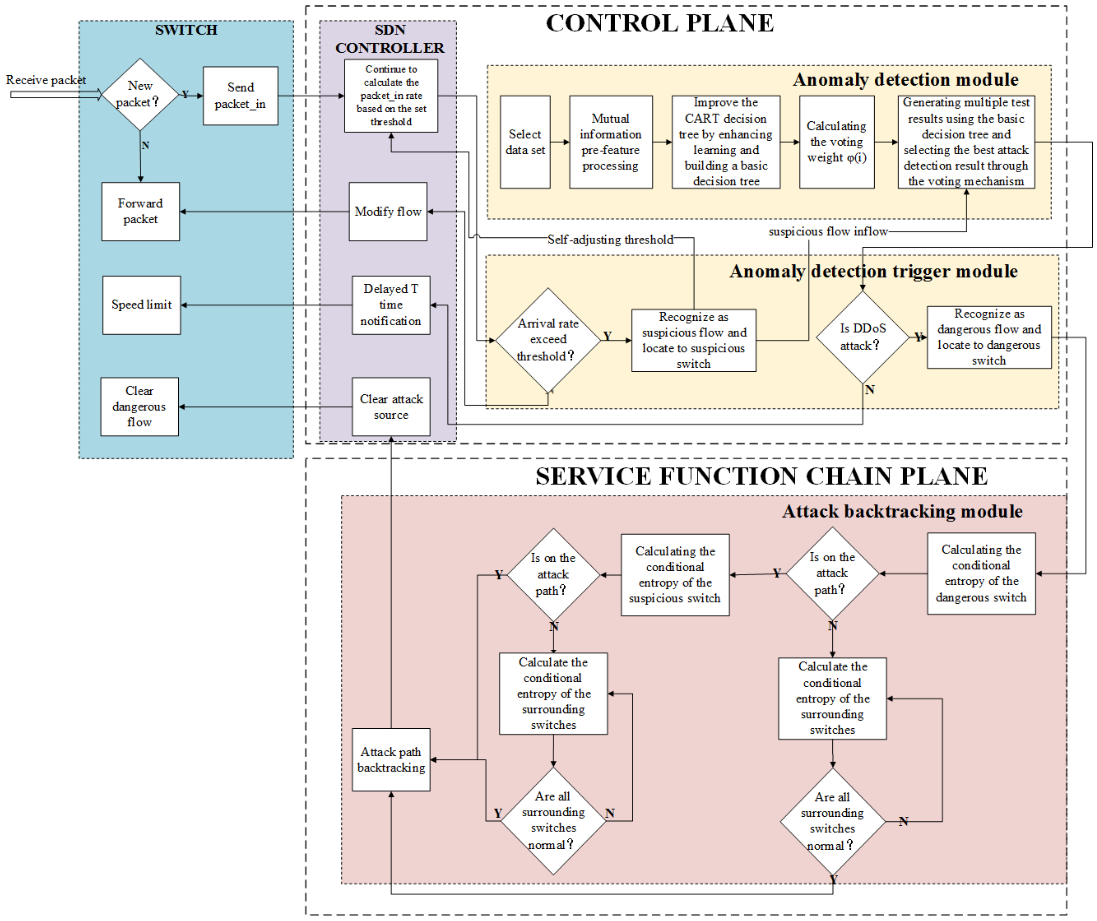
Before going into the details of DDTC operation, we first provide a higher-level description to render the overall scheme as shown in Figure 7.

1) At the beginning of the system work, the first step is to train the basic decision tree in the random forest. First, we select the dataset to perform mutual information feature preprocessing. Then, we construct multiple sub-data sets by boostrap resampling, and build a basic decision tree based on the sub-data set. Finally, the suspicious stream is obtained after the attack detection trigger module performs, and the DDoS attack detection is then completed.
2) After the system constructs the basic decision tree in the random forest, the switch starts to forward packets based on the matching flow table. When the flow enters the switch, if the match confirms that the flow is a new flow, the switch sends a packet_in message to the SDN controller. If a matching is found against the flow, the packet will be forwarded accordingly.
3) When the SDN controller accepts the packet_in message, it continues executing the attack detection trigger module. That is, the SDN controller continuously detects the packet_in message rate and compares it with the pre-defined threshold. If the detected rate exceeds the threshold in a normal case, the SDN controller performs the attack detection module and locates the abnormal switch. In addition, the SDN controller will notify the entire network after T cycles. Otherwise, it will correct the flow table for normal packet forwarding.
4) After forwarding the abnormal stream to the attack detection module, the system performs the attack detection. If the classification result shows a flow of the DDoS attack, the flow is then identified as a dangerous flow and the switch in which the flow is located is considered to be a dangerous switch. In addition, the SDN controller will advertise the switch as a dangerous switch throughout the network, so that other switches do not forward the flow to the dangerous switch. Otherwise, the abnormal flow is identified as a suspicious flow, and the switch in which it is found is identified as a suspicious switch.
5) When the system detects a dangerous flow, the system immediately performs an attack backtracking module. That is, the SDN controller notifies the SDNFV application and the NFV-O controller. Thus, the SDNFV application builds a service function chain for attack backtracking, and the NFV-O controller allocates resources to build a virtual backtracking module. After the deployment completes, the attack backtracking module performs decisions in the order of the dangerous switch, the switch around a dangerous switch, the suspicious switch, and the switch around a suspicious switch, and determines whether the switch is conditional entropy on the attack path. Finally, the attack backtracking is implemented.
6) The attack path completes the backtracking and clears the attack source.

The relevant terms and parameters of DDTC are described in Table 1, with the details explained in the following subsections.

A. ATTACK DETECTION TRIGGER MODULE

As an important factor that affects the efficiency of detection and the performance of system, the trigger mechanism of detection methods has not yet received much research attention. Previously, periodic trigger is a common detection



SWITCH
SDN CONTROLLER
CONTROL PLANE
Receive packet
New packet?
Y
N
Send packet_in
Continue to calculate the packet_in rate based on the set threshold
Forward packet
Modify flow
Speed limit
Delayed T time notification
Clear dangerous flow
Clear attack source
Anomaly detection module
Select data set
Mutual information pre-feature processing
Improve the CART decision tree by enhancing learning and building a basic decision tree
Calculating the voting weight φ(i)
Generating multiple test results using the basic decision tree and selecting the best attack detection result through the voting mechanism
Self-adjusting threshold
suspicious flow inflow
Anomaly detection trigger module
Arrival rate exceed threshold?
Recognize as suspicious flow and locate to suspicious switch
Is DDoS attack?
Recognize as dangerous flow and locate to dangerous switch
SERVICE FUNCTION CHAIN PLANE
Attack backtracking module
Is on the attack path?
Calculating the conditional entropy of the suspicious switch
Calculating the conditional entropy of the dangerous switch
Calculate the conditional entropy of the surrounding switches
Attack path backtracking
Are all surrounding switches normal?

FIGURE 7. An overall operation of DDTC.

TABLE 1. System parameters.

Term	Explanation
N	The number of abnormalities detected
δ	Threshold impact factor
T	The cycle for detecting an anomaly
Pin_num	The number of packet_in messages
$num\_threshold$	Threshold for calculating the speed of the packet_in message
$velocity\_threshold$	Threshold for determining whether the packet speed is abnormal
V_p	Packet_in message speed
D	Initial data set
k	The number of training subsets
p_i	The probability that the sample is correctly divided
P	Precision of attack detection
R	Recall of attack detection
ACC	Accuracy of attack detection
F1-Score	Mean value of precision and recall
CB	Classification benefit
$Gini\ coefficient$	Parameter for determining attribute classification effect
S_{di}	The i- switch that forwards the dangerous flow
S_{si}	The i- switch that forwards the suspicious flow
$\varphi(i)$	Voting weight of the i- attribute

mechanism in existing DDoS defense systems [42]. However, the period in the periodic trigger mechanism is difficult to determine. If the selection period is too long, the DDoS attack cannot be detected in time. If too short, the detection will start very frequently, which will waste resources such as CPU and bandwidth. Therefore, in order to improve the problem of periodic triggering, many existing solutions propose a trigger based on packet_in messages, which is called a packet_in trigger [14], [43]. The packet_in trigger mechanism has a faster response to DDoS attacks than the periodic triggering mechanism, reducing the workload of the controller and the switch. Therefore, the DDTC attack detection trigger module constructs a lightweight detection trigger mechanism through the trigger mechanism based on packet_in messages.

The packet_in message is a special message in the SDN. When the switch encounters a new flow, it will perform flow table matching. If no matching flow entry is found, the data packet is encapsulated into a packet_in message and sent to the SDN controller. In addition, in order to avoid the problem that it generates a large number of packet_in messages and causes the controller to be overloaded by DDoS attacks, we collect statistical information (IP address, Port number, etc.) of header fields through packet_in messages without collecting flow entries. Since the OpenFlow switch and SDN controller exist in the SDNFV architecture, the attack detection trigger module based on the packet_in message triggering mechanism can be naturally applied to the DDoS attack detection trigger.

