# A network segmentation architecture for flow aggregation and DDoS mitigation in SDN using RAPID flow rules

Himanshu
Indian Institute of Technology Delhi
Hauz Khas, New Delhi, India
eey212005@ee.iitd.ac.in

Kaushik Saha
Indian Institute of Technology Delhi
Hauz Khas, New Delhi, India
ksaha@ee.iitd.ac.in

Payali Das
Indian Institute of Technology Delhi
Hauz Khas, New Delhi, India
eez198563@ee.iitd.ac.in

Swades De
Indian Institute of Technology Delhi
Hauz Khas, New Delhi, India
swadesd@ee.iitd.ac.in

## ABSTRACT

**Denial-of-Service (DoS) attacks have always posed a major threat to networks, directly or as a cover for more sophisticated attacks. In the recent years, with advances such as large number of IoT nodes, amplifying platforms like Botnets-as-a-Service, etc., the number of DoS attacks has increased significantly, and the attacks have become more sophisticated. The new paradigm of Software-Defined Networking (SDN) enables a centralized view of the network, which has promising potential for efficient detection and mitigation of such attacks. This modern approach, however, exposes more areas of attacks, such as, Buffer Saturation, Link Flooding, Flow Table Overflow (FTO), and Controller Saturation. In this paper, we propose a novel, extremely lightweight, simple, yet effective, integrated approach for detection and mitigation of several DoS attacks in SDN networks. Our approach uses the centralized view of the network coupled with network segmentation, based on IP assignment, to generate a novel set of rules that can be used to manage the network in a way that allows for lesser number of overall rules, preventing Flow Table Overflow altogether, while generating novel statistics which, with proper utilization, add the capability of detection and traceback of origin of attacks, to the controller, allowing Rapid Protection In Dataplane-DDoS (RAPID). We evaluate our approach using Mininet and Ryu, and show that our approach is effective in detecting and mitigating several attacks, while maintaining the network performance.**

## CCS CONCEPTS

• **Security and Privacy** → **Network Security**; **System Security**.


*ICDCN '24, Jan 04–07, 2024, Chennai, India*

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

## KEYWORDS

Software-Defined Networking (SDN), OpenFlow, Ternary Content Addressable Memory (TCAM), DDoS

**ACM Reference Format:**
Himanshu, Kaushik Saha, Payali Das, and Swades De. 2024. A network segmentation architecture for flow aggregation and DDoS mitigation in SDN using RAPID flow rules. In *ICDCN '24: 25th International Conference on Distributed Computing and Networking, Jan 04–07, 2024, Chennai, India.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

With the advancements in size and complexity of networks, such as enterprise and IoT networks and data centers, it is becoming increasingly difficult to manage them. The management of these networks is further complicated by the fact that they are often heterogeneous, consisting of a variety of devices from different vendors. The management of these networks is often done manually, which is time consuming and error prone. To address these issues, Software Defined Networking (SDN) has emerged as a new paradigm for network management.

SDN decouples the control plane from the data plane, and provides a centralized view of the network. This centralized view of the network is utilized in implementing network management functionalities, including network monitoring, traffic engineering, load balancing, intrusion detection, and many more, in a more efficient and programmable manner. The programmability of networks provided by SDN allows for the development of new applications that can run on top of the control plane completely separate from the datapath, and be used to manage the network without having to replace the network infrastructure altogether. In addition to improved network manageability, SDN has opened new areas for enhancement of security, scalability, automation and Quality-of-Service (QoS)[5]. The benefits of SDN are deeply utilized in large networks like data centers, SD-WANs (Software-Defined Wide Area Networks), smart cities and IoT networks, smart manufacturing, smart grids and so on[6].

The SDN architecture consists of three planes: data plane, control plane and management plane. The data plane consists of the network devices like switches and routers, which forward packets according to the rules installed by the controller. The control plane consists of the SDN controller, which manages the network. The management plane consists of the applications that can be used

to manage the network. The SDN controller connects to the data plane via SouthBound Interfaces (SBIs) using protocols like OpenFlow, and to the management plane via NorthBound Interfaces (NBIs). In large networks, the control is often distributed across several controllers to improve scalability and reliability. These controllers communicate with each other via East-WestBound Interfaces (EWBIs).[12]

The controller manages the network by installing rules in the data plane, which can also be used to collect statistics from the data plane and monitor the network. These rules are installed in the Ternary Content Addressable Memory (TCAM) of the OpenFlow switches. The switches then forward packets according to the rules installed by the controller. If a packet does not match any of the rules, it is sent to the controller for further processing. The controller can then install a new rule in the data plane to handle the packet or take any other action. The switches keep count of the number of packets and bytes matched to the flows, and the controller can then use these statistics for various purposes, including attack detection.

While SDN improves the manageability of the network, and provides new promises for detection of traditional attack vectors, it also introduces new security challenges.[27] The centralized view of the network makes the controller a single point of failure. The SD-network is vulnerable to other Denial of Service attacks like Buffer Overflow, Controller Saturation, Link Saturation and more, as described extensively by authors in [5]. Further, the flow rules installation is a slow process as the new flows need to be inserted according to priority and require shifting of older ones, and hence, during the insertion time the matching is paused.[9] As the number of flow rules increases, this insertion and removal becomes more and more time consuming. According to [8], the first 500 rules are installed in 6 seconds, whereas installation of next 1500 rules takes more than 2 minutes. Hence, an IP spoofing based attack also becomes increasingly difficult to mitigate not only in a large network but also in smaller ones if not mitigated early on. Furthermore, the TCAM Memory is power hungry and hence, only a finite number of flows can be installed on it. The TCAM can be easily exhausted if the flow installation is not properly handled in genuine traffic itself, let alone attack traffic, leading to TCAM exhaustion Denial of Service (DoS).

### 1.1 Motivation

The centralized view of the network, does not have direct visibility into the data plane and this makes it more complex to detect attacks in the network. The literature discusses several approaches to detect and mitigate DDoS attacks, alongside other traditional attack vectors, but without proper utilization of the potential of flow installations these approaches can become unnecessarily complex and inefficient. Many of these approaches focus on either protecting the controller or the data plane. While the former often involves collecting flow statistics or generating necessary statistics on the controller itself [20], the latter is concentrated around improving detection and removal of malicious flows, flow aggregation, etc. However, none of the approaches has been able to put a cap to the maximum number of flows to be installed on the TCAM, while improving overall security as well, and hence, neither one is able to collectively prevent several DoS attacks proactively. As discussed in [16], there is a lack of a unified approaches to detect and mitigate low-rate (TCAM exhaustion) and flooding DDoS attacks together in SDN networks.

