International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-11 Issue-11, October 2022

# IP Source Lockdown to Detect and Mitigate Multi-Destination, Multi-Port, Multi-Protocol DDoS Attacks in SDN

**Jitendra Patil, Vrinda Tokekar, Alpana Rajan**

***Abstract:** Distributed Denial of Service (DDoS) attack is not a new attack and remains a challenging task. It has already been addressed by researchers and a lot of work has been done in this direction. Most of the work in Software-Defined-Network (SDN) environment focused on legacy DDoS attacks where targets are end servers. Legacy DDoS attack traffics are associated with a single destination and mostly the solutions are around this characteristic. In the case of SDN, the target is SDN controller plane whose overcharging brings the network to a complete halt. An attacker can achieve this by customizing Multi-Destination, Multi-Port, Multi-Protocol DDoS (MMMD) attack traffic to force the data plane to push more messages to the controller plane. In this paper, we have considered MMMD attack traffic which is just like normal traffic but has the potential to paralyze the complete SDN based networking infrastructure. In the contribution of this work, we have created MMMD traffic and proposed a model named "Simple, Lightweight DDoS Detection and Mitigation model in Software Defined Network" (SLDDM) to combat MMMD traffic in the SDN environment. SLDDM is based on the implementation of IP source-lockdown in SDN environment to detect and mitigate malicious traffic originating from spoof/legitimate IPs. The proposed model has been evaluated under different scenarios and compared with standard models in the literature. SLDDM brings down average response time in establishing https connections by legitimate hosts under attack scenario from 31 seconds to 0.054 seconds. It has been evaluated that the SLDDM keeps the SDN controller healthy and responsive to legitimate hosts under attack conditions.*

***Index Terms:** SDN Controller; DDoS; spoof IP; TCP-SYN; IP Source-Lockdown*

## I. INTRODUCTION

Software Defined Network (SDN) is a concept developed by researchers at Stanford University to decouple the data and control plane [1][2][3]. The Control plane has a centralized view of the complete network and it can be configured logically by the network administrator [4][5][6].

**Manuscript received on 19 September 2022 | Revised Manuscript received on 28 September 2022 | Manuscript Accepted on 15 October 2022 | Manuscript published on 30 October 2022.**

*Correspondence Author (s)
**Jitendra Patil*,** Department of Computer Engineering, Devi Ahilya Vishwavidyala, Indore (M.P), India. E-mail: jpatil@rrcat.gov.in
**Vrinda Tokekar,** Professor, Department of Information Technology, Institute of Engineering and Technology, Devi Ahilya University. Indore (M.P), India.
**Alpana Rajan,** Department of Computer Engineering, Devi Ahilya Vishwavidyala, Indore (M.P), India.

In SDN networking model, the data and controller plane operate separately. The Data plane maintains flow table entries received from a controller. Data planes are Open v Switch (OVS) in a virtual SDN networking that is licensed under the open-source Apache 2.0 [7]. It is designed to empower huge network automation with centralized logical programmatic control. SDN is a futuristic networking model with centralized control of the complete network. In its initial design, centralized functionality was more considered over security aspects. This enables attackers to saturate SDN controller functionality using DDoS attacks. The aim of DDoS attack on the legacy network was to target web services like potential services offered on the Internet. DDoS attacks on the controller plane of legacy networking were not effective, since the control plane and data plane are coexisted in legacy networking edge boxes (switches, hub, etc) and distributed in the network. But same DDoS attack can certainly saturate the SDN controller resources.

Mostly proposed models available in the open literature can be grouped into

i) Anomaly-based detection

ii) Pattern matching detection.

Anomaly-based detection uses entropy on destination IP/port numbers, Machine Learning, and Neural network techniques to distinguish abnormal network [8][9][10][11][12][13][14]. These techniques can detect a wide range of networking attacks but a generation of false-positive detections are always linked. Pattern matching detections are based on matching flow patterns for a certain type of attack and do not link with false-positive alarms [15]. Pattern matching detection is accurate but performs for a certain type of networking flow only. Mostly the solutions are focused on DDoS TCP flood attacks only. There is hardly any solution in the open literature, which may effectively combat, the Multi-Destination, Multi-Port, Multi-Protocol (TCP/UDP/ICMP) DDoS (MMMD) attack traffic in SDN environment. In this paper, we have proposed SLDDM model to detect and mitigate MMMD attack traffic in SDN environment. In addition to MMMD, it gives guard against spoof IP/MAC addresses and is able to filter malicious traffic from flash traffic.

### 1.1 Motivation of the study

The saturation of SDN controller resources by a traditional DDoS attack can easily hamper the flow of the complete network. Traditional DDoS attacks which are designed for legacy networks are potential attacks in SDN environment by misusing the basic functionality of the data plane.

Traditional DDoS attacks normally target services listening on TCP ports and thus most of the research work to protect SDN controllers in the literature considered TCP traffic only. This could be one of the probable reasons that available solutions in the literature focus on traditional DDoS attacks in SDN environment. But to saturate the SDN controller resources, any type of networking traffic will usually suffice. Today, there are many freeware tools and ready-to-use libraries which may empower the attacker to generate MMMD attacks targeting reward-winning SDN controllers. We have also studied that models proposed by the researchers introduce significant load on controller and focused on TCP-SYN flooding attack considering web services as the end service [16][17][18]. A practically implementable, lightweight, and deployable solution to nullify MMMD attacks traffic along with low load on the controller was the motivational factor for us to design SLDDM.

### 1.2. Research Contributions

Spoof IP/MAC in combination with MMMD and flash traffic further adds complexity in designing a solution. To address the MMMD traffic and related issues of spoof IP/MAC in presence in flash traffic for SDN networking, SLDDM has been designed. It operates using two modules, the first module named “Lock inPORT with Source IP (LISP)” and the second module named “Detect and Mitigate DDoS attack from Legitimate IP (DMDLI)”. LISP provides prevention from traffic generated by using spoof IPs/MAC. It initiates an action in advance to mitigate the major source of DDoS attack i.e. traffic generated by using spoof source IPs/MAC. This is made possible by locking the source IP with inPORT of the Open Flow Switch (OFS) on receiving the first packet by the controller from the host. LISP comes into action as soon as the host comes up in the network. More details on LISP are depicted in Section 4.1. MMMD attack may also be generated by using compromised legitimate hosts by using legitimate IP and legitimate MAC. Second module DMDLI takes care of attacks generated by compromised legitimate hosts. More details on DMDLI are depicted in Section 4.2. Performance of SLDDM compared with literature models and found noteworthy for deployment in the large production network. There are no solutions available in the author’s knowledge that can detect and mitigate MMMD attack traffic in SDN environment.

### 1.3 Organisation of the paper

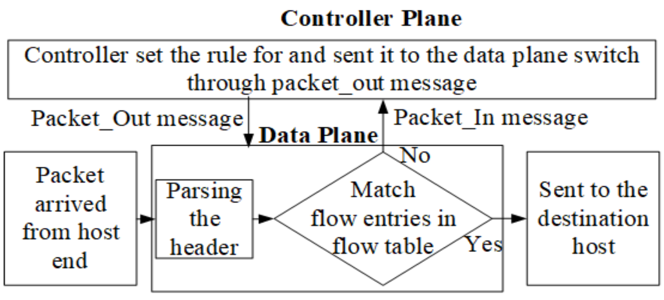
This paper has been organized into seven sections. Section 2 provides background on working of SDN and DDoS impact on default SDN configuration. In Section 3, we have analyzed the different proposed models by various researchers to combat DDoS attacks in SDN environment. Section 4 discusses the functionality of the proposed model SLDDM, its performance under no-attack and under-attack scenarios. In section 5, SLDDM has been compared with state-of-art models in the literature. Discussion and improvements on the proposed model are given in section 6. Section 7 concludes this paper with future enhancements.