The packet_in message triggering mechanism can be used for attack detection triggers, but there are still some disadvantages. For example, the static threshold in the packet_in message trigger mechanism cannot handle the variable attack size [10]. In the DDoS attack environment, the scale of DDoS attacks is random. In order to solve the above problem, we propose an improved packet_in trigger mechanism in the DDTC attack detection trigger module. The improved mechanism continuously adjusts the threshold by detection results to adapt to the ever-changing DDoS attack scale. The improved packet_in trigger mechanism consists of two modules: the packet_in message speed calculation module and the self-adjustment threshold module. The former is configured to calculate the rate of a packet_in message transmitted to the SDN controller. The latter is used to dynamically adjust the pre-defined threshold based on the detection result obtained in the last detection cycle. In addition, the *packet_in message speed calculation module* stores the packet_in message counter and records the number of packet_in messages, a predefined modulus, and a timer that records the arrival time interval of the packet_in message. The *self-adjustment threshold module* uses a self-adjusting threshold algorithm to adjust the pre-defined threshold. The self-adjusting threshold algorithm proposes a method of adjusting the threshold by counting the number of abnormalities N and adjusting threshold impact factor δ.

The detailed process of the packet_in trigger is shown in Algorithms 1 and 2. When the packet_in message arrives, the number of packet_in messages is called *Pin_num* and will increase by one. *num_threshold* is defined as a threshold and is initialized to a large number. Then, modulo operations are performed on *num_threshold* and *Pin_num* repeatedly until the resulting remainder is zero. At this time, the current time is recorded and referred to as t_{last}. When the remainder of the next modulo calculation is zero, the current time is recorded and is referred to as t_{next}. Finally, $T=t_{next}-t_{last}$ is the time interval and the period of abnormal detection. Therefore, the packet_in message rate can be calculated by Equation (1).

$$V_p=\frac{num\_threshold}{t_{next}-t_{last}} \tag{1}$$

After calculating the packet_in message rate is completed, the rate is compared with the pre-defined threshold called *velocity_threshold* to determine whether an abnormality is detected. If the calculated rate is greater than the set threshold, it is determined that there is an abnormality and the abnormal number N is increased by one. Then, the self-adjusting threshold module is performed before the abnormality detection of the next cycle is performed. *num_threshold*, T and N will be sent to the Self-adjusting threshold module. The module compares the combination of T and N to determine the degree of risk of anomalies in the system. And the pre-defined threshold is dynamically adjusted according to the comparison result. If N is increased to a certain amount, such as equal to 5, it indicates that the system has certain risks and the threshold should be lowered accordingly to help the system constantly detect abnormal conditions. When the system does not have an abnormal situation for a period of time, it proves that the system is currently not dangerous, and accordingly, increases the threshold size.

Here, once we find that the calculated rate is greater than the set rate threshold, it is determined that there is an abnormal situation and locates the corresponding switch as a suspicious switch.

Algorithm 1 Packet_in Message Abnormal Detection

```
Input: packet_in message
Output: whether the packet_in message is abnormal
 1: initialize num_threshold velocity_threshold N
 2: if a packet_in message arrives then
 3:     num= num+1
 4: end if
 5: if num mod num_threshold == 0 then
 6:     record the current time
 7:     calculate the time interval between the current time
        and the last time and set it to T
 8:     the velocity of packet_in message is calculated by the
        modulus dividing the time interval and set it to velocity
 9: else
10:     notify the network controller to handle the packet_in
        message
11: end if
12: if velocity > velocity_threshold then
13:     rocord the number of abnormalities and set it to
        N=N+1
14:     return suspicious switch;
15: else
16:     notify the controller to process packet_in messages
17: end if
```

Algorithm 2 Packet_in Message Abnormal Detection

```
Input: packet_in message
Output: whether the packet_in message is abnormal
 1: if N==5 then
 2:     Adjusted_threshold = threshold*δ
 3:     N=0
 4: else
 5:     adjusted_threshold=threshold
 6: end if
 7: if N==0 in 30T then
 8:     adjusted_threshold =threshold /δ
 9: end if
10: use adjusted_threshold for Algorithm 1
11: end
```

B. ATTACK DETECTION MODULE

After passing the attack detection trigger module, the traffic is divided into normal flow and suspicious flow. However, the suspicious flow cannot be identified as a DDoS attack. This is because attack detection trigger module also identifies the flash event as an abnormal condition [42]. Therefore, we use the attack detection module to detect the traffic generated by the DDoS attack.

Currently, many exiting researches focus on the DDoS attack detection by machine learning algorithms. Robinson et al. [44] ranked the machine learning algorithms based on DDoS classification performance. Random forest has good effects in many environments. Taking advantages of RF, the attack detection module can distinguish between benign flow entries generated by normal traffic and malicious flow entries generated by DDoS attack traffic. In the attack detection module, the SDN controller first obtains the information of the flow entry. Then, it extracts the feature value of the flow entry and sends to the trained base decision. RF is generated to detect the received stream feature to determine if it is a stream generated by a DDoS attack. So, the attack detection can be generally divided into two steps: the random forest training phase and the attack detection phase. The *random forest training phase* is responsible for constructing the basic decision tree for the training set. The *attack detection phase* is responsible for attack detection of the suspicious flow detected in the attack detection trigger module, and determines the flow generated by the DDoS attack. After the system is started, the RF training phase is first executed. Initially, multiple samples are divided by the Boostrap resampling method. Each sample then constructs a basic decision tree in the form of a segmentation of randomly selected attributes. Then the attack detection phase is performed. The detection of DDoS attacks is accomplished by generating predictions for each suspicious stream of each basic decision tree and selecting the best predictions by voting mechanisms [45].

Because of the integrated thinking of RF, there are good classification effects for many types of DDoS attacks. But there are still some shortcomings. On the one hand, the voting mechanism of RF is unreasonable. Some malicious attackers have the same voting weight, which will reduce the overall classification effect of RF. On the other hand, data correlation is an important factor that can affect the impact of random forests [43]. The lower the data correlation result in the better the RF construction. Therefore, in response to the above problems and improvements, we propose an improved RF algorithm for attack detection modules. In our attack detection module, we improve the random forest by combining reinforcement learning and mutual information. Specifically, we divide the attack detection module into three phases: feature preprocessing phase, decision tree improvement phase, and attack flow classification phase. The *feature preprocessing phase* filters the data set through mutual information. The *decision tree improvement phase* improves the classification ability of the decision tree through reinforcement learning. The *attack flow classification phase* is responsible for determining the flow of DDoS attacks from the suspicious flow classification. Firstly, k different training data subsets $d_0, d_1, d_2, \ldots d_{k-1}$ are obtained from the original data set D. Then, the feature preprocessing algorithm is used to remove the highly correlated features in the k data sets. The feature preprocessing phase cites Amiri et al.'s mutual information feature selection algorithm feature preprocessing stage [46]. In mutual information feature selection algorithm, each single input feature is added to select features set based on maximizing mutual information (MI) between selected input and output. This approach reduces the data correlation of the training set and train a random forest classifier with better classification results.

At present, there are two main methods to improve the performance of the random forest algorithm: one is to adjust the voting weight according to the decision performance, and the other is to improve the classification performance of a single decision tree. In the improvement stage of the decision tree, based on the idea of reinforcement learning, we improve the performance of the random forest algorithm by proposing the classification benefit (CB) coefficient and adding the CB coefficient to the decision tree attribute selection process. By improving the classification efficiency of each split node, the classification effect of a single decision tree is improved. For example, when considering the splitting of the *i-th* layer, the optimal splitting property is selected by evaluating the CB of the $i-1$ layer and combining *Gini coefficient*. First, the decision tree algorithm in our work selects the CART algorithm [47]. The algorithm splits by selecting the attribute of the smallest *Gini coefficient* until the end of the split. *Gini coefficient* is a probability that the samples in the sample set are misclassified. *Gini coefficient* for the data set S is:

$$\mathrm{Gini}(S)=1-\sum_{i=1}^{n} p_i^2 \tag{2}$$

where p_i^2 is the probability that the sample i is correctly divided.

Suppose that the data set S is divided into S_1 and S_2 by attribute A, and *Gini coefficient* for attribute A can be expressed as:

$$Gini_A(S)=\frac{|S_1|}{|S_2|}Gini(S_1)+\frac{|S_2|}{|S|}Gini(S_2) \tag{3}$$

For a single CART decision tree, this work assumes that the root node of the decision tree has level 1. The node level of the next level is the level of the parent node plus one. This work defines the CB of the *i-th* node obtained by F1-Score and ACC, F1-Score takes into account both the accuracy and the recall rate, which can achieve the best results. Accuracy considers the classifier's ability to classify the overall sample. Therefore, CB can be calculated from Equations (3)-(8) as follow:

$$\text{F1-Score}=\frac{2*P*R}{R+P} \tag{4}$$

$$\mathrm{P}=\frac{TP}{TP+FP} \tag{5}$$

$$\mathrm{R}=\frac{TP}{TP+FN} \tag{6}$$

TABLE 2. **Category information.**

	predictive positive class	predictive negative class
actual positive class	TP	FN
actual negative class	FP	TN

$$\mathrm{ACC} = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

$$\mathrm{CB} = \mathrm{ACC}^{*}\text{F1-Score} \tag{8}$$

As shown in Table 2, *TP* represents the number of samples that are actually positive and are correctly predicted to be positive. *TN* denotes the number of samples that are actually negative and are correctly predicted to be negative. *FP* is the number of samples that are actually negative but are incorrectly predicted to be positive. *FN* shows the number of samples that are actually positive but are incorrectly predicted to be negative.

