Detecting and Mitigating DDoS Attacks in SDN Using Spatial-Temporal Graph Convolutional Network

Yongyi Cao, Hao Jiang, *Member, IEEE*, Yuchuan Deng, Jing Wu, *Member, IEEE*, Pan Zhou, *Senior Member, IEEE*, and Wei Luo

Abstract—With the development of data plane programmable Software-Defined Networking (SDN), Distributed Denial of Service (DDoS) attacks on the data plane increasingly become fatal. Currently, traditional attack detection methods are mainly used to detect whether a DDoS attack occurs and it is difficult to find the path that the attack flow traverses the network, which makes it difficult to accurately mitigate DDoS attacks. In this article, we propose a detection method based on Spatial-Temporal Graph Convolutional Network (ST-GCN) over the data plane programmable SDN, which maps the network into a graph. It senses the state of switches through In-band Network Telemetry (INT) with sampling, inputs the network state into the spatial-temporal graph convolutional network detection model, and finally finds out the switches through which DDoS attack flows pass. Based on this, we propose a defense method combined with an enhanced whitelist and a precise dropping strategy, which can effectively mitigate DDoS attacks and minimize the impact on legitimate network traffic. The evaluation results show that our detection method can accurately detect the path that the DDoS attack flows pass through, and can effectively mitigate the DDoS attack. Compared to classic methods, our method improves the detection accuracy by nearly 10%. At the same time, the southbound interface load and CPU overhead brought by our detection and defense process are much lower than the classic methods.

Index Terms—DDoS, data plane programmable SDN, in-band network telemetry, spatial-temporal graph convolutional network

1 Introduction

DISTRIBUTED Denial of Service (DDoS) is an easy-to-initiate and highly destructive network attack. An attacker usually invades vulnerable nodes in the Internet which are then turned to a botnet and uses these massively distributed hosts to create massive packets with forged IP addresses to launch access attacks on victim servers. It can quickly consume resources of victim servers and crash them so that the victim servers are unable to respond to normal requests, that is, denial of service [1], [2], [3].

DDoS attacks are widespread and Software-Defined Networking (SDN) networks also suffer from such attacks [4], [5], [6]. In the OpenFlow era, DDoS attacks in SDN are mostly directed to the control plane. For example, an attacker generates a large number of attack flows which cannot match flow table rules with spoofing addresses. Each attack flow will cause the switch to send a $packet\_in$ message to the controller which is likely to block the southbound interface and greatly consume the resources of the controller [7]. However, with the widespread application of data plane programmable SDN, attacks on the data plane have gradually increased. For example, an attacker invades a host inside a SDN cloud center, making it a bot to attack other users in the network, exhausting resources like memory of other users and bandwidth of corresponding links [4], [8]. This is the type of DDoS attacks we focus on in this paper, i.e., flooding-based DDoS attacks on SDN.

Recent years, DDoS attacks have occurred more and more frequently. The servers of many giant Internet companies such as Apple, Amazon, Alibaba have suffered from DDoS attacks [9]. Consequently, DDoS attacks have caused huge economic losses to Internet companies and the entire society. In this regard, researchers have made a lot of efforts to detect and mitigate DDoS attacks, but the detection and defense of DDoS attacks still face some challenges such as forged IP addresses, forged traffic patterns [10], and encrypted messages [11], etc.

As for DDoS attack detection methods, signature-based detection methods cannot cope with encrypted data. Among anomaly-based detection methods, statistics-based methods usually only consider the change of the statistical property of one single indicator (e.g., packet arrival rate) which is easily bypassed through forged traffic patterns, and it is difficult for this type of method to distinguish between flash crowd traffic and DDoS attack traffic. More importantly, most existing detection methods cannot find

• *Yongyi Cao, Hao Jiang, Yuchuan Deng, and Jing Wu are with the Electronic Information School, Wuhan University, Wuhan, Hubei 430072, China. E-mail: {caoyongyi, jh, dengyc, wujing}@whu.edu.cn.*
• *Pan Zhou is with the Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China. E-mail: panzhou@hust.edu.cn.*
• *Wei Luo is with the China Ship Development and Design Center, Wuhan, Hubei 430064, China. E-mail: 171772014@qq.com.*

*Manuscript received 5 February 2021; accepted 23 August 2021. Date of publication 1 September 2021; date of current version 11 November 2022. (Corresponding authors: Hao Jiang and Jing Wu.)
Digital Object Identifier no. 10.1109/TDSC.2021.3108782*

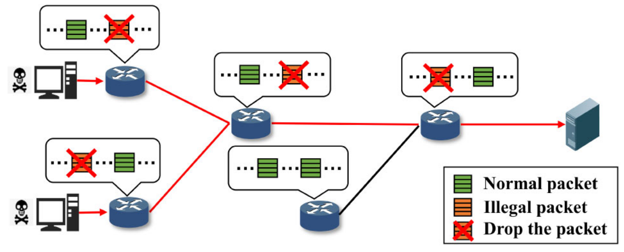
out the specific path of DDoS attack traffic in the network, which is not conducive to subsequent defense.

As for defense against DDoS attacks, a key issue is to distinguish between attack traffic and legitimate traffic, which is hard to accomplish. Some methods perform rate limiting of traffic on switches which do not distinguish between legitimate traffic and attack traffic. Some other methods such as the middle box direct all the network traffic to an external host for cleaning, which require additional hardware and will bring unpredictable delay. In addition, the defense strategy either deployed on all switches (distributed) or on the destination switch (centralized). This will cause unnecessary overhead if the controller indiscriminately issuing defense strategies to all switches. Because usually not all switches in the network contain DDoS attack traffic, however, the delay of centralized defense is higher and its defense efficiency is not as good as distributed defense [8].

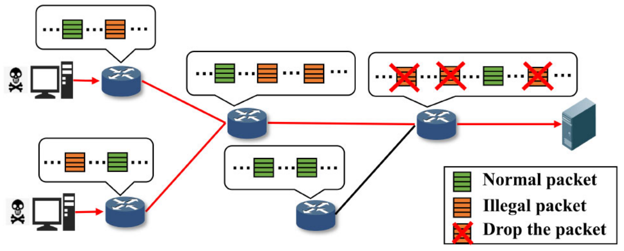
Though detecting and mitigating DDoS attacks is not easy, researchers found that when a DDoS attack occurs in SDN network, it usually has the characteristics of large traffic, increased source IP entropy, and a sharp increase in the number of single flows [12]. In other words, some of the network state will change rapidly when a DDoS attack occurs. Therefore, we can detect DDoS attacks by monitoring the change of network state. Luckily, it is not difficult to obtain fine-grained network state with In-band Network Telemetry (INT) technology based on programmable data plane. The fine-grained information can grasp the state of each switch in the network. As a result, compared with traditional SDN network, we can not only obtain more fine-grained temporal features of network state but also utilize the spatial features of network state under data plane programmable SDN. With this information, we can detect the DDoS attacks, trace the path that attack flows passed, i.e., attack path and formulate corresponding defense strategies.

In this paper, we propose a DDoS detection model based on Spatial-Temporal Graph Convolutional Network (ST-GCN) which can model the network topology as a graph and it extracts both temporal and spatial features of network state. And then we can finally find the attack path in the network. This detection model not only uses multiple statistical attributes, but also uses the topology structure of network traffic. Therefore, it is difficult for an attacker to bypass the detection through forging traffic patterns. Furthermore, we propose a method based on enhanced whitelist and precise dropping strategy to mitigate DDoS attacks as we identify the switches attack flows passed through. The enhanced whitelist protects most legitimate traffic and it can distinguish flash crowd traffic and DDoS attack traffic by detecting single flows. The traffic not being protected by the whitelist is the newly arrived legitimate traffic and DDoS attack traffic. And we protect the newly arrived legitimate traffic to the greatest extent and drop the attack traffic through the precise dropping strategy.

In the context of DDoS defense in SDN, existing methods usually only use the switch closest to the destination host (i.e., edge switch) for defense [13], as shown in Fig. 1b, which will aggregate all attack flows to one point and cause a great burden on the edge switch and the corresponding links. However, as shown in Fig. 1a, our mitigation method



(a) Distributed Defense



(b) Centralized Defense

Fig. 1. Comparison of distributed defense and centralized defense.

coordinates all switches in the attack path. In this way, we advance the defense process to the time when the attack flow enters the network, and greatly reduce the burden on each switch and corresponding links. Therefore, our method is more robust under highly intensive attacks.

The main contributions of this paper are summarized as follows:

- Modeling the network states. We use INT with sampling to collect network states while minimizing the consumption of computing resources of controller and bandwidth of southbound interface. The network states are then converted to a three-dimensional network state tensor which reflects the features of all switches at different time.
- Detecting DDoS attacks over data plane programmable SDN using ST-GCN. Existing DDoS attack detection methods mainly focus on temporal features of network states. In data plane programmable SDN, ST-GCN is used to find both temporal and spatial features of network state and finally find the attack path.
- Mitigating DDoS attacks by enhanced whitelist and precise dropping strategy. Since the switches containing attack flows can be identified during the attack detection phase, we propose a more targeted defense strategy to block as much attack flows as possible and minimize the chance of accidental killing of normal flows.

The rest of the paper is organized as follows. Section 2 introduces the the background of technologies and the threat model that we use in this paper. Section 3 is the system overview. The details of network state awareness and the DDoS attacks detection and mitigation are in Sections 4 and 5, respectively. We introduce the implementation and evaluation of the system in Section 6. Then, some other problems are discussed in Section 7. Section 8 introduces related work. We conclude this paper and talk about future work in Section 9.

2 Background

2.1 Data Plane Programmable SDN and INT

Based on the idea of decoupling the control plane and the data plane in traditional network equipments, researchers first proposed the concept of OpenFlow in 2008 [14]. McKeown *et al.* subsequently proposed the concept of SDN based on OpenFlow [15]. However, the scalability of the data plane model defined by OpenFlow protocol is poor [16]. With the development of Programmable Protocol-independent Packet Process (P4) language and programmable switches, the flexibility of SDN is no longer limited to the control plane, and the data plane also obtains programmable capabilities [17].

Before the emergence of programmable data plane, network administrators were only able to obtain lagging and inaccurate network telemetry information through terminal devices at the edge of the network. The advent of INT technology based on programmable data plane has largely alleviated the aforementioned dilemmas faced by network administrators. INT uses the customizable features of programmable devices to directly obtain packet-level telemetry data within the device when forwarding packets. Compared with out-band measurement method, the network state parameters obtained by INT are more granular, richer, and more sensitive to the changes of network status [18], [19].

The indicators collected by INT mainly includes switch ID, queuing delay, buffer delay, link delay, ingress/egress port ID, ingress/egress timestamp, in/out buffer depth of egress port, and five-tuples, i.e., *{source IP address, destination IP address, source port number, destination port number, protocol type}* [20], [21]. This information reflects the status of the network equipment when forwarding packets. Their values are highly related to the size of the network traffic, instantaneous throughput etc. When abnormal events occur in the network, values of these indicators will change accordingly. Identifying, extracting and analyzing how these indicators change can help us understand abnormal events in the network.

Using INT to obtain more fine-grained network state and the forwarding path of each packet, we can not only have a global view of the network but also obtain the state of all nodes in the network. In other words, if the network topology is regarded as a graph, we can obtain the connection as well as the features of all nodes in the graph.

2.2 Threat Model

OpenFlow switches are being replaced by more flexible programmable switches. However, SDN networks in the era of programmable data plane also face serious threats from DDoS attacks. Data center is currently the most widely used scenario for SDN, and DDoS attacks inside data centers are common. In this section, we use Header Space Analysis (HSA) [22], [23] to describe the process of DDoS attacks. In HSA, a packet header is regarded as a point in space $\{0,1\}^L$ which is called header space (L is the length of packets in bit). Based on header space, all network forwarding devices can be modeled as a switch transfer function T, which can simulate the forwarding process. Switch transfer function T can be formulated as

$$T(h,p) \rightarrow \{(h_1,p_1),(h_2,p_2)\ldots\}, \tag{1}$$

where h_i means the headers that T can process and p_i means the ports to which packets are forwarded. If the header of the packet arriving at the forwarding device presented by T matches h_i, then it is forwarded to the port p_i of the forwarding device.