## II. SDN AND DDoS

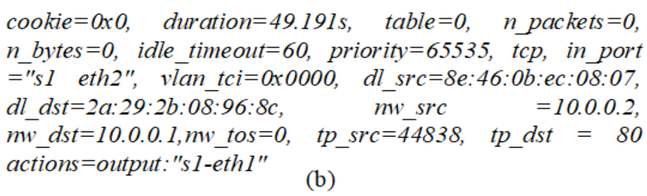
### 2.1 SDN

The working of SDN is depicted in Fig. 1(a) where the controller and data-plane operate in a separate box. Data planes are OpenFlow (OF) switches and maintain forwarding flow table entry associated with hard/idle timeout. Populating forwarding flow table entries in the data plane are done by the controller. All the communication between the controller and data plane takes place over OF protocol. OF defines the communication protocol in SDN environments that enables the SDN Controller to directly interact with the data plane [19]. When a new packet is received by the data plane and doesn’t match with existing forwarding flow table entries, the data plane encapsulates the received packet in OF packet_in message and sends it to the controller plane. The controller plane takes appropriate action and pushes forwarding flow table entry to the data plane over OF packet_out message [20, 21]. Once flow entry is installed in the data plane, all subsequent packets from hosts flow through the data plane and do not contact the controller until hard timeout or idle timeout encounters.

Sample flow table entry of TCP port 80 is shown in Fig. 1(b). Analysis of sample flow table entry at data plane exposes that data plane can be forced to send packet_in message to controller either by changing one or more parameters in the source packet. These parameters may be nw_src (source IP, srcIP), nw_dst (destination IP, dstIP), tp_src (source port number, srcP), tp_dst (destination port number, dstP) for TCP and UDP traffic. And for ICMP, it may be done by changing scrIP and dstIP.


Controller Plane
Controller set the rule for and sent it to the data plane switch through packet_out message
Packet_Out message
Packet_In message
Data Plane
No
Packet arrived from host end
Parsing the header
Match flow entries in flow table
Yes
Sent to the destination host

(a)


*cookie=0x0, duration=49.191s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=65535, tcp, in_port="s1 eth2", vlan_tci=0x0000, dl_src=8e:46:0b:ec:08:07, dl_dst=2a:29:2b:08:96:8c, nw_src =10.0.0.2, nw_dst=10.0.0.1,nw_tos=0, tp_src=44838, tp_dst = 80 actions=output:"s1-eth1"*

(b)

**Fig. 1. a) Functioning of SDN, b) Sample Flow table entry at Data Plane**

### 2.2 DDoS attack on SDN

To simulate the effect of DDoS attack on SDN an experimental setup has been created using Virtual Machine (VM) configured with 2 cores, 3 GB RAM, and powered with Ubuntu 14.04 over the base machine as i5-7400 @3.0 GHz 4 cores, 8 GB RAM. Fig. 2, depicts the attack scenario simulated over experimental setup. It has been observed that in presence of TCP-SYN flooding DDoS attack, SDN controller does not leave with enough computing resources to process any legitimate request [22]. In the case of SDN, DDoS saturate the functionality of SDN controller and subsequently, underlying networking infrastructure gets paralyzed. This way DDoS attacks may be launched to saturate the processing capacity of the data plane and controller plane [23, 24].

Results obtained by using experimental setup (please refer Fig. 2) to simulate TCP-SYN flooding DDoS attack are shown in Fig. 3. Attack traffic was generated by using hping3 and scapy under Mininet v.2.3.0d5 environment [25, 26]. Mininet is a popular SDN emulation tool and is mostly adopted by researchers to study SDN performance and security issues [27]. To analyze the CPU usage of the controller, TCP-SYN flooding traffic has been generated for different Packets Per Seconds (PPS) varying from 10 PPS to 2000 PPS. It is observed that the CPU usage of the controller under DDoS attack saturate at 1000 PPS. In view of observations made in Fig. 3, it is clear that the rate of packet_in messages, reaching the controller badly affects the performance of the controller. This eventually hampers the overall SDN-based networking infrastructure and end servers like webserver etc.

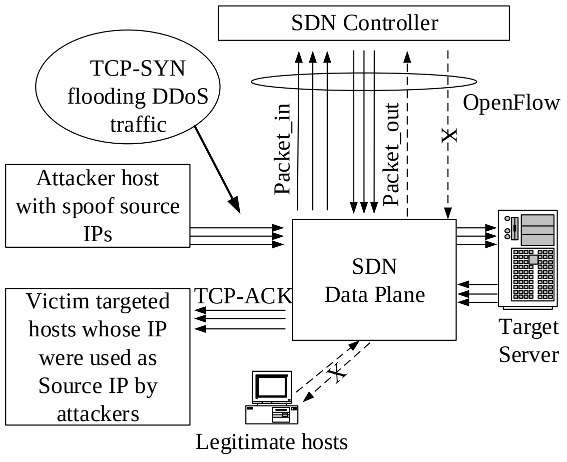

SDN Controller
TCP-SYN flooding DDoS traffic
OpenFlow
Packet_in
Packet_out
X
Attacker host with spoof source IPs
SDN Data Plane
TCP-ACK
Victim targeted hosts whose IP were used as Source IP by attackers
Target Server
X
Legitimate hosts

**Fig. 2. SDN controller under flooding attack**

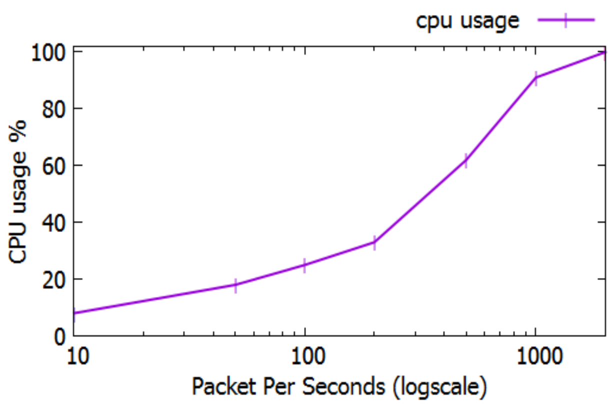


**Fig. 3. Effect of PPS on SDN controller CPU usage without deployment of any security model.**

## III. RELATED WORK

To mitigate the effect of DDoS attacks, many researchers have proposed various mechanisms to keep the SDN controller healthy and responsive to legitimate hosts. Following are some notable research studies to combat this problem. Mota el at. use Self-Organizing Maps (SOM) a machine learning technique for DDoS detection in SDN environment [28]. SOM model was proposed by Ostermann el at. [29]. Average and growth of various parameters extracted from flow table entries in OF switches were used to train SOM model. However, this work did not discuss the mitigation process and also the accuracy of detection is based on the training data set. This solution also required fetching flow table information at a high frequency which may affect the ongoing functionality of the data plane. Shin et al. has proposed AVANT-GUARD, a solution implemented at the data plane to detect and mitigate TCP-SYN flooding attack [30]. During the establishment of a TCP-three-way connection, the data plane switch act as a SYN proxy. Upon successful TCP-three-way connection, the data plane switch informs the controller that flow entry for this connection may be generated. AVANT-GUARD may be acceptable in general scenarios but it introduces long delays in establishing new TCP connections from legitimate hosts. This approach may invite buffer saturation attack at data plane and performance of AVANT-GUARD may be degraded [31]. Moreover, as per the original SDN architecture, nothing should be added to the data plane otherwise data plane will slowly swift towards existing legacy networking switches. There may the instances where attackers complete a TCP-three-way connection and go silent afterward to consume resources at the server end. Limitation in AVANT-GUARD about buffer saturation attack has been removed to some extent in LineSwitch [32]. LineSwitch followed the same mechanism to use proxy at the data plane level but with probabilistic blacklisting. Probabilistic blacklisting helped in managing the state of ongoing connections using less memory. However, spoof traffic generated by UDP and ICMP is not addressed by LineSwitch. SPHINX, proposed a solution to detect SYN flooding attacks which monitors the rate of packet-in messages reaching the controller [33]. If the rate of packet_in messages crosses the threshold value set by an administrator, it raises an alarm. This framework may generate false positive alarms [34]. Moreover, there is no work on mitigation to address ICMP and UDP spoof traffic.