After calculating CB, *Gini* coefficient* is defined to select the best attribute of the decision tree in a single split. *Gini* coefficient* is calculated by Equation (9), which is composed of *Gini coefficient* and CB. The smaller the *Gini coefficient* attribute and the larger of CB, the better the attribute splitting effect. Therefore, the attribute corresponding to the minimum value of *Gini* coefficient* is selected as the best split attribute of the current node during each split. According to this method, the optimal splitting attribute is selected in each split node to generate a decision tree, so as to improve the performance of decision tree classification.

$$Gini^{*} = \frac{Gini}{CB} \tag{9}$$

After improving the classification effect of a single decision tree, it is necessary to construct a more flexible voting mechanism to alleviate the reaction of the malicious decision tree or the useless decision tree to vote for the best classification result. In the attack flow classification phase, by CB, the optimal attribute of a single decision tree can be found. According to the ratio of CB of the optimal attribute of each decision tree to other decision trees, it can be determined whether the decision tree has better classification effect. Therefore, in the attack flow classification phase, the best classification model is selected by updating the voting weight of each decision tree. Firstly, the total CB^{t*} for the best attribute of a single decision tree is calculated from Equations (11) and (12). For example, let the decision tree M with c layers randomly select an attribute, then A collection of all CBs in a single decision tree is called CB^{t}. The CB^{t} of the optimal attribute in the decision tree m is CB^{t*}. Suppose there are v attributes in the decision tree. The equation below holds

$$CB^{\mathrm{t}} = CB_1 + CB_2 + CB_3 + \ldots CB_{\mathrm{m}} \tag{10}$$

$$CB^{\mathrm{t}*} = \max\left\{CB^{\mathrm{t}}_1, CB^{\mathrm{t}}_2, CB^{\mathrm{t}}_3, \ldots CB^{\mathrm{t}}_{\mathrm{n}}\right\} \tag{11}$$

where the value of m is $m = 1, 2, 3, 4 \ldots$ c-1, the value of n is $n = 1, 2, 3 \ldots .$ v

Therefore, the voting weight $\varphi(i)$ of a single decision tree is calculated by $CB^{\mathrm{t}*}$

$$\varphi(i) = \frac{CB^{t*}_i}{CB^{t*}_1 + CB^{t*}_2 + CB^{t*}_3 + \ldots CB^{t*}_k} \tag{12}$$

After completing the decision tree improvement phase, the predictive model generate multiple basic decision trees with better classification effects. Next, the attack detection phase is performed. The detection phase performs attack detection by passing the suspicious flow through the above basic decision tree. Suspicious flow detection is completed to generate multiple test results. Finally, all decision trees vote and select the results with the most votes. That is, the attack detection module completes the attack detection of the suspicious flow through the improved random forest. In addition, we provide a high-level description to present the attack detection module as follows:

1) The system starts to perform the Boostrap sampling on the training set, and selects k training subsets.
2) Apply mutual information feature selection algorithm for k training sets to remove redundant features.
3) Generate a CART decision tree from the training set and calculate the CB of each decision tree before classification.
4) Select the attribute of the minimum *Gini* coefficient* to split the process in a single decision tree splitting and construct an improved decision tree to enhance the classification performance of the decision tree.
5) Calculate the CB sum of the random attributes in a decision tree and compare it with the CB sum of the other attributes to select the largest CB sum in the decision tree.
6) The ratio of the best attribute $CB^{\mathrm{t}*}$ to all attributes CB^{t} is used as the voting weight of decision tree $\varphi(i)$.
7) Suspicious flow enters the above basic decision tree and performs attack detection. Finally, by voting weights, multiple decision trees ultimately select the final test results.
8) Therefore, through the random forest, the suspicious flow is divided into normal flow and malicious flow caused by DDoS attacks.

Consider the time complexity of the random forest algorithm of this scheme, let E be the size of the sample, F denote the number of attributes, and L represent the depth of the tree. For the improved random forest classifier in the CART decision tree stage, all attributes are used as split candidates, and an evaluation index *Gini coefficient* is calculated for it, and each layer needs to consider the upper layer of BC when splitting. So the time complexity of each layer is $O(E*F*L)$, and the time complexity of the L layer tree is $O(E*F*L^2)$. Therefore, the time complexity of the improved random forest algorithm is $O(E*F*L^2)$.

C. ATTACK BACKTRACKING MODULE

In order to improve the practicability of the defense system, the DDoS attack defense system needs to trigger the

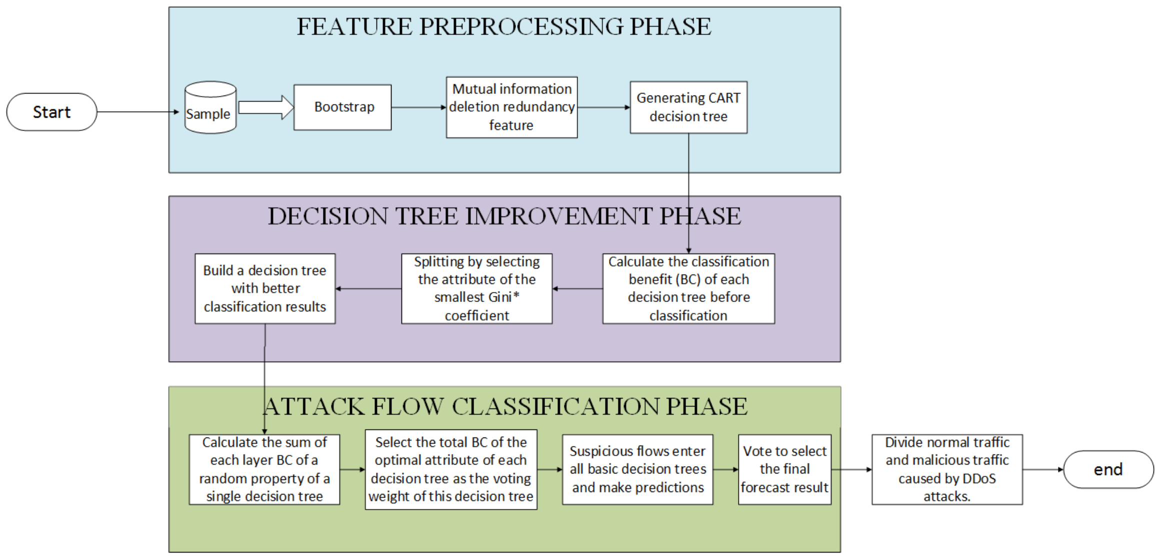

FEATURE PREPROCESSING PHASE
Start
Sample
Bootstrap
Mutual information deletion redundancy feature
Generating CART decision tree
DECISION TREE IMPROVEMENT PHASE
Build a decision tree with better classification results
Splitting by selecting the attribute of the smallest Gini* coefficient
Calculate the classification benefit (BC) of each decision tree before classification
ATTACK FLOW CLASSIFICATION PHASE
Calculate the sum of each layer BC of a random property of a single decision tree
Select the total BC of the optimal attribute of each decision tree as the voting weight of this decision tree
Suspicious flows enter all basic decision trees and make predictions
Vote to select the final forecast result
Divide normal traffic and malicious traffic caused by DDoS attacks.
end

FIGURE 8. **The flow chart for attack detection module.**

automatic mechanism to retrieve and recover the damaged service and resources to provide services that meet the service level agreement (SLA). Therefore, DDTC builds an attack backtracking module to clear the attack source and restore the system to a normal state. After passing the attack detection trigger module and the attack detection module, the DDTC divides the switches into normal switches, suspicious switches, and dangerous switches. In a normal system, the closer the switch is to the attack source, the higher the risk of becoming a dangerous switch. So, it is preferable to determine whether the switch with high dangerous level is on the attack path, and the attack backtracking effect is more obvious and takes up less load. Therefore, the DDTC attack backtracking module proposes a lightweight path backtracking mechanism.

Most of the existing attack backtracking schemes determine whether all switches are on the attack path, which increases the possibility that the defense system cannot handle a large number of attacks [14]. Since our work has classified the switches in the system, we only need to determine whether it is on the attack path according to the dangerous switches, and do not need to backtrack all the switches to complete the attack backtracking. Through the above module, we define the switch in which the dangerous flow is located as $S_{di}(i = 1, 2, 3\ldots)$ and define the switch in which the suspicious flow is located as $S_{si}(i = 1, 2, 3\ldots)$. The attack tracing method is described in detail below.

The specific steps of the lightweight attack backtracking module are as follows: when a DDoS attack is detected, the controller notifies the SDNFV application and the NFV-O to allocate resources to build a virtual backtracking module. The packet extraction header feature received by the suspicious switch port includes: the source IP address, the destination IP address, and the destination port, which are transmitted to the virtual backtracking module. First, we need to determine the switch in which the dangerous flow is located. We choose the information entropy to determine if the switch is on the attack path. The definition of conditional entropy is given below.