The forwarding process of a packet is denoted as

$$(s_1,r_1) \rightarrow \cdots \rightarrow (s_{n-1},r_{n-1}) \rightarrow (s_n,r_n), \tag{2}$$

where s_i represents the ith switch and r_i represents the packet processing rule, i.e., the flow table of ith switch.

Assume a network $G=(V,E)$, where V represents a set of switches i.e., nodes and E represents a set of links between switches i.e., edges. Let c represent the current switch or host, and d represent the destination switch or host. Then, we get

$$N_{c,d} = v, \quad where \quad c,d,v \in V, \tag{3}$$

which means the next hop of the path from node c to destination d is node v.

With network G, we use $L_{s,d}$ to represent all potential paths from source s to destination d and $p_{s,d}$ to represent a specific forwarding path from s to d in $L_{s,d}$. $h_{(v_i,v_{i+1})}$ represents a hop in the path. And we have

$$p_{s,d} = (v_s, v_{s+1}, \ldots, v_d), \quad where \quad N_{v_i,d} = v_{i+1}. \tag{4}$$

Denote the capacity of link (v_i,v_j) as $B^c_{(v_i,v_j)}$, the bandwidth consumed by DDoS attack traffic of link (v_i,v_j) is denoted as $B^d_{(v_i,v_j)}$, and the bandwidth consumed by all traffic including legal business traffic and DDoS attack traffic of link (v_i,v_j) is denoted as $B^t_{(v_i,v_j)}$. The bandwith consumption of any link in the network is subject to

$$0 \leq B^d_{(v_i,v_j)} \leq B^t_{(v_i,v_j)} < B^c_{(v_i,v_j)}. \tag{5}$$

Then, we use $U(h_{(v_i,v_{i+1})})$ to represent the proportion of DDoS traffic contained in the link between v_i and v_{i+1}, which can be calculated as

$$U(h_{(v_i,v_{i+1})}) = \frac{B^d_{(v_i,v_j)}}{B^t_{(v_i,v_j)}}. \tag{6}$$

We assume that the experimental scenario is a data center which employs SDN as an underlying network architecture and the bots compromised by the attacker and the victim are resided in the same network. The attacker has no information about the network topology. The attacker compromises some hosts in the data center to launch a DDoS attack. The target of the attacker is to exhaust the resources of one or some other server(s) or block some link in the data center so that normal requests can not be responded timely. Types of attacks include SYN flood, UDP flood and HTTP flood. When multiple hosts launch an attack at the same time, the attack flows enter the network from different edge switches and the different attack paths may partially overlap.

TABLE 1
Summary of Notations

Notation	Meaning
A	Adjacency matrix of graph G
D	Diagonalized degree matrix
E	Set of edges in graph G
f	The number of features of each node
F	A 3-D tensor of features of nodes
G	A graph mapped by the network topology
I_n	Identity matrix
l	The upper limit of dropping rate
L	Graph Laplacian matrix
Q	Set of IP addresses in the whitelist
V	Set of nodes in graph G
$\|V\|$	The number of nodes
$*$	The operation of convolution
$N_{c,d}$	The next hop of node c for destination d
$L_{s,d}$	All potential paths from source s to destination d
$P_{s,d}$	A specific forwarding path from s to d in $L_{s,d}$
P_{src}	Source IP address of a packet
P_{dst}	Destination IP address of a packet
r_{edge}	Dropping rate of edge switches
r_{mid}	Dropping rate of intermediate switches
$h_{(v_i,v_{i+1})}$	A hop in the path
$U(h_{(v_i,v_{i+1})})$	The proportion of DDoS traffic in the link traffic of a hop
$u_{(v_i,t)}$	The feature vector of node v_i at time slice t
$H(X)$	Entropy of some indicators of packets
R_{INT}	Sampling rate of INT
$T_k(\tilde{L})$	Scaled Chebyshev expansion polynomial of scaled graph Laplacian
λ_{max}	The maximum eigenvalue of the graph Laplacian matrix

For convenience, we summarize notations appear in this paper in Table 1.

2.3 Spatial-Temporal Graph Convolutional Network

In recent years, graph neural networks have developed rapidly due to their excellent capabilities for processing non-euclidean structured data [24], and a lot variants of it have been produced which are widely used in classifying nodes of citation networks [24], recommendation systems [25], traffic flow prediction [26], [27], [28], [29] and even security of e-commerce [30]. Like ordinary neural networks, graph neural networks are mainly used for classification and regression. In the task of classifying, people will abstract the target into a graph and extract the features of its nodes or edges as input, and the output is the classification result of the nodes, edges or graphs. In the study of traffic flow, researchers often use ST-GCN for traffic flow prediction. Inspired by them, we try to apply a ST-GCN in the field of network security.

Using ST-GCN, the spatial relationship between nodes and the temporal features of the network state can be fully utilized, and the detection accuracy is much higher than the probability-based path backtracking [31], [32]. On the other hand, compared to RNN-based graph neural networks, using convolution-based graph networks has many advantages such as fewer training parameters, light overfitting, and fast training speed. And researchers found the learning ability of convolution-based methods is not less than the methods based on RNN [33].

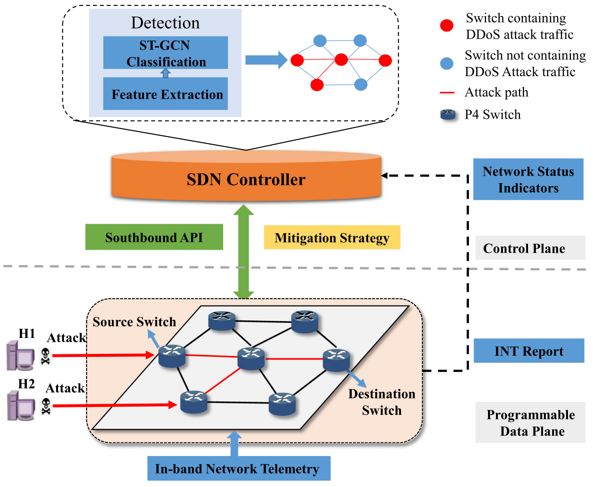


Fig. 2. System overview.

3 System Overview

For DDoS attacks detection over data plane programmable SDN, our goal is to find the attack path and locate the attack source, and then develop a defense strategy to precisely cut off the attack path. In this paper, we use INT to obtain fine-grained network state on programmable switches. Then, we map network state into a three-dimensional tensor and build a DDoS attack detection model based on the ST-GCN on the controller. Finally, we develop a targeted defense strategy based on the detection results and issue them to switches through the controller. Fig. 2 depicts an overview of the system, which contains three stages: network state awareness, DDoS attacks detection and precise mitigation.

1) Network state awareness. Based on P4 language and programmable data plane, we use INT to collect metadata of each switch. In order to reduce extra consumption of CPU computing resource and bandwidth, we adopt an adaptive interval sampling method when performing INT. The INT report and the packet header are parsed at the controller to obtain fine-grained network states. Then, the network is modeled as a spatial-temporal graph on the controller to capture the change of network state. Specifically, we convert the data collected into a three-dimensional tensor, which contains the feature vectors of each switch at different time, at the controller. A feature vector is composed of the states collected by INT and other properties calculated with these states.

2) DDoS attack detection. After we get the three-dimensional tensor, we use the tensor and the adjacency matrix of the network topology as input for the DDoS attack detection model. On the one hand, the detection model convolves the tensor from the perspective of time to extract temporal features of the data. On the other hand, the detection model convolves the tensor with adjacency matrix to extract spatial features of the data, that is, analyze how the states of adjacent nodes affect each other in space. By analyzing and learning temporal and spatial features of data, the model finds the switches which contain DDoS attack flows and finds the forwarding path of attack flows in the network.

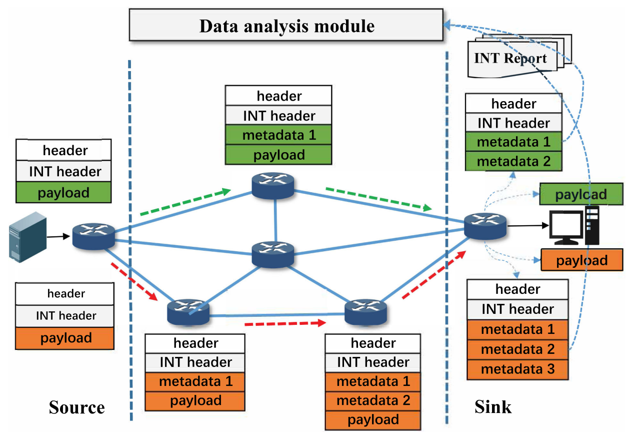


Fig. 3. Process of INT.

3) Precise mitigation. According to the detection result, the SDN controller finds the switches through which attack flows pass, locates the switches through which attack flows enter the network and issues a precise dropping strategy which can both mitigate DDoS attacks effectively and reduce the probability of killing normal packets by accident. Furthermore, we set up an enhanced whitelist on switches and add source and destination IP addresses of commonly used business flows to the whitelist to further prevent normal flows from being accidentally killed.

4 Network State Awareness

It's natural to regard network topology as a graph, switches in the network are nodes in the graph and the connections between switches are edges. We hope that detection model can consider both the topology of the network and the features of nodes when detecting DDoS attacks. Thus, we obtain the states of switches by INT as node features because it can reflect the quality of network service. Furthermore, the data obtained by INT needs to be organized into a suitable form as the input of detection model. And we call this process *network state awareness*.

4.1 INT With Sampling

It is difficult to obtain the state of switches in traditional networks. However, with data plane programmable SDN, state of switches can be obtained easily by INT [17], [18]. INT is a framework on data plane that can collect and report network state. The process of INT is shown in Fig. 3. In the framework of INT, the source switch inserts a special field i.e., telemetry instructions that can be understood by network devices, to the header of packets. When the packets are transmitted in the network, the programmable switch writes its real-time, fine-grained state called metadata to the packet as the telemetry instruction says. After the packets reach the destination switch which is the sink in Fig. 3, the INT report containing measurement information is sent to data analysis module, and the state of the network is obtained. At the same time, the payload of packets is forwarded to the host.

In real scenarios, the network traffic is huge and there are two problems with using INT on large-scale networks. The first is the cost of controller's CPU and memory is high, the second is that INT data requires additional cost of bandwidth [34]. To alleviate these problems, we adopt a sampling strategy when performing INT. We deploy INT on all switches to obtain the state of the entire network, and each packet will be written into metadata according to the INT instruction in packet header when forwarding. Because the state of switches are obtained through packets and the time interval between adjacent packets passing through the same switch is very small, the states of the same switch obtained by adjacent packets are very similar which means a part of data obtained is redundant and it will cause unnecessary consumption of network bandwidth and computing resources. In order to avoid the problem of redundance, we make a trade-off between the granularity of the state and bandwidth consumption, and adopt an adaptive sampling method.

From the perspective of information entropy, the goal of sampling is to reduce the redundance as much as possible. At the same time, in order to make the telemetry data meet the requirements of the subsequent analysis, we have to ensure its information entropy $H(X)$ dose not decrease too much, i.e., the telemetry data obtained in a certain period of time should be sufficiently diverse. Let $p(\frac{n_i}{S}) = p(x_i)$, $H(X) = -\sum_{x \in X} p(x) \log p(x)$, where S is the total number of packets and n_i is the number of occurrences of each source IP address. Our goal is to find a distribution $p^*(X)$ with the maximum $H(X)$ under certain constraints, which is then turned to a convex optimization problem. This can be formulated as

$$\begin{cases} p^*(X) = \arg\max_{X \in \Gamma}(H(X)) = \arg\max_{X \in \Gamma}\left(-\sum_{i=1}^{i=N} p(x_i) \log p(x_i)\right), \\ \sum_{i=1}^{n} p(x_i) f_k(x_i) = F_k, k \in (1, m), \\ \sum_{i=1}^{i=N} p(x_i) = 1, \end{cases} . \quad (7)$$

where F_k is a constant. To solve this problem, we adopt the method based on Lagrange multiplier and assuming that there are $m+1$ constraints, then we get

$$\begin{cases} \mathit{Maximum} \quad f = -\sum_{i=1}^{i=N} p(x_i) \log p(x_i), x_i \in (0,1), \\ \mathit{Subject\ \ to} \quad g_k = \sum_{i=1}^{n} p(x_i) f_k(x_i) = F_k, k \in (1, m), . \\ \qquad g_{m+1} = \sum_{i=1}^{i=N} p(x_i) = 1. \end{cases} \quad (8)$$

Get Lagrange equation

$$\begin{aligned} &\Lambda(p(x_1), \ldots, p(x_n), \lambda_1, \ldots \lambda_{m+1}) \\ &= f + \lambda_{m+1}(g - 1) + \sum_{k=1}^{m} \lambda_k (g_k - F_k). \end{aligned} \quad (9)$$

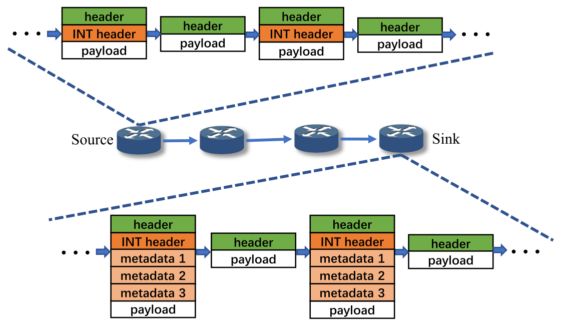


Fig. 4. A Rate-based sampling strategy ($R_{INT} = 2$).