Selective packet inspection to detect DoS flooding attack using SDN, combine technique for TCP-SYN flooding attack detection and mitigation [35]. This mechanism works only for TCP-SYN traffic. Secondly blocking spoof IP may not be effective, since a new set of spoof IPs may be used.

OPERETTA, an OpenFlow-based Remedy to mitigate TCP SYN FLOOD attacks against web servers. OPERETTA module is implemented in a controller and acts as an SYN proxy to the TCP connection generated by the client machine [36]. Upon successful verification of the TCP connection by OPERETTA module, it drops the connection by sending RST to the client machine and installing flow entry in the flow table. The client machine tries again to establish TCP connection, this time it does not need to negotiate with the controller. This model rejects the first attempt to establish TCP-three-way and forces the client machine to reinitiate the connection. This functionality of OPERETTA introduces a delay in establishing TCP connections. OPERETTA is fully deployed at the controller and acts as end server to verify each new TCP connection, overload the controller CPU under no attack condition. To overcome the limitations in OPERETTA, SLICOTS, a model which is based on watching the TCP-SYN traffic in the controller and decides whether the connection is valid or not is proposed by Mohammadi et al., [37]. Unlike OPERETTA, SLICOTS does not send SYN-ACK packet to the client machine, rather it maintains pending_list_table with status field along with other information like source MAC, destination MAC, source TCP port, and destination TCP port.

When a new TCP connection is received by the controller, SLICOTS extracts source MAC, destination MAC, source TCP port, and destination TCP port information and marked the status of this entry as "SYN". At the same time, SLICOTS publishes a temporary flow table entry with 3 seconds hard timeout which enables TCP connection request packet reaches to the server. It is expected that 3 seconds are enough to get SYN-ACK reply from the server end [38]. Upon receiving SYN-ACK from the server end, SLICOTS changes the value of status field to SYN-ACK and then finally waits for ACK packet from the client machine. In case ACK packet is received by SLICOTS, entry is removed from the pending_list table and install forwarding rules with a normal hard timeout. Why ACK packet reaches to controller though forwarding rule with a hard timeout of three seconds already in the flow table, is not clear. The mitigation process is based on MAC address instead of IP but blocking on MAC, would also block legitimate traffic. A model named SAFETY has been proposed to detect and mitigate TCP-SYN Flooding attacks by Kumar e. al., [39]. This model proposes two major components: a detection unit and a mitigation unit. The functionality of the detection unit is based on values in the counter. Upon TCP-SYN packet comes to the controller, counter increment by one, upon SYN-ACK, counter decrement by one, and upon RST, decrement by one. This way counter gets maintained in the database along with associated Destination IP, dpid (OVS switch id), and inPORT information. There is a threshold (Δt) to activate entropy calculation on this database. If K time's entropy value exceeds a certain set value, the presence of an attack scenario is declared. Once the presence of the attack scenario is declared, the mitigation unit gets activated. It identifies the victim and attacker IP and blocks the traffic at edge switches. SAFETY assumed the attacker traffic always towards a single destination IP address using TCP-SYN only. Spoof traffic from UDP and ICMP has not been considered. In practical scenarios, multiple destination attacks are possible, particularly in SDN environment to saturate the SDN controller. Deep learning approaches are being preferred by researchers to detect intrusion and botnet connections in the network. A deep learning approach to detect intrusion in the network has been proposed by. Vinayakumar R. et al.[40]. In this work deep neural network has been explored and hyper-parameter selection methods were used with KDDCup 99 dataset. Two-level deep learning framework has been explored to detect the presence of botnets in the Internet of Things underlying network [41]. A botnet is a potential mechanism to launch DDoS attacks. To prevent the prorogation of the botnet army, an intelligent botnet blocking approach in software-defined networks using honeypots has been proposed by Jafari et al. [42]. The aim of this approach is to establish the relationship between bots and bot-loaders and blocked the loader to collapse the complete botnet strategy. Ahuja et al. proposed an approach based on a hybrid of two machine learning (Support Vector classifier + Random Forest) techniques and achieved an accuracy of 98.8% [43]. Features extraction and dataset creation were the major contributions of this work. However, testing on MMMD attack traffic on this model has not been depicted. Proposed models discussed in this section, are focused on verification of TCP three-way connection to combat TCP flooding attacks. But in the real scenario, any type of IP packet traffic may be crafted to saturate the resources available with SDN controller. Taking this into consideration, SLDDM model has been designed to filtered MMMD type of attacking traffic. It is designed to protect SDN underlying network from spoof IP/MAC addresses and is capable to filter malicious traffic from flash traffic. It supports MMMD attack traffic with a low requirement of CPU usage and no false positive alarms, making it unique and adaptable to real SDN networks.

## IV. PROPOSED MODEL – SLDDM

This model is designed to safeguard the overall SDN-based large networking infrastructure in an organization. It has two modules named "Lock inPORT with Source IP" (LISP), and "Detect and Mitigate DDoS attack from Legitimate IP" (DMDLI). LISP helps in mitigating MMMD attacks generated by using spoof IPs and DMDLI helps in mitigating MMMD attacks generated by legitimate IPs.

In designing and implementation of SLDDM, we have studied and performed the following tasks:

i) Discover the vulnerabilities under which packet_in requests may be generated from the data plane.

ii) Possibility of crafting a packet to overcharge the SDN normal functionality.

iii) Designed a lightweight solution SLDDM to mitigate the multi-destination spoof traffic generated by using multiple protocols like TCP, UDP, and ICMP.

iv) Examine the effectiveness of SLDDM in protecting SDN controllers and average response time in establishing http connection by legitimate hosts under different attack scenarios.

v) SLDDM performance has been compared with state-of-arts models SAFETY and OPERETTA available in the literature.

### 4.1 LISP- Lock in PORT with Source IP

It has been assumed that networking switches are connected by trunk ports. Here trunk ports are like normal ports connecting other switches. Malware, Trojans at hosts may generate DDoS attacks which may bring down the complete SDN-based networking to halt. Figure 4 depicts the flow of LISP algorithm.

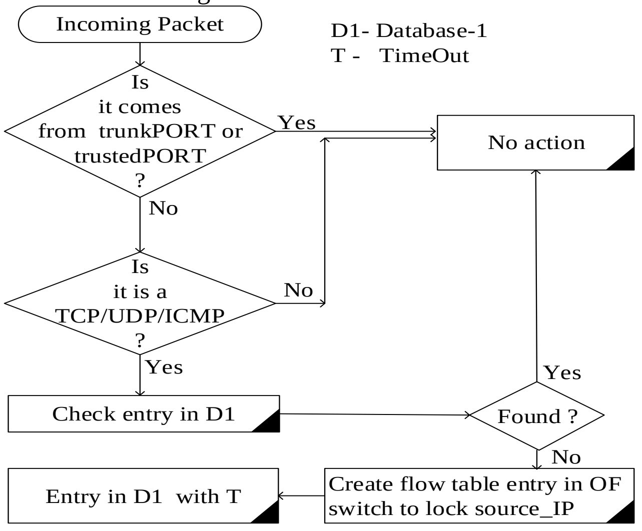

Incoming Packet
D1- Database-1
T - TimeOut
Is it comes from trunkPORT or trustedPORT ?
Yes
No
No action
Is it is a TCP/UDP/ICMP ?
No
Yes
Check entry in D1
Found ?
Yes
No
Entry in D1 with T
Create flow table entry in OF switch to lock source_IP