For the random variables X and Y, the conditional entropy of X for Y is defined as:

$$\begin{aligned} H(X|Y) &= \sum_y p(y)H(X|Y=y) \\ &= -\sum_y p(y)\sum_x p(x|y)\log_2 p(x|y) \end{aligned} \tag{13}$$

Then we calculate one conditional entropies H_1 where H_1 is conditional entropy of the source IP address with respect to the destination IP address. Firstly, we calculate the conditional entropy on S_{di} and analyze the existing results according to the set upper and lower thresholds δ_1 and δ_2 as follows:

1) When the conditional entropy is between δ_1 and δ_2, the switch is not on the attack path. Then, we judge the conditional entropy of other S_{di}. Until all dangerous switches are found, the judgment of the suspicious switch will be made.
2) When the conditional entropy is not between δ_1 and δ_2, we check its neighbors and record the attack path. We continuously determine whether the condition entropy of the surrounding switch is normal, until the condition entropy of all the switches around the switch is normal and the next S_{di} judgment is performed.

After the judgment of S_{di}, the judgment of M is performed, and the method is the same as S_{si}. Finally, the attack path of all S_{di} and S_{si} records are counted, and the attack path is backtracked to find the attack source. The flow diagram of the attack backtracking module is shown in Figure 9.

In addition, before the attack backtracking, the system's attack mitigation function is also implemented by flow classification. The system's attack mitigation function is

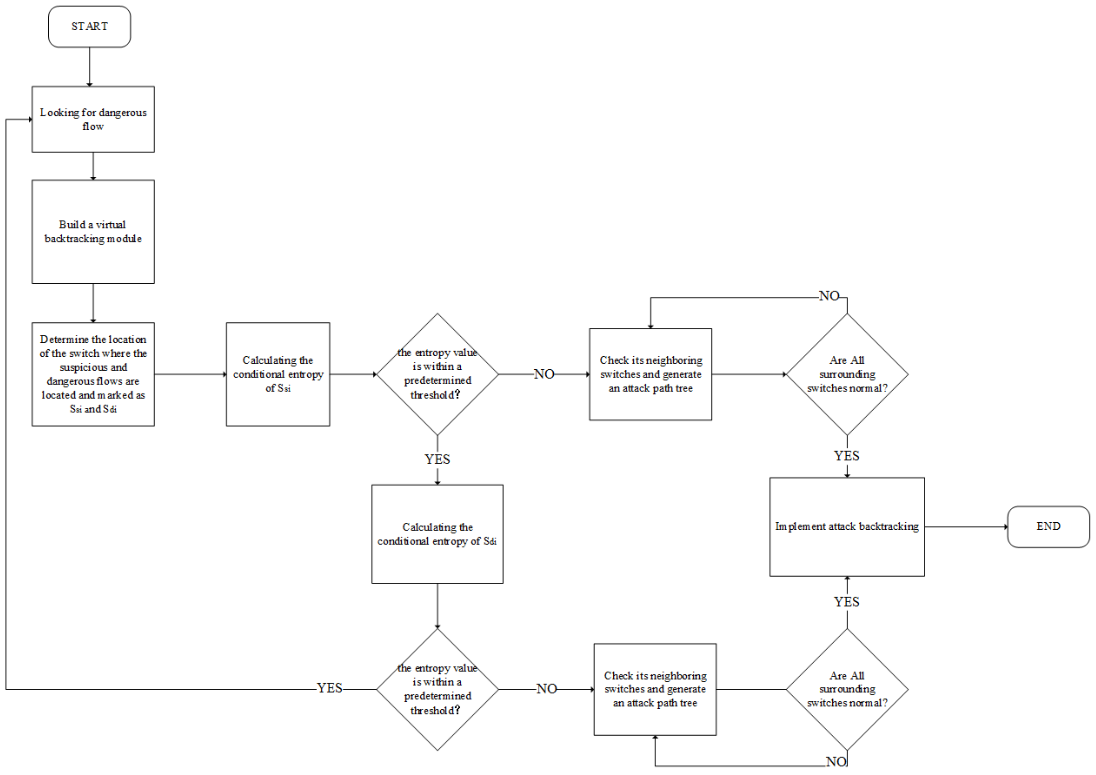

START
Looking for dangerous flow
Build a virtual backtracking module
Determine the location of the switch where the suspicious and dangerous flows are located and marked as Ssi and Sdi
Calculating the conditional entropy of Ssi
the entropy value is within a predetermined threshold?
NO
YES
Check its neighboring switches and generate an attack path tree
Are All surrounding switches normal?
Calculating the conditional entropy of Sdi
Implement attack backtracking
END

FIGURE 9. Attack backtracking module flow diagram.

implemented by reducing the interaction with dangerous switches through normal switches. When the flow is divided into normal flow and suspicious flow, the controller delays the T cycles and notifies all switches that there is a suspicious switch and marks the suspect switch. Other switches in the forwarding process identify the forwarded switch as a suspicious switch by identifying the tag. Because all switches are preferentially forwarded to the normal switch, this approach limits the rate of suspicious switches, which helps reduce the risk of suspicious switches. Moreover, once the SDN controller detects dangerous traffic, the controller immediately marks the switch where the dangerous flow is located as a dangerous switch and prohibits other switches from forwarding. Therefore, this method stops the access of dangerous switches. Ultimately, this approach effectively mitigates the risk of DDoS attacks.

VI. SIMULATIONS AND PERFORMANCE EVALUATION

In this section, the performance evaluation of the proposed defensive strategy DDTC is presented. The dataset, simulation environment, attack types, performance metrics and comparative analysis are presented in the following subsections.

A. DATASET

Due to the complexity of the DDoS attack environment, different defense DDoS attack scenarios have different focuses. In DDTC, data sets that favor the simulation of real-world environments with multiple flooding DDoS attacks show better defense in DDTC. To evaluate the performance of DDTC, we use the UNB ISCX dataset [48]. The UNB ISCX dataset is a dataset that is more in line with real network characteristics. It contains streams such as TCP, UDP, ICMP, etc. Therefore, the UNB ISCX data set satisfies the conditions of multiple DDoS attack types and marks DDoS attack network traffic. To analyze the characteristic of flows, 13 basic flow features are extracted to detect DDoS, i.e., the probability of source IP address, probability of destination IP address, length of each packet, protocol type, total packet byte, the average packet byte, variance of packet byte, standard deviation of packet byte, average of bandwidth, average of packet number, variance of packet number, the non-zero fragment of flow, the maximum fragment of flow, and the number of packets in first time fragment. We divide the entire data set into ten data sets with the equal size. The first partition and the last partition are used as training data and test data, respectively, while the other eight parts are used for the online model update process.

B. SIMILATION ENVIRONMENT

In this paper, we refer to the literature [17], [26], [49], [50]and build a simulation environment to verify the performance of the proposed DDTC. We use *Mininet* [51] as the network emulator to create a realistic virtual network, running real kernel, switch and application code. It is also a great way to develop and experiment with OpenFlow and SDN systems. Our *Mininet* environment is integrated with *Ryu* [52] as the

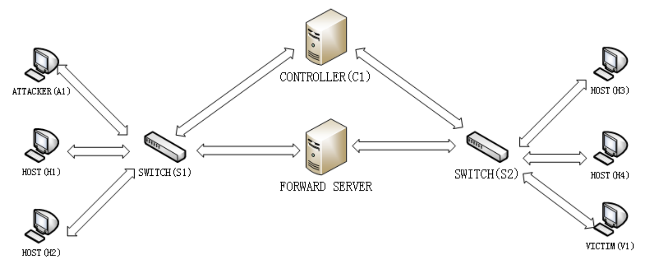


FIGURE 10. The network topology used in the experiment.

SDN controller. In addition, the network functions of the NFV and service chain are implemented in *Docker* [53] containers; they share the kernel with the host system, but isolation is provided between applications, as in virtual machines. The experiments are conducted on a computer with Intel i7-4710MQ 2.5 GHz (4 cores) processor and 8G RAM.

Figure 10 depicts the topology used for the experiment. There are two switches ($S1$, $S2$) that are managed by the controller ($C1$). The switches are controlled by the controller and forwarded through the forwarding server. Each switch is connected to three hosts. One of the hosts ($A1$) on the S1 switch was attacked by a malicious user and began DDoS attacks against $C1$, causing the controller to fail to process other hosts named Victims ($V1$). There are also four hosts ($H1, H2, H3, H4$) that are neither victims nor attackers, generating legitimate traffic on the network. An attacker in this topology simulates a real DDoS attack by generating a malicious packet flow from a different IP address. Although this is not a large-scale topology, an attacker can generate malicious packet streams that appear to come from different IP addresses. Therefore, the environment can simulate a larger topology in practice. In addition, the experimental setup makes it easier to measure and evaluate performance-related phenomena to more accurately determine the effectiveness of our approach.

C. ATTACK TYPES

In the experiment, we suppose that multiple hosts try to attack one host using $TFN2K$ [54] with IP Spoofing. $TFN2K$ is a well-known DDoS attack tool, which has been widely used to attack several large famous sites. At the same time, $TFN2K$ can produce most types of attacks: UDP flood attack, DNS Amplification attack, NTP attack, and mixed attack containing various kinds of attack.

