CCS: A Cross-Plane Collaboration Strategy to Defend Against LDoS Attacks in SDN

Meng Yue, Qingxin Yan, Zichao Lu, and Zhijun Wu

***Abstract*—Software-Defined Networking (SDN) actualizes the separation of control and forwarding, innovates network functionality with a logically centralized controller, and facilitates network-wide collaboration. Contemporary SDN infrastructure exposes potential bottlenecks which are prone to engaging low-rate denial of service (LDoS) attacks. Currently, a great deal of detection methods are deployed in the controller, and the controller needs to poll the switch frequently, which brings heavy load to the controller and the southbound link. According to the analysis of existing researches, we focused on the how to decrease the frequent polling of the controller and improve the detection rate. In this paper, we adopted the idea of cross-plane collaboration and proposed a two-phase detection framework, which carried out the lightweight detection method in the data plane and the in-depth detection based on Bayesian voting mechanism in the control plane. Once LDoS attacks are detected, the controller recalculates routes for the bottleneck nodes using the optimized Dijkstra algorithm to complete mitigation. Theoretical analyses and extensive experiments are conducted to validate the performance of our proposed method. Test results show that our method outperforms other traditional methods in terms of the detection rate of 99.1%, the detection delay of 1.3s and the communication overhead of 1068 Byte/s, the average CPU utilization of controller remains at approximately 3.5%. The proposed method takes a step forward to enhance the security of SDN.**

***Index Terms*—Software-defined networking, low-rate denial of service attacks, cross-plane collaboration.**

I. Introduction

THE ADVENT of Software-Defined Networking (SDN) has aroused great concern and research upsurge in the network industry [1], [2]. SDN enables the separation of the control plane and the data plane, helps to create a "global view" of the entire network, improves visibility and policy consistency. Contemporary network further tends to simplification and automatization. Through the layered network architecture, the network administrator can program the network system as required and push forward the underlying implementation without interacting with hundreds of devices, thousands of lines of code and complex protocols. The functionality pioneered by SDN precipitate service agility, programmability, better performance characteristics and lower latency to become the considerable factors of new network construction. OpenFlow [3] has become the mainstream protocol of SDN in data center, which defines the communication standard between SDN controller and switch. Low-rate Denial of Service (LDoS) attacks [4], [5] mainly exploit the vulnerabilities of various adaptive mechanisms in the network, such as the congestion control mechanism in the TCP protocol and the queue management mechanism of the router. The adaptive protocol is designed to focus on the effectiveness, fairness and stability of the system in the steady state, and its security is not much considered, which leads to its own vulnerability. In order to achieve the optimal performance, network protocols often assume that the network is stable most of the time and make efforts to ensure the performance of the network in the stable state, but ignore the transient performance of the network. LDoS attacks exploit the relatively low transient performance of the network by periodically launching attack flows of a certain strength to make the network constantly fluctuate between failure and stability, thus reducing the overall performance of the network. LDoS attacks generate periodic high-speed short-time pulses with characteristics of low average speed and high concealment, which enormously increases the difficulty of detection. The Internet of Things (IoT), cloud computing platform and big data center has a significant risk of encountering such attack [6], [7], [8].

SDN carries the distinctive properties that allow the LDoS attackers to concentrate on mining these vulnerabilities [9], [10], [11], which constitutes a potential threat to the entire SDN [12], [13]. Currently, the corresponding detection technologies in SDN mainly face with the following problems.

(1) Countermeasures are routinely deployed on the controller, these methods make the controller poll the data collected by the switch for each test in order to ensure the real-time detection. However, this brings a significant communication overhead between the switch and the controller, and can even severely congests the southbound interface link.

(2) Extensive researches have been conducted based on the advantages of centralized control and flexible data collection in SDN [14], [15] and do not fully utilize the global perspective

Manuscript received 30 December 2022; revised 28 June 2023 and 24 January 2024; accepted 4 February 2024. Date of publication 7 February 2024; date of current version 12 July 2024. This work was supported in part by the National Natural Science Foundation of China (62172418, U1933108, U2133203), the Natural Science Foundation of Tianjin China (21JCZDJC00830), the Scientific Research Project of Tianjin Municipal Education Commission (2019KJ117), the Fundamental Research Funds for the Central Universities of CAUC (3122020076) and the Additional Funds for the National Natural Science Foundation of China by Civil Aviation University of China (3122022PT05). The associate editor coordinating the review of this article and approving it for publication was J. J. Yang. *(Corresponding author: Meng Yue.)*

Meng Yue and Zhijun Wu are with the College of Safety Science and Engineering, Civil Aviation University of China, Tianjin 300300, China (e-mail: myue_23@163.com; zjwu@cauc.edu.cn).

Qingxin Yan and Zichao Lu are with the College of Electronic Information and Automation, Civil Aviation University of China, Tianjin 300300, China (e-mail: yanqingxin_1228@163.com; lzc273716@163.com).

Digital Object Identifier 10.1109/TNSM.2024.3363490

of the controller. For example, a considerable number of researchers put the focal point on using machine learning algorithms to detect attacks, and the number of training features also shows a growing trend. In addition, training the network and calculating numerous parameters can lead to a significant detection delay and additional computational efforts.

(3) Another serious challenge of intrusion detection is derived from the diversity of network traffic. If observed only at the traffic level, normal network communication behavior may also appear to be "abnormal". For example, mass network activity and retransmission can lead to a surge in network traffic [16], [17], which may have some similarities with attacks, and ultimately leads to a high false positive rate.

In this paper, to solve the problem of the controller frequently polls, we proposed a collaborative detection framework between the control plane and the data plane, and executed lightweight detection at the data plane. When the switches find an exception, they report the specified information to the controller through the south interface protocol OpenFlow. The controller collects the information reported by all abnormal switches and makes a global judgment.

In the lightweight detection phase of the data plane, the switches collect the statistical information of ports and flows. In order to reflect the anomaly characteristics of the flow table and the flow, we defined four indicators then designed a threshold-based detection method. Once the switch catches the exceptions of these characteristics, it will send encapsulated messages to the controller through the south interface to indicate where there the unusual circumstance appears and provide a reliable feature basis for the controller. The controller extracts the information reported by the switch for statistical feature analysis to get the overall state of the downstream nodes of the controller, and then uses the Bayesian voting mechanism to detect whether the network is subjected to the attack. The in-depth detection sufficiently takes advantage of the characteristics of the global perspective of the controller. The controller provides intention or policy-based management for the entire network. It serves as a centralized or distributed intelligent entity with an overall view of the network, according to which routing and handover decisions can be made. Once the controller detects the LDoS attack, it starts the mitigation mechanism for the bottleneck node, calculates the backup path by using the optimized Dijkstra algorithm, and issues the flow rules to the corresponding switches to complete the transfer of the original traffic on the bottleneck link. The cross-plane collaborative detection method reduces the load pressure on the controller and the southern communication link, ensures real-time detection, and enables the controller to make decisions in a timely manner.

The main contributions of this paper can be summarized as follows.

(1) We proposed a cross-plane collaborative detection architecture. First, lightweight detection is carried out in the data plane. If there is an exception, the controller conducts a global in-depth detection. This method effectively reduces the communication overhead caused by frequent polling of the controller.

(2) We deployed the lightweight detection method in the data plane and we proposed four new features to characterize the behavior of LDoS attacks, so as to effectively distinguish anomalies. These features are lightweight and will not significantly increase the burden of the switches.

(3) A global in-depth detection method based on path aggregation is deployed in the controller, and the detection model is established by using Bayesian voting mechanism. This method depends on the global perspective of the controller, and can accurately identify attacks and locate the bottleneck nodes that are affected.

(4) We used the improved Dijkstra algorithm to reroute the victim bottleneck nodes, so as to complete the mitigation and recover the link performance. The controller monitors the local behavior of nodes, quantifies the traffic size of each destination and the bandwidth capacity of the node outlet port as weights, and forms the final decision.

Our research gives full play to visibility and policy consistency, and has practical significance in combating LDoS attacks.

The rest of the paper is organized as follows. Section II presents the related works, summarizes the current research status and subsistent bottlenecks, and presents the motivation of this paper. Section III describes the principles of LDoS attacks in SDN scenarios. Section IV expounds the architecture of our proposed framework. Section V covers the experiments to verify the proposed detection method and mitigation method, and conducts comparisons with other methods. Section VI concludes this paper and discusses the future work.

II. Related Works

LDoS attacks were first found in 2001 by Asta Networks after six-month monitoring on Internet2 Abilene backbone [18]. Then, Kuzmanovic and Knightly first published its principle at the SIGCOMM in 2003 [5]. Later in 2004, the website www.qq.com was attacked by such attack [19]. Although LDoS is a traditional attack, new network scenarios (such as SDN, cloud data center networks) provide greater space for such attack [20], [21], [22], [23].

LDoS attacks traffic has the characteristic of intermittency, with a relatively low average rate, making it difficult to defend against such attacks with existing methods. LDoS attacks can avoid detection and prevention more effectively, which brings new challenges to the research of attack prevention. Traditional middle-box based DoS attack defense mechanisms lack monitoring flexibility. The characteristics of the separation of control and forwarding as well as programmable network behavior provide new ideas for the detection and defense of LDoS attacks in SDN [24]. Although there have been some researches on LDoS attacks against SDN architecture [25], the overall situation is still comparatively insufficient.