Take partial derivative of $p(x_1), p(x_2), \ldots, p(x_n)$

$$\begin{aligned} &\frac{\partial (f + \lambda_{m+1}(g-1) + \sum_{k=1}^{m} \lambda_k (g_k - F_k))}{\partial p(x_i)} = 0, \\ &-1 - \log_2 p(x_i) + \lambda_{m+1} + \sum_{k=1}^{m} \lambda_k f_k(x_i) = 0. \end{aligned} \tag{10}$$

Because of $1 = \sum_{i=1}^{n} p(x_i) = \frac{1}{Z(\lambda_1, \ldots \lambda_m)} \sum_{i=1}^{n} \exp (\sum_{k=1}^{m} \lambda_k f_k(x_i))$

$$\begin{cases} p(x_i) = \dfrac{1}{z(\lambda_1, \ldots, \lambda_m)} \exp\left(\displaystyle\sum_{k=1}^{m} \lambda_k f_k(x_i)\right), \\ z(\lambda_1, \ldots, \lambda_m) = \displaystyle\sum_{i=1}^{n} \exp\left(\sum_{k=1}^{m} \lambda_k f_k(x_i)\right). \end{cases} \tag{11}$$

Suppose the values of N and S are constants. We can summarize when $p(x_1) = p(x_2) = \ldots p(x_n)$, $\sum_{i=1}^{i=n} p(x_i) = 1$, $H(X)$ takes the maximum value. For a random variable X, the entropy $H(X)$ is the largest when $p(X)$ is uniformly distributed. Therefore, in order to save enough information, packets of every IP address should be uniformly sampled. In practice, we find that adopting the method of equal interval sampling achieves good performance.

When performing sampling, the INT source node inserts INT instructions into the header of a packet in every R_{INT} packet. If a packet contains INT instructions, all specified INT metadata of each switch it passes will be written into it. Otherwise, no metadata will be written as shown in Fig. 4.

R_{INT} is the interval of sampling i.e., the number of unsampled packets between two samples. And R_{INT} is calculated as

$$R_{INT} = \eta \cdot \frac{N}{\Delta T}, \tag{12}$$

where ΔT is a time period, N is the number of packets passing through the switch during ΔT, and η is a constant. R_{INT} changes according to the volume of network traffic. The number of packets passed by a programmable switch will be recorded. When the number of packets per unit time increases, the sampling interval R_{INT} will increase accordingly, thereby reducing the overhead caused by INT.

INT with sampling does not need extra sniffing packets, nor can it affect forwarding of normal packets. Compared with out-band telemetry method, the measurement delay of INT is extremely low. And the data can also maintain a finer granularity because of the adaptive frequency of sampling.

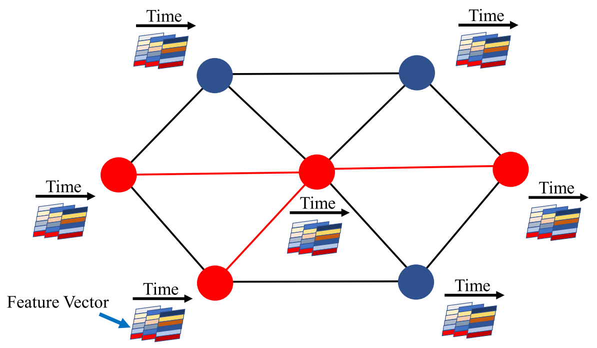


Fig. 5. Modeling the network.

4.2 Network Modeling and Data Conversion

The network is denoted as a graph $G = (V, E, A, F)$ of which features of nodes change over time, as shown in Fig. 5. V is a set of nodes, i.e., programmable switches in this case. E is a set of edges, i.e., links between programmable switches. A is the adjacency matrix of graph G and $A \in \mathbb{R}^{|V| \times |V|}$. F is a three-dimensional tensor containing feature vectors of all nodes at different time, $F \in \mathbb{R}^{f \times |V| \times t}$ where f is the number of features of each node, $|V|$ is the number of nodes, t is the number of time slices. F can be obtained after data conversion.

As shown in Fig. 5, the feature of each node i.e., state of each switch is different which implies the spatial distribution of attack flows. Also, the states of switches change over time due to the constant forwarding of packets. With the beginning and continuation of DDoS attacks, the states of switches change according to some rules. We will explore these rules in Section 5, before that, we have to solve the problems of how to determine the state we use as features of switches and how to convert the data obtained from INT into three-dimensional tensor F.

We have introduced the states of switches that can be collected by INT, and we directly use some of them, i.e., queuing delay, buffer delay, link delay and in/out buffer depth. In addition, we also consider other indicators. One of the most significant features of DDoS attacks is the large number of forged source IP addresses and port numbers, which will cause the entropy of the corresponding indicators to rapidly increase in a short time. Based on this, many studies on DDoS attacks rely on entropy to distinguish DDoS attack flows from normal flows [35], [36], [37]. Entropy is a measure of uncertainty, that is, if the entropy of source IP addresses of packets rapidly increases in a short time, it means that the traffic comes from more different source IP addresses during this time, which indicates that the network is probably suffering from a DDoS attack. Taking the entropy of source IP address as an example, it can be calculated as $H(X) = -\sum_{i=1}^{N} p(\frac{n_i}{S}) logp(\frac{n_i}{S})$, where N is the number of different source IP addresses, the definitions of n_i and S are as mentioned above. We calculate the entropy of source IP addresses of packets and use it as part of the feature vector. In addition, the statistical characteristics of flash crowd traffic and DDoS attack traffic are very similar. However, there are also significant differences in some indicators, such as the number of packets sent per IP address and bidirectional flow. Therefore, we also take the number of unidirectional and bidirectional flows in the switch as part of the feature vector.

After obtaining states of switches, we need to aggregate and convert the data obtained by INT to the three-dimensional feature tensor F. F is the input of detection model, and the detection model learn the rules of how DDoS attacks affect the state of network in both temporal and spatial perspectives according to F. In order to convert the data to a three-dimensional feature tensor F, we classify the data D_{INT} according to switch ID first. Then, we set the size of time slice T_{slice} (such as 200ms) and calculate the mean value or entropy of all state collected in each time slice, so as to obtain the feature vectors $D_{i,j}$ of all switches in a time slice where i means the ith switch and j means at the jth time slice. We reapeat this process and finally get a three-dimensional feature tensor F, which includes the feature vectors of all switches in multiple time slices, and the feature vectors are composed of multiple states. Algorithm 1 describes the process of data conversion.

Algorithm 1. Data Conversion

```
    Require: Data collected by INT process $D_{INT}$
    Ensure: Three-dimensional feature tensor $F$
1:  Initiate $T_{slice}$, i;
2:  sort $D_{INT}$ by switch ID $\rightarrow \{D_1, D_2, \ldots, D_n\}$;
3:  $t = T_{total}/T_{slice}$;
4:  while $i < n$ do
5:    divide $D_i$ into $t$ sets $\{D_{i,1}, D_{i,2}, \ldots, D_{i,t}\}$;
6:    average of every subset in $\{D_{i,1}, D_{i,2}, \ldots, D_{i,t}\}$
7:  $\rightarrow U_i = \{u_{i,1}, u_{i,2}, \ldots, u_{i,t}\}, U_i \in \mathbb{R}^{t \times f}$;
8:    i++;
9:  end while
10: form $\{U_1, U_2, \ldots, U_n\}$ to $F, F \in \mathbb{R}^{f \times |V| \times t}$;
```

5 Detection and Mitigation of DDoS Attacks

5.1 Detection Model of DDoS Attacks

DDoS attacks are continuous in time and have some characteristics in perspective of time. Furthermore, when DDoS attack flows enter the network and are forwarded between switches, it must have different effects on the state of different switches, and the differences in the states of switches must follow certain rules in space. Taking these into consideration, we try to simultaneously model and analyze switch states from perspectives of time and space when a DDoS attack occurs, and recognize the switches that are affected by the attack. We construct a DDoS attack detecting model using CNN and graph convolutional network. The temporal features of data are extracted by CNN and the spatial features of data are extracted by graph convolution considering it is natural to describe the spatial relationship of the network with a graph.

5.1.1 Temporal Convolution

In fact, RNN-based methods are the most widely used methods in time series processing [33]. However, Bai *et al.* proved that simple convolutional models can perform better than RNN-based models such as LSTM on a variety of tasks and datasets [33]. Yu *et al.* also mentioned that RNN-based models have problems such as time-consuming iterations, complex gate mechanisms and slow response to dynamic changes, while CNN-based models are faster to train, simpler in structure and do not overly rely on previous time steps [26]. We use a CNN-based method to extract temporal features of data. The result shows that a simple CNN can achieve good performance.

Inspired by Bai *et al.* [33], we construct a time convolutional module with two convolutional layers and use this module to extract temporal features of data. As mentioned in Section 4, we construct a two-dimensional span $U_{v_i} = [u_{(v_i,t)}, u_{(v_i,t-1)}, \ldots, u_{(v_i,t-s+1)}]$ for each node in graph G, $u_{(v_i,t)}$ is the feature vector of node v_i $(i = 1, 2, \ldots, |V|)$ at time slice t, and then we perform two convolutions on the two-dimensional span of each node using time convolutional module. It should be pointed out that in order to extract features at different scales of time, the sizes of kernels of different convolutional layers are different. With kernels of different size, features of different time scales can be extracted and the results are more stable.

The time convolutional module can be formalized as

$$y = ReLU(\Phi_2 * ReLU(\Phi_1 * U_{v_i})), \tag{13}$$

where U_{v_i} is the input, y is the output, Φ is the parameter of the convolution kernel, $*$ represents the operation of convolution, and $ReLU$ is the activation function we use.

5.1.2 Spatial Convolution

Defferrard *et al.* uses truncated Chebyshev expansion polynomials to define local convolution kernels that can be efficiently learned on graphs [38]. Using this convolution kernel, the signal x on the graph can be filtered. This process can be formalized as $y = g_\theta(L)x = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})X$, where $T_k(\tilde{L})$ represents the scaled Chebyshev expansion polynomial of scaled graph Laplacian, and θ represents Chebyshev polynomial coefficients. The definition of scaled graph Laplacian is $\tilde{L} = \frac{2L}{\lambda_{max} - I_n}$, where λ_{max} is the maximum eigenvalue of the graph Laplacian matrix, I_n is the identity matrix, and $L = I_n - D^{-1}AD^{-1}$ is a normalized Laplacian matrix, where A is an adjacency matrix and D is a diagonalized degree matrix, that is, $D_{ii} = \sum_j A_{ij}$. The Chebyshev expansion polynomial is defined recursively as $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, where $T_0 = 1$ and $T_1 = x$. The convolution kernel of this method is K-localized, and the computational complexity of the entire convolution operation is $O(K|E|)$.