**Fig. 4. Flow of Lock in PORT with Source IP**

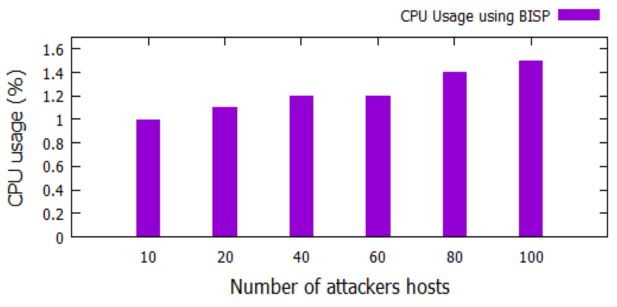


**Fig. 5. CPU usage using LISP under attack**

LISP does not wait for some attack to happen; rather it takes proactive action in advance to lock in PORT with Source IP. This way all traffic generated using spoof IPs gets dropped at the data plane only. LISP helps in conserving the computing resources at the data and controller plane. It monitors TCP/UDP/ICMP packets from hosts connected to in PORTs. And checks its presence in Database-1(D1). D1 maintains src IP, Datapath identifier (dpid) and in PORT information. Datapath identifiers are 64-bit numbers that distinctively identify the instances OF switches in SDN. The presence of this information in D1 is associated with time Out (T); 'T' can be configured as per the requirement of an organizational network. 'T' value may be large for static IPs and maybe less for Dynamic Host Control Protocol (DHCP) assigned IPs. If the incoming packet has a presence in the D1, that signifies this packet would have reached the controller in past. This time reached the controller because of hard/idle timeout is over and needs to create a new flow table entry. In this case, LISP does not take any action because LISP has information that this packet came from locked IP. If the packet does not have a presence in D1, it means this packet has come for the first time or the first packet after time Out 'T'. In this case, LISP locks src IP with in PORT and creates an entry in D1 with timeout 'T'. This entry will remain in the database till 'T' expires. If packets come from trunk ports or trusted ports for servers, it will simply ignore. It is assumed that all trunk ports are connected by switches only and all other ports are connected by hosts. The effect of LISP on CPU usage of the SDN controller has been evaluated on 480 host network under Mininet environment. Fig. 5 depicts the effect of LISP on controller CPU usage where 10 to 100 hosts were made to generate spoof source IP traffic at 500 PPS.

Fig. 5 shows the additional computing load of LISP on the controller CPU in SDN network. Observations have been made that the controller does not get saturated by 100 attacking hosts. The reason could be controller has to process only 100 packets which were the first packet generated by 100 hosts. Average CPU usage is normally about 1% at controller without any traffic and LISP also maintains average usage at about 1%. LISP could achieve this performance because it had to lock only 100 attacker hosts to their IPs in the initial stage and consequently all the malicious traffic generated by these hosts got dropped at the data plane only. It may possible, hosts may have multiple VMs. In that case, LISP may need to be modified to support multiple source IPs. Algorithm 1 – DetectFirstPaket and Algorithm 2 – flowEntryToLock has been designed to achieve the functionality of LISP. Algorithm 1 and algorithm 2, are implemented using python as a module in the POX controller.

To lock the source IP with in PORT by which the host is connected, two flow table entries are used. The first flow entry will allow all incoming packets from in PORT (i.e. from the connected host) to carry the source IP registered by LISP. This flow entry is assigned with higher priority (e.g. 100) than the second flow entry. The second flow entry drops all traffic generated by the host connected by in PORT and assigned with lower priority (e.g. 90). This way, these two flow entries, lock the source IP with in PORT and make the host connected with in PORT to communicate with the registered source IP only.

## 4.2 DMDLI - Detect and Mitigate DDoS attack from Legitimate IP

LISP allows one source IP from each port. This mechanism blocks spoof source IP traffic at the data plane. The possibility of spoof traffic may be generated by legitimate IPs cannot be ignored. The overall aim of attackers are to saturate controller processing capability by generating heavy packet_in traffic to the controller. DMDLI is designed to handle such spoof traffic, its flow is depicted in Fig. 6.

**Algorithm 1 –DetectFirstPaket**
Input: packet_in(Pi), _DB($SRC_{IP}$ , dpid, inPORT , T)
Output: Generate inPORT-$SRC_{IP}$ locking at data plane

```
1:  if (Pi .inPORT == trunkPORT) ∨ (Pi.inPORT ==
    trustedPORT) then
2:    noACTION
3:  else
4:      if (Pi.TYPE == TCP) ∨ (Pi.TYPE == UDP) ∨ (Pi.TYPE
      == ICMP) then
5:      if (Pi.SRC_IP ∈ _DB(SRC_IP, inPORT) then
6:          noACTION
7:      else
8:          flowEntryToLock()
9:      endif
10:   else
11:     noACTION
12.   endif
13: endif
```

**Algorithm 2 –** flowEntryToLock
Input: dpid, inPORT, $SRC_{IP}$, _DB($SRC_{IP}$ , dpid, inPORT , T)
Output: Generate inPORT-$SRC_{IP}$ locking at data plane

```
1: procedure flowEntryToLock (dpid, inPORT, SRC_IP, _DB)
2:   ovs-ofctl add-flow <dpid> priority=100, in_port=inPORT, ip, srcIP = SRC_IP, action=normal
3: ovs-ofctl add-flow <dpid> priority=90, in_port= inPORT, ip, action = drop
4: _DB (SRC_IP ,dpid, inPORT , T) ← _DB(SRC_IP , dpid, inPORT , T)
5: end procedure
```

/* ovs-ofctl is a command line tool for monitoring and administration of flow table entries. Statement at #2 and #3 in Algorithm2 are used to create flow entry with different priorities in OpenFlow switch. */

It is assumed that trunk ports and servers are connected to trusted data plane ports. Therefore, DMDLI does not put a check on trunk ports and all ports by which servers are connected. DMDLI analyses the packet coming to the controller, if it is from the trunk or trusted ports, ignores it else checks for TCP/UDP. If it is a TCP/UDP and has a presence in Database-2 (D2). D2 also maintains the same information as D1 but is associated with different timeOut tags i.e. ‘tm’. DMDLI counts the number of occurrences for this source IP (nSIP) in D2. If it is greater than threshold K, blocking of source IP at data plane gets initiated, follows by dropping off all ongoing connections from this IP if any. And if nSIP in D2 is found lesser than threshold K, DMDLI creates an entry of that source IP in D2. ‘K’ is a threshold constant to sense the nature of traffic from particular IP whether it is malicious or not. The availability of information about nSIP in D2 is getting controlled by the timeOut parameter ‘tm’. The database maintains only fresh entries not older than ‘tm’ value. Algorithm 3 – “Detect and Mitigate DDoS attack from Legitimate IP” implements the flow of DMDLI. Algorithm 3 has been implemented in Python as a module in the POX controller.

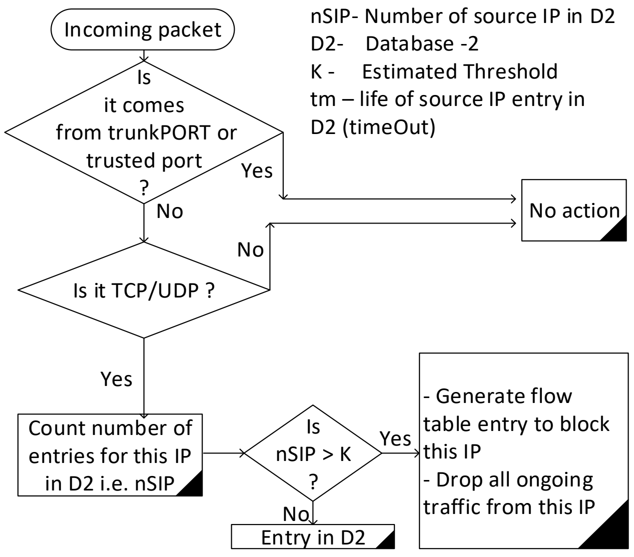

Incoming packet
nSIP- Number of source IP in D2
D2- Database -2
K - Estimated Threshold
tm – life of source IP entry in D2 (timeOut)
Is it comes from trunkPORT or trusted port ?
Yes
No
No action
No
Is it TCP/UDP ?
Yes
Count number of entries for this IP in D2 i.e. nSIP
Is nSIP > K ?
Yes
No
Entry in D2
- Generate flow table entry to block this IP
- Drop all ongoing traffic from this IP