TCP SYN Flood Attack: The protocol type of an attack packet is TCP, and the TCP flag is set to SYN flag. Other attributes are randomized.

DNS Amplification Attack: The protocol type of attack packets is DNS, and the destination port is set to 53. Attack packet size is 60 bytes. Other attributes are randomized.

NTP Attack: The protocol type of attack packets is NTP, and the destination port is set to 123. Attack packet size is 90 bytes. Other attributes are randomized.

Generic Attack: All attributes are selected randomly in their respective ranges. This can be stated as an unfamiliar type of attack since packet characteristics do not manifest a statistical coherence.

In DDTC, we mainly study DDoS attacks against SDN controllers. The most common DDoS attacks for SDN controllers are flooding DDoS attacks such as TCP/SYN flooding, UDP flooding, and ICMP flooding and hybrid attacks [21]. Therefore, we simulate a flooding DDoS attack in a simulated environment.

D. PERFORMANCE METRICS

Different performance metrics measured by different modules in DDTC. In the attack detection trigger module, when DDTC faced with DDoS attacks, the following performance indicators are used for evaluation.

Mean response time: the mean time interval from the attack's starting time to the detection module's starting time.

Mean utilization ratio of CPU: An indicator for assessing network load by averaging the CPU load when no attack occurred and the CPU load at the time of the attack.

In the attack detection module, the indicators in the information retrieval are often used for evaluation.

True Positive (TP): the number of attacks correctly detected as attacks.

True negative (TN): the number of normal traffic correctly detected as normal

False Negative (FN): the number of attacks incorrectly detected as normal

False Positive (FP): the number of normal traffic incorrectly detected as attack

True Positive Rate (TPR): $TPR = \frac{TP}{TP+FN}$

False Positive Rate (FPR): $FPR = \frac{FP}{FP+TN}$

Accuracy (ACC): $ACC = \frac{TP+TN}{P+N}$

In the attack backtracking module, the conditional entropy H(Source IP | Destination IP) is usually used for evaluation.

H (Source IP | Destination IP): uncertainty of source IP under known destination IP conditions.

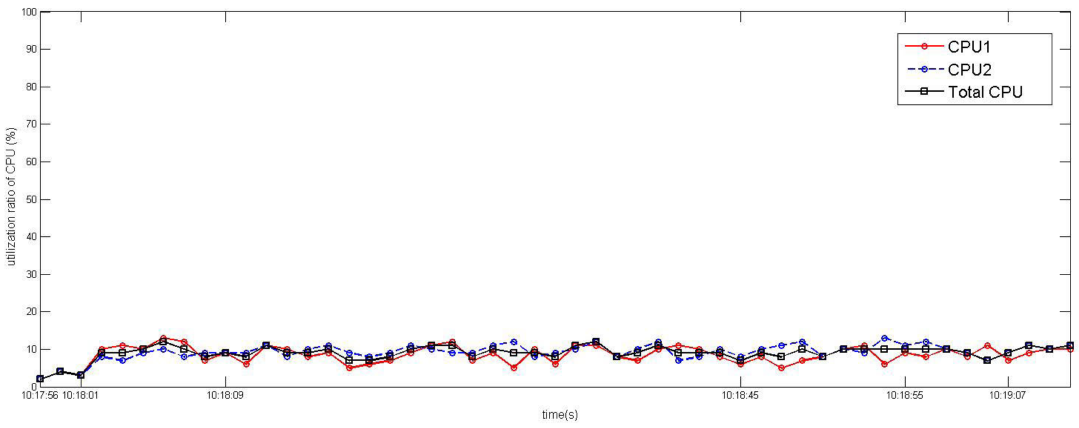
E. COMPARATIVE ANALYSIS

It is important to compare the performance of each module in the proposed DDTC with the latest or classic related mechanisms. In order to achieve this purpose, we ensure that the other modules remain unchanged during each experiment. Each of our experiments only changes the modules that need to be evaluated.

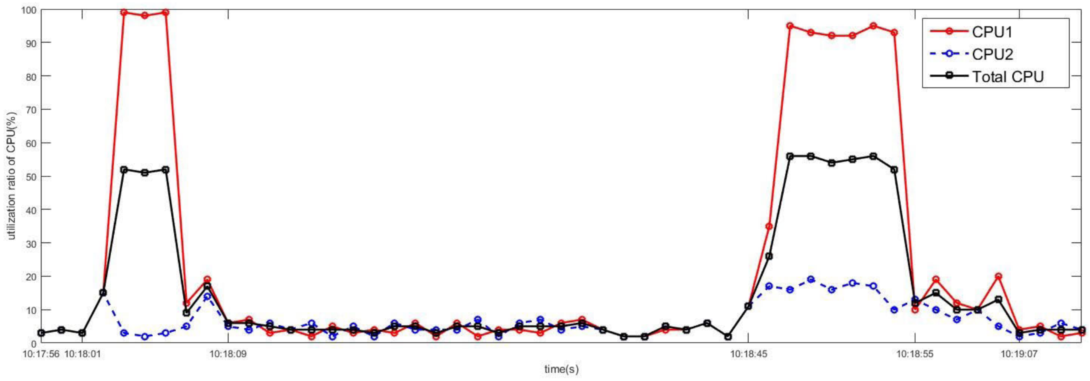
1) EVALUATING OVERALL PERFORMANCE

First, we study how DDTC works under normal conditions and in a DDoS attack environment. In a normal environment, all CPUs run at a lower utilization rate. In a DDoS attack environment, the total CPU runs at an average utilization rate of 50% when the detection trigger module detects a DDoS attack. In addition, we compare DDTC with the SD-Anti-DDoS [14] of the SDN architecture.

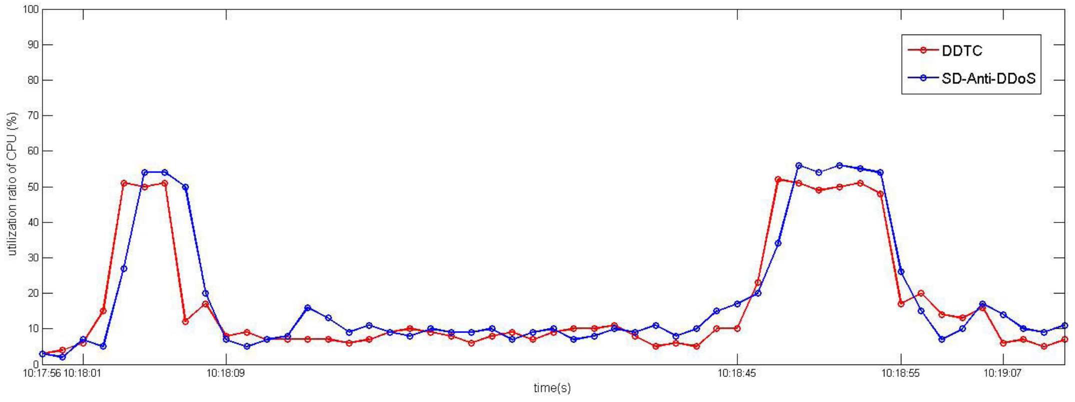
The CPU utilization of DDTC in a normal environment is shown in Figure 11(a). In this simulation, we determine



(a) Utilization ratio of CPU of controller in normal environment



(b) Utilization ratio of CPU of controller in attack environment



(c) Comparison of average CPU utilization between DDTC and SD-Anti-DDoS

FIGURE 11. Utilization ratio of CPU of controller over time.

the defensive effect of DDTC by observing the change in utilization of the two CPUs over time. In the experiment we chose the time period of 10:17:56 10:19:10. The results show that the traffic classification strategy makes the DDTC CPU utilization lower in the normal environment, only 8%. Therefore, it is proved that DDTC has a lower load in a normal environment.

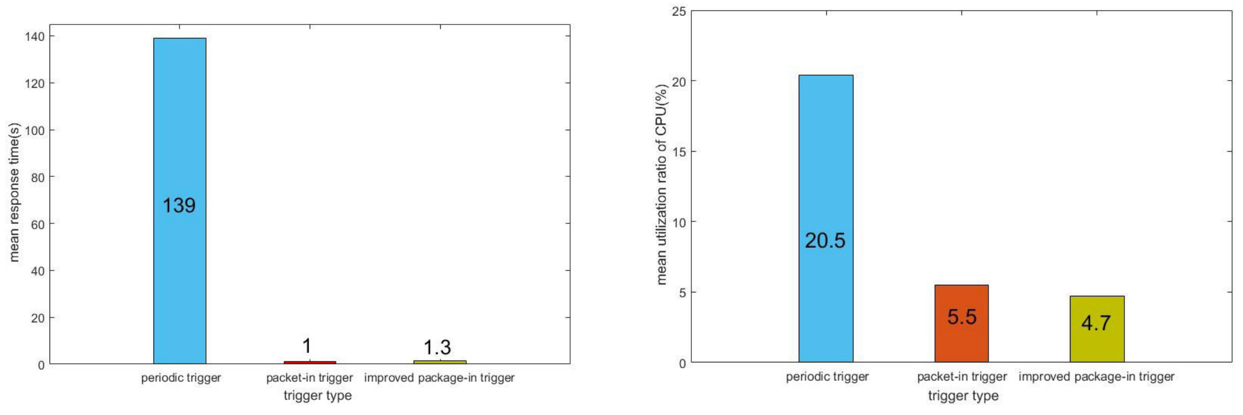
We also analyze the CPU utilization in the attack environment and compare the results in Figure 11(b) with Figure 11(a), demonstrating that the DDTC is effective against DDoS attacks and has self-healing capabilities. The specific time of each module in the DDTC is described in Table 4. At the beginning of the initial state, the random forest model is first trained using a training data set, starting at 10:18:01 for

TABLE 3. Actions of each module over time in simulation.