Kipf *et al.* pointed out that the result of convolution of high-order neighboring nodes can be achieved by stacking multiple first-order graph convolutional layers [24]. It also mentioned that because the parameters of the neural network can be trained, the convolution operation can be simplified to $g_\theta * x \approx \theta(I_n + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})x$ by assuming $\lambda_{max} \approx 2$ and reducing some parameters. At the same time, in order to alleviate the problems of instability, gradient explosion and gradient vanishing caused by this assumption and simplified operations in deep networks, the article introduces a renormalization process, $I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, where $\tilde{A} = A + I_N$, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. Finally, convolution of

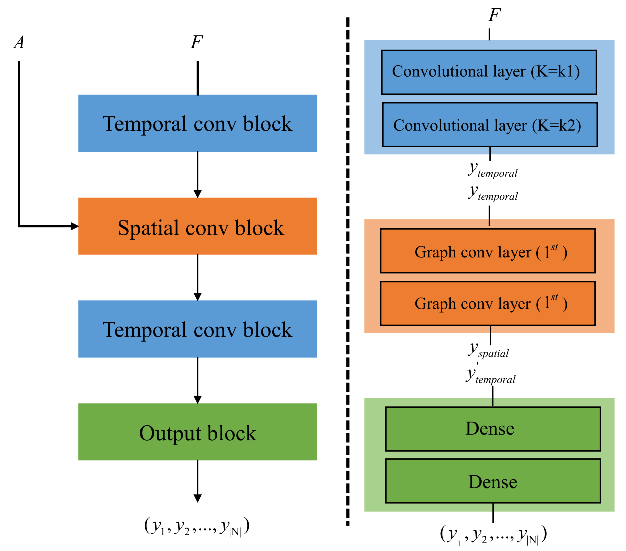


Fig. 6. Spatial-temporal graph convolutional detecting model.

signal x on the graph can be expressed as $g_\theta * x \approx \theta(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}})x$.

In our model, the spatial convolution module consists of two first-order graph convolutional layers, that is, our model will convolve the second-order neighboring nodes of each node in the graph. This process can be formalized as

$$y = ReLU(\Phi_{G2} *_G \times ReLU(\Phi_{G1} *_G \times N_t)), \tag{14}$$

where $*_G$ represents the graph convolution, Φ_{G1} and Φ_{G2} are the parameters of the graph convolution kernel, and $N_t = [u_{(v_1,t)}, u_{(v_2,t)}, \ldots, u_{(v_{|V|},t)}]$ is the input of graph convolution which contains the feature vectors of all switches at time slice t, y is the output of graph convolution, and $ReLU$ is the activation function we use.

5.1.3 Spatial-Temporal Graph Convolutional Detecting Model

The DDoS detection model extracts both temporal and spatial features of data to determine whether the network is under a DDoS attack currently, and which switches contain attack flows.

Yu *et al.* and Guo *et al.* use a spatial-temporal graph convolutional network to extract both temporal and spatial features of data and predict the traffic on road [26], [27]. In fact, road traffic is similar to network traffic. Intersections in a road network are like switches in a network, and roads connecting intersections are like links between switches. Inspired by these works, we construct an attack detection model with CNN and graph convolution network to extract temporal and spatial features of data, respectively.

Our model consists of two temporal convolution modules, a spatial convolution module and an output module as shown in the left of Fig. 6. Specifically, the temporal convolution module and the spatial convolution module contain two convolution layers and two graph convolution layers respectively, and the output module contains two dense layers, as shown in the right of Fig. 6. Obviously, the temporal convolutional module and the spatial convolution module are used to extract temporal and spatial features of data respectively. It is worth mentioning that we add another time convolution module after the spatial convolution module so that each node can not only get its own information at different time slices, but also get information of neighboring nodes at different time slices. This improves the learning ability and stability of our model. The output module gives the final classification result of switches.

5.2 Mitigating DDoS Attacks

As mentioned earlier, we could be aware of which switch in the network contains attack flows using ST-GCN, i.e., we can find the path that attack flows pass through in the network and locate the switches through which attack flows enter the network. The path can be formulated as

$$p_{i,j} = \sum (v_i, r_i) \rightarrow (v_j, r_j),\ where\ v_i, v_j \in V_{attack}, \tag{15}$$

where V_{attack} is the set of switches through which DDoS flows pass, and $p_{i,j}$ is the path that attack flows pass through. When there are DDoS flows pass through both switch v_i and switch v_j and the forwarding rules are met, we can say that edge (v_i, v_j) is a hop in the attack path. For mitigating DDoS attacks, there are two principles. One is to shield as many attack flows as possible, and the other is to try not to kill normal flows by mistake. By finding the attack path $p_{i,j}$, we can adopt a more targeted strategy of dropping packets to mitigate the DDoS attack.

In order to reduce the chance of killing normal packets by mistake when dropping packets, we setup and deploy a whitelist on all switches based on prior knowledge. Specifically, the IP addresses of commonly used business hosts will be added to the whitelist. Then, the whitelist will be distributed to all switches by controller. If both the source and destination IP addresses of the packet match the IP addresses in the whitelist, the packet will be forwarded by switches, otherwise, the packet will be dropped based on the rule described later. The matching process can be formalized as

$$(P_{src} \in Q) \bigcap (P_{dst} \in Q), \tag{16}$$

where P_{src} is the source IP address of a packet, P_{dst} is the destination IP address, Q is the set of IP addresses in the whitelist.

Most attackers will forge IP addresses when launching DDoS attacks to hide themselves. These spoofing IP addresses cannot match the IP addresses in the whitelist. However, if an attacker compromises a commonly used business host and uses its real IP address, the attack flow will be protected by a whitelist. To avoid this situation, we add additional restrictions. Considering that DDoS attack flows are usually one-way, we stipulate that if the source and destination IP addresses of a flow are in the whitelist and the flow is bidirectional, then this flow can be protected by the whitelist. Otherwise, it will not be protected by the whitelist.

As mentioned above, most legitimate traffic is protected by the whitelist. Then, discarding all the packets of remaining attack traffic is the simplest and most effective method

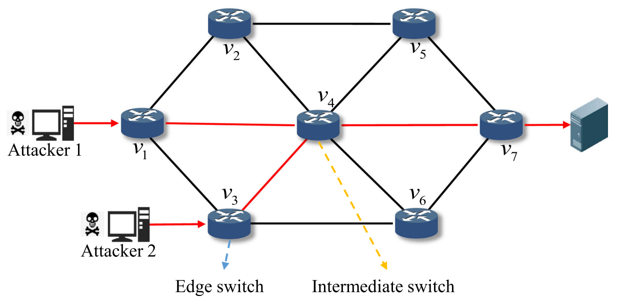


Fig. 7. Mitigation of DDoS attacks.

to mitigate DDoS attacks [39]. However, legitimate traffic that contains newly arrived or infrequently used IP addresses cannot be protected by the whitelist. If all packets that cannot match IP addresses in the whitelist are discarded, the above-mentioned legitimate businesses will also be completely paralyzed. In order to reduce negative impact on these businesses, we adopt a specific strategy of dropping packets.

According to the characteristics of network traffic, we make the following assumption, i.e., the closer the hop is to the attack source on an attack path, the greater the proportion of DDoS traffic in the link traffic. It can be formulated as

$$U(h_{(v_i,v_{i+1})}) < U(h_{(v_j,v_{j+1})}), \quad where \ \ i < j. \tag{17}$$

Suppose that two attack paths (v_1, v_4, v_7) and (v_3, v_4, v_7) are detected in the network, and they overlap on link (v_4, v_7) as shown in Fig. 7. For (v_1, v_4) and (v_3, v_4), the proportions of attack traffic are $U(h_{(v_1,v_4)})$ and $U(h_{(v_3,v_4)})$, respectively, then the proportions of attack traffic on link (v_4, v_7) can be denoted as $U(h_{(v_4,v_7)}) = (B^d_{(v_1,v_4)} + B^d_{(v_3,v_4)} + B^d_{others})/(B^t_{(v_1,v_4)} + B^t_{(v_3,v_4)} + B^t_{others})$, it should be pointed out that (v_4, v_7) may also contain traffic from other links besides (v_1, v_4) and (v_3, v_4), such as (v_2, v_4). Since these links do not contain attack traffic, $B^d_{others} = 0$ and $B^t_{others} \geq 0$. And $B^t_{others} = 0$ if (v_4, v_7) does not contain traffic from other links besides (v_1, v_4) and (v_3, v_4). Therefore, $U(h_{(v_4,v_7)})/U(h_{(v_1,v_4)}) \leq 1$ and $U(h_{(v_4,v_7)})/U(h_{(v_1,v_4)}) \leq 1$. Obviously, for an attack path $p_{s,d} = (v_s, v_{s+1}, \ldots, v_d)$, $where\, N_{v_i,d} = v_{i+1}$, $U(h_{(v_{i-1},v_i)}) \geq U(h_{(v_{i+1},v_{i+2})})$ holds for every link on the path, that is, the previous assumption holds.

As show in Fig. 7, the attack traffic on links (v_1, v_4) and (v_3, v_4) is much larger than normal business traffic because of the DDoS attack, i.e., $U(h_{(v_1,v_4)}) >> 1 - U(h_{(v_1,v_4)})$ and $U(h_{(v_3,v_4)}) >> 1 - U(h_{(v_3,v_4)})$. Generally, the switches through which attack flows enter the network contain the most attack flows, so the dropping rate of them should be larger to mitigate the attack. For convenience, we will call them edge switch S_e later on. While other switches in $p_{i,j}$, which are called intermediate switches S_m, should keep smaller dropping rates to reduce the impact on normal requests. The key issue is to find the attack path $p_{i,j}$ and the edge switches S_e, and assign proper dropping rates of different switches in that path. The dropping rate can be determined by the change of some indicators of the switch. As for switches that only contain normal flows, there is no need to drop packets.

Taking the characteristics of DDoS attacks into account, we comprehensively consider the entropy H of source IP addresses of packets and the change of the number of packets N passing through an edge switch per unit time to determine the dropping rate r_{edge} of the edge switches. This can be formalized as

$$r_{edge} = k(\Delta H, \Delta N), \tag{18}$$

where ΔH is the difference between the entropy of source IP addresses of packets per unit time, and ΔN is the difference in the number of packets passing through the switch per unit time and $r_{edge} \in (0, 1)$. When there is a DDoS attack and the values of H and N collected in some switches increase rapidly, then the dropping rate r_{edge} of that switch is expected to increase rapidly to mitigate the attack. However, an excessively high dropping rate is bounded to affect normal requests, so the dropping rate should have an upper limit that significantly smaller than 1. Therefore, the dropping rate r_{edge} is determined by

$$k(w) = \frac{1}{(1 + e^w)} - (1 - l), \tag{19}$$

where w is the weighted sum of ΔH and ΔN and $w > 0$. When a DDoS attack happens, l is the upper limit of dropping rate, thus, $r_{edge} \in [l - 0.5, l)$.

Intermediate switches S_m may be connected to other switches which only contain normal flows, so they may contain fewer attack flows and r_{mid} of them should be smaller than edge switches. When r_{edge} of edge switches is significantly smaller than the upper limit l, the attack is not so heavy and DDoS attacks can be mitigated by edge switches. In order not to affect normal requests, the intermediate switches should not drop packets. Otherwise, intermediate switches need to cooperate with edge switches to mitigate the DDoS attack with $r_{mid} = m \cdot r_{edge}$, where $m \in (0, 1)$ and m is a coefficient of which the value depends on the distance between the intermediate switch and edge switch and the distance is the shortest hops from the intermediate switch to edge switch in the attack path. The farther away from edge switch, the smaller the dropping rate performed by the intermediate switch. Therefore, we get

$$r_{mid} = \begin{cases} 0, & l - r_{edge} > \varepsilon \\ m \cdot r_{edge}, & l - r_{edge} \leq \varepsilon \end{cases}, \tag{20}$$

where ε is a small constant that measures the difference between r_{edge} and l.

The value of m can be obtained by

$$m = \frac{N_{hops} - n_{hops}}{N_{hops}}, \tag{21}$$

where N_{hops} is the distance from edge switch to destination switch in the attack path, and n_{hops} is the distance from edge switch to the corresponding intermediate switch in the attack path.