### 1.2 Key contributions

In light of this, we propose a novel approach for managing SDN networks. Our approach couples the centralized view of network with IP-based network segmentation to generate a set of rules that can be used to manage the network efficiently, with lesser number of overall flow rules, while extending to the controller, the capability of traceback of attack origins of an ongoing DDoS attacks, allowing Rapid Protection In Datapath-DDoS (RAPID). Since the controller protection specific approaches are already discussed in the literature, we concentrate more on the data plane protection in this paper with overview of how the same approach reduces the amount of effort in controller protection as well. The rest of the paper is organized as follows:
Section 2 discusses the literature review.
Section 3 discusses the proposed approach.
Section 4 discusses the evaluation of the proposed approach using Ryu and Mininet and comparison with state-of-the-art method for DDoS detection and mitigation.
Section 5 concludes the paper.

## 2 LITERATURE REVIEW

DDoS is a general attack targeting switches, servers and controllers in an SDN network.[27] The attack aims at exhausting resources like controller capacity, link bandwidth, switch buffer, TCAM space, server resources, etc., which may occur simultaneously as well.

Following the attack vector discussions in [1, 7, 11, 18], we refer to TCAM exhaustion and ARP, ICMP, TCP SYN and UDP flooding attacks in our evaluation because, without loss of generality, these attacks are able to cover the variety of non-application layer, network DDoS attacks discussed in the literature:

- UDP flooding attack is essentially a layer 4 volumetric attack, which aims at exhausting the network bandwidth, switch capacity, etc.
- ARP and ICMP attacks are layer (respectively 2.5 and) 3 volumetric, protocol attack, which can be further amplified using IP or MAC-IP spoofing if the attackers use existing IP addresses, the genuine receiving hosts will send reply packets to the spoofed hosts which can be exploited by spoofing a server IP address and having the hosts in the network attack the server, increasing possibility of TCAM exhaustion with seemingly genuine rules as side effect.
- TCP SYN flood targets the TCP servers present in the network. The attackers send SYN packets to the server, which responds with SYN-ACK packets. The attackers do not respond with ACK packets, which causes the server to keep the connection open for a long time, exhausting the server resources. Coupled with IP or MAC-IP spoofing, the SYN flood can be amplified as well: with non-existent spoofed IP addresses, the server resources will be quickly depleted, whereas with existing spoofed hosts, the genuine hosts will

send TCP RST packets to the spoofed hosts, converting it to a volumetric attack.
- Further, MAC and IP spoofing is used in TCAM exhaustion attacks as well, which can be launched independently as a low-rate attack or as a side-effect of a high-rate flooding attack discussed earlier.

Several methods have been proposed in the literature for flow aggregation, and detection and mitigation of low-rate and flooding DDoS attacks. We broadly classify and discuss these in the following subsections.

## 2.1 Flow aggregation

The authors in [15] identify flow aggregation as a 0-1 Knapsack problem and propose a heuristic algorithm to solve it. In [13], a packet header-based hash table is used for link aggregation. In [21], a QoS-aware flow aggregation is proposed to perform match-fields based aggregation to minimize the effect of aggregation on QoS by ensuring that the long-lived flows are not aggregated. However, TCAM exhaustion attacks are not discussed by either of these.

## 2.2 Statistical and Entropy-based mitigation

### *TCAM exhaustion mitigation*

. Authors in [25] introduce FTOP to detect and mitigate Low-rate Flow Table Overflow (LFTO) and Flash Crowd TCAM exhaustion. FTOP polls the flows periodically and predicts future flow count by KF estimator. If the estimate exceeds a threshold, a Random Forest classifier is applied on the flows and then malicious flows are identified and removed. SFTO-Guard [24] uses flow count and port Shannon entropy features to detect LFTO attacks with Light-GBM. FTMaster [23] goes further in feature extraction and entropy calulation based on degree of distortion of IP and ports to detect and mitigate LFTO attacks. The team also proposes FTODefender [26] which works similarly by polling flows periodically and evaluating flow table similarly (to a benign flow table), source IPs entropy in the flow table (using IP to flow count mapping probabilities) and variation of growth rate of foreign source and destionation port entries. There are several similar approaches but these, however, are completely reactive approaches based on flow counts and hence, are not suitable for high-rate attacks.

The authors in [14] have proposed a novel method for detection of FTO attacks at low and high rates and mixed attacks. The method takes a count of rules which 'belong to' a port on a switch and calculates proactive rule numbers for the ports and switch, which, if exceeds the flow capacity of the switch, implies an attack, and the ports, which have high factor of standard deviation from the average port proactive rate of the switch, are blocked. This approach has a similar basis to our approach, but is strictly limited to the TCAM exhaustion attacks and has not taken other DDoS attacks into consideration at all. Furthermore, the approach becomes reactive in the sense that only when the proactive rule count exceeds the flow capacity, the ports are blocked. This still leaves the controller and critical servers in the network vulnerable to targeted attacks.

### *TCP SYN, ICMP and UDP flood mitigation*

. Several approaches try to use TCP protocol characteristics like like OPERETTA uses connection establishment acknowledgment (ACK), SLICOTS installs temporary flows with TCP RTO timeout value as hard timeout, SRL uses MSS value, RTT approach in uses TCP RTT, etc., to detect TCP SYN flood attacks. SYN-Guard uses MAC frame sequence number for MAC-spoofing detection. However, these approaches have several flaws, including the fact that they are not able to detect low-rate attacks.

AEGIS [20] introduces several novel statistical metrics like chance of syn flood, deviation between SYN and SYN-ACK packet counts, service fairness, controller health, etc. to detect TCP SYN flood on the controller. It goes one step further in classifying the heavy traffic into IP and MAC-IP spoofing attacks and Flash Crowd traffic. Furthermore, the authors propose a method for context-aware estimation for the number of backup controllers required. However, the approach has some drawbacks as discussed later in section 4.

## 2.3 Learning-based methods

The authors in [10] and [11] use LSTM models to detect ICMP, TCP SYN and UDP flood attacks. In [22], the authors have used NSL-KDD dataset with K-means for processing training data and KNN for traffic detection. The features extracted are avg byte rate, avg flow duration time, percentage of symmetric flows, rate of variation of asymmetric flows, fraction of flows with small counts, etc. We also use symmetry as a basis for our approach, but the aforementioned paper discussed does not take into account the TCAM exhaustion. In [17], the authors generated genuine traffic using iperf in mininet, while the attack traffic was generated using hping3. Several other methods are proposed in literature using various techniques like KNN, CNN, LSTM, DL, etc., but due to lack of a proper standard SDN dataset, a majority of learning datasets used in the research studies are legacy datasets, unreliable for SDNİf self-generated testbeds are used, The test networks used are unable to generate a realistic traffic to train on, which is a major drawback of these approaches.[19]

ADAM[3] extracts layer 3 and layer 4 features from the network traffic and use Entropy Vector as a training dataset for their model. The method is different from usual Learning-based methods in the sense that it uses network bandwidth and link congestion as its primary detection feature. If the bandwidth exceeds a threshold, the link is considered to be congested. At this point, the controller redirects the traffic flow towards itself until a certain number of packets are received. These packets are then examined with the features extracted and an entropy profile is created, which is used for training KNN models for predicting outlier EVs. But once again, this is high-rate attack detection method and will not apply to TCAM exhaustion attacks.