**Fig. 6. Detection and Mitigation mechanism for DDoS attacks generated from Legitimate IPs**

**Algorithm 3 - Detect and Mitigate DDoS attack from Legitimate IP**

Input: packet_in(Pi), _DB($SRC_{IP}$ , dpid, inPORT )
Output: attack free environment

```
1:  if (Pi.inPORT == trunkPORT) ∨ (Pi.inPORT ==
    trustedPORT) then
2:     noACTION
3:  else
4:     if (Pi.TYPE == TCP) ∨ (Pi.TYPE == UDP) then
5:        Count(Pi.SRC_IP) ← _DB(SRC_IP ,dpid, inPORT)
6:        if (Pi.SRC_IP.count > K) then
7:           blockIP(SRC_IP, dpid, inPORT)
8:        else
9:           _DB(SRC_IP ,dpid, ,inPORT, tm) ← SRC_IP , dpid,
             inPORT, tm
10:       endif
11:    else
12:       noACTION
13.    endif
14: endif
```

## 4.3 Time and Space Complexity of algorithms

The time complexity of algorithm-1 doesn’t contain any loop structure. It has some if-else statements and calls sub-procedure at line number 8. Sub procedure which is algorithm-2 also doesn’t involve any loop structure. Thus, the overall time complexity of algorithm-1 is O(1). In presence of some local variables and if-else statements in algorithm-1 and algorithm-2 which require a constant amount of space incur O(1) space complexity. Algorithm-3 operates independently of algorithm-1 and algorithm-2 is the implementation of the DMDLI component of SLDDM. In absence of any loop structure and no sub-procedure, algorithm-3 contributes O(1) time complexity. There is no ‘n’ space variable, only some local variables which require constant space contributes O(1) space complexity.

## 4.4 Estimation of timeOut ‘tm ’and threshold constant ‘K’

Parameter timeout ‘tm’ in DMDLI helps in maintaining the fresh entry in D2. With LISP in existence, a host can send traffic only from single IP. Packets reaching the controller by a host may be because of a new flow request or a hard/idle timeout in the flow entry table occurred. If more than 10 PPS have been observed in ‘tm’ time, it may be considered malicious traffic. 10 PPS with the same source IP where spoof IPs are already blocked by LISP, a request for 10 new connections in a second is highly unreasonable. Less than 10 PPS may also be considered to classify malicious traffic but for the safe side to avoid any false detection threshold ‘K’ has been fixed at 10 PPS. In the experimental topology with 480 hosts, 50 hosts were generating normal traffic and 10 hosts were generating spoof traffic. With a ‘tm’ of 1 second and ‘K’ with 10 PPS, all the 10 malicious hosts were detected and got blocked in an average time of 0.79 seconds.

## 4.5 Performance analysis

To evaluate the performance of the proposed SLDDM model, a network setup has been created in a virtualized environment using two VMs. These VMs are operating on different base machines connected over 1 Gbps Ethernet network. Fig. 7 depicts the experiment setup topology interconnections [44]. VM1 has been used to run the POX controller and on VM2, SDN networking linear topology has been implemented using Mininet as an emulator. Parameters and their values used in emulated setup are shown in Table 1. Some control/ link layer discovery protocol (LLDP) packets that get exchanged between the data plane and controller plane intermittently, are not considered in the emulated setup. Performance analysis of SLDDM has been done in two parts Part I and Part II.

***Part I:*** To evaluate the impact of SLDDM on controller CPU and memory usage. In this case, no malicious traffic was pushed on SDN experimental setup. Normally hard timeOut of flow entry is 30 seconds, therefore graphs shown in Fig. 8 and Fig. 9 were plotted on a scale of 30 seconds. HTTP connections were generated using wget. Wget is a utility to fetch html pages from the webserver. CPU and memory activity by the POX controller (python2.7) has been measured by the Python “psrecord” graphical tool.

i) SDN Controller CPU usage by HTTP connections initiated by 20 legitimate hosts without deployment of SLDDM is shown in Fig. 8. Load on controller CPU ramped to 18% for a fraction of time, this peak was during the establishment of 40 forwarding flow entries (i.e 20 pairs) for 20 HTTP connections. After the installation of forwarding flow entries, no packets of normal traffic hits the controller till idle/hard timeout expires or request for new connection and thus rest of the time load on CPU observed in between 2-3%.

ii) Controller CPU and memory usage by HTTP connections initiated by 20 legitimate hosts with the deployment of SLDDM is shown in Fig. 9. Controller CPU usage, in this case, is almost the same as without deployment of SLDDM. It indicates that SLDDM does not significantly overload the resources available with SDN controller.

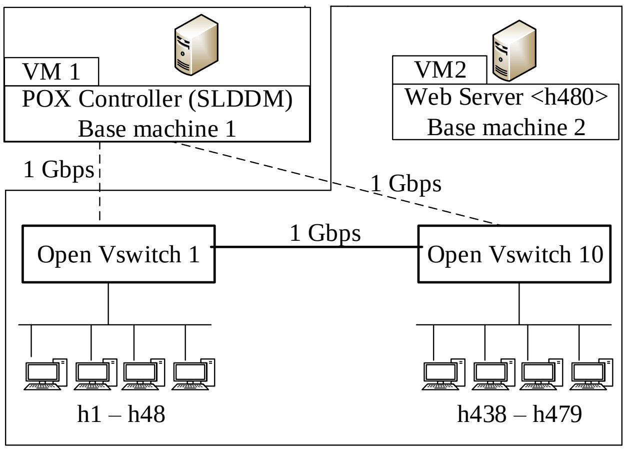


**Fig. 7. Experiment setup network topology**

**TABLE 1. Setup Parameters To Perform Performance Analysis**

| Components | Software/ Configuration |
|---|---|
| Base machine | i5-7400 @3.0 GHz 4 cores, 8 GB RAM - 2 numbers |
| VMs | 2 numbers |
| VM configuration | 2 cores, 3 GB, Ubuntu 14.04 |
| Victim/legitimate/Attacker OS | Ubuntu 14.04 |
| Attacker traffic | 0-2000 PPS by 50 hosts |
| Genuine traffic | 10-50 PPS by 20 hosts |
| Network | All links 1 Gbps |
| Total Hosts | 480 numbers |
| Open vSwitch (OVS) | 10 numbers (v2.9.2) |

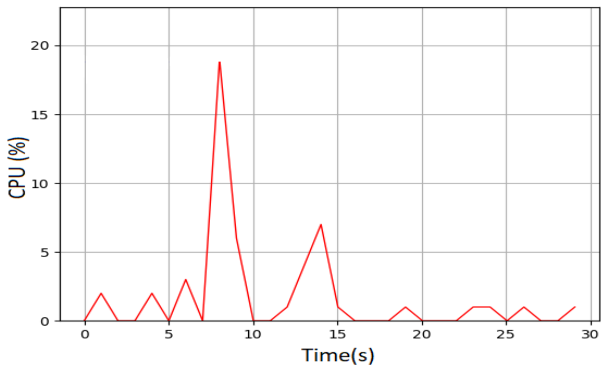


**Fig. 8. SDN controller CPU usage during HTTP connections generated by 20 legitimate hosts without SLDDM and no on-going attack**

***Part II:*** The impact of SLDDM on SDN controller CPU usage and response time to the legitimate hosts, in establishing http connections, under heavy flooding attack, initiated by using TCP-SYN, UDP, and ICMP packets have been evaluated. Attacks are simulated using spoof IPs with random source port numbers using hping3 and scapy. The effectiveness of SLDDM has been evaluated under the following six cases:

*i) Average CPU usage against TCP-SYN flood attack*

Fig. 10 shows that after POX controller is integrated with SLDDM, CPU usage of the controller does not get overloaded and remain almost near to the no-attack condition (i.e ~ 2 – 3%)

*ii) Established time experienced by legitimate hosts to connect web server during TCP-SYN flood attack.*

As per the graph shown in Fig. 11, legitimate hosts could establish connection to the webserver in about 0.06 seconds.