In the mitigation stage, we first use a whitelist Q to protect normal packets. Then, we find out the attack path $p_{i,j}$ according to the results of the detection model, and make switches in $p_{i,j}$ adopt targeted packet dropping strategies. Through these methods, we mitigate DDoS attacks without

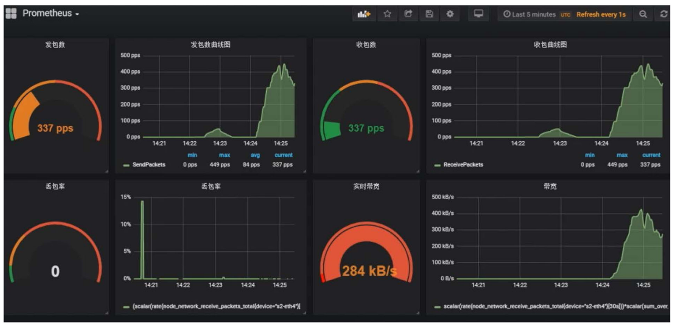


Fig. 8. INT execution diagram.

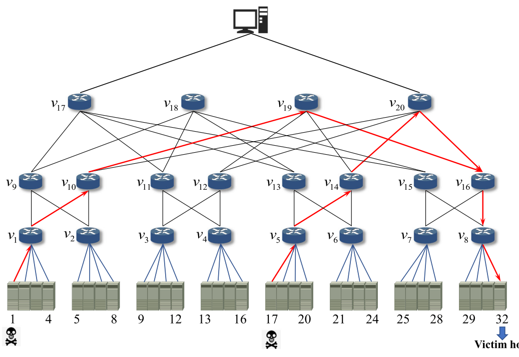


Fig. 9. Topology of experimental network.

affecting normal services. The processes are described in Algorithm 2.

Algorithm 2. Mitigation of DDoS Attacks

```
Require: Packets sent from source $D_{sent}$
Ensure: Proportion a packet being dropped $r_{drop}$
1:  Initiate $Q$, $S_e$, $V_{attack}$
2:  if $P_{src} \in Q$ and $P_{dst} \in Q$ then
3:    $r_{drop} = 0$
4:  else
5:    if $ID_{switch} \in S_e$ then
6:      $r_{drop} = \frac{1}{1+e^w} - (1-l)$
7:    else if $ID_{switch} \in V_{attack}$ then
8:      $m = \frac{N_{hops} - n_{hops}}{N_{hops}}$
9:      $r_{drop} = m \cdot [\frac{1}{1+e^w} - (1-l)]$
10:   else
11:     $r_{drop} = 0$
12:   end if
13: end if
14: return $r_{drop}$
```

6 Implementation and Evaluation

6.1 Implementation of System

We build an experimental network with 20 programmable switches and some hosts. It is noted that one of the hosts is used as an SDN controller. INT module is implemented using *P4* language on programmable switches. The *INT_header* will be added to some packets at source switch, and the corresponding metadata will be written to the packets when being forwarded. Fig. 8 is a screenshot showing the execution of INT. The structure of packet header and actions performed on the packet by switches are defined in file *INT_header.P4*. We also count the number of packets received in a period of time to help implementing adaptive sampling and enhanced whitelist. When a packet arrives at the sink switch, the payload of it will be forwarded to the host, and the INT report which contains all metadata written by switches will be parsed by analysis module at the controller. The controller is developed based on ONOS and the analysis module is developed by Python, which can extract the network states and timestamps collected by packets as they pass through each switch.

The module of data conversion and the graph convolutional network detection module are developed by Python and resided on the controller. The module of data conversion turns the set of switch states into a three-dimensional tensor and passes it to the detection module. Then, the detection module outputs the classification result of all nodes. Finally, the dropping rate of each switch is determined by the controller. The enhanced whitelist mechanism is implemented by *P4* language and issued to all switches by controller.

6.2 Experimental Setup and Parameters Setting

We use CAIDA [40] traffic traces as background traffic and construct DDoS attacks with different attack path by hping3. We deploy the detecting and mitigating system on the experimental network. We use a server configured as an 8-core 3.70GHz Intel Xeon (R) E3-1240 v6 CPU, 32GB RAM as the controller. Considering that data center is one of the most common application scenarios of SDN networks, and the fat-tree structure is widely used in data center networks, we choose the fat-tree structure shown in Fig. 9 as the experimental topology. The topology contains 20 nodes and 32 edges. There are 8 switches directly connected to the hosts and they are edge switches. The upper layers of switches in Fig. 9 are the aggregation layer and the core layer, respectively. The actual network equipment layout corresponds to the fat-tree structure mentioned above, as shown in the right of Fig. 10. The left side of Fig. 10 is the picture of a machine cabinet in our computer room, including the controller and three programmable switches, and the top is the operation interface of the controller.

We send attack traffic at different rates from different hosts to construct different attack paths. The maximum bandwidth that our experimental network can withstand is 10Gb/s. When training the detection model, a complete attack process consists of multiple attack phases and intermittent phases. The path of the attack flow is different in different attack phases, and only background traffic exists in the intermittent phases. When testing, we launch DDoS attack from 2 different hosts to attack the same victim successively as shown in Fig. 9, where host 1 and host 17 are bots, and host 32 is the victim. The attack paths are $(h_1, v_1, v_{10}, v_{19}, v_{16}, v_8, h_{32})$ and $(h_{17}, v_5, v_{14}, v_{20}, v_{16}, v_8, h_{32})$.

As mentioned before, in practice, our detection model contains two temporal convolution layers, a spatial convolution layer and a output layer. We use cross entropy as loss, and use back propagation to minimize the loss. When training our detection model, the learning rate is 0.001 and the training iterates 200 times. Also, we use Adam algorithm to optimize the training process. Our method is supervised learning so that we input the three-dimensional tensor

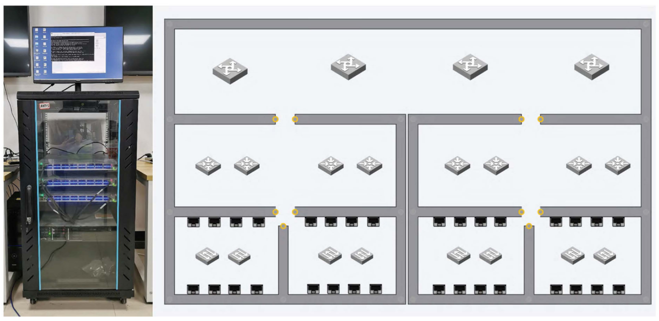


Fig. 10. Diagram of network equipment layout.

with the labels of each switch into the model when training detection model.

6.3 Evaluation Metrics

We use different metrics to evaluate the system:

1) *Accuracy*. We use accuracy to evaluate the detection model which can be calculated by $accuracy = \frac{TP+TN}{TP+FP+TN+FN}$, where TP represents true positive, i.e., the number of switches been judged that contain attack traffic and actually contain attack traffic, FP means false positive, i.e., the number of switches been judged that contain attack traffic but actually contain no attack traffic, TN means true negative, i.e., the number of switches been judged that do not contain attack traffic and actually contain no attack traffic, FN means false negetive i.e., the number of switches been judged that do not contain attack traffic but actually contain attack traffic.

2) *False Positive Rate (FPR)*. We also use FPR to evaluate the performance of detection model besides accuracy, which can be calculated as $FPR = \frac{FP}{FP+TN}$.

3) *Ratio of received normal packets*. We use ratio of received normal packets to evaluate the mitigation strategy. The ratio of received normal packets can be calculated as $r_{normal} = \frac{number\ of\ received\ normal\ packets}{number\ of\ normal\ packets\ been\ sent}$.

4) *CPU utilization of controller*. We use CPU utilization of the controller and a switch to evaluate the overhead of detection.

5) *Latency*. We also use latency of network state awareness to evaluate the overhead of detection.

6) *Southbound interface communication load*. We use southbound interface communication load to evaluate the overhead of detection and mitigation.

6.4 Performance Evaluation

6.4.1 Accuracy of DDoS Attacks Detection

As mentioned earlier, we extract and identify the temporal and spatial features of the switch state, and use the results to detect whether each switch in the network contains attack flows. Specifically, the detection model outputs the classification results of all switches. Compared with previous methods, our model can use both temporal and spatial features to detect DDoS attack flows. Therefore, our model can fully utilize the information hidden in data. In order to verify the ability to detect DDoS attack flows of our model, we use the same data to train and test the proposed ST-GCN and other models such as SVM, CNN and GCN, and take the average of their results for comparison. The accuracy and FPR of these methods are shown in Figs. 11 and 12, respectively.

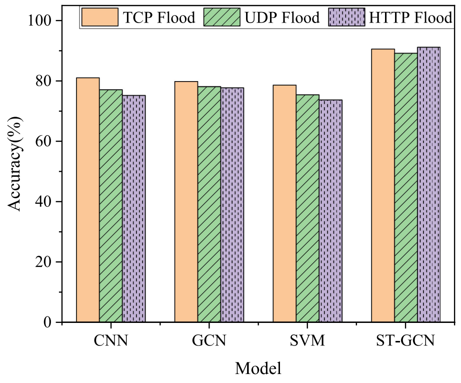


Fig. 11. Accuracy of detection.

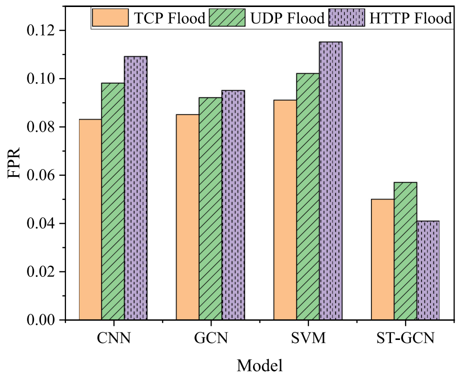


Fig. 12. FPR of detection.

Obviously, our model outperforms other methods. The accuracy of our model is the highest, and the detection accuracy for three attacks of TCP Flood, UDP Flood and HTTP Flood are 90.5%, 89.1% and 91.1% respectively. The detection accuracy of SVM is always the lowest, 78.6%, 75.4% and 73.7%, respectively. The FPR of our model is the lowest. For these three attacks, the FPRs of our model are 0.050, 0.057 and 0.041 respectively. The FPRs of the other methods are significantly higher than our model. The FPR of the other methods for TCP Flood attack is around 0.085, for UDP Flood attack is around 0.096, and for HTTP Flood attack is the highest, above 0.095. GCN is a generalization of CNN on graphs. Therefore, applying CNN alone on graph can only extract temporal features of data, and only applying GCN can only extract the spatial features of data. The expressive ability of SVM is limited and cannot learn the spatial features of data. Our detection model can extract and use both temporal and spatial features of data, and the learning and expression capabilities are stronger.

It should be noted that testing contains three parts and each part lasts for 1 minute. The first and third parts of data come from two DDoS attacks with different paths as shown in Fig. 9, and the second part of data contains background traffic only. We found during the test that the detection accuracy of our model is basically stable at 90% in all three phases, and if only GCN or CNN are used, the detection accuracy is significantly reduced when an attack occurs, in other words, only using GCN or CNN to learn the features of attack flows is poor, as shown in Fig. 13. In conclusion, it

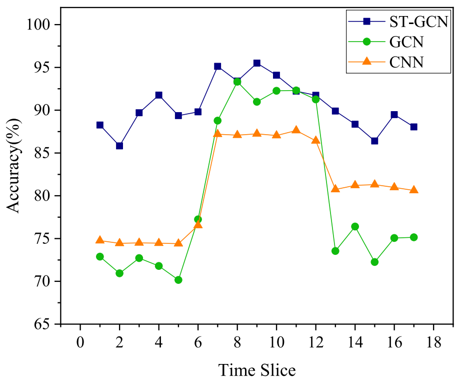


Fig. 13. Accuracy in different stage.

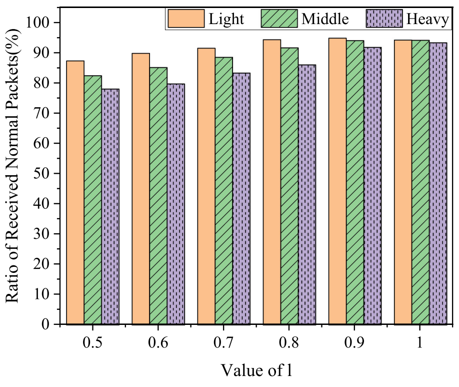


Fig. 14. Effect of different values of l.

is not enough to extract features from only temporal or spatial perspective and determine whether there is a DDoS attack, while processing data from both temporal and spatial perspectives can greatly improve the learning ability of detection model.