Flood Control [4] use time-differential port-level statistics instead of flow statistics for the detection. A machine-learning classifier is then applied on these features to classify a port as attacking or genuine. These statistics are also used by us for different features. However, the aforementioned paper does not discuss the TCAM exhaustion attacks.

# 3 PROPOSED APPROACH

We propose a novel approach which couples the centralized view of network available at the controller with IP-based network segmentation to generate a set of rules that can be used to manage

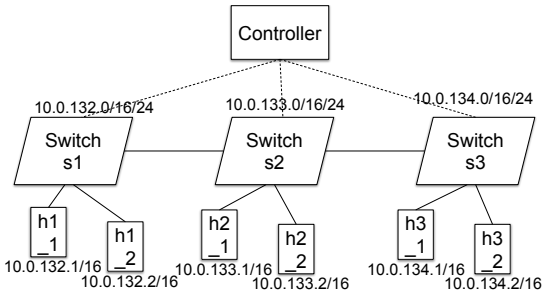


**Figure 1: System Model.**

the network in a way that allows for lesser number of overall rules, while extending the capability of traceback of origins of ongoing DDoS attacks, to the controller. We utilize up to 8 tables in the OpenFlow pipeline to achieve this. The flow diagram of the proposed approach is shown in Algorithm 1.

## 3.1 Assumptions

It is assumed that:

- The network is purely Software-Defined, and managed by a single controller, or a set of controllers that are working in a coordinated manner. The traditional networks can be present but these undermine the applicability of the approach, and therefore, are not advisable. If an attacker however hides in the traditional network, the approach can still trace the attack to the SDN ports on which the traditional network is connected without knowledge of the network administrator, but a complete traceback to attacker will not be possible.
- The SouthBound Interface (SBI) is OpenFlow, and protected with ample security measures, such as TLS, to prevent any attacks via the SBI itself.
- The Link Discovery Service (LDS) is running on the controller using Link-Layer Discovery Protocol (LLDP) or OpenFlow Discovery Protocol (OFDP) to discover the network topology. It is further assumed that this service is protected from attacks like topology tampering, Man-in-the-Middle (MitM), etc. This is a reasonable assumption as the LDS is a critical component of the controller; the LLDP packets can be modified by the controller, and therefore, protected with proper encryption mechanisms in place.
- The controller and switches are trusted. The flow rule installations and flow statistics are accurate and not tampered with. Further, the time delays in the flow statistics collection are same for all the switches, and hence, the flow stats replies sent by the switches are received by the controller at the same timestamp.
- The network uses IPv4 addressing scheme. The approach can be modified to extend to IPv6.

## 3.2 Network Segmentation

We divide the traffic flow into these types based on the Network and Transport Layer Protocol adopted by the packets:

a. **IP Symmetric**: The IP packets using layer 4 protocols for which a reply packet is expected, like ICMP, TCP, some UDP applications like DNS, DHCP, etc.
b. **IP Asymmetric**: The IP packets using layer 4 protocols for which a reply packet is not expected.
c. **Non-IP**: The non-IP packets, like ARP, etc. These packets are handled by the controller itself.

This characterization can be achieved using metadata fields in any table in the OF pipeline. The flows for the first two types of traffic are managed separately. For the Non-IP traffic, the flows either drop or forward to the controller. Without loss of generality, it has been assumed for evaluation that the layer 4 traffic is ICMP, TCP and UDP only. The scope of this paper has been limited to detection and mitigation of IP Symmetric packet attacks.

When the network goes online, the controller identifies the subnet mask of the network portion, say parent subnet, assigned to it, and divides this portion into sub-subnets or SSNs further. These SSNs are initially assigned randomly to the switches. For simplicity, it is at least required that the parent subnet assigned to the network is such that it covers all the ports on all switches in the controller-assigned network, and therefore, a switch can be identified on the basis of SSNs it uses. If the number of ports in a switch is higher than the range available from one SSN, it is assigned multiple SSNs. The SSNs are denoted as IP/SNLength/SSNLength For example, in a network represented by CIDR `10.0.0.0/16`, the switches may be assigned `10.0.1.0/16/24`, `10.0.2.0/16/24`, and so on, where 16 is the length subnet mask and 24 is the length of sub-subnet mask. Alternatively, the sub-subnet mask can also be assigned with multiple DHCP proxies per switch instead of single (all virtually on controller in both cases) as discussed later in the approach, but for simplicity, this configuration is used.

The goal is to assign to hosts connected directly to a switch, IP addresses from the sub-subnet network, and mask length equal to the switch's subnet mask.

The IP addresses are now divided into two types:

1. **Internal IP**: IPs from the switch's sub-subnet range.
2. **External IP**: IPs outside the switch's sub-subnet range.

A DHCP proxy application is set up on the controller, which maintains a database of SSNs assigned to the switches, alongside the MAC-IP-Port mapping of the hosts. Further, this application is responsible for handling MAC-IP-Port assignment for hosts with static IP addresses.

The controller then generates and installs a set of rules based on the switches to SSN mapping, called SubNet Flow Rules (SNFRs), on each switch. The rules are defined in such a way that the controller can identify the SSN of the source and destination of an IP-based packet, and the type of the packet, i.e., symmetric or asymmetric. The aim of the SNFRs is to generate statistics which can be utilized in traceback in case of an attack. A table-wise exemplary description of the SNFRs for switch $s_2$ from Fig 1 is given in Fig 2. In this example, the IP address of DHCP proxy application (or the controller) is set as `10.0.254.1/16` .

In addition, the flows for non-local addresses are configured as a separate set but in a similar fashion - symmetric, asymmetric, non-IP, etc., except that the flows will be more directed towards some

gateway(s), whose information will be provided to the controller in advance.