Yue et al. [26] have developed two LDoS attack models that effectively limit TCP throughput and improve attack potency, urging the defenders to develop corresponding methods from the perspective of attack and defense game. Cao et al. [20] used LDoS attack pulse flow to destroy the shared link in

the control path and data path, causing the delay of the control flow and indirectly affecting the core services of SDN. Wu et al. [27] verified that LDoS attacks can cause changes in the essence of network traffic, and also change the eigenvalue Holder index of network traffic. Based on this, they proposed a multifractal-based LDoS attack detection method, and detected LDoS attacks according to the abnormal changes of the Holder index. Xiang et al. [28] innovatively proposed to use two new information metrics, the generalized entropy metric and the information distance metric, to detect low-speed DDoS attacks by measuring the difference between legitimate traffic and attack traffic. Xie et al. [29] analysed LDoS attacks exploiting the limited Ternary Content-Addressable Memory (TCAM) finiteness of SDN switches, and designed a flow table overflow prediction module and a flow table item deletion policy for such attacks, so as to effectively detect and suppress such attacks. Tang et al. [21] proposed the Histogram-Based Gradient Boosting and Finding Peaks (HGB-FP) algorithm framework. In addition to accurately identifying LDoS attacks, it can also locate attackers through the peak attribute of the traffic. In [30], Tang et al. used machine learning algorithm to determine whether LDoS attacks occur by extracting traffic characteristics, and locate attack sources and victims through time-frequency analysis. In order to detect and mitigate DoS attacks against SDN, Gao et al. [31] put forward a defense framework called FloodDefender, which uses new frequency characteristics to detect attacks. And the mitigation module implements three new technologies: table-miss engineering to prevent communication bandwidth from running out, packet filter to filter out attack traffic and save computing resources of the control plane, and flow rule management to eliminate most useless flow entries in the switch flow table. At present, traditional attack detection methods focus on detecting whether DoS attacks occur in a single aspect, while ignoring to find the path of the attack flow through the network. This increases the difficulty to defend against DoS attacks. Cao et al. [32] presented a detection method based on Spatial-Temporal Graph Convolutional Network (ST-GCN). It senses the state of the switch through in-band network telemetry (INT) sampling, inputs the network state into the spatiotemporal graph convolutional network detection model, and finally finds the switch through which DDoS attack traffic passes. Han et al. [33] put forward a monitoring algorithm based on flow density and a classification algorithm based on machine learning, which are deployed on the data plane and control plane respectively to achieve efficient collaboration intelligence. Wang et al. [34] proposed a controller scheduling mechanism called BWManager, introduced a priority mechanism based on bandwidth prediction, and designed a new weighted loop algorithm to handle requests with different priorities. Imran et al. [35] proposed a simple and lightweight detection and mitigation system named DAISY to protect SDN from DoS attacks by blocking malicious traffic from attackers. El Houda et al. [36] conducted a multi-stage defense scheme against DDoS attacks. They measured the randomness of traffic with entropy, and classified it according to entropy. Li et al. [37] used the random forest algorithm to select the characteristics of the training data for the BP neural network, and improved the Bat Algorithm to build an LDoS attack detection model. Xie et al. [38] proposed SoftGuard, which utilizes adaptive Fast Fourier Transform to determine whether the aggregated TCP throughput has periodicity to confirm the low-rate TCP attack, and uses the average Euclidean Distance to accurately identify the attack flow. By installing mitigation rules in the entrance switch, the identified attack flow can be effectively suppressed.

Through comparative analysis of existing literature, we find that most of the defense mechanisms take advantage of the centralized control and flexible acquisition of the controller but fail to fully use the global perspective of the controller. Some machine learning based methods have a high detection rate, but there may be a certain detection delay. In addition, the controller needs to frequently poll the traffic statistics from the data plane switch for attack detection, which will increase the load of the southbound channel. In order to solve the problems above, we proposed a cross plane collaborative overall detection mechanism that extracts new attack features and conducts lightweight detection and depth detection in the data plane and control plane respectively. This technology examines network behavior from both local and global perspectives, producing comprehensive conspicuousness. The data plane is based on the abnormal behavior of the a single switch's flow table and controls the abnormal behavior of the traffic, while the control plane is based on the aggregation phenomenon of abnormal nodes throughout the network. The cooperation mechanism improves the overall efficiency of the system, reduces the frequent polling of the controller, and can achieve more accurate detection.

III. Brief Review of LDoS Attack in SDN

LDoS attacks under SDN not only affect the throughput of normal TCP data traffic, but also interfere with control traffic [20]. Therefore, this attack is more harmful to SDN than to traditional networks [21]. Commonly, the attacker first probes the bottleneck link as target using network measurement technologies [20], [22]. Here, the bottleneck link is a link with small available bandwidth and shared by the control traffic and the data traffic. The aggregation of multiple flows facilitates small available bandwidth, such as TCP incast (many-to-one communication model) [39], the under-provising in cloud data center network [22]. The in-band development of controller in SDN forms the shared link. Once a bottleneck link is determined, the attacker then launches attack data traffic to fill up the bottleneck buffer. In this case, due to TCP congestion control on the data channel, the link utilization rate is extremely low. Moreover, control packets delivered on the control channel may also be delayed or dropped, which causes the network functions enabled by the controller to be almost paralyzed. We illustrate the attack scenario via an example in Fig. 1.

In Fig. 1, we build an in-band developed SDN network consisting of four switches $\{S_1, S_2, S_3, S_4\}$ and eight host users $\{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8\}$, where h_2 and h_3 communicate through links $h_2-S_1-S_2-h_3$, while h_1 and h_4 communicate through links $h_1-S_1-S_2-h_4$. We assume that the control path

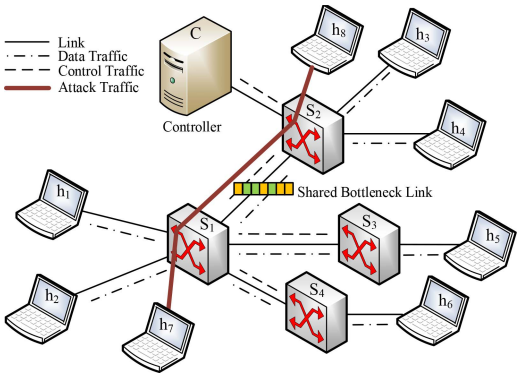


Fig. 1. LDoS attack scenario in SDN.

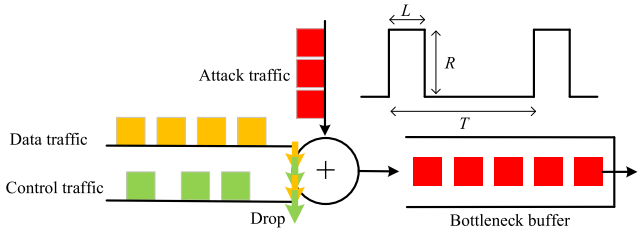


Fig. 2. LDoS attack model.

between S_2 and controller C is S_2–C, the link between S_1 and S_2 is the shared bottleneck link, h_1 and h_2 send normal traffic to h_3 and h_4. Assume h_7 is the attacker who sends LDoS attack traffic to h_8 through the bottleneck link. The attack packets cause congestion in the bottleneck link, and normal traffic passing through the bottleneck link will be affected. The traffic from h_1 and h_2 (they belong to region of S_1) to h_3 and h_4 needs to pass through the bottleneck link, so the throughput of h_1 and h_2 will decrease sharply. LDoS attacks can also affect the delivery of control traffic. S_1, S_3 and S_4 are affected switches, since the bottleneck link is also used by the control paths $S_3 - S_1$ and $S_4 - S_1$. Consequently, the control messages delivered on these paths may be delayed or dropped.

As shown in Fig. 2, the LDoS attack can be modeled by a series of on-off square bursts with rate R, width L, and period T [5]. The attacker can implement such attacks using well-configured attack parameters <R, L, T> [24]. R should exceed the bottleneck link capacity at least, to induce packet loss. L should be the same scale of RTT, which makes it long enough to induce timeout but short enough to avoid detection. Commonly T should be set to an RTO, by doing so, when flows attempt to exit timeout, they are faced with another loss. According to Fig. 2, although R exceeds the bottleneck link, the average rate ($L \times R/T$) is still low. Therefore, it is called LDoS attack.

IV. Defense Method Design

A. Overall Architecture Design

The purpose of cooperative detection is to reduce the frequent polling of the switch information by the controller, and only take further action when the switch detects an exception. We used the available computing resources of the switch to complete lightweight detection on the data plane. And when exceptions are detected, we further performed in-depth detection on the control plane. The overall architecture design is shown in Fig. 3.

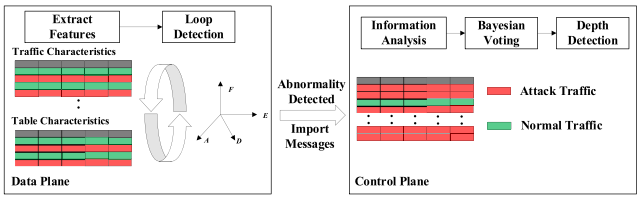


Fig. 3. The overall architecture design.

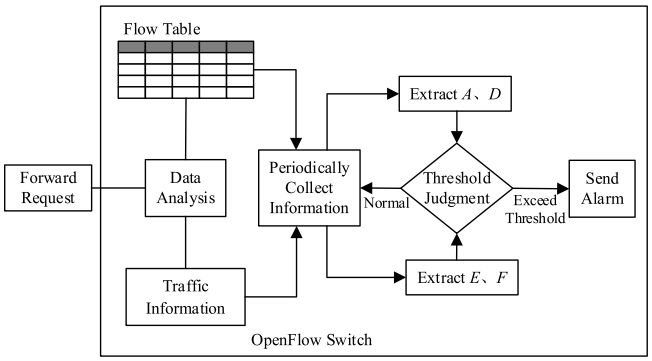


Fig. 4. Lightweight detection architecture design.

We extracted the flow table and flow characteristics for attack detection, so that the detection method can catch the abnormal situation. The Bayesian voting mechanism is used for depth detection based on global topology, which enables the controller to accurately detect LDoS attacks.

B. Lightweight Detection Method

1) Data plane detection architecture: In the SDN environment with OpenFlow as the communication and interaction protocol, the OpenFlow switches collect port and flow statistics without additional collection devices. In SDN, when the switch receives a packet, the packet needs to match the flow table and be forwarded according to the flow rules. For a new data flow, that is, there are no match items in the flow table, the switch will send Packet_ in messages to the controller, the controller will send Packet_ out messages to issue flow rules and complete the forwarding of the traffic. Reference [40] pointed out that the data plane has certain programming capability. The existing OpenFlow switches have one or more Central Processing Units (CPUs). These processors have extraordinary computing power and generous unused resources, making it possible to deploy detection methods in the data plane. The lightweight detection architecture is shown in Fig. 4.

Firstly, the switch periodically collects flow table information and flow information and extracts features. This method detects these indicators in real time. Once the threshold is exceeded, an exception may occur in the network. At this time, the switch will send alarm information to the controller.

2) Feature extraction: In order to reflect the abnormal characteristics of the flow table, we defined two indicators: flow table matching mean *A* and matching popularity *D*. The formulas are as shown in Eq. (1) and Eq. (2).

$$A = \sum_{i=1}^{N} C_i / N \tag{1}$$

$$D = \max(C_i) \tag{2}$$

where C_i is the matching times of the *i*-th flow rule of the port, and N is the total number of flow rules. Since LDoS attacks mainly affect TCP long connections, A is featured by a steady decrease under abnormal conditions. The LDoS attack is a periodic high-speed pulse. It generally uses small packet attacks (to improve attack efficiency) and matches the same flow rules (in order to make the attack packets enter the queue quickly and avoid the delay caused by the controller). Therefore, the flow table shows that the value *D* of a flow is extremely high during the flow table timeout period.