6.4.2 Effectiveness of DDoS Attacks Mitigation

We measure the effectiveness of mitigation by the ratio of normal packets received after we started the mitigation module. The attack and mitigtion process is carried out three times. The sender always sends legal packets at a constant rate, but launches attack with three different intensities at 100Mbps, 200Mbps and 300Mbps, corresponding to three levels of light, medium, and strong respectively. We compared the effect of different values of parameter l, which controls the range of dropping rate of edge switches, as shown in Fig. 14. As mentioned earlier, $r \in [l - 0.5, l)$ and $l \in [0.5, 1)$, otherwise, the value of dropping rate may be meaningless. In the experiment, we select six values of l from 0.5 to 1 for comparison.

As shown in Fig. 14, for light and medium attacks, the effect of mitigation is the best when $l = 0.9$, while under heavy attack the effect of mitigation is the best when $l = 1$. In addition, the stronger the attack, the more obvious the effect of the value of l on mitigation. This is because l determines the lower and upper boundaries of dropping rate. The dropping rate is calculated based on the change of the network state. When the network state changes slowly, or when the attack is light, the dropping rate determined by our method generally does not reach the upper limit. However, when under strong attack, the dropping rate may not reach the ideal value if the value of l is too small. Also, because we deploy a whitelist mechanism, the proportion of illegal packets in packets that cannot match the whitelist is high. If the value of l is setted too small, attacks will not be effectively defended.

We set $l = 0.9$ and compare the changes in attack traffic and legitimate traffic at sender and receiver respectively. The results are shown in Fig. 17. The process lasts for 60s. The sender sends legal packets at the rate of 30Mbps constantly. The attack traffic is sent at the rate of 300Mbps from 10th second. The attack lasts for 40s. Fig. 17a shows the speed of legitimate traffic and attack traffic sent by the sender. Fig. 17b shows the the speed of legitimate traffic and attack traffic received. It can be seen that after the mitigation is started, the received attack traffic decreases rapidly, and normal traffic recovers quickly, but it cannot reach the state when there is no attack. On the one hand, the edge switches that attack flows enter the network get a high dropping rate to filter out most of the attack traffic. On the other hand, the whitelist mechanism protects the vast majority of legitimate traffic. However, due to detection errors and the possibility of damaging some new normal packets during defense, legitimate traffic cannot be fully received after defense is turned on, but the ratio of received normal packets is close to the ideal level.

6.4.3 Overhead of Detection

In order to evaluate the performance of our method in terms of load on southbound communication interface, network state awareness latency and CPU consumption of controller, we have compared the above items of our method with SVM-based detection method [20] and entropy-based detection method [13].

It's common that using southbound interface to send packet headers to the control plane, then analyze the headers and classify network traffic on the control plane. However, the southbound interface should be used generally for communication and instruction transmission between data plane and control plane. Excessive southbound interface load will affect normal communication between the control plane and the data plane. In programmable switches, we use INT with sampling to realize network state awareness. Adaptive sampling reduces overhead while ensuring the effectiveness of features. Fig. 15 shows the changes of link load of three methods in the whole attack cycle. When the attack occurrs, the load on the southbound interface of the INT-based method changes slightly, and remains below 40 Mbps, while the load on the southbound interface of the other two methods rises rapidly to 40 Mbps. The results show that the cost of our method is much lower than that of other methods which is a benefit of using INT.

The sensing delay of INT is smaller compared with outband telemetry. And we choose two other methods based on OpenFlow protocol, which were compared under three different types of attacks as shown in Fig. 16, the network state awareness delay of control group, i.e., SVM-based detection method and entropy-based detection method, is more than 800ms, while our method delay is around 750ms, which benefits from the INT with sampling and the extremely high forwarding efficiency of programmable switches.

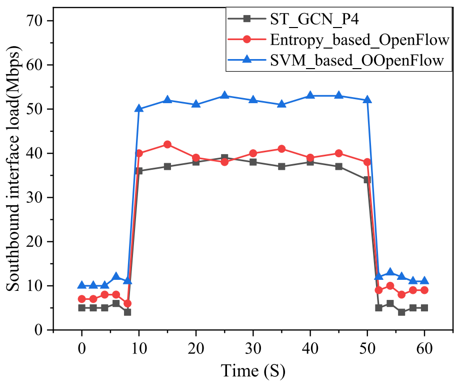


Fig. 15. Southbound interface communication load.

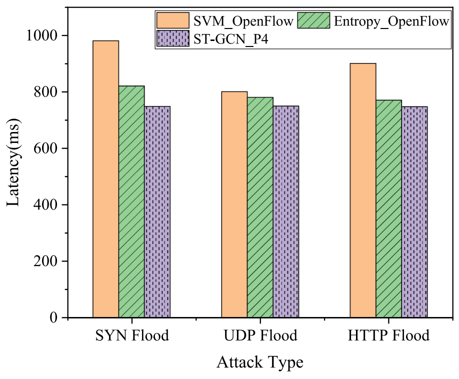


Fig. 16. Latency of network state awareness.

The CPU resource of controller is precious in SDN network because the controller is in charge of the management and configuration of all switches in the network. Therefore, we hope that DDoS detection module does not consume the CPU resource of the controller excessively. As shown in Fig. 19, We compared the changes in the controller CPU utilization when performing DDOS attack detection under the two modes of INT without sampling as shown in Fig. 19a and INT with adaptive sampling as shown in Fig. 19b. When the network is attacked by three DDoS attacks respectively, the CPU utilization of the controller with adaptive sampling is much lower than that without sampling. For TCP Flood, the CPU utilization without sampling approaches 90% at 15s and 55s. For UDP Flood, it also reaches 80% at the above two time nodes. While the CPU utilization under three attacks is less than 60% with adaptive sampling. It can be seen that the INT with sampling effectively reduces the burden on the controller CPU under various types of attacks.

In addition, we compared the CPU utilization of programmable switches when performing INT with/without sampling, as shown in Fig. 18. Obviously, the CPU utilization of the switch when sampling is performed is significantly lower than that of when sampling is not performed, especially when the switch is under DDoS attacks.

6.4.4 *Overhead of Mitigation*

Our defense methods are different from the previous methods as both the control plane and the data plane are programmable thanks to P4 language and programmable switches. When DDoS traffic is detected, most of the previous defense methods focus on traffic cleaning of the last

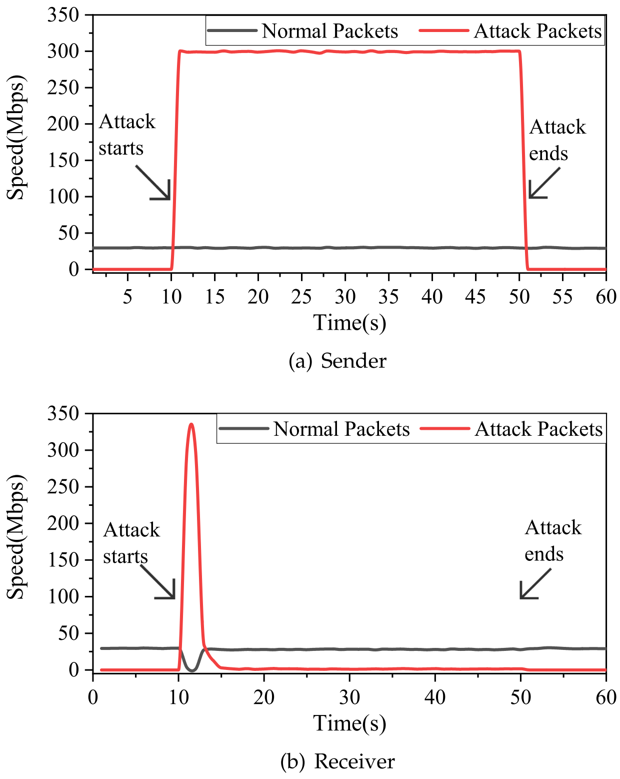
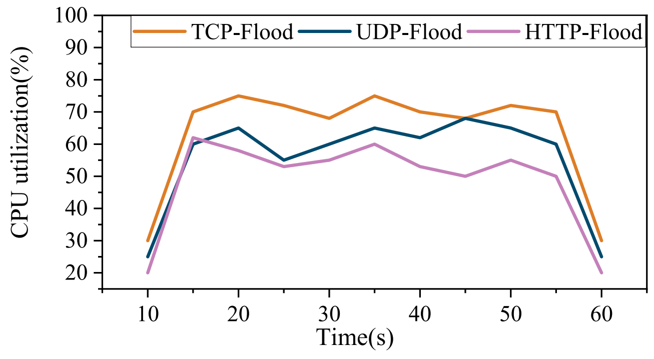


Fig. 17. Speed of legitimate traffic and attack traffic.

switch on the attack path, while our method tries to find the attack path based on the results of the detection model, and discards packets with different dropping rates on different switches on the attack path to mitigate attacks. In other words, our defense strategy is distributed cooperative and it is executed simultaneously on multiple switches, so the cost of defense is distributed. In the experiment, there are totally 5 switches on the attack path in the first phase of testing which are v_1, v_{10}, v_{19}, v_{16} and v_8. The distributed defense strategy is performed on all five switches, while the centralized defense strategy is only performed on switch v_8.

As shown in Tables 2 and 3, we compare the southbound traffic of each switch under the distributed defense strategy and the southbound traffic of the last switch under the centralized defense strategy during the defense process to evaluate switch overhead under different mitigating methods. It can be seen from Tables 2 and 3 that the total cost of all switches under the distributed defense strategy is greater than that of the last switch under the centralized defense strategy, but the cost of each switch under distributed defense strategy is much lower. Since we want to keep the cost of the southbound interface in a low range to ensure the normal communication between the controller and the switch, it is clear that the distributed defense strategy is better.

As shown in Fig. 20, we compared the changes in the received packets ratio of each switch in an attack cycle under distributed defense and centralized defense. By analyzing the changes in the curve, we found that the time spent on performing distributed defense is significantly shorter than that of centralized defense. In other words, the delay caused by the distributed defense strategy is much smaller than the centralized defense strategy. Our defense


(a) Switch utilization of network state awareness without sampling

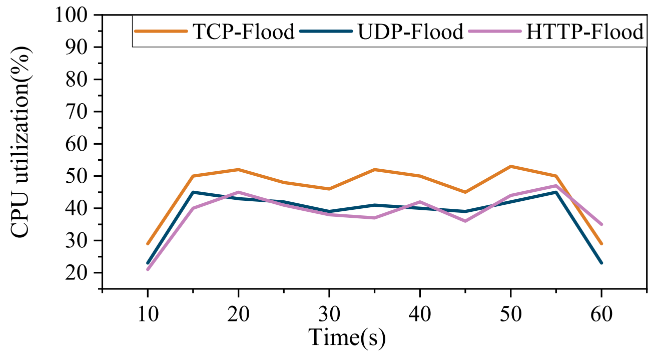
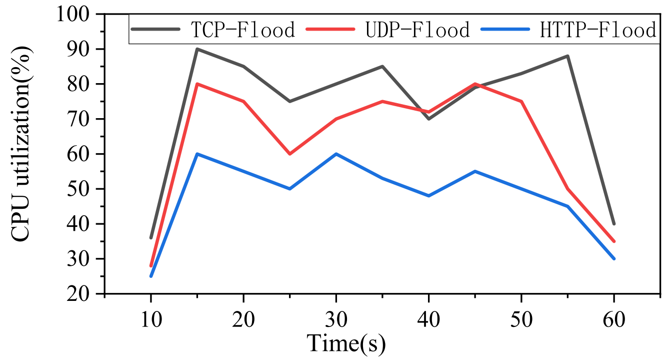

(b) Switch utilization of network state awareness with adaptive sampling

Fig. 18. Switch CPU utilization of network state awareness.


(a) CPU utilization of network state awareness without sampling

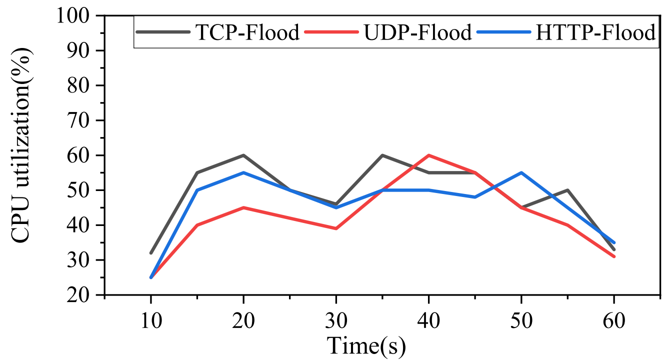

(b) CPU utilization of network state awareness with adaptive sampling

Fig. 19. Controller CPU utilization of network state awareness.