### 3.3 IP Symmetric protection

The flow rules (figure 2) are defined such that the following features can be obtained for each switch (in the figure, the red rectangle represents `in_port` flow rules, blue `output:p` flow rules and green are rules used in calculating $C_{ij}$ features below):

- $C_{11}$ := table 4 cookie `0x3110000000000001` packets
- $C_{12}$ := table 4 cookie `0x3120000000000001` packets
- $C_{21}$ := table 5 cookie `0x3110000000000001` packets
- $C_{22}$ := table 5 cookie `0x3120000000000001` packets
- $P_p^{rx}$ := packet counts of table 1 `in_port:p` rules (fig. 2)
- $P_p^{tx}$ :=

$$\sum_{f \in F} c_f\,, \quad \text{where } c_f = \text{Packet count of flow } f,$$
$$\forall\, F = \{f \mid f \in \text{flows on the switch in table 6 and 7 which have an action } \texttt{output:p}\}.$$

- $R_{2in}$ :=

$$\sum_{q \in Q} P_q^{tx}\,, \quad \forall\, Q = \{q \mid q \in \text{peer ports on neighbors}\}.$$

In $C_{ij}$, the subscript $i$ is for source IP, and $j$ for destination IP. Subscript value 1 refers to internal IP, and 2 refers to external IP For example, $C_{12}$ represents packets with internal IP as source and external IP as destination, thereby going "out" of the switch, $C_{22}$ - packets "through" the switch, and so on. The port features $rx$ and $tx$ are from the perspective of the switch, looking out via its ports: $rx$ - packets received by the switch from its ports, $tx$ - packets transmitted by the switch to its ports. Further, link information is maintained by the controller from which $R_{2in}$ can be calculated.

With this information, the following metrics are evaluated for each switch and port:

- Link balance:

$$\Delta_{1k} = \frac{C_{21k} + C_{22k} - R_{2in.k}}{C_{21k} + C_{22k} + R_{2in.k}} \tag{1}$$

- Port balance:

$$\Delta_{2p} = \frac{P_p^{rx} - P_p^{tx}}{P_p^{rx} + P_p^{tx}} \tag{2}$$

The flow statistics are collected every $T$ seconds or every $N$ packets, whichever is earlier. The values of $T$ and $N$ are selected such that the controller is not overloaded with Packet In messages and controller-switch links are not saturated. A meter is also installed on the miss-entry of table 6 and 7 to limit the Packet In rate below the link capacities.

Under normal conditions, in a network, these values are close to zero. In fact $\Delta_1$ is perfectly zero if no packets are lost in link transit. However, in case of a DDoS attack with IP spoofing or MAC-IP spoofing, four cases are possible as in Table 1.

When an external IP is spoofed to send heavy malicious traffic, $C_{21}$ or $C_{22}$ increase on edge switch, whereas $R_{2in}$ does not since the switch is not receiving this traffic from the neighbors. This leads to a rise in $\Delta_1$ value.

| Spoofed IP | Target IP |
|---|---|
| External | External |
| Internal | External |
| External | Internal |
| Internal | Internal |

**Table 1: Possible attack configurations.**

Further, during the attack the attacker port is transmitting much more than it is receiving, hence, $P_p^{rx}$ increases much more than $P_p^{tx}$. This leads to a rise in $\Delta_2$ value.

The rise in $\Delta_1$ and $\Delta_2$ values can be used to detect the attack. Further, the $\Delta_1$ and $\Delta_2$ values can be used to trace the attack to the switch ports on which the attack is originating. The controller can then install a flow rule on the switch to drop the packets from the attacker, or take any other action. The steps for the detection and mitigation of the attack are given in Algorithm 1.

Out of several configurations tested for ICMP flood and TCP SYN flood attack, one is of utmost importance, whose characteristics are as follows:

- The attackers lie on path between the Victim and the Spoofed IP SSN switch.
- The host spoofs an existing genuine host on the switch.
- The victim is a TCP Server.

In this configuration, the values of $\Delta_1$ and $\Delta_2$ were found to be the lowest; when the attacker sends SYN, the Server replies with a SYN-ACK to the spoofed host which replies back with an RST. Hence this configuration was used for selection of thresholds.

The thresholds were selected as follows:

**Algorithm 1:** RAPID Detection and Mitigation

```
Data: Switches; Ports; Links; Flow statistics
Result: Output result
Repeat every T seconds or N packets
    Get flow statistics for all Switches
    Calculate Δ1 for all Switches
    Calculate Δ2 for all Ports
    Sort Switches in descending order of Δ1
    Sort switch Ports in descending order of Δ2
    forall switch k in switches do
        if Δ1 of switch k is greater than 0.15
        then
            forall port p in switch k do
                if Δ2 of port p is greater than 0.5
                then
                    if port p is edge port then
                        Block port p
                    else
                        if Δ1 > 0.33 then
                            Install meter on port p
                        Traceback via the link on port p
```