In addition, we define the entropy value *E* of bottleneck link queue change, the formula is shown in Eq. (3).

$$E = -\sum_{i=1}^{n} \frac{L_1 + L_2 - P}{C} \log_2 \frac{L_1 + L_2 - P}{C} \tag{3}$$

where L_1 is the cache queue at one end of the bottleneck link, L_2 is the queue length of the bottleneck link transmission, *P* is the length of congestion packet loss, and *C* is the capacity of the bottleneck link. The core idea of LDoS attacks is to attack the bottleneck link or router and cause instant congestion. This leads to the loss of many TCP packets, thus forcing the network to send congestion signals. As a result, the source side activates the TCP congestion control mechanism, which adaptively adjusts the size of the sending window and try to recover to a stable state. Periodic attacks can cause TCP performance shocks and severely reduce TCP throughput. Meanwhile, the queue length is extremely unstable and fluctuates significantly. Under abnormal conditions, entropy *E* will increase. A threshold-based detection method can be formed according to the above four features.

LDoS attacks show certain new features under SDN, such as the impact on the control flow. In order to reflect the abnormal characteristics of control flow, the influence factor of control flow transmission capability *F* is defined, using the formula shown in Eq. (4).

$$F_l = \frac{\lambda_l}{\frac{r_l}{c_l} + \frac{c_l}{c_l - r_l}} \tag{4}$$

where λ_l refers to the delay of link l, c_l refers to the physical bandwidth of link l, and r_l refers to the available bandwidth of link l. LDoS attacks can increase the delay of control flow and reduce the available bandwidth of link l, causing *F* to increase under abnormal conditions.

The above feature-based detection relies on the threshold decision scheme. We use an adaptive mechanism based on EWMA (exponentially weighted moving average) to dynamically update the threshold:

$$Th(i) = (1-w) \times Th(i) + w \times Th(i-1) \tag{5}$$

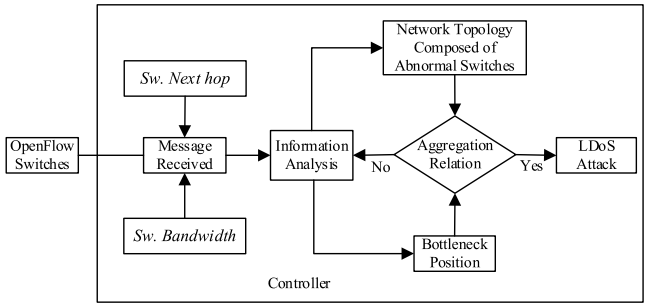


Fig. 5. Control plane detection architecture.

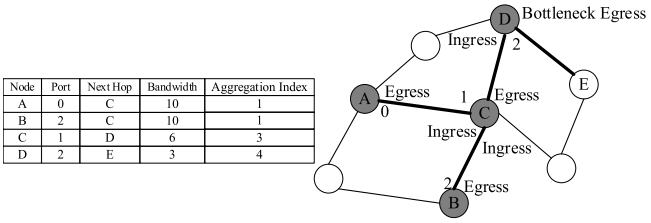


Fig. 6. Example of aggregation relation.

where *Th*(*i*) denotes the current threshold for a specific feature, *Th*(*i*–1) denotes the prior threshold and *w* denotes the weight ranging from 0 to 1.

C. Global Detection Method

1) Control plane detection architecture: The control plane is responsible for network control and information collection. The control plane's global view function can enable real-time monitor and adjustment of the network through the OpenFlow protocol, and can also collect the global information. OpenFlow allows the controller to dynamically specify the forwarding behavior of the SDN switch by installing flow rules. The lightweight detection made by the data plane reduces the load pressure on the controller and the south communication link. This ensures the real-time detection and enables the controller to make decisions rapidly. The detection architecture of the control plane is shown in Fig. 5.

After the data plane completes lightweight detection, the switch that detects the exception sends a report message to the controller. The message includes the next hop node (*Sw. Next hop*) of the abnormal port and the available bandwidth (*Sw. Bandwidth*) of the port and reflects the overall state of the downstream nodes of the controller. According to this, the controller can obtain two pieces of information: 1) the network topology formed by the abnormal switches, 2) the location of bottleneck nodes, that is, where is the minimum available bandwidth on a path in the network topology. If these switches that report exceptions have a certain topology relationship (defined as an aggregation relationship, expressed by the aggregation index), and the minimum available bandwidth is at the end of the path, the controller determines that LDoS attacks occurr. Fig. 6 shows an example of the aggregation index.

As shown in Fig. 6, all nodes form a topology. From the global perspective of the controller, A, B, C, D are the switch

nodes that report exceptions. The exception ports of A, B, C, and D are Ports 0, 2, 1, and 2 respectively, and the bottleneck port of D is port 2. Node A and node B's next hop is exception node C. At this time, the aggregation index is defined as 1 and the next hop of C is exception node D, and then the aggregation index is defined as 3 and the next hop of D is abnormal node E. At this time, the aggregation index is defined as 4. The four nodes form an abnormal topology with the bottleneck link at the end of the abnormal topology. At this time, it meets the conditions that the controller determines the occurrence of LDoS attacks, and sends control information to the switch for real-time mitigation. We used Bayesian voting mechanism to model the aggregation index.

2) Bayesian voting algorithm: In this paper, we utilized the Bayesian voting mechanism to establish the aggregation index relationship. The Bayesian voting algorithm places all test nodes at the same confidence level, meaning that malicious nodes will also vote positive for abnormal nodes. If there are a great number of abnormal nodes, the detection rate will be terribly inferior. To address this issue, we introduced weights to improve the algorithm. From the global perspective of the controller, there are N_1 normal nodes and N_2 abnormal nodes. The weight values of normal nodes and abnormal nodes are as shown in Eq. (6) and Eq. (7).

$$W_1 = \frac{N_1}{N_1 + N_2} Sw.Bandwidth_1 \tag{6}$$

$$W_2 = \frac{N_2}{N_1 + N_2} Sw.Bandwidth_2 \tag{7}$$

Among them, $Sw.Bandwidth_1$ and $Sw.Bandwidth_2$ respectively indicate the available bandwidth of the normal port and the abnormal port reported by the switch to the controller. The total number of votes of node v_i is shown in Eq. (8).

$$\sum_{j=1}^{N_1+N_2} V_{ji} = W_1 * \sum_{j=1}^{N_1} V_{ji} + W_2 * \sum_{j=1}^{N_2} V_{ji} \tag{8}$$

wherein, $W_1 * \sum_{j=1}^{N_1} V_{ji}$represents the total number of votes of normal nodes, and $W_2 * \sum_{j=1}^{N_2} V_{ji}$represents the total number of votes of abnormal nodes. The average number of votes of node v_i is shown in Eq. (9).

$$v_i\_ave\_number = \frac{\sum_{i=1}^{N_1+N_2} v_i\_number}{N_1 + N_2} \tag{9}$$

wherein, $\sum_{i=1}^{N_1+N_2} v_i\_number$ indicates the number of votes of node v_i. The overall average voting value of the network topology is shown in Eq. (10).

$$all\_ave\_voting = \frac{\sum_{i=1}^{N_1+N_2} V_{ji}}{N_1 + N_2} \tag{10}$$

Therefore, the mean Bayesian voting value of node v_i is shown in Eq. (11).

$$Bayes\_voting(v_i) = \frac{v_i\_ave\_number * all\_ave\_voting + \sum_{j=1}^{N_1+N_2} V_{ji}}{v_i\_ave\_number + v_i\_number} \tag{11}$$

Algorithm 1 Bayesian Detection Algorithm for Node v_i

```
Input: Available bandwidth Sw. Bandwidth of next hop
       node of node v_i
       Information base of normal node N_i= NULL
       Information base of abnormal node A_i= NULL
if time≠0
   Collect information of all next hop nodes R_i
end if
for j=1 to Number of next hop nodes
   if Sw. Bandwidth ϵ N_i
      v_ij = 1,
   else if v_ij = −1
      end if
   end if
end for
for i = 1 to Number of nodes
   Calculate the voting value voting(v_i) of node v_i
   if |voting(v_i) − Bayes_voting(v_i)| converges to param-
   eter σ
      max(|voting(v_i) − Bayes_voting(v_i)|) corresponding
      to the bottleneck
      Determine LDoS attacks occur
   end if
end for
```

We suppose that $voting(v_i)$ represents the voting value of node v_i, if the difference $|voting(v_i) - Bayes\_voting(v_i)|$ converges to the parameter σ (i.e., it aggregates towards a specific value), it is considered that the aggregation relationship is satisfied. Parameter σ represents a preset benchmark, which can be set according to experimental statistics and network measurements in practice [41], [42]. In addition, this Bayesian voting algorithm is substituted into the weight formula (5)(6). This weight is proportional to the bandwidth. If the value of the bandwidth of node v_i is relatively small, and the corresponding weight value is also small, the final difference will be larger, and then the maximum value corresponds to the bottleneck position (Maximum aggregation index). After these two conditions are met, the controller will determine that LDoS attacks occur.

Algorithm 1 shows the specific process of Bayesian voting algorithm. The controller views nodes' behavior from the global perspective and takes the bandwidth reported by the switch as an indicator. If the connection of node 1 to node 2 is detected as normal, then node 1 to node 2 will vote positively, v_{ij}=1. Conversely, $v_{ij} = -1$. Then we calculate the voting value $voting(v_i)$ of node v_i. If $|voting(v_i) - Bayes\_voting(v_i)|$ converges to parameters σ and $\max(|voting(v_i) - Bayes\_voting(v_i)|)$ (i.e., the aggregation index is the largest) corresponds to the bottleneck location, at this moment, the controller determines that LDoS attacks occurr.

D. Mitigation Mechanisms Based on the Re-Routing Strategy

The existing link recovery and mitigation schemes are generally divided into two methods, active and passive [43].

Active link recovery refers to the process where the controller calculates the corresponding backup path and stores it in the relevant switch nodes before a failure occurs. When the link detects an abnormality and generates congestion, it quickly reroutes the data packets along the calculated backup path. However, active recovery and mitigation solutions will occupy the TCAM resource of the switch. The passive link recovery scheme refers to the operation that when a link abnormality is detected, the controller calculates and issues routing commands to the switch node. This paper adopts a passive link recovery and mitigation scheme, which only takes the next action when an abnormality is detected in the link and this scheme does not occupy switch resources. The paper's core objective is to improve routing efficiency, shorten link anomaly time, and restore link performance.