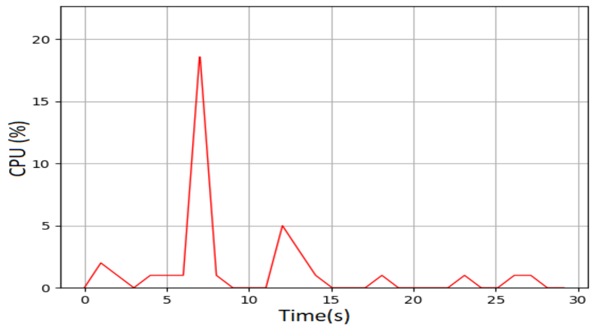


**Fig. 9. SDN controller CPU usage during HTTP connections generated by 20 legitimate hosts with SLDDM and no on-going attack**

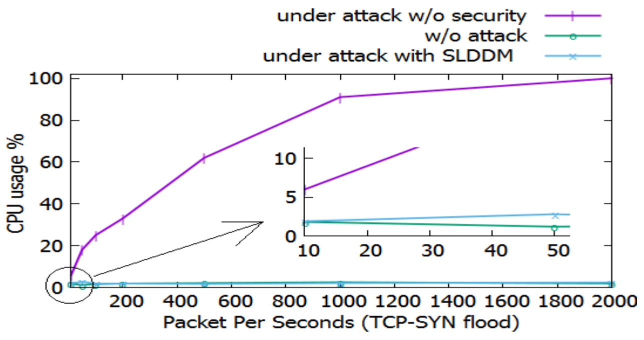


**Fig. 10. Controller CPU usage against TCP-SYN flood in presence of SLDDM**

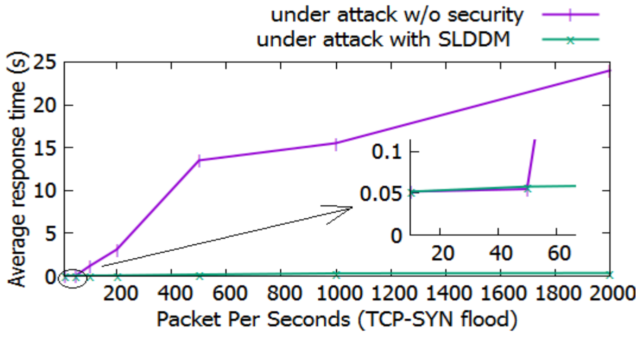


**Fig. 11. Response time to legitimate hosts under TCP-SYN flood attack**

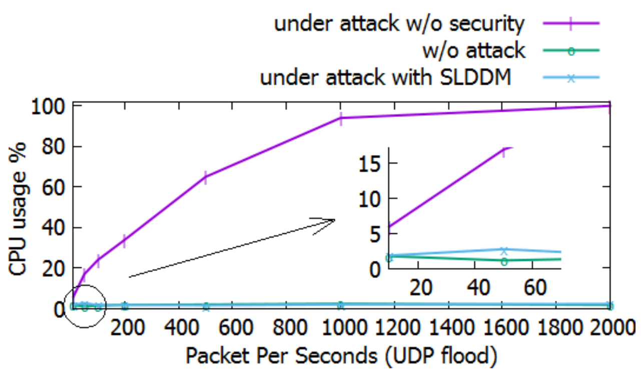


**Fig. 12. Controller CPU usage against UDP flood attack.**

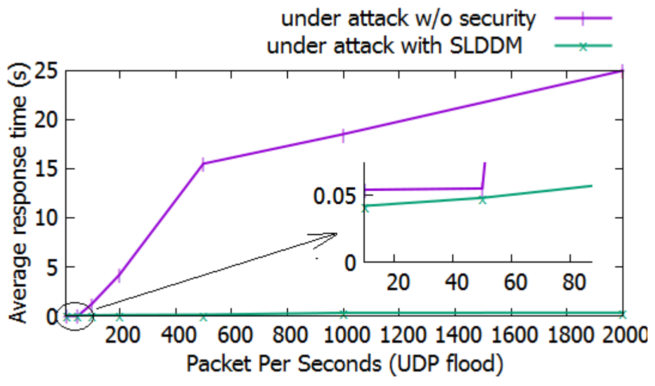


**Fig. 13. Average response time to legitimate http client hosts under UDP flood attack**

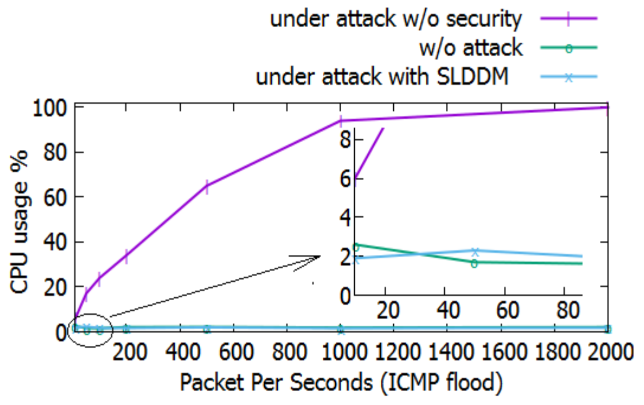


**Fig. 14. Controller CPU usage against ICMP flood attack.**

*iii) Average CPU usage against UDP flood attack*

As it can be seen in Fig. 12 that the controller CPU usage using SLDDM under UDP flood attack is the same as TCP-SYN flood (i.e ~ 2 – 3%).

*iv) Established time experienced by legitimate hosts to connect web server during UDP flood attack.*

Fig. 13 shows that legitimate hosts have an average established time of 0.06 seconds which is same as in the case of TCP-SYN flood attack.

*v) Average CPU usage against ICMP spoof flood attack*

Fig. 14 shows that the CPU usage under ICMP flood attack is same as in the case of TCP-SYN flood attack.

*vi) Established time experienced by legitimate hosts to connect web server in presence of ICMP flood attack.*

Fig. 15 shows that under ICMP flooding attack, legitimate hosts received an average response in about 0.06 seconds which is the same as in the case of TCP-SYN flood attack.

Results obtained during the analysis of "Impact of SLDDM on SDN controller under different attack conditions" are found to be near to the no-attack condition. This could be because the traffic generated by spoof source IPs are dropped at inPORT only and thus could not reach the SDN controller.

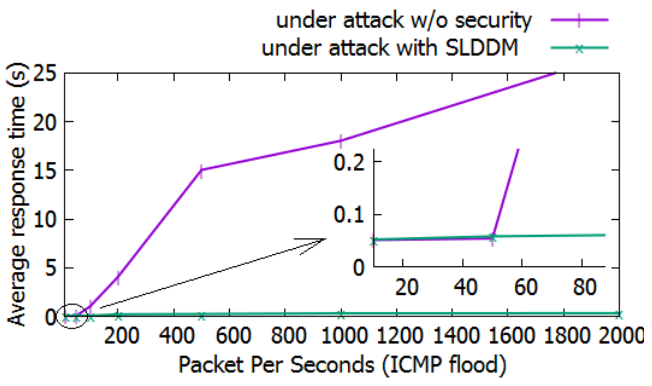


**Fig. 15. Average response time to legitimate http client hosts under ICMP flood attack**

**Table 2. Safety – Attack Setup Parameters**

| Resources | Configuration |
|---|---|
| Web Server | Apache 2.4.7 |
| Victim/Benign/Attacker OS | Ubuntu 14.04 64 bit |
| Victim/Benign/Attacker Configuration | i5-7400 @3.0 GHz 4 cores |
| Victim Service | HTTP Service |
| Victim/legitimate/Attacker OS | Ubuntu 14.04 |
| Attack traffic | 2000-3000 PPS |
| Benign traffic | 30-50 PPS |
| Network | 1 Gbps |

## 4.6 Spoof IP/MAC and flash traffic

LISP locks source IP at inPORT and only the host connected with inPORT is allowed to communicate with registered source IP only. This way all the spoof IPs generated by compromised hosts get dropped at inPORT only. Source MAC address can also be bonded with inPORT but it would have added complexity and increased running time of LISP. Locking the source IP with inPORT also protects against spoofing of source MAC addresses. LISP enables in PORT to allow registered IP only. For example, an attacker generates malicious traffic using a spoof MAC address with registered source IP (otherwise spoof packet will be dropped at inPORT). This configuration of spoof MAC with registered IP will be dropped at inPORT because registered IP is already bound with legitimate MAC and switch does not allow to bind two MAC addresses with single IP.

Normal/flash traffic does not get blocked at inPORT because they are encapsulated with legitimate source IP and legitimate MAC address. This way functioning of LISP which is based on lockdown of source IP with inPORT provides protection against spoof IP/MAC traffic and at the same time allows normal/flash traffic.