| | T | cookie | match | prior | instructions | pkts |
|---|---|---|---|---|---|---|
| 0 | 0 | 0x0 | [{'eth_dst': '01:80:c2:00:00:0e', 'eth_type': 35020}] | 65535 | [['CONTROLLER']] | 4541 |
| 1 | 0 | 0x1000000000000000 | [{'eth_type': 34525}] | 1 | [[]] | 1069 |
| 2 | 0 | 0x2130000000000001 | [{'eth_type': 2054, 'arp_tpa': '10.0.254.1'}] | 512 | [['meter:255', 'CONTROLLER']] | 6 |
| 3 | 0 | 0x2010000000000001 | [{'eth_type': 2054, 'arp_spa': ('10.0.133.0', '255.255.255.0')}] | 256 | [['meter:255', 'CONTROLLER']] | 5 |
| 4 | 0 | 0x2020000000000001 | [{'eth_type': 2054}] | 0 | [['meter:255']] | 25 |
| 5 | 0 | 0x3130000000000001 | [{'eth_dst': '08:00:27:b8:0f:8d', 'eth_type': 2048, 'ipv4_dst': '10.0.254.1'}] | 1024 | [['CONTROLLER']] | 0 |
| 6 | 0 | 0x31f0000000000001 | [{'eth_type': 2048, 'ipv4_dst': '255.255.255.255'}] | 768 | [['CONTROLLER']] | 0 |
| 7 | 0 | 0x3100000000000001 | [{'eth_type': 2048, 'ipv4_dst': '0.0.0.0'}] | 768 | [['CONTROLLER']] | 0 |
| 8 | 0 | 0x30f0000000000001 | [{'eth_type': 2048, 'ipv4_src': '255.255.255.255'}] | 512 | [['CONTROLLER']] | 0 |
| 9 | 0 | 0x3000000000000001 | [{'eth_type': 2048, 'ipv4_src': '0.0.0.0'}] | 512 | [['CONTROLLER']] | 0 |
| 10 | 0 | 0x3000020000000001 | [{'eth_type': 2048, 'ip_proto': 17}] | 256 | [['goto:2']] | 0 |
| 11 | 0 | 0x3000000000000001 | [{'eth_type': 2048}] | 0 | [['goto:1']] | 1337713 |
| 12 | 0 | 0xe0000f0000000001 | [{'in_port': 3}] | 1280 | [[]] | 34354 |
| 13 | 0 | 0xf0ffffffffffffff | [{}] | 0 | [[]] | 0 |
| 14 | 1 | 0xe000000000000001 | [{'in_port': 4, 'eth_type': 2048}] | 256 | [['goto:3']] | 99686 |
| 15 | 1 | 0xe000000000000001 | [{'in_port': 1, 'eth_type': 2048}] | 256 | [['goto:3']] | 203494 |
| 16 | 1 | 0xe000000000000001 | [{'in_port': 2, 'eth_type': 2048}] | 256 | [['goto:3']] | 679862 |
| 17 | 1 | 0xe000000000000001 | [{'in_port': 3, 'eth_type': 2048}] | 256 | [['goto:3']] | 354670 |
| 18 | 1 | 0xe000000000000001 | [{'in_port': 4294967293, 'eth_type': 2048}] | 256 | [['goto:3']] | 1 |
| 19 | 2 | 0xe000020000000001 | [{'in_port': 4, 'eth_type': 2048, 'ip_proto': 17}] | 256 | [['goto:3']] | 0 |
| 20 | 2 | 0xe000020000000001 | [{'in_port': 1, 'eth_type': 2048, 'ip_proto': 17}] | 256 | [['goto:3']] | 0 |
| 21 | 2 | 0xe000020000000001 | [{'in_port': 2, 'eth_type': 2048, 'ip_proto': 17}] | 256 | [['goto:3']] | 0 |
| 22 | 2 | 0xe000020000000001 | [{'in_port': 3, 'eth_type': 2048, 'ip_proto': 17}] | 256 | [['goto:3']] | 0 |
| 23 | 2 | 0xe000020000000001 | [{'in_port': 4294967293, 'eth_type': 2048, 'ip_proto': 17}] | 256 | [['goto:3']] | 0 |
| 24 | 3 | 0x3010000000000001 | [{'eth_type': 2048, 'ipv4_src': ('10.0.133.0', '255.255.255.0')}] | 256 | [['goto:4']] | 107433 |
| 25 | 3 | 0x3020000000000001 | [{'eth_type': 2048}] | 0 | [['goto:5']] | 1230280 |
| 26 | 4 | 0x3110020000000001 | [{'eth_type': 2048, 'ipv4_dst': ('10.0.133.0', '255.255.255.0'), 'ip_proto': 17}] | 272 | [['goto:6']] | 0 |
| 27 | 4 | 0x3110000000000001 | [{'eth_type': 2048, 'ipv4_dst': ('10.0.133.0', '255.255.255.0')}] | 256 | [['goto:6']] | 0 |
| 28 | 4 | 0x3120020000000001 | [{'eth_type': 2048, 'ip_proto': 17}] | 16 | [['goto:7']] | 0 |
| 29 | 4 | 0x3120000000000001 | [{'eth_type': 2048}] | 0 | [['goto:7']] | 107433 |
| 30 | 5 | 0x3110020000000001 | [{'eth_type': 2048, 'ipv4_dst': ('10.0.133.0', '255.255.255.0'), 'ip_proto': 17}] | 272 | [['goto:6']] | 0 |
| 31 | 5 | 0x3110000000000001 | [{'eth_type': 2048, 'ipv4_dst': ('10.0.133.0', '255.255.255.0')}] | 256 | [['goto:6']] | 106715 |
| 32 | 5 | 0x3120020000000001 | [{'eth_type': 2048, 'ip_proto': 17}] | 16 | [['goto:7']] | 0 |
| 33 | 5 | 0x3120000000000001 | [{'eth_type': 2048}] | 0 | [['goto:7']] | 1123565 |
| 34 | 6 | 0x31000000a0085021 | [{'eth_type': 2048, 'ipv4_dst': '10.0.133.2'}] | 512 | [['output:4']] | 98807 |
| 35 | 6 | 0x31000000a0085011 | [{'eth_type': 2048, 'ipv4_dst': '10.0.133.1'}] | 512 | [['output:3']] | 7907 |
| 36 | 6 | 0xf3ffffffffffffff | [{}] | 0 | [['meter:1', 'CONTROLLER']] | 1 |
| 37 | 7 | 0x31000000a0086001 | [{'eth_type': 2048, 'ipv4_dst': ('10.0.134.0', '255.255.255.0')}] | 512 | [['output:2']] | 657847 |
| 38 | 7 | 0x31000000a0084001 | [{'eth_type': 2048, 'ipv4_dst': ('10.0.132.0', '255.255.255.0')}] | 512 | [['output:1']] | 573150 |
| 39 | 7 | 0xf4ffffffffffffff | [{}] | 0 | [['meter:2', 'CONTROLLER']] | 1 |

**Figure 2: Flow rules on switch $s_2$ in the network topology in Figure after the attack from $h_{21}$ to $h_{31}$.**

- $\Delta_1$ standard threshold was selected as 0.15, slightly less than 0.19 - estimated maximum in the above attack.
- $\Delta_1$ strict threshold was selected as 0.33 from theoretical analysis of other attack configurations.
- $\Delta_2$ threshold was selected as 0.5, for a situation where $P_p^{rx} = 3 \times P_p^{tx}$, *i.e.*, the switch receives from the port thrice as much as it transmits to the latter.

## 3.4 IP Asymmetric protection

Handling of IP Asymmetric traffic is not possible with the same features. However, in conjunction with administrator policies, the flow rules can be used in tracing back the sources of attack.

## 3.5 Controller and Non-IP protection

The non-IP and the DHCP packets are forwarded directly to the controller. The controller can implement port-wise counters based protection mechanisms for this traffic. For example, a port can be limited to send only a maximum number of DHCP requests in a lease period to prevent DHCP starvation attack. Similarly, the controller can monitor ARP requests and replies to detect ARP spoofing attacks.

In our implementation, the ARP requests are handled by the controller itself and never forwarded directly to the recipients, but rather an ARP proxy-like mechanism is used. The controller sends ARP request with its own MAC and IP as source addresses. Thus, any ARP replies received must have the same as destination addresses.

Further, since the controller is in control of DHCP communications as proxies or as a server, it has knowledge of which switch does a requested IP address belong to, and can flood only that switch. All switches have flow rules to drop ARP packets with external source IP addresses. This avoids excessive unnecessary flooding and the removes the requirement of STP for loops. Further, it limits the range of IP addresses available for ARP spoofing.

Furthermore, the controller can keep counts of the number of IP addresses claimed by each port in ARP and DHCP requests and monitor the same for any abnormal behavior.