In the SDN architecture, Jiang et al. [44] extended the Dijkstra algorithm. When solving the single source shortest path problem, they not only considered the weight of edges, but also the weight of nodes. This method improves the routing efficiency. Li et al. [45] used an optimized genetic algorithm for routing and set objective functions for multiple QoS parameters. The proposed algorithm effectively reduced the packet loss rate and latency of the link. Tanha et al. [46] used a more complex routing strategy based on the ant colony algorithm, where individuals in the ant colony correspond to the routing strategy. This process involves continuous iteration and updating while also setting budget constraints, and then continuously updates the routing strategy to achieve optimal results. This method often requires more computing time while the users' demand for network services is becoming increasingly time sensitive, making it unrealistic to use the controller for excessive computation in the presence of attacks.

After the controller captures the bottleneck position, it will recalculate the backup path for the bottleneck nodes and instruct the corresponding switches to install flow rules to complete the transfer of the data flow from the original bottleneck link. The controller collects the flow table information of the abnormal switches after calculating the route and rephrases the forward port in the forward table according to the calculated new path. Then the controller sends the new forward table entry into the abnormal switches, guaranteeing that all the data groupings are forwarded out by the alternate path, and ensuring the normal communication of subsequent clients. The mitigation performance is measured by the recovery time, the available bandwidth and delay of the bottleneck link, the throughput and delay of the control packets. This paper used the optimized Dijkstra algorithm to reroute data packets. The rerouting strategy is shown in Fig. 7.

In Fig. 7, the attacker attacks the bottleneck link $S_1 - S_2$, so that the request of the normal user $User_1$ cannot be responded. The controller quickly calculates a new path according to the Dijkstra algorithm, that is, $User_1 - S_1 - S_4 - S_5 - S_6 - User_2$. This effectively mitigates the LDoS attack.

At present, the most widely used controllers in SDN, such as Ryu and Pox, use Dijkstra as default routing algorithm. The SDN controller has a global perspective, and through the LLDP Protocol (Link Layer Discovery Protocol, LLDP)

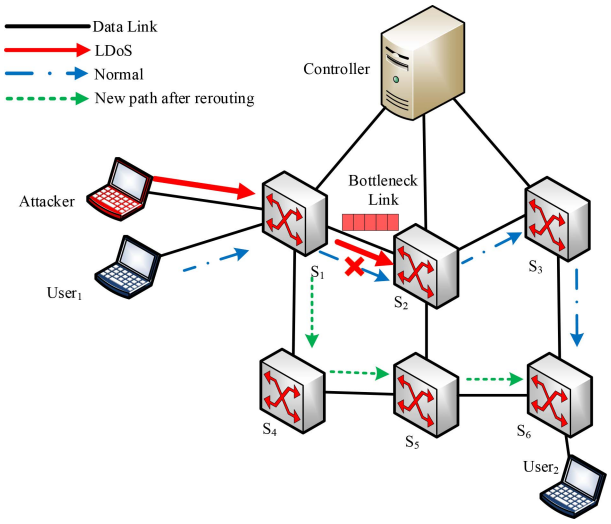


Fig. 7. The rerouting strategy.

it can obtain basic information of the network, such as network topology, bandwidth, latency, etc., forming an overall resource view. The controller uses the Dijkstra algorithm to calculate the shortest path from the source node to the destination node, but this shortest path is not necessarily the optimal path. The development of SDN and the complex and ever-changing needs of the real network environment have increased the difficulty of selecting the optimal path. In path selection problems, a single metric parameter cannot be selected as a constraint to make reliable path planning [47]. This will lead to some unnecessary congestion and reduce link utilization. At the same time, we should not only consider the reachability of the link, but also take the real-time changing network parameters into account in the path planning. Then we should use the characteristics of centralized management of the controller to plan the most suitable forwarding path. Therefore, on the one hand, we should save the cost of path consumption. On the other, it is necessary to consider the performance changes of the selected link in real-time, measure the traffic distribution throughout the entire network, and improve the reliability of routing.

Obviously, the traditional Dijkstra algorithm cannot meet these requirements, especially when the network is under attack, the link performance changes greatly and is extremely unstable. Therefore, improving link utilization is particularly important. The traditional link recovery scheme does not consider factors such as traffic scheduling and historical failures when calculating backup routes. If a suitable backup path is not selected, it may exacerbate network congestion. When making routing decisions, the controller often faces the following problem: the performance indicators of the path to the destination may vary over time. So the routing decision system must be able to perceive changes in some performance indicators, such as capacity, link utilization, and bandwidth. The centralized controller receives the relevant states of nodes in the network and forms the final routing decision. In order to implement a bottleneck node mitigation strategy, it is necessary to know the traffic size of each destination and the

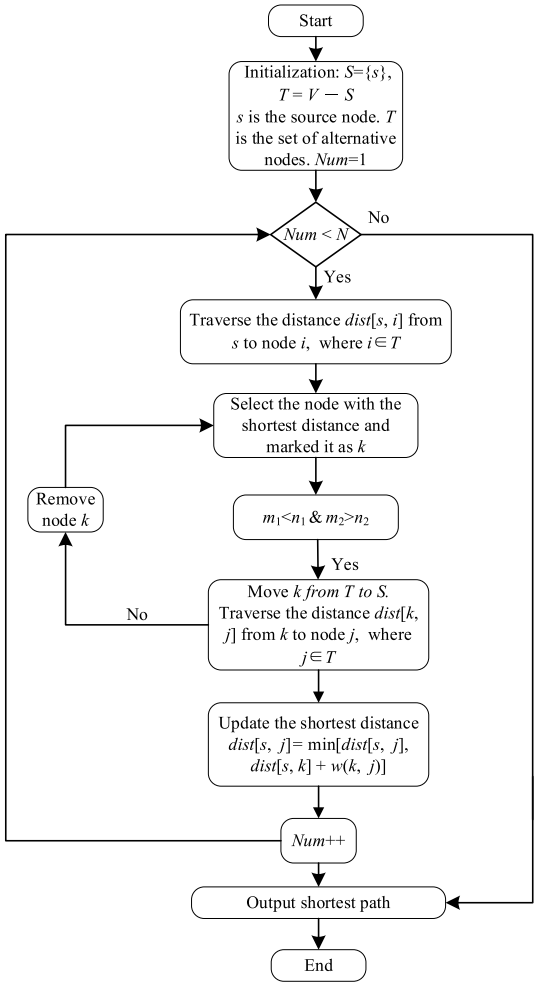


Fig. 8. Improved Dijkstra algorithm.

bandwidth capacity of the node's exit port. In this paper, the performance indicators of real-time traffic are included in the decision-making process. Two parameters m_1 and m_2 are defined to represent the traffic size of each destination and the bandwidth capacity of the node's exit port, respectively. These two parameters are used as decision indicators for the controller to select the nodes included in the optimal path. The optimized flowchart is shown in Fig. 8.

As shown in Fig. 8, V represents the set including all N nodes in the network, the set S only contains the source node s, and T represents all other nodes. Our algorithm first traverses the distance from each node i in T to s and selects the node k with the shortest distance. Then, it determines the current traffic size m_1 of node k and the bandwidth m_2 of the traffic exit port, so that the throughput of the bottleneck link is n_1 and the bottleneck link capacity is n_2. If $m_1 < n_1$ and $m_2 > n_2$, it moves k from T moves into S. Next, it traverses the distance from nodes j left in T to k, and update the shortest distance from s to j according to $dist[s, j] = \min[dist[s, j], dist[k, j]+w(k, j)]$. If $m_1 < n_1$ and $m_2 > n_2$ is not satisfied, k is removed, and the left nodes is checked again. Loop the above process until all nodes are moved to S. Finally, it gets the shortest path.

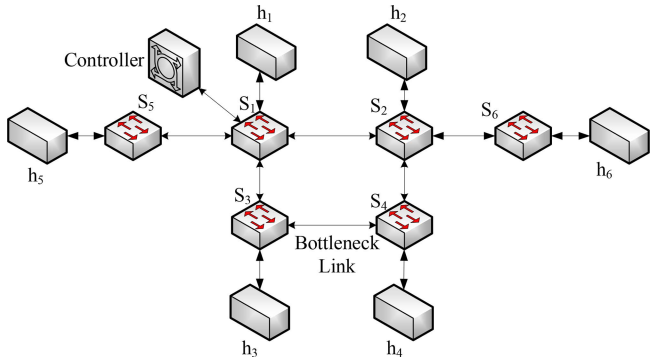


Fig. 9. Experimental topology.

V. Experimental Results and Analysis

A. Experimental Environment Configuration

The experimental platform is mainly composed of the Mininet simulation tool and the Pox open-source controller. Mininet is a standard work simulator for SDN. It supports multiple versions of OpenFlow protocols and can create a virtual network containing controllers, switches, hosts and links. It is the first choice of most researchers at present. The Pox controller is written in Python language and has a complete set of programmable interfaces that perfectly support the OpenFlow protocol. Developers can use the interface provided by the controller for free design and research. The hardware used in the experiment mainly includes two physical hosts with quad-core Intel Xeon CPU E5504 and 64GB RAM, which deploy Pox controller and Mininet simulation network respectively. The experimental topology is shown in Fig. 9.

According to the attack principle reviewed in Section III, the experimental scenario commonly considers two folds. One is that the target link is shared by data traffic and control traffic, and both the control plane and the data plane are affected. The other is configuring the link bottleneck and using a TCP-oriented periodic on-off "square-wave" attack traffic to decrease the attack rate. As shown in Fig. 9, the experimental platform consists of six Open vSwitch switches, one Pox controller and six hosts. In Fig. 9, h_1 is the attacker, and h_2, h_3, h_4, h_5 and h_6 are all legitimate users. h_1 sends attack traffic to h_4, h_2 sends normal traffic to h_3, and h_5 sends background traffic to h_6. We set the bottleneck link between S_3 and S_4 with 15Mbps bandwidth and 10ms one-way delay. The bandwidth of other links is 100Mbps, and the link delay is 1ms. The background traffic is generated by the D-ITG [48] tool, while the normal traffic is generated by the traffic generation tool Scapy [49]. We used Socket to continuously initiate connections to simulate attack flow. Unlike the traditional flooding DoS attack, the LDoS attack does not launch large-scale attack flows, but precisely designs attack parameters to conceal its behaviors. Here, we set the attack parameters <R, T, L> to <15Mbps, 1.2s, 400ms>. In addition, we set the size of the switch flow table to 1500 [50]. Totally, the above network settings and attack parameters are typical and commonly used by existing studies [20], [21].