## 4.7 Security analysis of SLDDM

In this section, the security aspects of SLDDM have been discussed. It is assumed in this paper that SDN controller is well hardened and powered with local firewalls to provide restricted and authenticated access. Only verified and harden data planes can only communicate with the control plane. Security analysis of the proposed model SLDDM are discussed below:

*i) Connection with SLDDM:* SLDDM does not operate in daemon mode and there is no open TCP/UDP port to establish the connection. It is coded as an integral part of the POX controller.

*ii) Session hijacking while communicating with the data plane:* SLDDM uses secure OF protocol to send forwarding table entries to the data plane, it is a one-way communication i.e from SLDDM to the data plane.

*iii) Malicious traffic to overload SLDDM:* Malicious traffic with a legitimate source IP and spoof source IP is found to be filtered by SLDDM. (please refer to sections 4.5 and 4.6). There are no pre-shared keys or TCP/UDP ports are open which enables the functioning of SLDDM in SDN controller safe and secured.

## V. COMPARISON OF SLDDM MODEL WITH OPERETTA, SAFETY

Results published by SAFETY in Fig. 18, Fig. 19, Fig. 21 ,and Fig. 22 have been referred the basis for comparing the results obtained from SLDDM. Under the same scenarios-1 and scenario-2 as depicted in SAFETY, we tried to keep the same simulation attack setup parameters. Setup parameters and their values are given in Table 2.

***Scenario-1:*** *Network topology consists of 41 hosts, out of which 28 attackers <h1-h28> sending TCP-SYN packets varying from 60 to 260 targeting to the web server using spoof IP of <h31-h40>. There are two legitimate hosts <h29, h30> to analyse the response time from web server <h41>.*

*i) Average delay to establish an HTTP request*

Graphs in Fig. 17 shows that the average response time experienced by hosts <h29, h30> under attack condition by 28 hosts in presence of SLDDM is in the range of 0.054 to 0.38 seconds. The reason could spoof IP got locked at inPORT by LISP and DMDLI could complete its task in about 0.011 seconds. The average response time by SLDDM is much lower than OPERETTA i.e 6 seconds, and SAFETY i.e. 3 seconds (please refer Fig. 16). The average response time with SLDDM is almost near to the no-attack situation.

*ii) Average CPU utilization by the controller*

Average CPU usage with varying malicious SYN packets with SLDDM under scenario-1 is in the range of 1% to 2% (please refer Fig. 19). With the no-attack condition, it is assumed that all 40 hosts are generating normal traffic. CPU usage with SLDDM is near to the no-attack condition and much lower than OPERETTA and SAFETY (please refer Fig. 18).

***Scenario-2:*** *Number of attackers and legitimate hosts are same as scenario-1. In this case, numbers of attacker are varying from 2 to 28 and each attacker sends fix number of 100 SYN packets.*

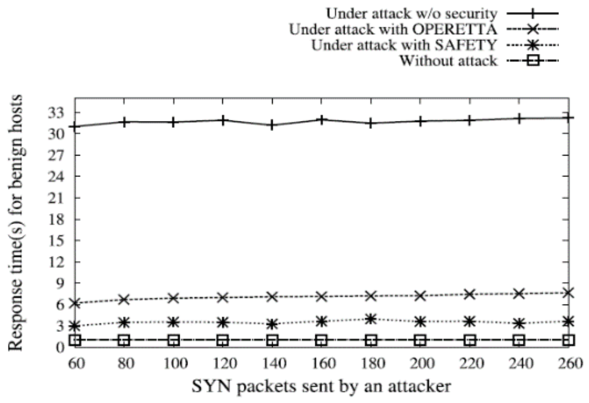


**Fig. 16. Refer Fig. 18 of SAFETY: Average delay to establish an HTTP connection with varying attackers.**

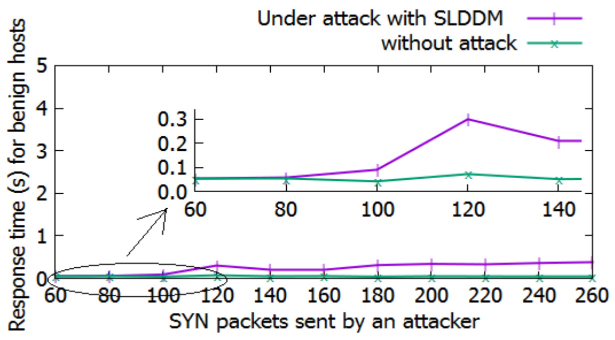


**Fig. 17. Average delay to establish an HTTP connection with varying attackers using SLDDM.**

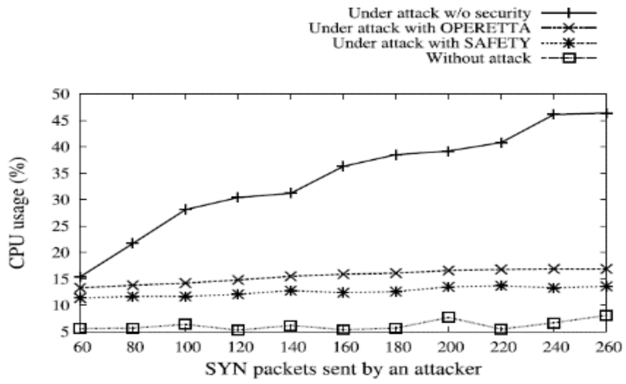


**Fig. 18. Refer Fig. 19 of SAFETY: Average CPU consumption with varying malicious SYN packets.**

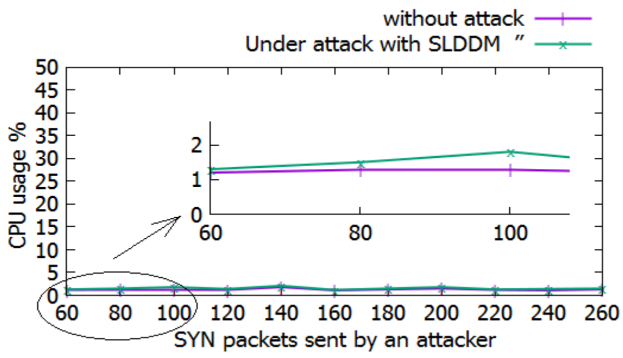


**Fig. 19. Average CPU usage with varying malicious SYN packets in presence of SLDDM**

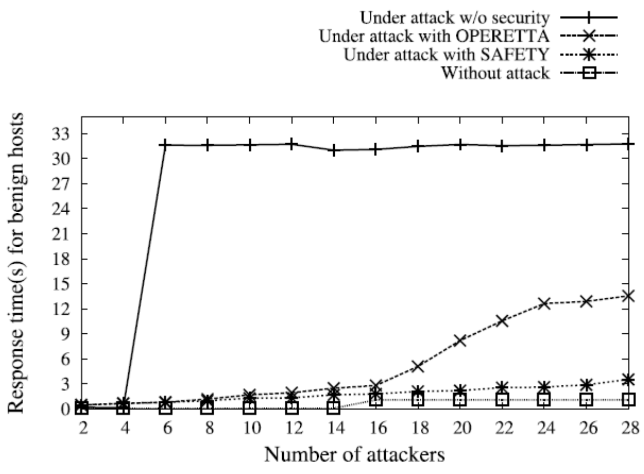


**Fig. 20. Refer Fig. 21 of SAFETY: Average delay to establish an HTTP connection with varying attackers.**

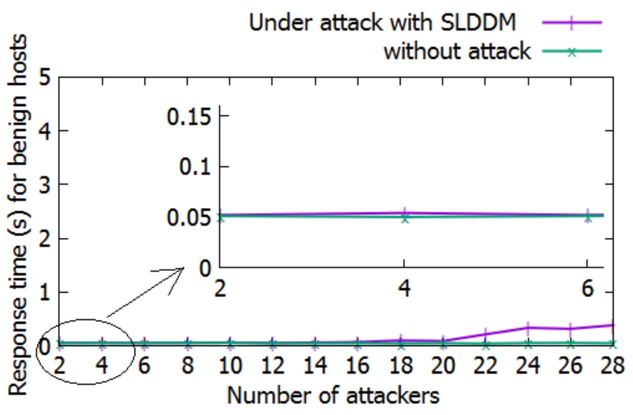


**Fig. 21. Average delay to establish an HTTP connection with varying attackers in presence of SLDDM**

i) Average time to establish an HTTP request

As shown in Fig. 20, the average response time for benign hosts without a security model has been recorded as 31 seconds. Response time for benign hosts with OPERETTA increases with an increase in the number of attackers and the same has been observed in SAFETY but much lower than OPERETA. As per results obtained shown in Fig. 21 SLDDM brings response time further down to almost near to no-attack condition i.e. in the range of 0.054 seconds.