A new IP symmetric attack is possible against the approach discussed above in following steps:

1. The attacker compromises two hosts in the network, say $A$ and $B$.
2. $B$ connects to the a host $T$ and hence the ARP details of $B$ are cached on $T$. $B$ disconnects soon after this.
3. $A$ waits for some time for the `output:`$p_B$ flow installed on $B$'s switch to time out and then spoofs the address of $B$ and sends flood to $T$. Since the flows are no longer there, the packets are forwarded to the controller.
4. The controller tries to find $B$ but it is not connected to the network. The flood keeps coming until handled using the features discussed in Section 3.3.

To mitigate this, since the table miss entry is set to drop packets, and the unmatched packets are rather segregated in the OpenFlow pipeline, the rules which forward to the controller can be metered based on network demand, controller-switch link bandwidth and controller capacity. Further, apart from collecting the statistics every $T$ seconds, a collection every $N$ packets is also be made for a rapid response. The evaluation of these variables and the packet rates of the aforementioned meters has been left to the network administrator.

### 3.6 Flow Table protection

The flows installation scheme in RAPID falls under the category of destination-based flows. Hence, we compare the flow count with a simple destination IP-based flow installation scheme.

For a network with $K$ switches, and $N$ hosts and $L+N$ ports per switch, the number of flows on any switch can be limited by:

$$F \sim O(K \times N) \tag{3}$$

With RAPID, each switch will have flows in the order of:

- $(L+N) \times 2$ `in_port` rules for the all its ports
- $N$ `output:p` rules for the hosts connected to it.
- $K-1$ `output:p` rules for connecting to the other switches.
- $4 \times 2$ SNFR rules.
- About 14 other auxillary rules.

Hence, the maximum number of flows on any switch in a RAPID network architecture will be in the order of:

$$F \sim O(L+N+K)\ , \tag{4}$$

which is much less than the destination-based flow installation scheme for large networks, which makes RAPID scalable. For example, in a network with $K = 100$ switches, $N = 10$ hosts, and $L+N = 48$ ports per switch, the number of flows on any switch will be in the order of $O(148)$, compared to $O(100 \times 10)$ flows in the destination-based flow installation scheme.

## 4 EVALUATION

### 4.1 Experimental Setup

The tests were performed using an Ubuntu-20.04 Server in VirtualBox VM with 4GB RAM and 2 CPU cores as the controller running Ryu applications. Since the network architecture discussed is different from usual norms, a dataset for benchmarking could not be shortlisted, and hence, a Mininet-emulated network was used.

| Metric | Simple | RAPID |
|---|---|---|
| Transfer (GBytes/s) | 51.9 | 48.7 |
| Bandwidth (Gbits/s) | 44.6 | 41.8 |

**Table 2: Simple vs RAPID bandwidth comparison**

The network topology used for evaluation of above configuration is shown in Fig 1. The network consists of 3 switches, 3 hosts, and 1 controller. The hosts are connected to the switches as shown in the figure. The controller is connected to the switches using a dedicated interface on the controller VM. The host $h_{21}$ spoofs IP address from SSN range of switch $s_1$ and launches attack on the victim host $h_{31}$, which is the acting TCP server here.

### 4.2 Bandwidth comparison

Since the OpenFlow pipeline has lengthened, the bandwidth of the datapath links must be affected. So we compare the bandwidth of the datapath links with and without RAPID. For comparison, we run a `simple_switch_14` controller provided by ryu which installs flows on a switch to act as a traditional switch. The bandwidth is tested using `iperf` commands. The results in Table 2 show about 6.2% reduction in the bandwidth.

### 4.3 Attack detection and mitigation

For evaluation, the results were compared with AEGIS controller implementation.[20] AEGIS is selected because it claims to provide a complete solution to TCP SYN flooding attack using IP and MAC-IP spoofing. The base code for both RAPID and AEGIS is same, and hence, the comparison is fair.

The traffic rate and attack rate were selected as 100 packets/sec and 10000 packets/sec to stay within the VirtualBox VM-Host link bandwidth limits. The attack was launched for 45 seconds, 20 minutes later than genuine traffic. All hosts in the network were communicating with the server, when the attack was launched.

The genuine traffic was generated as Poisson TCP traffic using Distributed Internet traffic generator or D-ITG[2] using following command:

```
ITGSend -T TCP -t \$((1000 * 3600)) \
    -a SERVER -rp PORT -o 1000 -O 100 \
    -l logs/sender.log -x logs/receiver.log
```

The attack traffic was generated using hping3, which the server listening at port 8888 using following command:

```
timeout 45 hping3 -SVd 1 -w 64 -p 8888 SERVER \
    -a SPOOFED -c $((1000*1000)) -i u100
```

AEGIS calculates a Figure-of-Merit (FOM) for the controller. The FOM takes into account:

- The number of SYN and SYN-ACK packets received by the controller, and processed by the controller.
- The time series differences in the reasons for Packet Ins received by the controller.
- The CPU and memory usage of the controller.

The test conditions are selected in favor of AEGIS wherever possible:

- The ARP details of the non-existent spoofed host are saved in the TCP server's ARP cache to futher increase the flooding with SYN-ACK replies from the server to the controller which is unaware of the spoofed host's existence. This has both positive and negative impact on AEGIS, as it increases the controller CPU and memory usage and the time series differences, but also decreases one of their features, i.e., Difference between SYN and SYN-ACK, D. But due to the high rate of attack, the positive outweighs the negative.
- The attacker is spoofing a non-existent IP `10.0.132.10/24` in sub-subnet range of switch $s_1$. Here, AEGIS and RAPID are both impervious to the IP range selected.
- AEGIS is also impervious to the attack location. Hence, The attacker is placed at the switch $s_2$, which is most unfavorable for RAPID.