TABLE 2
Southbound Traffic of Each Switch in Distributed Defense

Switch ID	SYN flood	UDP flood	HTTP flood
Switch v_1	10.5 Mbps	9.8 Mbps	12.5 Mbps
Switch v_{10}	8.5 Mbps	8.3 Mbps	10.4 Mbps
Switch v_{19}	5.2 Mbps	5.0 Mbps	7.3 Mbps
Switch v_{16}	4.5 Mbps	4.3 Mbps	6.2 Mbps
Switch v_8	4.3 Mbps	4.0 Mbps	6.0 Mbps
Total	33.0 Mbps	31.4 Mbps	42.4 Mbps

TABLE 3
Southbound Traffic of the Last Switch in Centralized Defense

Switch ID	SYN flood	UDP flood	HTTP flood
Switch v_8	28.7 Mbps	25.6 Mbps	40.7 Mbps

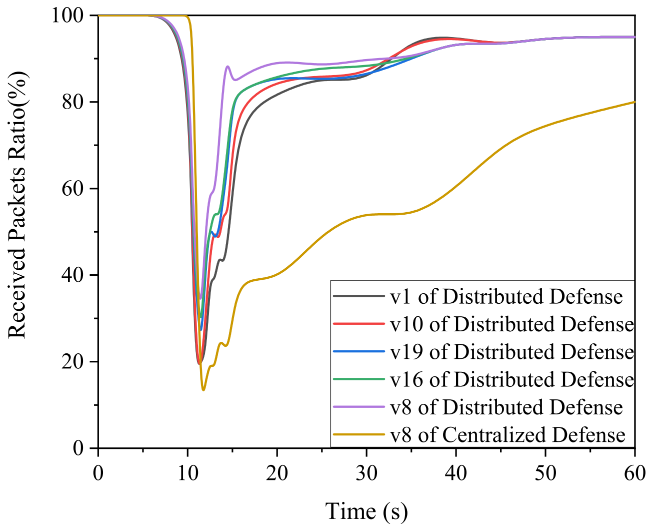


Fig. 20. Received packets ratio under different defense strategies.

delay has roughly dropped from 60s to 30s. The efficiency of defense has roughly doubled.

7 Discussion

Dynamic Attack With Path Change. In the scenario of data center, the number of switches is large and the connection between switches are complicated which means the paths in the network are diverse. An attacker may change the edge switch where attack traffic enters to change the attack path, which may lead to the decrease of accuracy of detection model. In order to solve this problem, the training data set of detection model should be updated continuously, and the detection model should be retrained periodically with new training data to enhance the detection capability of it. Furthermore, the model and its training method can be optimized to improve its generalization ability.

DDoS Attacks With Unspoofed Addresses. Not all DDoS attacks spoof the source IP addresses. However, DDoS attacks with unspoofed addresses will also cause changes in network state, so our method is equally effective. As for mitigation, DDoS attacks with unspoofed addresses will bypass ordinary whitelist, however, we also detect bidirectional flows to activate the whitelist which avoid being bypassed by attackers as mentioned in Section 5.2.

DDoS Attacks With Multiple Targets. In the previous discussion, we mainly considered the situation of a single victim host. However, if there are multiple targets in a DDoS attack, or multiple DDoS attacks overlap in time, we need to make some small adjustments to the defense strategy mentioned earlier. The key to the problem is how to determine the dropping rate of the overlapped switches if the attack paths overlap, because the dropping rate partly depends on the distance from the switch to the victim. Since most of the legitimate traffic has been protected by the whitelist, the dropping rates of these switches can be calculated separately for each target, and finally perform the highest dropping rate only.

Impact of DDoS Attacks on Controller and Southbound Interface. The DDoS attack traffic contains a large number of packets with forged IP addresses which cannot match the flow table entries in the switch. Therefore, the switch will continuously send $packet\_in$ messages to the controller to determine how to deal with these packets. Too many $packet\_in$ messages caused by packets with forged IP addresses will block the southbound interface between the switches and the controller, and these messages will also consume a lot of resources of the controller such as CPU and memory. Our mitigation method drops packets, which cannot match the flow table entries, in proportion, and the dropping rate is determined by the intensity of the detected DDoS attack and the location of the switches. Since most of the packets which are not protected by the whitelist are dropped, the number of $packet\_in$ messages that the switches have to send is greatly reduced. Therefore, the burden on the southbound interface and the controller is also reduced.

8 Related Works

Due to the flexibility, scalability, and centralized control of SDN controllers, most DDoS attack detection in SDN are performed on the controller [36], [41], [42]. A classic method is to calculate the entropy of some indicators (e.g., source/destination IP addresses of packets) on the controller to evaluate the dispersion of network traffic [36], because DDoS attackers usually forge source IP addresses which causes the source IP addresses of attack flows to be highly diverse so that we can use the change of entropy to detect attack flows. The accuracy of this type of detection system depends on the threshold of entropy. However, selection of threshold mainly depends on expert knowledge which is certainly a drawback. Another type of methods uses the scalable computing power of the controller to extract key features of network traffic from the packet header, and uses probability-based statistical methods or machine learning algorithms to detect DDoS attacks [43], [44], [45].

A SVM-based network intrusion detection system [45] can accurately determine whether a DDoS attack has occurred, but there are problems such as long detection time and low detection efficiency. Niyaz [37] proposed a multi-vector DDoS detection system based on deep learning, which implemented SAE-based detection on the SDN controller, and used deep learning to reduce and classify a large number of features derived from the packet header.

Conti M. *et al.* [46] used Moving Target Defense (MTD) technology to defend against DoS attacks. Specifically, they used SDN controller to regularly change the IP addresses of hosts in the network, and simultaneously used real IP addresses and virtual IP addresses to reduce the burden of the controller. In addition, they also used change point detection technology to detect and mitigate DoS attacks. However, they only considered the changes of packet arrival rate, which may accidentally hurt flash crowd traffic, and this method is not suitable for DDoS attacks. In another paper [47], Conti M. *et al.* also used the change point detection technology to detect DDoS attacks. This method also only considers the changes of packet arrival rate, so it cannot distinguish between DDoS attacks and flash crowd traffic.

The aforementioned DDoS detection method only detect the occurrence of DDoS attacks in the network, but cannot find the attack path of DDoS flows and accurately locate the entrance of attack flows [48]. Therefore, the defense strategy based on these methods are relatively crude [49]. When the detection result indicates that a DDoS attack occurs, large-scale traffic cleaning is performed, which will kill a large amount of normal business flows at the same time [50]. With the development of programmable data plane and P4, there are many DDoS attack detection or anomaly detection methods based on programmable switches [51], [52], [53], [54]. These methods use the programmability of switches to deploy detection algorithms directly on data plane, and their overhead and detection delay are much lower than detection on the controller. In addition, these detection methods will not cause excessive southbound communication load. However, these detection methods, which are hardware-based, usually rely on threshold, so their detection accuracy is relatively low and require further fine-grained detection on the controller.

Some researchers have also proposed DDoS source tracing methods and path backtracking algorithms [31], [55], such as calculating the entropy of source IP addresses of packets in each switch port per unit time to find out the port through which DDoS attack traffic passes, and backtrack hop by hop to find the attack path or using the characteristics of some field in the header to locate the source of the attack. These methods are able to provide accurate attack path which is the basis of subsequent defense strategies, and the rate of manslaughter is greatly reduced. However, the accuracy of these detection methods is low, and the detection delay is large which means it cannot defend DDoS attacks in time [56]. Ling *et al.* [57] proposed a method of anomaly detection and path backtracking under SDN, which changes the traffic rate of switches on the attack path by adjusting the size of TCP window to defense DDoS attacks. However, the path backtracking method is based on IP address, so the path of an IP spoofing attack may not be detected.

Compared with existing methods, on the one hand, our detection method uses multiple statistical characteristics of network traffic (i.e., switch state) which not only avoids excessive dependence on single characteristic of network traffic (e.g., packet arrival rate, entropy), but also adapts to different types of flooding-based DDoS attacks. On the other hand, our detection method uses GCN to classify the switches and finds the attack path, which is the basis of subsequent accurate distributed mitigation of DDoS attacks.

GCN brings new possibilities to the processing of non-euclidean structured data. GCN was quickly used to solve various problems like network modeling [58], recommendation systems [25] and even security of e-commerce [30]. Furthermore, many researchers combine graph neural networks with other neural networks (e.g., CNN, LSTM) and proposed ST-GCN to extract and analyze information contained in data from both temporal and spatial perspectives. For example, some researchers use the ST-GCN to predict traffic flow [26], [27], [28], [29], Yan *et al.* regarded a person as a graph to recognize actions, human joints are nodes in the graph and connections between joints are edges [59]. GCN provides new ideas for solving many problems and researchers make progress in many fields with the help of GCN. However, as far as we know, the application of graph convolution network in attack detection is not much, although the structure of network is also non-euclidean. Especially under data plane programmable SDN, the restoration and awareness of network topology does not cost a lot. And we can use GCN to detection attacks in the network and trace the attack path easily. Therefore, our work is a meaningful and important exploration.

9 Conclusion and Future Work

In this paper, we propose a DDoS attack detection model under data plane programmable SDN based on ST-GCN. This model extracts the characteristics of data from both temporal and spatial perspectives, and then identifies the switches containing attack flows which means find the attack path. Based on the results of detection, we propose an effective method to mitigate DDoS attacks, which combines an enhanced whitelist and a precise dropping strategy. In implementation and evaluation, we verified the detection capabilities of our model and the effectiveness of the mitigate strategy. The results show that the accuracy of DDoS attacks detection of our model is high and the performance on mitigating is superb. The detection accuracy of our model has improved by nearly 10% compared with the classical methods, and the delay of defense is reduced. In addition, the overhead of our method is lower compared with existing methods.

In more general scenarios, the cost of obtaining labels required by supervised learning can be very high. In the future, we hope to optimize the detection model and its training method to improve the generalization of our model.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants U19B2004 and 61972448, the Introduce Special Funds for Innovation from High-End Scientific Research Institutions under Grant 181129112748101, and the National Defense Basic Scientific Research program of China under Grant JCKY2018207C121.