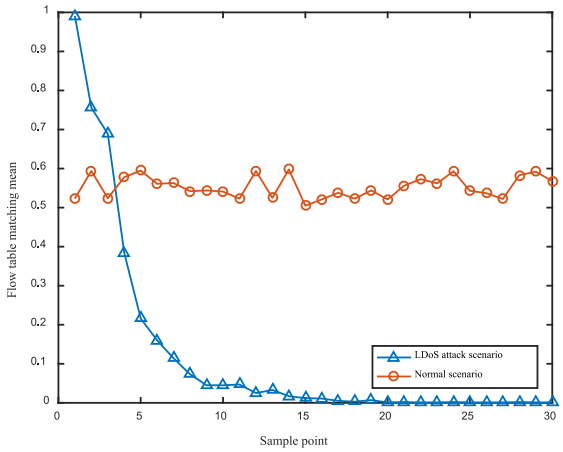


Fig. 10. Flow table matching mean.

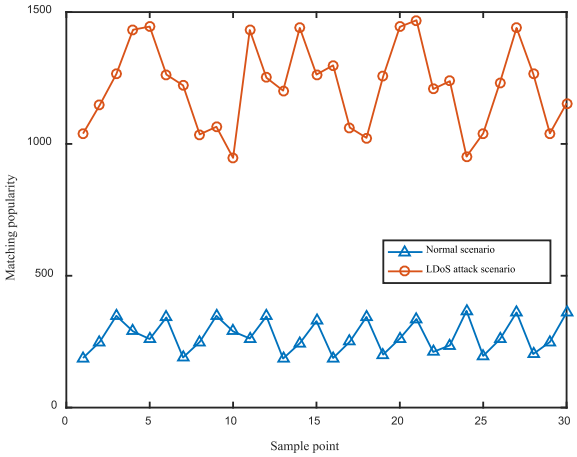


Fig. 11. Matching popularity.

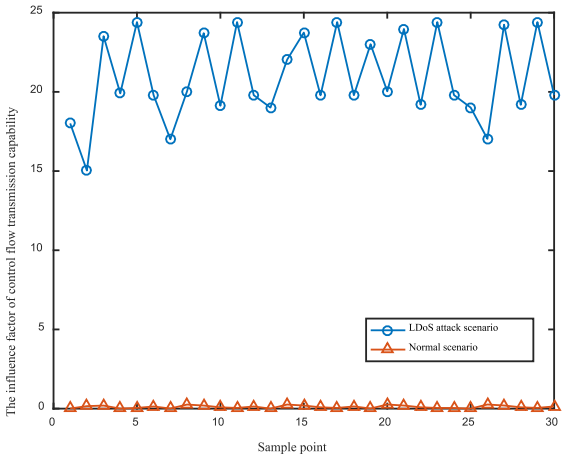


Fig. 12. The influence factor of control flow transmission capability.

B. Detection Performance Analysis

In this section, we analyzed the detection performance of lightweight detection method through quantitative indicators and the performance of depth detection based on the Bayesian voting algorithm.

1) Lightweight Detection in Data Plane: In the experiment, we collected 30 groups of flow table matching mean A under normal and attack scenarios respectively. The variation of A is shown in Fig. 10.

Fig. 10 shows that in the attack scenario A has decreased. These sampling points represent different flow rules. The attacker successfully matches the flow rules after sending a large number of packets in the first period, resulting in an initial increase of A. Before these flow rules expire, the attacker will send the same packets to the switch. At this time, there are flow rules installed in the last period, and no matching is performed. Therefore, in the attack scenario, the matching times of flow rules will decrease. Under normal circumstances, the specific access requirements of users have tremendous randomness, while packets vary in size and may match different flow rules. Therefore, the matching times of the i-th flow rule will be relatively high.

We collected 30 groups of matching popularity D under normal and attack scenarios respectively. The variation of D is shown in Fig. 11.

Fig. 11 shows that the matching popularity D of a certain flow rule in the attack scenario appears at a high value. This is due to the attacker sending a large number of data packets in the first period, which match a significant number of flow rules. As a result, the flow table fills up quickly, and the maximum number of times a flow rule is matched can be close to the upper limit of the flow rule capacity. Under normal circumstances, the packet requirements of legitimate users will match different flow rules randomly, so the matching popularity of a flow will appear at a low value.

We collected 30 groups of data on the influence factor F of control flow transmission capability under normal and attack scenarios respectively. The variation of F is shown in Fig. 12.

SDN architecture with in-band deployment [26] has attracted an increasing number of attention due to its positive flexibility and economic deployment. However, in the SDN with in-band control [51], [52], the control flow and data flow share a physical link for forwarding. Since the attack can increase the delay of the control flow and reduce the available bandwidth of the link, the transmission capability impact factor F of the control flow in the attack scenario remains at a high value (15-25). In normal scenarios, legitimate users send requests to match flow rules, and the controller sends commands to instruct the switch to take action, resulting in legal delay and bandwidth occupation, and then F appears at a tremendously low value.

We collected 30 groups of the entropy value E of bottleneck link queue changes under normal and attack scenarios respectively. The variation of E is shown in Fig. 13.

Fig. 13 shows that the entropy value E of the bottleneck link queue change remains at a high value under attack scenarios. LDoS attack traffic causes periodic packet loss of normal TCP traffic and creates different levels of network congestion. This continuously triggers the TCP congestion control mechanism, making the congestion control window at the normal TCP sending end always in a small state, thus resulting in a sharp drop in the TCP throughput of the victim host. The link will be in a cycle of congestion - packets loss - recovery – congestion.

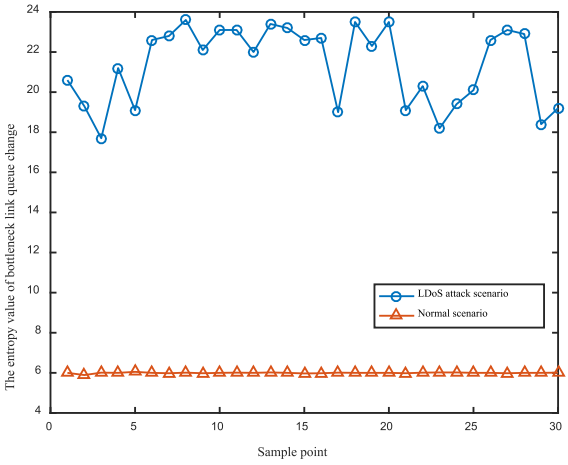


Fig. 13. The entropy value of bottleneck link queue change.

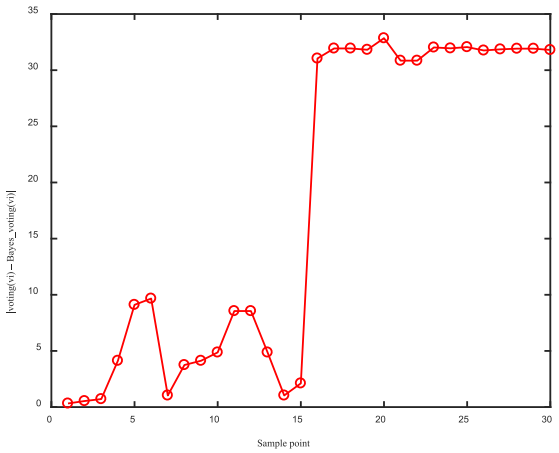


Fig. 14. |*voting*(v_i) – *Bayes_voting*(v_i)| under one experiment.

This makes the queue length extremely unstable and change greatly. Under abnormal circumstances, the entropy E will increase. In normal scenarios, the packet loss of the link is within a reasonable range. A threshold-based detection method can be set according to the above four features.

2) Depth Detection in Control Plane: In the experiment, we collected 30 groups of the value of |*voting*(v_i) – *Bayes_voting*(v_i)| as shown in Fig. 14.

At the initial stage of detection, the value of |*voting*(v_i) – *Bayes_voting*(v_i)| fluctuates from around 0.32 to 9.66. After we injected attacks at the 15th second, it increases sharply and then stabilizes. We repeated the experiments, each time using the same experimental settings, and collected 20 groups of test results when attacks occur, which are shown in Fig. 15.

Fig. 15 shows that the value of |*voting*(v_i) – *Bayes_voting*(v_i)| is stable at about 31, so we set the value of σ to 31. We compared the method proposed in this paper with other detection methods by conducting extensive experiments, which further proves that our method has excellent performance in real-time and detection rate. Also, it reduces communication overhead and CPU utilization of the controller.

Table I presents the real-time performance and detection rate. Here, real-time refers to the detection time cost.

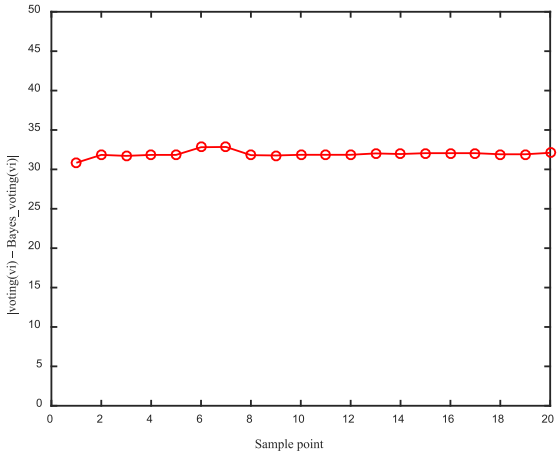


Fig. 15. The value of |*voting*(v_i) – *Bayes_voting*(v_i)| when LDoS attacks occur.

TABLE I
DETECTION PERFORMANCE

Approach	P_D(%)	P_{FN}(%)	P_{FP}(%)	Detection time (s)
XGBoost	98.8	1.8	2.3	2.8
AdaBoost	97.9	2.1	1.9	3.3
HGB-FP [21]	96.2	3.0	4.0	2.7
BA-ANN [37]	98.7	1.4	3.5	1.6
SoftGuard [38]	96.0	5.1	2.2	3.2
Our method	99.1	1.0	1.4	1.3

Our detection method was compared with other existing methods on the same experimental platform. XGBoost and AdaBoost are commonly used machine learning algorithms that boost multiple weak learners into one strong learner, with high detection rates but longer detection time. The HGB-FP algorithm has higher complexity, longer detection time, and higher false alarm rate due to the input of a large number of feature values. The BA-ANN algorithm selects the features to be trained through the ANN neural network model, and constructs a detection model through the Bat iterative algorithm. The time complexity is small, but there is a high false alarm rate. The SoftGuard algorithm utilizes fast Fourier transform to analyze convection, and the time-frequency domain conversion increases a certain detection delay. It can be seen that our method not only has a high detection rate P_D, a low false negative probability P_{FN} and a low false positive probability P_{FP}, but also has preferable real-time performance. The detection time is shorter, and the controller no longer interacts with the switch frequently during global deep detection, which greatly shortens communication delay and maintains a relatively low detection time. This method is suitable for cross-plane detection.