## 4.4 Results

The attack is launched in the above network configuration with controllers running RAPID and AEGIS. The respective scanning periods for both approaches are kept 1 second. The results are shown in following figures:

- Figure 3 shows the $\Delta_1$ values for the switches in RAPID. The $\Delta_1$ value for switch $s_2$ is higher than the other switches, and therefore, the attack is detected at switch $s_2$.
- Figure 4 shows the $\Delta_2$ values for the ports of switches $s_1$ and $s_3$. The port $s_2$ of switch $s_1$ suggests an attack. Using the link information, we trace the attack to switch $s_2$.
- Figure 5 shows the $\Delta_2$ values for the ports of switch $s_2$. The $\Delta_2$ value for port $h_{21}$ is higher than the other ports, and therefore, the attack is detected at port $h_{21}$. RAPID installs a drop rule on the port $s_2$ of switch $s_1$, with a hard timeout of `15s`, and a meter with hard timeout of `30s`, to mitigate the attack. In the figure, multiple peaks can be seen, because the attack continues even after the hard timeout of `15s` and is mitigated again after the hard timeout of `15s`.
- Figure 6 shows the comparison of RAPID and AEGIS detection metrics. The detection time for RAPID is `1s` for the selected rates, whereas for AEGIS it is `9s`. The detection time for AEGIS is higher because it waits for the FOM quality to degrade below 60%, which does not fall rapidly. Further, AEGIS fails to mitigate the attack in this scenario of IP-spoofing because the number of IP-spoofing MACs is less than the 20% threshold set by AEGIS. To aid AEGIS, the threshold was reduced to 10%, but it still failed to mitigate the attack once the attacker adapted with MAC-IP spoofing and random traffic generation because traceability is not implemented by AEGIS.
- Figure 7 shows the comparison of RAPID and AEGIS CPU and Memory usage. The CPU and Memory usage for RAPID and AEGIS are comparable before the attack. During the attack, the CPU and memory usage patterns of the two are comparable but since AEGIS fails to detect the attack, the CPU usage for AEGIS remains high for the entire duration of the attack.

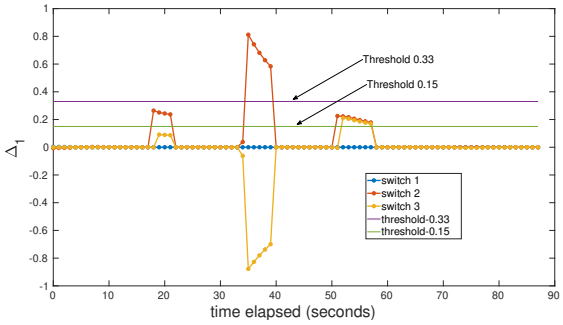


**Figure 3: $\Delta_1$ values for the switches under attack.**

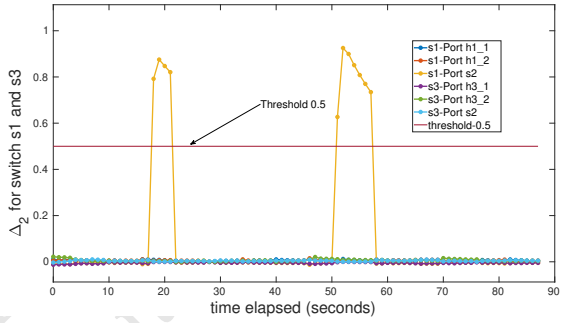


**Figure 4: $\Delta_2$ values of ports of non-attacker switches s1,s3.**

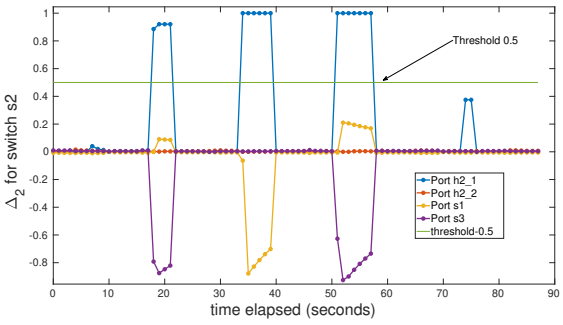


**Figure 5: $\Delta_2$ values of ports of attacker's switch s2.**

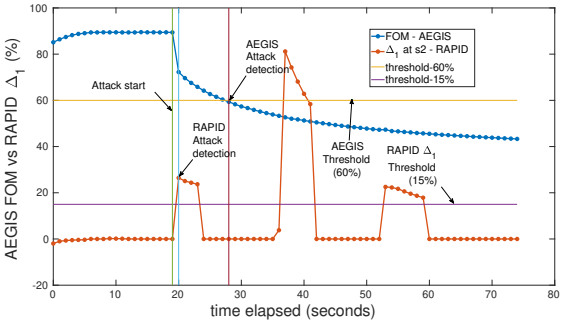


**Figure 6: RAPID vs AEGIS detection metrics.**

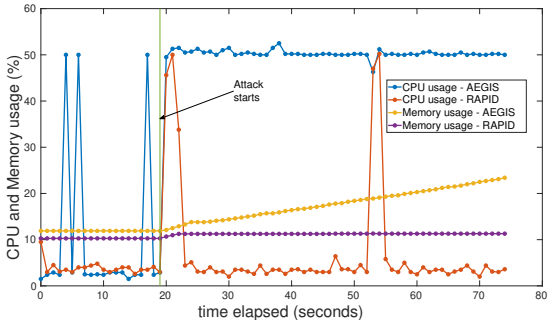


**Figure 7: RAPID vs AEGIS CPU-Memory usage.**

Further, AEGIS mitigation metrics require peer-to-peer flows to be installed, which clearly leads to much higher flow count than any destination-based flow scheme, including RAPID.

## 5 CONCLUSION

We propose a network management methodology, RAPID, which combines centralization in SDN with IP-based segmentation to detect and mitigate several types of DDoS attacks in the network, including but not limited to, TCAM exhaustion, ICMP flooding, TCP SYN flooding, using MAC, IP or MAC-IP spoofing, to give a proactive and reactive approach to DDoS mitigation.

We compare the performance of RAPID with a simple controller network scenario and find that the added detection and traceability come with a bandwidth trade-off of about 6.2%.

Further, we compare the detection and mitigation of a TCP SYN attack with AEGIS, a state-of-the-art SDN-based TCP SYN DDoS mitigation methodology, and find that RAPID is faster in detecting and mitigating the attack. We also find that the flow count requirement of RAPID is lower than that of AEGIS in large networks.

### Future Work

We plan to extend RAPID to detect and mitigate other types of DDoS attacks, including but not limited to, ARP poisoning, UDP flooding, DNS amplification, and NTP amplification.

## REFERENCES


[1] B. Alhijawi, S. Almajali, H. Elgala, H. Bany Salameh, and M. Ayyash. 2022. A survey on DOS/ddos mitigation techniques in SDNS: Classification, comparison, solutions, testing tools and datasets. *Computers and Electrical Engineering* 99 (2022), 107706.