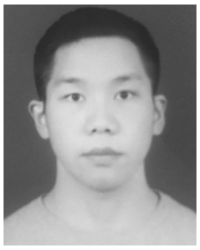
References

[1] W. Dou, Q. Chen, and J. Chen, “A confidence-based filtering method for DDoS attack defense in cloud environment,” *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1838–1850, 2013.
[2] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, “Bohatei: Flexible and elastic DDoS defense,” in *Proc. 24th USENIX Secur. Symp.*, 2015, pp. 817–832.
[3] J. Mirkovic and P. Reiher, “A taxonomy of DDoS attack and DDoS defense mechanisms,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 39–53, 2004.
[4] J. Wang, R. Wen, J. Li, F. Yan, B. Zhao, and F. Yu, “Detecting and mitigating target link-flooding attacks using SDN,” *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 6, pp. 944–956, Nov./Dec. 2019.
[5] N. Arboleda, “AWS hit by DDoS attack dragging half of web down,” 2020. Accessed: May 20, 2020. [Online]. Available: https://www.crn.com.au/news/aws-hit-by-ddos-attack-dragging-half-of-web-down-532842
[6] C. Cimpanu, “macOS systems abused in DDoS attacks,” 2020. Accessed: May 20, 2020. [Online]. Available: https://www.zdnet.com/article/macos-systems-abused-in-ddos-attacks/
[7] Q. Yan, F. 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, Firstquarter 2015.
[8] T. Sasaki, C. Pappas, T. Lee, T. Hoefler, and A. Perrig, “SDNsec: Forwarding accountability for the SDN data plane,” in *Proc. 25th Int. Conf. Comput. Commun. Netw.*, 2016, pp. 1–10.
[9] T. B. Scholl, “Methods and apparatus for distributed backbone internet DDoS mitigation via transit providers,” U.S. Patent 8 949 459, Feb. 3, 2015.
[10] G. Kirubavathi and R. Anitha, “Botnet detection via mining of traffic flow characteristics,” *Comput. Elect. Eng.*, vol. 50, pp. 91–101, 2016.
[11] J. Zhou, Z. Xu, A. M. Rush, and M. Yu, “Automating botnet detection with graph neural networks,” 2020, *arXiv:2003.06344*.
[12] X. Yang, B. Han, Z. Sun, and J. Huang, “SDN-based DDoS attack detection with cross-plane collaboration and lightweight flow monitoring,” in *Proc. IEEE Global Commun. Conf.*, 2017, pp. 1–6.
[13] 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.
[14] N. McKeown *et al.*, “OpenFlow: Enabling innovation in campus networks,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.
[15] S. Das, G. Parulkar, and N. McKeown, “Why OpenFlow/SDN can succeed where GMPLS failed,” in *Proc. Eur. Conf. Exhib. Opt. Commun.*, 2012, pp. 1–3.
[16] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, “Advanced study of SDN/OpenFlow controllers,” in *Proc. 9th Central Eastern Eur. Softw. Eng. Conf. Russia*, 2013, pp. 1–6.
[17] P. Bosshart *et al.*, “P4: Programming protocol-independent packet processors,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, 2014.
[18] C. Kim, A. Sivaraman, N. Katta, A. Bas, A. Dixit, and L. J. Wobker, “In-band network telemetry via programmable dataplanes,” in *Proc. ACM Conf. Special Interest Group Data Commun.*, 2015, pp. 1–9.
[19] N. Van Tu, J. Hyun, and J. W.-K. Hong, “Towards ONOS-based SDN monitoring using in-band network telemetry,” in *Proc. 19th Asia-Pacific Netw. Operations Manage. Symp.*, 2017, pp. 76–81.
[20] J. Ye, X. Cheng, J. Zhu, L. Feng, and L. Song, “A DDoS attack detection method based on SVM in software defined network,” *Secur. Commun. Netw.*, vol. 2018, 2018, Art. no. 9804061.
[21] Y. Cao, J. Wu, B. Zhu, H. Jiang, Y. Deng, and W. Luo, “A cross-plane cooperative DDoS detection and defense mechanism in software-defined networking,” in *Proc. Int. Conf. Smart Comput. Commun.*, 2019, pp. 231–243.
[22] P. Kazemian, G. Varghese, and N. McKeown, “Header space analysis: Static checking for networks,” in *Proc. 9th USENIX Symp. Netw. Syst. Des. Implementation*, 2012, pp. 113–126.
[23] J. Wang, Y. Wang, H. Hu, Q. Sun, H. Shi, and L. Zeng, “Towards a security-enhanced firewall application for OpenFlow networks,” in *Proc. Int. Symp. Cyberspace Safety Secur.*, 2013, pp. 92–103.

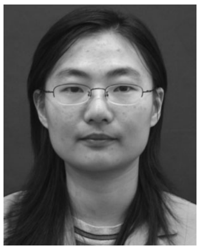
[24] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.
[25] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 974–983.
[26] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," 2017, *arXiv:1709.04875*.
[27] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 922–929.
[28] L. Bai *et al.*, "STG2Seq: Spatial-temporal graph to sequence model for multi-step passenger demand forecasting," 2019, *arXiv:1905. 10069*.
[29] C. Park *et al.*, "STGRAT: A spatio-temporal graph attention network for traffic forecasting," 2019, *arXiv:1911.13181*.
[30] A. Li, Z. Qin, R. Liu, Y. Yang, and D. Li, "Spam review detection with graph convolutional networks," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, 2019, pp. 2703–2711.
[31] A. Yaar, A. Perrig, and D. Song, "Pi: A path identification mechanism to defend against DDoS attacks," in *Proc. Symp. Secur. Privacy*, 2003, pp. 93–107.
[32] Y.-C. Wu *et al.*, "DDoS detection and traceback with decision tree and grey relational analysis," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 7, no. 2, pp. 121–136, 2011.
[33] V. K. Shaojie Bai, J. Zico Kolter, "Convolutional sequence modeling revisited," 2018.
[34] D. Suh, S. Jang, S. Han, S. Pack, and X. Wang, "Flexible sampling-based in-band network telemetry in programmable data plane," *ICT Exp.*, vol. 6, no. 1, pp. 62–65, 2020.
[35] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," in *Proc. Int. Conf. Comput. Netw. Commun.*, 2015, pp. 77–81.
[36] R. Wang, Z. Jia, and L. Ju, "An entropy-based distributed DDoS detection mechanism in software-defined networking," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, 2015, pp. 310–317.
[37] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based DDoS detection system in software-defined networking (SDN)," 2016, *arXiv:1611.07400*.
[38] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.
[39] N. Z. Bawany, J. A. Shamsi, and K. Salah, "DDoS attack detection and mitigation using SDN: Methods, practices, and solutions," *Arabian J. Sci. Eng.*, vol. 42, no. 2, pp. 425–441, 2017.
[40] The CAIDA UCSD anonymized internet traces, 2020. Accessed: Nov. 20, 2020. [Online]. Available: https://www.caida.org/data/passive/passive_dataset.xml
[41] K.-Y. Chen, A. R. Junuthula, I. K. Siddhrau, Y. Xu, and H. J. Chao, "SDNShield: Towards more comprehensive defense against DDoS attacks on SDN control plane," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2016, pp. 28–36.
[42] Y. Cui *et al.*, "SD-Anti-DDoS: Fast and efficient DDoS defense in software-defined networks," *J. Netw. Comput. Appl.*, vol. 68, pp. 65–79, 2016.
[43] R. Swami, M. Dave, and V. Ranga, "Software-defined networking-based DDoS defense mechanisms," *ACM Comput. Surv.*, vol. 52, no. 2, pp. 1–36, 2019.
[44] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Proc. Int. Conf. Wireless Netw. Mobile Commun.*, 2016, pp. 258–263.
[45] R. Kokila, S. T. Selvi, and K. Govindarajan, "DDoS detection and analysis in SDN-based environment using support vector machine classifier," in *Proc. 6th Int. Conf. Adv. Comput.*, 2014, pp. 205–210.
[46] M. Conti and A. Gangwal, "Blocking intrusions at border using software defined-internet exchange point (SD-IXP)," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw.*, 2017, pp. 1–6.
[47] M. Conti, A. Gangwal, and M. S. Gaur, "A comprehensive and effective mechanism for DDoS detection in SDN," in *Proc. IEEE 13th Int. Conf. Wireless Mobile Comput. Netw. Commun.*, 2017, pp. 1–8.
[48] S. Dong, K. Abbas, and R. Jain, "A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments," *IEEE Access*, vol. 7, pp. 80813–80828, 2019.
[49] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on SDN based network intrusion detection system using machine learning approaches," *Peer-to-Peer Netw. Appl.*, vol. 12, no. 2, pp. 493–501, 2019.
[50] 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.
[51] S. Garg, A. Singh, G. S. Aujla, S. Kaur, S. Batra, and N. Kumar, "A probabilistic data structures-based anomaly detection scheme for software-defined Internet of Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3557–3566, Jun. 2021.
[52] M. Kuka, K. Vojanec, J. Kučera, and P. Benáček, "Accelerated DDoS attacks mitigation using programmable data plane," in *Proc. ACM/IEEE Symp. Archit. Netw. Commun. Syst.*, 2019, pp. 1–3.
[53] M. Dimolianis, A. Pavlidis, and V. Maglaris, "A multi-feature DDoS detection schema on P4 network hardware," in *Proc. 23rd Conf. Innov. Clouds Internet Netw. Workshops*, 2020, pp. 1–6.
[54] M. Zhang *et al.*, "Poseidon: Mitigating volumetric DDoS attacks with programmable switches," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2020, pp. 1–18.
[55] A. John and T. Sivakumar, "DDoS: Survey of traceback methods," *Int. J. Recent Trends Eng.*, vol. 1, no. 2, 2009, Art. no. 241.
[56] C. Jin, H. Wang, and K. G. Shin, "Hop-count filtering: An effective defense against spoofed DDoS traffic," in *Proc. 10th ACM Conf. Comput. Commun. Secur.*, 2003, pp. 30–41.
[57] Z. Ling, J. Luo, D. Xu, M. Yang, and X. Fu, "Novel and practical SDN-based traceback technique for malicious traffic over anonymous networks," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1180–1188.
[58] K. Rusek, J. Suárez-Varela, A. Mestres, P. Barlet-Ros, and A. Cabellos-Aparicio, "Unveiling the potential of graph neural networks for network modeling and optimization in SDN," in *Proc. ACM Symp. SDN Res.*, 2019, pp. 140–151.
[59] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 7444–7452.
[60] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2013, *arXiv:1312.6203*.
[61] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2014–2023.
[62] L. Zheng, Z. Li, J. Li, Z. Li, and J. Gao, "AddGraph: Anomaly detection in dynamic graph using attention-based temporal GCN," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 4419–4425.
[63] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1003–1012.
[64] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1243–1252.
[65] K. K. Thekumparampil, C. Wang, S. Oh, and L.-J. Li, "Attention-based graph neural network for semi-supervised learning," 2018, *arXiv:1803.03735*.
[66] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710. 10903*.
[67] K. Malialis and D. Kudenko, "Distributed response to network intrusions using multiagent reinforcement learning," *Eng. Appl. Artif. Intell.*, vol. 41, pp. 270–284, 2015.
[68] O. Osanaiye, K.-K. R. Choo, and M. Dlodlo, "Distributed denial of service (DDoS) resilience in cloud: Review and conceptual cloud DDoS mitigation framework," *J. Netw. Comput. Appl.*, vol. 67, pp. 147–165, 2016.
[69] C. Li *et al.*, "Detection and defense of DDoS attack–based on deep learning in OpenFlow-based SDN," *Int. J. Commun. Syst.*, vol. 31, no. 5, 2018, Art. no. e3497.

Yongyi Cao received the bachelor's degree in communication engineering from Wuhan University, Wuhan, China, in 2018, and the master's degree in circuits and systems from Wuhan University, Wuhan, China, in 2021. He is currently working at Alibaba. His research interests include software-defined networking, machine learning, and network security.

Hao Jiang (Member, IEEE) received the BEng degree in communication engineering and the MEng and PhD degrees in communication and information systems from Wuhan University, Wuhan, China, in 1999, 2001, and 2004, respectively. From 2004 to 2005, he undertook his postdoctoral research work with LIMOS, Clermont-Ferrand, France. He was a visiting professor with the University of Calgary, Canada, ISIMA, and Blaise Pascal University, France. He is currently a professor with Wuhan University. He has authored more than 60 papers in different journals and conferences. His research interests include mobile ad hoc networks and mobile.



Yuchuan Deng received the bachelor's degree in computer science and technology from Sichuan University, Chengdu, China, in 2019. He is currently working toward the master's degree in information and communication engineering at Wuhan University, Wuhan, China. His research interests include software-defined network, system security, and machine learning.



Jing Wu (Member, IEEE) received the BEng degree in communication engineering and the PhD degree in communication and information systems from Wuhan University, Wuhan, China, in 2002 and 2007, respectively. From 2004 to 2005, she undertook her postdoctoral research work with LIMOS, Clermont-Ferrand, France. She is currently an associate professor with Wuhan University. Her research interests include wireless communication networks, network simulation, and intelligence data processing.

Pan Zhou (Senior Member, IEEE) received the PhD degree from the School of Electrical and Computer Engineering, Georgia Institute of Technology (Georgia Tech), Atlanta, Georgia, in 2011. He is currently a full professor and PhD advisor with Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology (HUST), Wuhan, China. He received the "Rising Star in Science and Technology of HUST" in 2017, and the "Best Scientific Paper Award" in the 25th International Conference on Pattern Recognition (ICPR 2020). He is currently an associate editor of the *IEEE Transactions on Network Science and Engineering*. His research interests include security and privacy, big data analytics, machine learning, and information networks.

Wei Luo received the BSc degree in computer science from the Wuhan University of Technology, Wuhan, China, in 2002, and the MSc and PhD degrees in computer science from the Huazhong University of Science and Technology, Wuhan, China, in 2005 and 2008, respectively. Currently, he is a researcher with the Department of Innovation Center, China Ship Development and Design Center, Wuhan, China. His research interests include reliability-aware scheduling, cluster and fault tolerant computing, parallel and distributed systems, and machine learning.