Next, we assess the extra communication overhead induced by our method. The communication overhead is defined as the throughput of the southbound interface. Test results are shown in Fig. 16.

Our method only sends a warning message to the controller when abnormal flow is detected on the data plane, then the controller begins to perform in-depth detection. In this way, the communication volume of the southbound interface approaches 0 under normal circumstances. When LDoS attacks

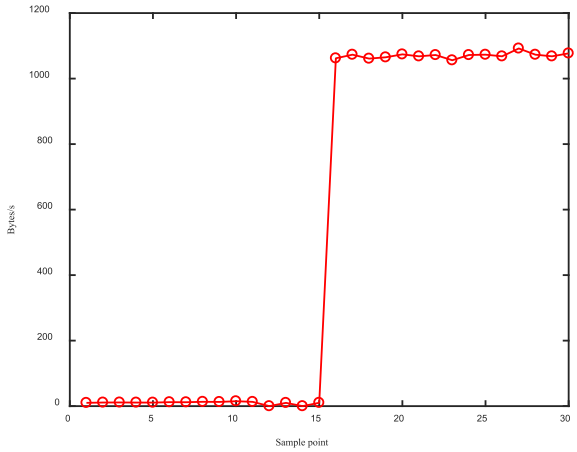


Fig. 16. Communication overhead of the southbound interface.

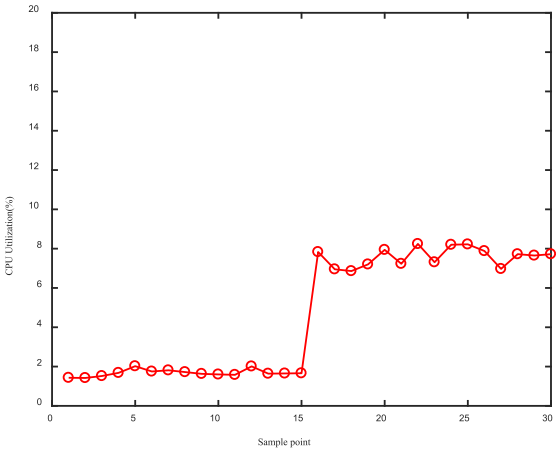


Fig. 17. CPU utilization of controller.

are launched at the 15th second, the coarse-grained detection method in the data plane first detects the abnormality and triggers in-depth detection in the control plane. This increases the communication overhead of the southbound interface..

The communication overhead further effects the CPU utilization of the controller. We test the impact on the controller CPU in the circumstance where we executed the cross plane collaborative detection strategy as shown in Fig. 17.

Fig. 17 shows that the CPU utilization of controller remains at a relatively low value (below 10%). The lightweight detection method is executed on the data plane, occupying only the computational resources of the switch, and not imposing significant load pressure on the controller. When the switch detects the abnormality at the 15th second, it will only report it to the controller. The controller then analyzes and calculates the received information and performs global in-depth detection. At the same time, the real-time detection is high, so the CPU usage time of the controller is short and thus the CPU utilization rate remains at a low value. The CPU of the controller gradually rises to 10%.

In addition, we compared the average communication overhead and CPU utilization with those of other methods as shown in Table II.

TABLE II
MEASUREMENT OF CONTROLLER LOAD

Approach	Communication overhead (Bytes/s)	Average CPU utilization (%)
SAIA [29]	2132	8.5
FloodDefender [31]	1459	11.0
BWManager [34]	2026	20.0
DAISY [35]	1923	10.1
Our method	1068	3.5

Our proposed method implements feature extraction in the data plane, consuming only the computation resources of the switches and not putting great load pressure on the controller. The controller will initiate the detection algorithm based on Bayesian voting when the switches detect abnormality with less computation. DAISY is applicable to threshold detection, so this method has a low CPU utilization in the initial stage. However, once an attack occurs, it will block the defense. Every time an attack is detected, the blocking time will be extended, resulting in a long CPU utilization time. The detection and defense methods of both BWManager and FloodDefender are based on neural networks. These two algorithms have high complexity and are deployed in the controller, resulting in slightly higher CPU utilization of the controller. The CPU resources occupied by SAIA attack detection and defense system showed a linear increase. In the experiment, we configured LDoS attacks on host h_4 and captured all data packets from the southbound interface using wireshark. This paper separates the lightweight detection stage from the deep detection stage, enables the analysis of traffic to be completed on the data plane and identifies the occurrence of abnormal situations in the first place. This reduces the southbound communication overhead of SDN. The in-depth detection stage is deployed on the control plane, fully utilizing the computational power and global perspective of SDN architecture. The communication overhead of our method is significantly lower than the other four methods.

C. Mitigation Performance Analysis

The controller identifies the LDoS attack, discovers the bottleneck position and launches the mitigation mechanism by implementing the Dijkstra algorithm. Our attack mitigation method inevitably brings extra time overhead. Fortunately, existing studies have validated that Dijkstra algorithm has the advantages of fast convergence and robustness [53], [54], [55]. For example, [56] indicated that calculating intra area routes using Dijkstra algorithm took no more than tens of milliseconds on common routers, and [57] indicated that the average convergence time decreased with the number of failure links grows. To validate the performance of our proposed method, we tested the recovery time of the bottleneck link. The test results are shown in Fig. 18.

The link recovery time includes the time to calculate the route, the time to install the forwarding rule to the switch on the backup path, and the time to migrate the flow to the backup path. Fig. 18 shows that the recovery time of bottleneck link maintains a value of about 40ms. The controller does not need to interact with the switch, and it actively installs new

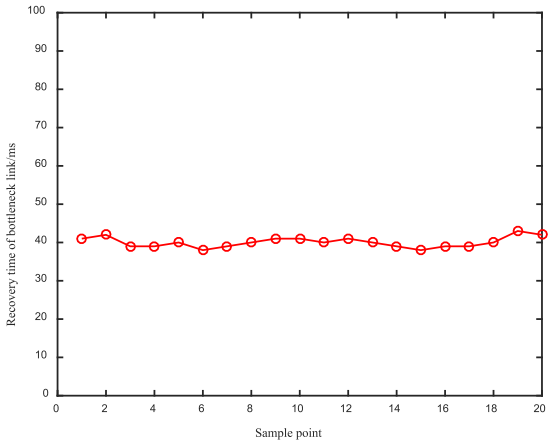


Fig. 18. Recovery time of bottleneck link.

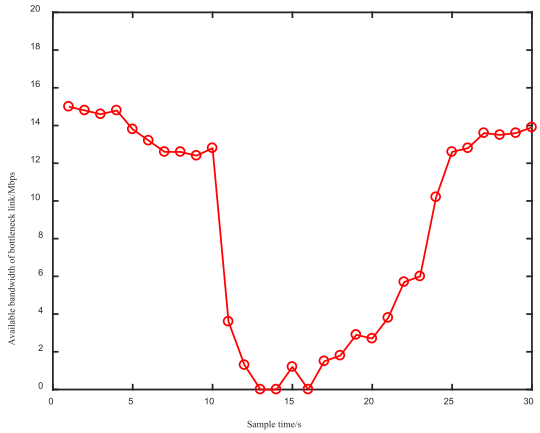


Fig. 19. Available bandwidth of bottleneck link.

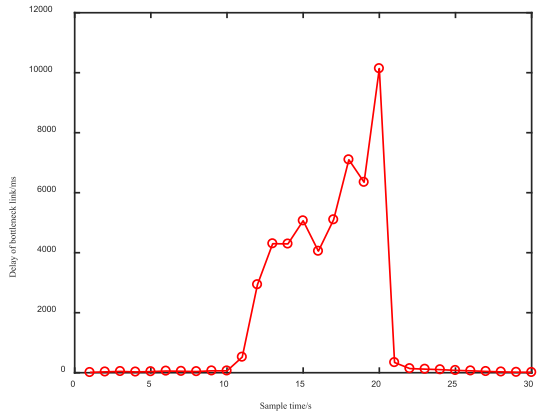


Fig. 20. Delay of bottleneck link.

flow rules to the switch, ensuring that SDN can recover its performance as soon as possible after encountering attacks.

The available bandwidth of the bottleneck link is shown in Fig. 19 and the delay of the bottleneck link is shown in Fig. 20. In the experiment, the data from the 1st to 10th second is collected after sending normal traffic. From Fig. 19 and Fig. 20, it can be seen that the available bandwidth and delay of the bottleneck link are within the normal range of change. Under normal circumstances, the average throughput

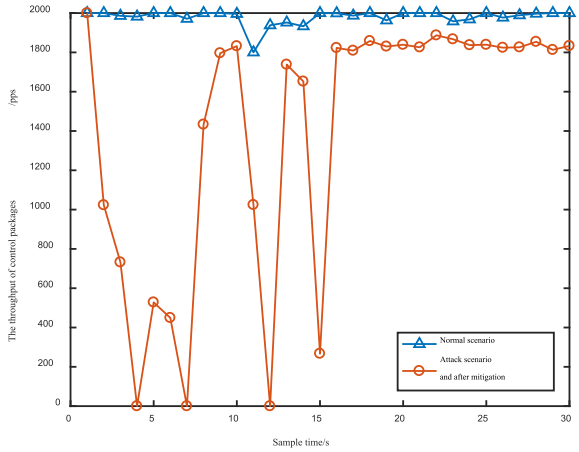


Fig. 21. Throughput of control packets.

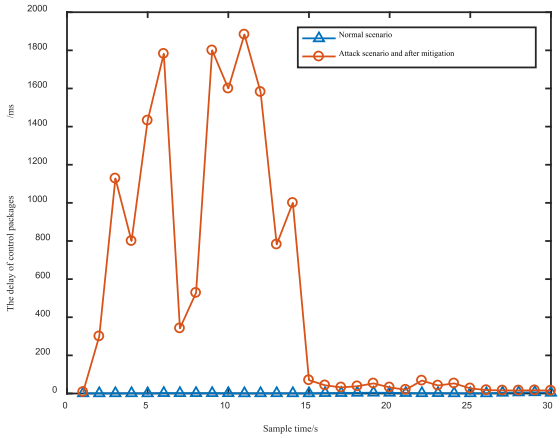


Fig. 22. Delay of control packets.

of the bottleneck link is about 13Mbps, and the average delay is about 15ms. The data from the 10th to 20th second is collected after sending attack traffic. The average throughput of the bottleneck link under the attack is about 1.5Mbps, and the maximum delay can reach about 10000ms. This indicates that LDoS attacks will affect normal TCP connections through the bottleneck link. The data from the 20th to 30th second is collected after the mitigation mechanism is activated. The average available bandwidth of the bottleneck link is about 11Mbps, and the average delay is about 22ms. The above results indicate that the mitigation mechanism proposed in this paper helps to reduce network congestion, thereby increasing the available bandwidth of the bottleneck link and reducing the transmission delay of the link.