[2] S. Avallone, S. Guadagno, D. Emma, A. Pescape, and G. Ventre. 2004. D-ITG distributed Internet traffic generator. In *First International Conference on the Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings.* IEEE, Los Alamitos, CA, USA, 316–317. https://doi.org/10.1109/QEST.2004.1348045

[3] Tianyang Cai, Tao Jia, Sridhar Adepu, Yuqi Li, and Zheng Yang. 2023. ADAM: An Adaptive DDoS Attack Mitigation Scheme in Software-Defined Cyber-Physical System. *IEEE Transactions on Industrial Informatics* 19, 6 (2023), 7802–7813. https://doi.org/10.1109/TII.2023.3240586

[4] Tapadhir Das, Osama Abu Hamdan, Shamik Sengupta, and Engin Arslan. 2022. Flood Control: TCP-SYN Flood Detection for Software-Defined Networks using OpenFlow Port Statistics. In *2022 IEEE International Conference on Cyber Security and Resilience (CSR).* IEEE, Rhodes, Greece, 1–8. https://doi.org/10.1109/CSR54599.2022.9850339

[5] R. Deb and S. Roy. 2022. A comprehensive survey of vulnerability and information security in SDN. *Computer Networks* 206 (2022), 108802.

[6] M. Rahouti et al. 2022. SDN Security Review: Threat Taxonomy, implications, and open challenges. *IEEE Access* 10 (2022), 45820–45854.

[7] S. G. Rawat et al. 2023. A survey of ddos attacks detection schemes in SDN environment. In *2023 International Conference on Computer, Information and Telecommunication Systems (CITS).* IEEE, Genoa, Italy.

[8] X. Ding et al. 2017. Unified NVTCAM and STCAM Architecture for improving packet matching performance. *ACM SIGPLAN Notices* 52, 5 (2017), 91–100.

[9] Y. Wan et al. 2021. T-cache: Efficient policy-based forwarding using SMALL TCAM. *IEEE/ACM Transactions on Networking* 29, 6 (2021), 2693–2708.

[10] Sya Raihan Heggi, Parman Sukarno, and Satria Akbar Mugitama. 2022. LSTM-NB: DoS Attack Detection On SDN With P4 Programmable Dataplane. In *2022 International Conference on Advanced Creative Networks and Intelligent Systems (ICACNIS).* IEEE, Bali, Indonesia, 1–6. https://doi.org/10.1109/ICACNIS57039.2022.10055615

[11] Y. Jia, F. Zhong, A. Alrawais, B. Gong, and X. Cheng. 2020. FlowGuard: An intelligent edge defense mechanism against IOT ddos attacks. *IEEE Internet of Things Journal* 7, 10 (2020), 9552–9562.

[12] M. B. Jimenez, D. Fernandez, J. E. Rivadeneira, L. Bellido, and A. Cardenas. 2021. A survey of the main security issues and solutions for the SDN architecture. *IEEE Access* 9 (2021), 122016–122038.

[13] Ronaldo RR Junior, Marcos AM Vieira, Luiz FM Vieira, and Antonio AF Loureiro. 2022. Intra and inter-flow link aggregation in SDN. *Telecommunication Systems* 79, 1 (2022), 95–107.

[14] Dezhang Kong, Chunming Wu, Yi Shen, Xiang Chen, Hongyan Liu, and Dong Zhang. 2022. TableGuard: A Novel Security Mechanism Against Flow Table Overflow Attacks in SDN. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference.* IEEE, Rio de Janeiro, Brazil, 4167–4172. https://doi.org/10.1109/GLOBECOM48099.2022.10001437

[15] T. Kosugiyama, K. Tanabe, H. Nakayama, T. Hayashi, and K. Yamaoka. 2017. A flow aggregation method based on end-to-end delay in SDN. In *2017 IEEE International Conference on Communications (ICC).* IEEE, Paris, France.

[16] W. H. Muragaa. 2023. A hybrid scheme for detecting and preventing single packet low-rate ddos and flooding ddos attacks in SDN. In *2023 IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA).* IEEE, Benghazi, Libya.

[17] S Neelavathy Pari, E C Ritika, B Ragul, and M Bharath. 2023. AI-based Network Flooding Attack Detection in SDN using Multiple Learning Models and Controller. In *2023 12th International Conference on Advanced Computing (ICoAC).* IEEE, Chennai, India, 1–7. https://doi.org/10.1109/ICoAC59537.2023.10249017

[18] M. M. Oo, S. Kamolphiwong, and T. Kamolphiwong. 2017. The design of SDN based detection for distributed denial of service (ddos) attack. In *2017 21st International Computer Science and Engineering Conference (ICSEC).* IEEE, Bangkok, Thailand.

[19] Ritu Raj and Sandeep Singh Kang. 2022. A Review on DDoS attack Detection in SDN using ML. In *2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N).* IEEE, Greater Noida, India, 550–553. https://doi.org/10.1109/ICAC3N56670.2022.10074330

[20] N. Ravi, S. M. Shalinie, C. Lal, and M. Conti. 2021. AEGIS: Detection and mitigation of TCP SYN Flood on SDN controller. *IEEE Transactions on Network and Service Management* 18, 1 (2021), 745–759.

[21] N. Saha, S. Misra, and S. Bera. 2022. Q-Flag: Qos-aware flow-rule aggregation in software-defined IOT Networks. *IEEE Internet of Things Journal* 9, 7 (2022).

[22] Liang Tan, Yue Pan, Jing Wu, Jianguo Zhou, Hao Jiang, and Yuchuan Deng. 2020. A New Framework for DDoS Attack Detection and Defense in SDN Environment. *IEEE Access* 8 (2020), 161908–161919. https://doi.org/10.1109/ACCESS.2020.3021435

[23] Dan Tang, Chenjun Gao, Wei Liang, Jiliang Zhang, and Keqin Li. 2023. FTMaster: A Detection and Mitigation System of Low-rate Flow Table Overflow Attacks via SDN. *IEEE Transactions on Network and Service Management* , (2023), 1–1. https://doi.org/10.1109/TNSM.2023.3270339

[24] Dan Tang, Dongshuo Zhang, Zheng Qin, Qiuwei Yang, and Sheng Xiao. 2023. SFTO-Guard: Real-time detection and mitigation system for slow-rate flow table overflow attacks. *Journal of Network and Computer Applications* 213 (2023), 103597.

[25] Dan Tang, Zhiqing Zheng, Keqin Li, Chao Yin, Wei Liang, and Jiliang Zhang. 2023. FTOP: An Efficient Flow Table Overflow Preventing System for Switches in SDN. *IEEE Transactions on Network Science and Engineering* , (2023), 1–13. https://doi.org/10.1109/TNSE.2023.3297650

[26] Dan Tang, Zhiqing Zheng, Chao Yin, Bing Xiong, Zheng Qin, and Qiuwei Yang. 2023. FTODefender: An efficient flow table overflow attacks defending system in SDN. *Expert Systems with Applications* , (2023), 121460.

[27] Wenxiu Zhang, Shan Jing, and Chuan Zhao. 2023. A Survey of SDN Data Plane Attacks and Defense Strategies. In *Proceedings of the 2023 2nd International Conference on Networks, Communications and Information Technology.* ACM, Wuhan, China, 59–65.