State	Event	Time
Initial State	Application starts Random forest classifier training starts Random forest classifier training ends	10:17:56 10:18:01 10:18:09
Attack detection trigger module	Attack detection trigger module starts Detect abnormality	10:18:09 10:18:45
Attack Backtracking Module	Attack backtracking module starts Successful backtracking Flow table clean module starts	10:18:55 10:19:07 10:19:08

TABLE 4. Attack backtracking module result.

	Event	location
Results of attack backtracking	Attack destination Attack source Attack path	V1 A1 V1,S2,S1,A1



(a) Mean response time

(b) Mean utilization ratio of CPU

FIGURE 12. Utilization ratio of CPU of controller over time.

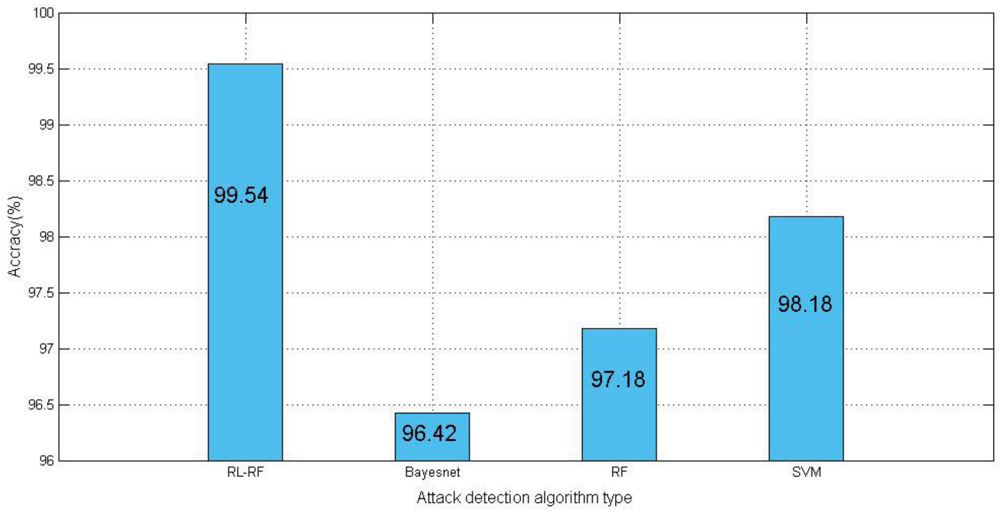
8 seconds. At 10:18:45, the attack detection trigger module finds that the speed of the packet_in message is abnormal, and generates a start command for the attack detection module. The attack detection module starts at 10:18:45. The test is completed at 10:18:55. After that, the attack backtracking module starts at 10:18:55 and successfully tracks the attack path at 10:19:07. After the attack backtracking module is completed, the system clears the dangerous switch and returns to normal.

The CPU utilization comparison between DDTC and SD-Anti-DDoS is shown in Figure 11(c). In this simulation, we compare the average CPU utilization of DDTC with SD-Anti-DDoS. The results show that DDTC with SDNFV technology has lower CPU utilization, which is between 5% and 51%. First, in DDTC, not all functions require SDN controllers for allocation management, which increases the security of DDTC attacks against DDoS. Secondly, based on the traffic classification strategy, the idea of priority processing according to risk is realized, so that it is not necessary to process all switches at the same time, and the load of the SDN controller is alleviated. Thirdly, the improvements in the various module methods also help reduce CPU utilization during DDTC operation. Moreover, the random forest training phase and attack detection trigger module of DDTC are earlier than the corresponding functional modules in the SD-Anti-DDoS mechanism. However, DDTC also has problems such as long attack detection processes. This is due to the large time complexity of the algorithm in the attack detection module.

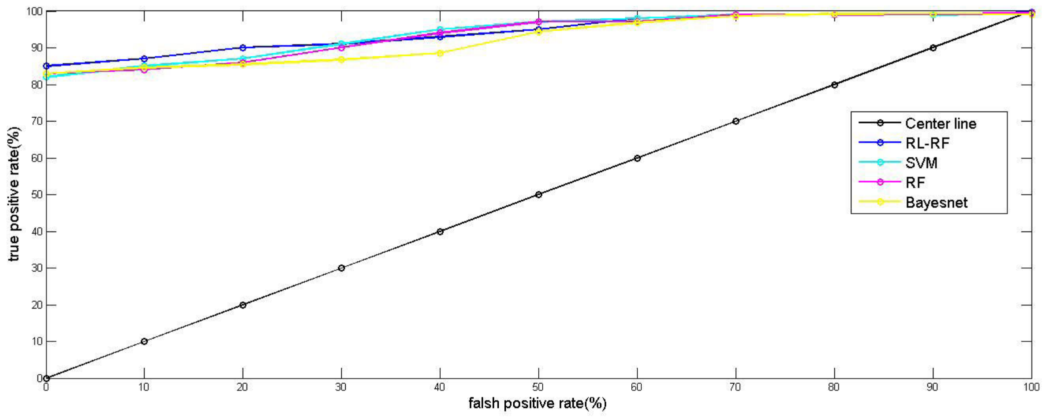
2) PERFORMANCE TEST OF THE ATTACK DETECTION TRIGGER MODULE

In what follows, we analyze the attack detection trigger module in the DDTC system and compare it with the common attack detection trigger mechanism, especially the periodic trigger mechanism and the existing packet_in trigger schemes. Our attack detection trigger mechanism is compared with the literature [23] and SD-anti-DDoS [14].

The mean response time comparison of the cycle trigger, the packet_in trigger, and the improved packet_in trigger is shown in Figure 12(a). The results show that the packet_in trigger response time is much smaller than the periodic



(a) Accuracy comparison



(b) Comparison of TPR and FPR

FIGURE 13. Comparison of detection accuracy of the algorithm.

trigger. However, the improved packet_in trigger response time is slightly higher than the existing packet_in trigger. The self-adjusting threshold algorithm needs to continuously increase its own threshold according to system changes. Since the self-adjusting threshold algorithm obtained less feedback at an early stage, the improved packet_in trigger was not as effective as the existing solution in the early days of the system.

In addition, the mean utilization of the CPU of the periodic trigger and the packet_in trigger is shown in Fig 12(b). The results show that the packet_in triggers CPU utilization much less than the periodic trigger. Furthermore, the improved packet_in trigger has a better performance in CPU utilization. The improved packet_in trigger has a better performance in CPU utilization. Since the packet_in trigger in the work attack detection trigger module has a lower CPU load and network load, due to continuous adjustment of the detection threshold. No additional determination relative to the existing packet_in trigger is required.

Therefore, the packet_in trigger can significantly reduce the response time of the attack while having less controller load. Experimental results show that the improved packet_in flip-flop can achieve lower load and shorter response time. The improved packet_in trigger can thus be used to detect DDoS attacks against SDN controllers.

3) PERFORMANCE TEST FOR ATTACK DETECTION

We also evaluate the random foresting algorithm based on reinforcement learning (RL-RF) in DDTC and the existing attack detection algorithm by comparing the performance indicators ACC and ratio of TPR to FPR. The selected attack detection algorithms include random forest, Bayesnet, and SVM.

The accuracy comparison of various algorithms is shown in Figure 13(a). The results show that the RL-RF algorithm has the highest accuracy, reaching 99.54%. Compared with the random forest algorithm without combined reinforcement learning, it has better effect.