Since LDoS attacks can interfere with the transmission of control traffic, we validate the recovery effect of the mitigation mechanism on control flow in experiments. Fig. 21 and Fig. 22 show the throughput and delay of control packages. Throughput is defined as packets per second (pps) on the southbound interface. Delay is defined as one-way delay of a packet transmitted from the controller to its directly connected switch.

In order to test the throughput of the control flow, we made the controller generate 2000 control packets per second

TABLE III
COMPARISON OF MITIGATION MODEL PERFORMANCE

Method	Recovery time	Available bandwidth (Recovery percentage)	Delay
Extended Dijkstra [44]	43ms	75.6%	37ms
NSGAII [45]	58ms	76.1%	30ms
ACO-R [46]	72ms	82.3%	28ms
Our method	40ms	84.6%	22ms

to switch S_1 in the experiment. As shown in Fig. 21, under normal circumstances, the throughput can reach 2000 packets per second. The data before the 15th second is the collected data after sending attack traffic, and the data after the 15th second is the collected data after activating the mitigation mechanism. In the case of attack, the throughput of the control packets becomes extremely unstable and may even drop to 0 due to the timeout retransmission mechanism. After the controller detects the attack, it initiates a mitigation mechanism to reduce network congestion. After the attack defense method takes effect, the throughput of the control packets quickly returns to normal level, and stabilizes at about 1831pps, which is 91.5% of the normal situation.

As shown in Fig. 22, the average delay of the control packets under normal circumstances is 3.21ms. The average delay of the control packets under attack is about 300 times higher than that under normal scenarios. The delay is mostly less than 8ms without attacks, and fluctuates widely between 8ms and 1800ms under attacks. The data after the 15th second is collected after the deployment of the mitigation mechanism, and the average delay of the control packet is reduced to about 30ms. The above results indicate that the mitigation mechanism proposed in this paper helps to reduce network congestion, thereby improving the throughput of the control packets and reducing the transmission delay of the control traffic.

In order to further verify the performance of the mitigation method proposed in this paper, it was compared with other routing algorithm models, and the results are shown in Table III.

The Extended Dijkstra method [44] extends the original Dijkstra algorithm by considering the weights of nodes and edges simultaneously. During network changes, some nodes may not be suitable as a certain hop point for the optimal path, resulting in low computational complexity. Therefore, the recovery time for bottleneck links is shorter, but the available bandwidth and delay recovery ratio are lower. The NSGAII [45] method obtains QoS parameters on the link through monitoring, and then establishes a multi-objective function optimization model based on the bandwidth requirements of the new data flow to find the optimal path, thereby alleviating network congestion. However, due to the mutual constraints between multiple objectives, the algorithm structure is more complex and the recovery time is longer. During the optimization process, the progress of the optimization of each objective is inoperable, resulting in relatively low percentages of available bandwidth recovery and delay recovery. ACO-R [46] deployed a routing algorithm based on the ant colony algorithm which includes continuous iteration and update, thereby requiring more computational time. Therefore, the recovery time on bottleneck links is slightly longer, while the available bandwidth recovery and the delay recovery effect are both higher. Due to the fact that LDoS attacks can reduce the throughput of a certain node, this paper uses the traffic size of each destination and the bandwidth capacity of the node's exit port as decision indicators for path selection. After the controller detects the attack, it immediately reroutes the bottleneck node based on the information reported by the switch, ensuring the real-time and efficient recovery without occupying too much CPU of the controller. The recovery time of the link is about 40ms and the available bandwidth and delay of the link can be restored to a higher proportion.

VI. CONCLUSION

SDN conducts an unconventional concept which realizes the separation of control and forwarding, breaks the closure between different devices in the traditional network, and shortens the network deployment cycle. LDoS attack is one of the fundamental threats faced by the SDN. This paper proposed a cross-plane cooperative detection method against LDoS attacks. We deployed lightweight detection methods in the data plane. The data plane can conveniently count each flow separately. We determined whether there is an exception based on four characteristics. We conducted in-depth detection based on Bayesian voting mechanism according to the information reported by the switch. After detecting the attack, the controller recalculates the route to the bottleneck node to complete mitigation. The proposed method makes full use of the global perspective of the controller, improves the detection rate, and effectively reduces the controller load and detection delay. In the future, four aspects are worth researching. 1) Taking advantages of SDN to further improve the other performance of Dijkstra algorithm (e.g., link failure tolerant, load balance, et al.). 2) Combining moving target defense with SDN. The global view of the controller in SDN can promote the effectiveness of moving target defense, thereby increasing the complexity and cost of attacks and reducing the attack success rate. 3) Constructing attack mitigation strategies by resource allocation. When the attack traffic is detected, the network traffic can be migrated from the victim bottleneck switch to other switches, and then the idle resources of these switches can be used to mitigate attacks. 4) Further exploring other possible low rate DoS attack models in SDN, precisely characterizing their behaviors and developing effective countermeasures.

REFERENCES

[1] Y. Maleh, Y. Qasmaoui, K. El Gholam, Y. Sadqi, and S. Mounir, "A comprehensive survey on SDN security: Threats, mitigations, and future directions," *J. Reliab. Intell. Environ.*, vol. 9, pp. 201–239, Jun. 2023, doi: 10.1007/s40860-022-00171-8.

[2] I. Maity, S. Misra, and C. Mandal, "SCOPE: Cost-efficient QoS-aware switch and controller placement in hybrid SDN," *IEEE Syst. J.*, vol. 16, no. 3, pp. 4873–4880, Sep. 2022, doi: 10.1109/JSYST.2021.3124280.

[3] R. Aryan, A. Yazidi, F. Brattensborg, Ø. Kure, and P. E. Engelstad, "SDN Spotlight: A real-time OpenFlow troubleshooting framework," *Future Gener. Comput. Syst.*, vol. 133, pp. 364–377, Aug. 2022, doi: 10.1016/j.future.2022.03.014.

[4] V. De M. Rios, P. R. M. Inácio, D. Magoni, and M. M. Freire, "Detection and mitigation of low-rate denial-of-service attacks: A survey," *IEEE Access*, vol. 10, pp. 76648–76668, 2022, doi: 10.1109/ACCESS.2022.3191430.

[5] A. Kuzmanovic and E. W. Knightly, "Low-rate TCP-targeted denial of service attacks: The shrew vs. the mice and elephants," in *Proc. SIGCOMM: Appl., Technol., Architect., Protoc. Comput. Commun.*, 2003, pp. 75–86.

[6] I. Cvitić, D. Perakovic, B. B. Gupta, and K.-K. R. Choo, "Boosting-based DDoS detection in Internet of Things systems," *IEEE Internet Things J.*, vol. 9, no. 3, pp. 2109–2123, Feb. 2022, doi: 10.1109/JIOT.2021.3090909.

[7] X. Sun, R. Torres, and S. Rao, "Preventing DDoS attacks on Internet servers exploiting P2P systems," *Comput. Netw.*, vol. 54, no. 15, pp. 2756–2774, Oct. 2010, doi: 10.1016/j.comnet.2010.05.021.

[8] J. Cheng, R. Xu, X. Tang, V. S. Sheng, and C. Cai, "An abnormal network flow feature sequence prediction approach for DDoS attacks detection in big data environment," *Comput. Mater. Contin.*, vol. 55, no. 1, pp. 95–119, 2018, doi: 10.3970/cmc.2018.055.095.

[9] C. Yoon et al., "Flow wars: Systemizing the attack surface and defenses in software-defined networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3514–3530, Dec. 2017, doi: 10.1109/TNET.2017.2748159.

[10] J. C. C. Chica, J. C. Imbachi, and J. F. B. Vega, "Security in SDN: A comprehensive survey," *J. Netw. Comput. Appl.*, vol. 159, Jun. 2020, Art. no. 102595, doi: 10.1016/j.jnca.2020.102595.

[11] B. Mladenov and G. Iliev, "Studying the effect of internal DOS attacks over SDN controller during switch registration process," in *Proc. Int. Symp. Netw. Comput. Commun.*, Shenzhen, China, 2022, pp. 1–4.

[12] M. U. Farooq, M. Rashid, F. Azam, Y. Rasheed, M. W. Anwar, and Z. Shahid, "A model-driven framework for the prevention of DoS attacks in software defined networking (SDN)," in *Proc. IEEE Int. Syst. Conf.*, Vancouver, BC, Canada, 2021, pp. 1–7.

[13] J. Galeano-Brajones, J. Carmona-Murillo, J. F. Valenzuela-Valdés, and F. Luna-Valero, "Detection and mitigation of DoS and DDoS attacks in IoT-based stateful SDN: An experimental approach," *Sensors*, vol. 20, p. 816, Feb. 2020.

[14] K. Hong, Y. Kim, H. Choi, and J. Park, "SDN-assisted slow HTTP DDoS attack defense method," *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 688–691, Apr. 2018, doi: 10.1109/LCOMM.2017.2766636.

[15] T. Lukaseder, L. Maile, B. Erb, and F. Kargl, "SDN-assisted network-based mitigation of slow DDoS attacks," in *Proc. 14th Int. Conf. Secur. Privacy Commun. Netw.*, 2018, pp. 102–121.

[16] S. Axelsson, "The base-rate fallacy and its implications for the difficulty of intrusion detection," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 3, pp. 186–205, Aug. 2000, doi: 10.1145/357830.357849.

[17] R. Sommer, "Viable network intrusion detection in high-performance environments," Ph.D. dissertation, Dept. Comput. Sci., Technische Universität München, Munich, Germany, 2005.

[18] M. Delio, "New breed of attack zombies lurk," Accessed: Jan. 1, 2024. [Online]. Available: https://www.wired.com/2001/05/new-breed-ofattack-zombies-lurk

[19] Q. Zhu, Z. Yizhi, and X. Chuiyi, "Research and survey of low-rate denial of service attacks," in *Proc. 13th Int. Conf. Adv. Commun. Technol. (ICACT)*, Gangwon, Korea (South), 2011, pp. 1195–1198.

[20] J. Cao et al., "The Crosspath attack: Disrupting the SDN control channel via shared links," in *Proc. 28th USENIX Secur. Symp.*, 2019, pp. 19–36.

[21] D. Tang, S. Zhang, Y. Yan, J. Chen, and Z. Qin, "Real-time detection and mitigation of LDoS attacks in the SDN using the HGB-FP algorithm," *IEEE Trans. Services Comput.*, vol. 15, no. 6, pp. 3471–3484, Nov./Dec. 2022, doi: 10.1109/TSC.2021.3102046.