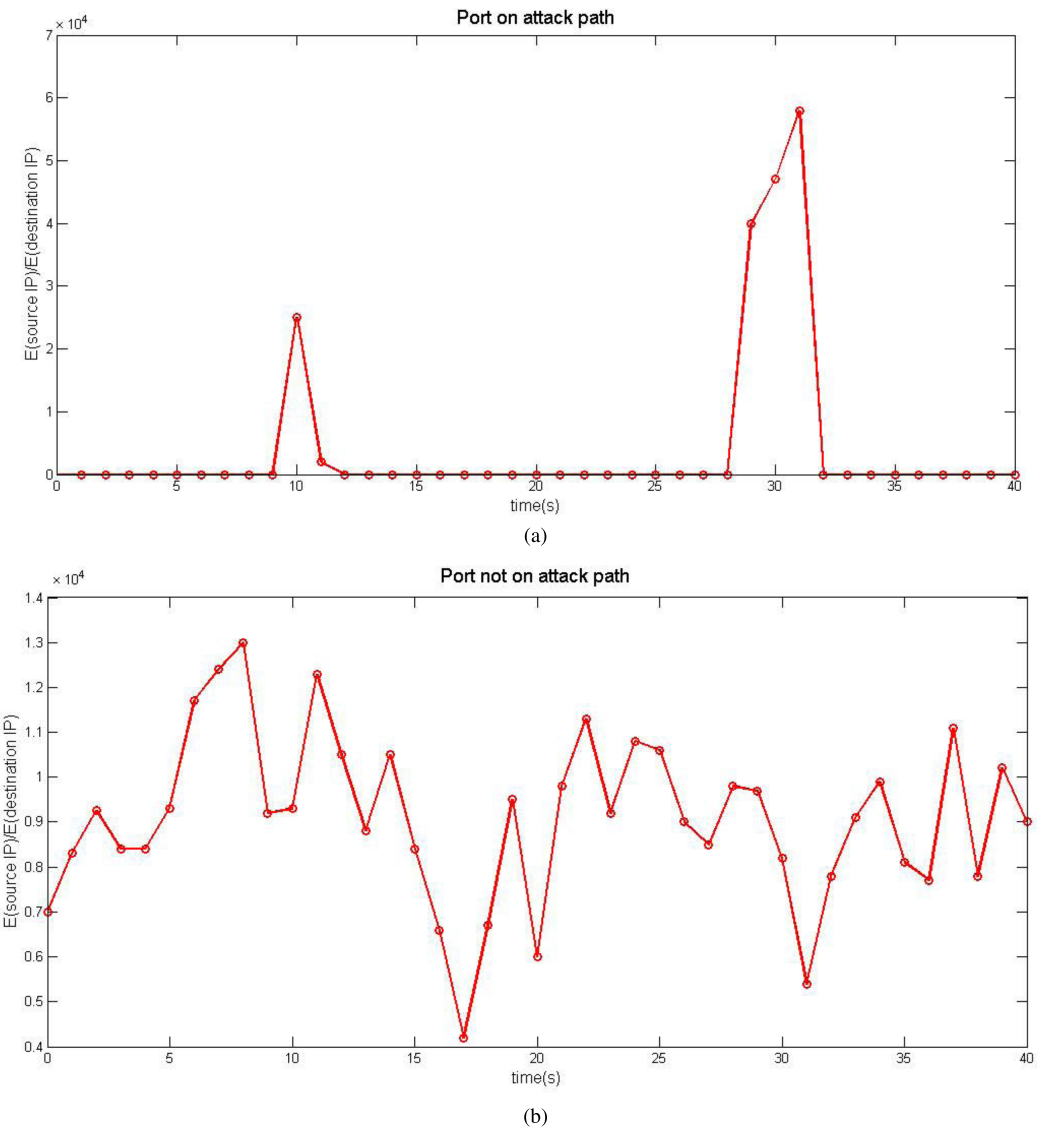


FIGURE 14. **Comparison of ports on the attack path and ports not on the attack path.**

In addition, the ratio of TPR to FPR of various attack detection algorithms is shown in Figure 13(b). The results show that multiple detection algorithms have better results with increased detection time. Specifically, the RL-RF algorithm has a higher ratio in the initial state than the other detection algorithms, about 85%. That is, when the detection time is short, the RL-RF algorithm performs the detection of the DDoS attack well, compared to other detection algorithms.

Thus, the RL-RF algorithm is used in DDTC and can accurately detect DDoS attacks against SDN controllers.

4) ATTACK BACKTRACKING EFFECT TEST

Finally, we assess whether DDTC can trace back to the source of the attack. The result proves that the attack backtracking module in DDTC can complete the attack path backtracking. Table 4 describes the traceability results of the DDTC defense process used in Figure 11(b).

First, it is known that $V1$ is the victim host, and then the attack path is determined by conditional entropy. Therefore, it is found that $S1$, $S2$ and $A1$ are on the attack path. At this point, the complete attack path is $A1 \rightarrow S1 \rightarrow S2 \rightarrow V1$.

The conditional entropy of the switch $S1$ port in the absence of attack and the presence of attack $S1$ port is shown in Figure 14. The results show that the variation of conditional entropy on the attack path is very large, while the change of conditional entropy on the non-attack path is small, and the corresponding value is also small.

Therefore, the attack backtracking module in the DDTC can complete the path backtracking, thereby accurately clearing the DDoS attack source in the system.

VII. CONCLUSION

In this work, the DDTC based on the SDNFV architecture has been proposed for DDoS attacks in an SDN environment. The DDTC has three modules, namely attack detection trigger, attack detection, and attack backtracking. In the attack detection trigger module, the packet_in message trigger algorithm has been improved. The attack detection trigger module reduces the load by continuously adjusting the defined threshold. The results have shown that the algorithm only accounts for about 86% of the load of the existing packet_in message trigger algorithms. In the attack detection module, the random forest classifier has been improved by introducing reinforcement learning, so that the accuracy of the classification module reaches 99.54%. In addition, it has also been very effective for unfamiliar attack types. But we analyzed that the algorithm has a high time complexity which needs to be further improved. In addition, we have verified that the attack backtracking module can perform path backtracking well without causing a large load.

REFERENCES

[1] M. Batty, K. W. Axhausen, F. Giannotti, A. Pozdnoukhov, A. Bazzani, M. Wachowicz, G. Ouzounis, and Y. Portugali, "Smart cities of the future," *Eur. Phys. J. Special Topics*, vol. 214, no. 1, pp. 481–518, Nov. 2012.

[2] S. R. Bhavani, J. Senthilkumar, A. G. Chilambuchelvan, D. Manjula, R. Krishnamoorthy, and A. Kannan, "CIMIDx: Prototype for a cloud-based system to support intelligent medical image diagnosis with efficiency," *JMIR Med. Informat.*, vol. 3, no. 1, p. e12, 2015.

[3] A. Volkov, A. Khakimov, A. Muthanna, R. Kirichek, A. Vladyko, and A. Koucheryavy, "Interaction of the IoT traffic generated by a smart city segment with SDN core network," in *Proc. Int. Conf. Wired/Wireless Internet Commun.*, 2017, pp. 115–126.

[4] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[5] M. C. Dacier, H. König, R. Cwalinski, F. Kargl, and S. Dietrich, "Security challenges and opportunities of software-defined networking," *IEEE Security Privacy*, vol. 15, no. 2, pp. 96–100, Mar./Apr. 2017.

[6] A. Shoeb and T. Chithralekha, "Resource management of switches and Controller during saturation time to avoid DDoS in SDN," in *Proc. IEEE Int. Conf. Eng. Technol. (ICETECH)*, Mar. 2016, pp. 152–157.

[7] J. Cui, M. Wang, Y. Luo, and H. Zhong, "DDoS detection and defense mechanism based on cognitive-inspired computing in SDN," *Future Gener. Comput. Syst.*, vol. 97, pp. 275–283, Aug. 2019.

[8] H. D. Zubaydi, M. Anbar, and C. Y. Wey, "Review on Detection Techniques against DDoS attacks on a software-defined networking controller," in *Proc. Palestinian Int. Conf. Inf. Commun. Technol. (PICICT)*, 2017, pp. 10–16.

[9] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," in *Proc. Int. Conf. Comput.*, 2015, pp. 77–81.

[10] X. You, Y. Feng, and K. Sakurai, "Packet in message based DDoS attack detection in SDN network using OpenFlow," in *Proc. 5th Int. Symp. Comput. Netw. (CANDAR)*, 2017, pp. 522–528.

[11] Q. Yan, R. Yu, Q. Gong, and J. Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 602–622, 1st Quart., 2016.

[12] K. Kalkan, G. Gur, and F. Alagoz, "Defense mechanisms against DDoS attacks in SDN environment," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 175–179, Sep. 2017.

[13] A. F. M. Piedrahita, S. Rueda, D. M. F. Mattos, and O. C. M. B. Duarte, "Flowfence: A denial of service defense system for software defined networking," in *Proc. Global Inf. Infrastruct. Netw. Symp. (GIIS)*, 2015, pp. 1–6.

[14] Y. Cui, L. Yan, S. Li, H. Xing, P. Wei, Z. Jian, and X. Zheng, "SD-Anti-DDoS: Fast and efficient DDoS defense in software-defined networks," *J. Netw. Comput. Appl.*, vol. 68, pp. 65–79, Jun. 2016.

[15] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 413–424.

[16] S. Murtuza and K. Asawa, "Mitigation and detection of DDoS attacks in software defined networks," in *Proc. 11th Int. Conf. Contemp. Comput. (IC3)*, 2018, pp. 1–3.

[17] L. S. R. Sampaio, P. H. A. Faustini, A. S. Silva, L. Z. Granville, and A. Schaeffer-Filho, "Using NFV and reinforcement learning for anomalies detection and mitigation in SDN," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2018, pp. 432–437.

[18] C. C. Machado, L. Z. Granville, and A. Schaeffer-Filho, "ANSwer: Combining NFV and SDN features for network resilience strategies," in *Proc. Comput. Commun.*, Jun. 2016, pp. 391–396.

[19] W. Zhang, G. Liu, A. Mohammadkhan, J. Hwang, K. K. Ramakrishnan, and T. Wood, "SDNFV: Flexible and dynamic software defined control of an application-and flow-aware data plane," in *Proc. 17th Int. Middleware Conf.*, 2016, p. 2.

[20] C. Bu, X. Wang, M. Huang, and K. Li, "SDNFV-based dynamic network function deployment: Model and mechanism," *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 93–96, Jan. 2018.

[21] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2046–2069, 4th Quart., 2013.

[22] Z. Chen, F. Jiang, Y. Cheng, X. Gu, W. Liu, and J. Peng, "XGBoost classifier for DDoS attack detection and analysis in SDN-based cloud," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Jan. 2018, pp. 251–256.

[23] 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.

[24] Y. Wang, T. Hu, G. Tang, J. Xie, and J. Lu, "SGS: Safe-guard scheme for protecting control plane against DDoS attacks in software-defined networking," *IEEE Access*, vol. 7, pp. 34699–34710, 2019.

[25] K. Kalkan, G. Gür, and F. Alagöz, "SDNScore: A statistical defense mechanism against DDoS attacks in SDN environment," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2017, pp. 669–675.

[26] K. Kalkan, L. Altay, G. Gür, and F. Alagöz, "JESS: Joint entropy-based DDoS defense scheme in SDN," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2358–2372, Oct. 2018.

[27] A. Mohammadkhan, G. Liu, W. Zhang, K. K. Ramakrishnan, and T. Woodv, "Protocols to support autonomy and control for NFV in software defined networks," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2015, pp. 163–169.

[28] B. Rashidi, C. Fung, and M. Rahman, "A scalable and flexible DDoS mitigation system using network function virtualization," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Apr. 2018, pp. 1–6.

[29] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 1st Quart., 2016.

[30] R. Mijumbi, J. Serrat, and J.-L. Gorricho, "Self-managed resources in network virtualisation environments," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2015, pp. 1099–1106.

[31] J. Cao, Y. Zhang, W. An, X. Chen, J. Sun, and Y. Han, "VNF-FG design and VNF placement for 5G mobile networks," *Sci. China Inf. Sci.*, vol. 60, no. 4, 2017, Art. no. 040302.

[32] M.-A. Kourtis, M. J. McGrath, G. Gardikis, G. Xilouris, V. Riccobene, P. Papadimitriou, E. Trouva, F. Liberati, M. Trubian, J. Batallé, H. Koumaras, D. Dietrich, A. Ramos, J. Ferrer Riera, J. Bonnet, A. Pietrabissa, A. Ceselli, and A. Petrini, "T-NOVA: An open-source MANO stack for NFV infrastructures," *IEEE Trans. Netw. Service Manage.*, vol. 14, no. 3, pp. 586–602, Sep. 2017.

[33] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[34] G. Fanelli, J. Gall, and L. Van Gool, "Real time head pose estimation with random regression forests," in *Proc. CVPR*, 2011, pp. 617–624.

[35] M. Chen, C. Zhang, and S. Chen, "Semantic event extraction using neural network ensembles," in *Proc. Int. Conf. Semantic Comput. (ICSC)*, 2007, pp. 575–580.

[36] X. Xu and W. Chen, "Implementation and performance optimization of dynamic random forest," in *Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discovery (CyberC)*, 2017, pp. 283–289.

[37] M. P. Paing and S. Choomchuay, "Improved random forest (RF) classifier for imbalanced classification of lung nodules," in *Proc. Int. Conf. Eng., Appl. Sci., Technol.(ICEAST)*, 2018, pp. 1–4.

[38] A. Hau, P. Zhang, Z. Zheng, M. Zhu, Y. He, Q. Li, B. Zhang, F. Huang, G. Zhau, and J. Li, "Land price prediction based on random forest," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2018, pp. 2948–2951.

[39] S.-H. Choi, D. Hwang, and Y. Choi, "Wireless intrusion prevention system using dynamic random forest against wireless MAC spoofing attack," in *Proc. IEEE Conf. Dependable Secure Comput.*, Aug. 2017, pp. 131–137.

[40] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. Kompella, "Towards an elastic distributed SDN controller," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 7–12, Oct. 2013.

[41] C. Liang, R. Kawashima, and H. Matsuo, "Scalable and crash-tolerant load balancing based on switch migration for multiple open flow controllers," in *Proc. 2nd Int. Symp. Comput. Netw.*, 2014, pp. 171–177.

[42] G. Garg and R. Garg, "Detecting anomalies efficiently in SDN using adaptive mechanism," in *Proc. 5th Int. Conf. Adv. Comput. Commun. Technol.*, 2015, pp. 367–370.

[43] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the source," in *Proc. IEEE Int. Conf. Netw. Protocols*, Nov. 2002, pp. 312–321.

[44] R. R. R. Robinson and C. Thomas, "Ranking of machine learning algorithms based on the performance in classifying DDoS attacks," in *Proc. IEEE Recent Adv. Intell. Comput. Syst. (RAICS)*, Dec. 2015, pp. 185–190.

[45] Y. Feng, H. Akiyama, L. Lu, and K. Sakurai, "Feature selection for machine learning-based early detection of distributed cyber attacks," in *Proc. IEEE 16th Int. Conf. Dependable, Autonomic Secure Comput., 16th Int. Conf. Pervasive Intell. Comput., 4th Int. Conf. Big Data Intell. Comput. Cyber Sci. Technol. Congr. (DASC/PiCom/DataCom/CyberSciTech)*, Aug. 2018, pp. 173–180.

[46] F. Amiri, M. R. Yousefi, C. Lucas, A. Shakery, and N. Yazdani, "Mutual information-based feature selection for intrusion detection systems," *J. Netw. Comput. Appl.*, vol. 34, no. 4, pp. 1184–1199, 2011.

[47] B. Choubin, H. Darabi, O. Rahmati, F. Sajedi-Hosseini, and B. Kløve, "River suspended sediment modelling using the CART model: A comparative study of machine learning techniques," *Sci. Total Environ.*, vol. 615, pp. 272–281, Feb. 2017.

[48] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, 2012.

[49] Computer Networks Group. *Reinforcement-Containernet*. Accessed: Apr. 2018. [Online]. Available: https://github.com/ComputerNetworks-UFRGS/reinforcement-containernet

[50] M. S. Bonfim, K. L. Dias, and S. F. L. Fernandes, "Integrated nfv/sdn architectures: A systematic literature review," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–39, 2018.

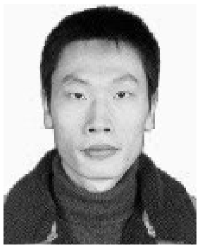
[51] *Mininet*. Accessed: Oct. 2017. [Online]. Available: http://mininet.org/

[52] *Ryu*. Accessed: Dec. 2016. [Online]. Available: https://osrg.github.io/ryu/v

[53] *Docker*. Accessed: Dec. 2016. [Online]. Available: https://osrg.github.io/ryu/v

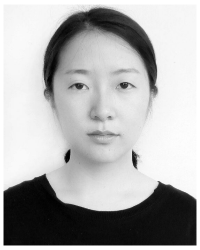
[54] CERT, "TFN2K causes denial-of-service attack," *Netw. Secur.*, vol. 2000, no. 2, pp. 1–2, 2000.

CHUANFENG XU received the bachelor's degree in optoelectronic information science from the Changshu Institute of Technology, in China, in 2017. He is currently pursuing the master's degree in computer application technology with the School of Mathematics and Information, Fujian Normal University. His research interests include wireless and mobile computing systems, computer networks, information and network security.



HUI LIN received the B.S. degree in computing science from Fujian Normal University, China, in 1999, and the M.E. degree in communication and information engineering from the Chongqing University of Posts and Telecommunications, China, in 2007. He is currently an Associate Professor with the School of Mathematics and Information, Fujian Normal University, China. He is currently pursuing the Ph.D. degree with the College of Computer Science, Xidian University. His research interests include software-defined networking, network function virtualization, and information and network security.

YULEI WU received the B.Sc. degree (Hons.) in computer science and the Ph.D. degree in computing and mathematics and from the University of Bradford, U.K., in 2006 and 2010, respectively. He is currently a Senior Lecturer with the Department of Computer Science, University of Exeter, U.K. His expertise is in networking. His research has been supported by the Engineering and Physical Sciences Research Council (EPSRC) of U.K., the National Natural Science Foundation of China, University's Innovation Platform, and industry. His main research interests include intelligent networking technologies, network slicing and softwarization, the future Internet architecture and technologies, green networking, wireless networks, network security and privacy, and analytical modeling and performance optimization. He is a Fellow of the Higher Education Academy (HEA). He contributes to major conferences on networking as various roles including a steering committee chair, a general chair, a program chair, and a technical program committee member. He is an Editor of the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, *Computer Networks* (Elsevier), and IEEE ACCESS.



XUANCHENG GUO received the bachelor's degree in computer science from the Fujian Institute of Engineering, China, in 2017. She is currently pursuing the master's degree in computer application technology with the School of Mathematics and Information, Fujian Normal University, China. Her research interests include software-defined networking, and information and network security.

WENZHONG LIN received the M.E. degree in electrical engineering from Fuzhou University, China, in 1990, and the Ph.D. degree from Nagasaki University, Japan, in 2016. He is currently a Professor with Minjiang University, China. His research interests include the Internet of Things, computer control technology, and servo drive technology.