[22] H. Liu, "A new form of DoS attack in a cloud and its avoidance mechanism," in *Proc. ACM Workshop Cloud Comput. Secur. Workshop*, 2010, pp. 65–76.

[23] M. Sides, A. Bremler-Barr, and E. Rosensweig, "Yo-Yo attack: Vulnerability in auto-scaling mechanism," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 103–104, Aug. 2015, doi: 10.1145/2829988.2790017.

[24] Y. Tong, B. Zhihua, G. Zhen, Y. Lina, and Z. Lei, "Detection and defense mechanism of LDoS attack in SDN environment," *J. Front. Comput. Sci. Technol.*, vol. 14, no. 4, pp. 566–577, 2020, doi: 10.3778/j.issn.1673-9418.1905043.

[25] K. S. Sahoo, D. Puthal, M. Tiwary, J. J. P. C. Rodrigues, B. Sahoo, and R. Dash, "An early detection of low rate DDoS attack to SDN based data center networks using information distance metrics," *Future Gener. Comput. Syst.*, vol. 89, pp. 685–697, Dec. 2018, doi: 10.1016/j.future.2018.07.017.

[26] M. Yue, J. Li, Z. Wu, and M. Wang, "High-potency models of LDoS attack against CUBIC +RED," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4950–4965, 2021, doi: 10.1109/TIFS.2021.3117066.

[27] Z. Wu, L. Zhang, and M. Yue, "Low-rate DoS attacks detection based on network multifractal," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 5, pp. 559–567, Sep./Oct. 2016, doi: 10.1109/TDSC.2015.2443807.

[28] Y. Xiang, K. Li, and W. Zhou, "Low-Rate DDoS attacks detection and traceback by using new information metrics," *IEEE Trans. Inf. Forensics Security*, vol. 6, pp. 426–437, 2011, doi: 10.1109/TIFS.2011.2107320.

[29] S. Xie, C. Xing, G. Zhang, and J. Zhao, "A table overflow LDoS attack defending mechanism in software-defined networks," *Secur. Commun. Netw.*, vol. 2021, Jan. 2021, Art. no. 6667922, doi: 10.1155/2021/6667922.

[30] D. Tang, Y. Yan, S. Zhang, J. Chen, and Z. Qin, "Performance and features: Mitigating the low-rate TCP-targeted DoS attack via SDN," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 428–444, Jan. 2022, doi: 10.1109/JSAC.2021.3126053.

[31] S. Gao, Z. Peng, B. Xiao, A. Hu, Y. Song, and K. Ren, "Detection and mitigation of DoS attacks in software defined networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1419–1433, Jun. 2020, doi: 10.1109/TNET.2020.2983976.

[32] Y. Cao, H. Jiang, Y. Deng, J. Wu, P. Zhou, and W. Luo, "Detecting and mitigating DDoS attacks in SDN using spatial-temporal graph convolutional network," *IEEE Trans. Dependable Secure Comput.*, vol.19, no. 6, pp. 3855–3872, Nov./Dec. 2022, doi: 10.1109/TDSC.2021.3108782.

[33] B. Han, X. Yang, Z. Sun, J. Huang, and J. Su, "OverWatch: A cross-plane DDoS attack defense framework with collaborative intelligence in SDN," *Secur. Commun. Netw.*, vol. 2018, Jan. 2018, Art. no. 9649643, doi: 10.1155/2018/9649643.

[34] T. Wang, Z. Guo, H. Chen, and W. Liu, "BWManager: Mitigating denial of service attacks in software-defined networks through bandwidth prediction," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 4, pp. 1235–1248, Dec. 2018, doi: 10.1109/TNSM.2018.2873639.

[35] M. Imran, M. H. Durad, F. A. Khan, and H. Abbas, "DAISY: A detection and mitigation system against denial-of-service attacks in software-defined networks," *IEEE Syst. J.*, vol. 14, no. 2, pp. 1933–1944, Jun. 2020, doi: 10.1109/JSYST.2019.2927223.

[36] Z. A. El Houda, L. Khoukhi, and A. S. Hafid, "Bringing intelligence to software defined networks: Mitigating DDoS attacks," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 4, pp. 2523–2535, Dec. 2020, doi: 10.1109/TNSM.2020.30148703.

[37] X. Li, N. Luo, D. Tang, Z. Zheng, Z. Qin, and X. Gao, "BA-BNN: Detect LDoS attacks in SDN based on bat algorithm and BP neural network," in *Proc. IEEE Int. Conf. Parallel Distrib. Process. Appl. Big Data Cloud Comput., Sustain. Comput. Commun., Soc. Comput. Netw. (ISPA/BDCloud/SocialCom/SustainCom)*, New York City, NY, USA, 2021, pp. 300–307.

[38] R. Xie, M. Xu, J. Cao, and Q. Li, "SoftGuard: Defend against the low-rate TCP attack in SDN," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Shanghai, China, 2019, pp. 1–6.

[39] Y. Zhang and N. Ansari, "On architecture design, congestion notification, TCP incast and power consumption in data centers," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 39–64, 1st Quart., 2013, doi: 10.1109/SURV.2011.122211.00017.

[40] J. Mao, B. Han, Z. Sun, X. Lu, and Z. Zhang, "Efficient mismatched packet buffer management with packet order-preserving for OpenFlow networks," *Comput. Netw.*, vol. 110, pp. 91–103, Dec. 2016, doi: 10.1016/j.comnet.2016.09.016.

[41] M. Panda and M. R. Patra, "Network intrusion detection using naïve Bayes," *Int. J. Comput. Sci. Netw. Secur.*, vol. 7, no. 12, pp. 258–263, Dec. 2007.

[42] D. M. Farid and N. Haibi, "Combining naïve Bayes and decision tree for adaptive intrusion detection," *Int. J. Netw. Secur. Appl.*, vol. 2, no. 2, pp. 12–25, May 2010, doi: 10.5121/ijnsa.2010.2202.

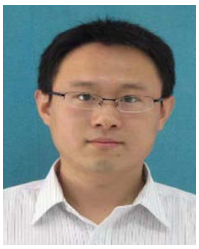
[43] C. Li et al., "QL-STCT: An intelligent routing convergence method for SDN link failure," *J. Commun.*, vol. 43, no. 2, pp. 131–142, 2022, doi: 10.11959/j.issn.1000-436x.2022038.

[44] J.-R. Jiang, H.-W. Huang, J.-H. Liao, and S.-Y. Chen, "Extending Dijkstra's shortest path algorithm for software defined networking," in *Proc. 16th Asia–Pac. Netw. Oper. Manage. Symp.*, 2014, pp. 1–4.

[45] D. Li, X. Wang, Y. Jin, and H. Liu, "Research on QoS routing method based on NSGAII in SDN," *J. Phys. Conf. Ser.*, vol. 1656, no. 1, 2020, Art. no. 012027, doi: 10.1088/1742-6596/1656/1/012027.

[46] M. Tanha, D. Sajjadi, R. Ruby, and J. Pan, "Traffic engineering enhancement by progressive migration to SDN," *IEEE Commun. Lett.*, vol. 22, no. 3, pp. 438–441, Mar. 2018, doi: 10.1109/LCOMM.2018.2789419.

[47] V. Kumar, S. Jangir, and D. G. Patanvariya, "Traffic load balancing in SDN using round-robin and Dijkstra based methodology," in *Proc. Int. Conf. Adv. Technol. (ICONAT)*, 2022, pp. 1–4.
[48] A. Botta, A. Dainotti, and A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios," *Comput. Netw.*, vol. 56, no. 15, pp. 3531–3547, 2012, doi: 10.1016/j.comnet.2012.02.019.
[49] P. Biondi. "Scapy." Scapy.net. Accessed: Jan. 1, 2024. [Online]. Available: https://scapy.net/
[50] S. Y. Qiao, C. C. Hu, H. Li, X. H. Guan, and J. H. Zou, "A mechanism of taming the flow table overflow in OpenFlow switch," *Chin. J. Comput.*, vol. 41, no. 9, pp. 2003–2015, 2018, doi: 10.11897/SP.J.1016.2018.02003.
[51] W. Braun and M. Menth, "Software-defined networking using OpenFlow: Protocols, applications and architectural design choices," *Future Internet*, vol. 6, no. 2, pp. 302–336, May 2014, doi: 10.3390/fi6020302.
[52] L. Xu, J. Huang, S. Hong, J. Zhang, and G. Gu, "Attacking the brain: Races in the SDN control plane," in *Proc. 26th USENIX Secur. Symp. (USENIX Secur.)*, 2017, pp. 451–468.
[53] J. Moy, "OSPF Version 2," Internet Eng. Task Force, RFC-2328, Apr. 1998.
[54] M. Goyal et al., "Improving convergence speed and scalability in OSPF: A survey," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 2, pp. 443–463, 2nd Quart., 2012, doi: 10.1109/SURV.2011.011411.00065.
[55] J. Wu, F. Dai, X. Lin, J. Cao, and W. Jia, "An extended fault-tolerant link-state routing protocol in the Internet," *IEEE Trans. Comput.*, vol. 52, no. 10, pp. 1298–1311, Oct. 2003, doi: 10.1109/TC.2003.1234527.
[56] A. Shaikh and A. Greenberg, "Experience in black-box OSPF measurement," in *Proc. 1st ACM SIGCOMM Workshop Internet Meas.*, 2001, pp. 113–125.
[57] A. Nastiti, A. Rakhmatsyah and M. A. Nugroho, "Link failure emulation with Dijkstra and bellman-ford algorithm in software defined network architecture (case study: Telkom university topology)," in *Proc. 6th Internet Conf. Inf. Commun. Technol.*, Bandung, Indonesia, 2018, pp. 135–140.



Meng Yue received the Ph.D. degree in information and communication engineering from Tianjin University, China, in 2017. He is an Associate Professor with the College of Safety Science and Engineering, Civil Aviation University of China. His current research interests include aeronautical telecommunication network and cyber security.

Qingxin Yan received the B.S. degree from the North University of China, China. She is currently pursuing the master's degree in information security with the College of Electronic Information and Automation, Civil Aviation University of China. Her research interests include security of information security.

Zichao Lu received the B.S. degree from the North China Institute of Aerospace Engineering, China, in 2021. He is currently pursuing the master's degree in information security with the College of Electronic Information and Automation, Civil Aviation University of China. His research interests include security of information security.

Zhijun Wu received the Ph.D. degree in cryptography from the Beijing University of Posts and Telecommunications, China, in 2004. He is a Professor with the College of Safety Science and Engineering, Civil Aviation University of China, where he is the supervisor of Ph.D. candidates. His current research interests include network security and cloud computing.