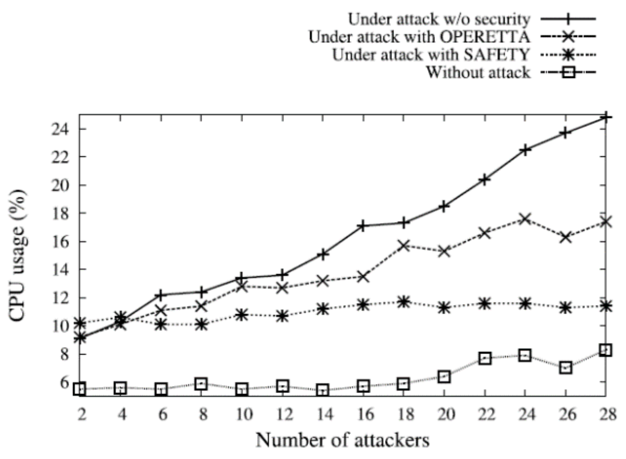


**Fig. 22. Refer Fig. 22 of SAFETY: Average CPU consumption with varying attackers.**

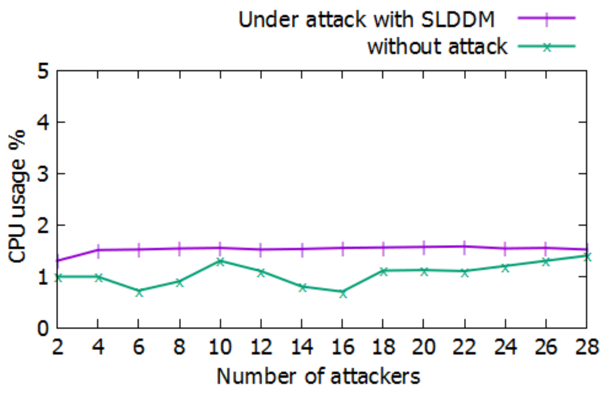


**Fig. 23. Average CPU usage with varying attackers in presence of SLDDM**

*ii) Average CPU utilization by the controller*

Average CPU utilization by the controller obtained by using SLDDM is shown in Fig. 23. Average CPU usage with varying attackers with SLDDM under scenario-2 is about 1.6%. It is almost the same as in the case of varying malicious traffic (i.e. scenario-1). In this case, also, it is assumed that all 40 hosts are generating normal traffic under no-attack conditions. CPU usage by SLDDM is much lower than OPEREETA and SAFETY (please refer Fig. 22).

Overall results obtained by SLDDM are exciting, this may be because the major component used in generating flood attacks are normally by spoof source IPs which gets mitigated by LISP efficiently at the data plane only. And attacks that could be generated by using compromised legitimate hosts, are detected and mitigated by DMDLI with very low computing requirements.

## VI. DISCUSSION AND IMPROVEMENTS

Overall results obtained by SLDDM in section 4.5 are exciting. SLDDM has been evaluated for TCP/UDP/ICMP with multi destination. Major components used in generating flood attacks are normally by using spoof source IPs which get mitigated by LISP (discussed in section 4.1) efficiently at the data plane only. And attacks that could be generated by using compromised legitimate hosts IPs, are detected and mitigated by DMDLI (discussed in section 4.2) with very low computing requirement at SDN controller (please refer Fig. 8 and Fig. 9). Table 3 shows the average delay experienced by legitimate hosts to establish http connection under attack conditions with OPERETTA, SAFETY, and SLDDM. It has been observed that SLDDM outperforms OPERETTA and SAFTY under various scenarios mentioned in section 5.

**Table 3. Http Average Delay (in seconds) to Legitimate Hosts Under Attack Condition**

| Without Security | OPERETTA | SAFETY | SLDDM |
|---|---|---|---|
| 31 | 6 | 3 | 0.054-0.38 |

OPERETTA was designed to combat TCP-SYN FLOOD attacks only. SYN proxy mechanism was adapted in OPERETTA to verify the source IP. Once source IP is verified, forces the user end host to reinitiate the new TCP connection.

This functionality introduces a delay in establishing TCP connections to the server. Furthermore, OPERETTA performs poorly, in case there are no ongoing attacks. Another model named SAFETY is based on maintaining the table of packets involved in establishing TCP three-way connection and after a certain time (Δt) applying entropy for K times to ascertain the attack. This makes it perform better than OPERETTA. SAFETY was designed to support a single victim destination host whereas in a real-world scenario multiple destination victim hosts are possible. Secondly, verified source hosts by SAFETY may also engage in launching attacks. Considering the combination of multi networking parameters to launch attacks, the SLDDM model has been designed to detect and mitigate attacks generated by TCP/UDP/ICMP targeting multiple destinations.

Compromised hosts may complete TCP three-way connection and afterward goes silent. This problem exists in OPERETTA and is partially solved by SAFETY to wait for the first data packet but an attacker may go silent after sending a few data packets and thus the problem will remain unnoticeable in SAFETY also. OPERETTA and SAFETY did not discuss this special case. This problem is analysed in SLDDM and found it may get detected and blocked if the rate of such connections exceeds than threshold K generating from particular IP. If such attacks launch from legitimate IP and legitimate MAC at an extremely slow rate, then such attacks will remain unnoticeable by SLDDM also. As a solution to this problem, we proposed to integrate Artificial Intelligence (AI) or Machine Learning (ML) techniques on a separate server to analyse and detect attacks that are very similar to normal data traffic.

## VII. CONCLUSION

The functionality of the proposed model has been tested against MMMD attacks using TCP, UDP, and ICMP protocols. Under various scenarios, depicted in section 4.5, the performance of SLDDM has been found encouraging. SLDDM has been found a scalable, effective, and lightweight solution as compared to solutions proposed in OPERETTA and SAFETY model. It has been rigorously tested against various combinations of traffic at different PPS and found effective in mitigating the attacks with minimal CPU usage. It brings down the average response time in establishing https connections by legitimate hosts under attack scenarios from 31 seconds to 0.054 seconds. In addition to MMMD attacks, it performed shield against spoof IP/MAC addresses and was able to filter malicious traffic from flash traffic. It could have been further improved by shifting its modules to a separate server. Results obtained in the experimental setup are very near to the no-attack situation which makes it practical and deployable in a real network.

Future work: i) Deployment of SLDDM on actual hardware and performance analysis on real traffic. ii) Augmentation in LISP functionality to handle multiple legitimate IPs connected from a single Ethernet port. iii) Induction of intelligence in DMDLI to identify the malicious traffic generated from compromised legitimate hosts. iv) Attempts will be made to integrate AI and ML like techniques to make SLDDM self-resilience.

## REFERENCES

1. Gorkemli B., Parlakisik A. M., Civanlar S., Ulas A., Tekalp A. M.: Dynamic management of control plane performance in software-defined networks. Proc. IEEE NetSoft Conf. Workshops, Seoul, South Korea, , pp. 68–72 (2016) [CrossRef]
2. Mohammadi R., Javidan R.: An adaptive type-2 fuzzy traffic engineering method for video surveillance systems over software defined networks. Multimedia Tools Appl., pp. 1–16 (2016) [CrossRef]
3. Online document, Open Networking Foundation, http://www.opennetworking.org, Accessed Nov 2019
4. Kreutz D., Ramos F. M. V., Veríssimo P. E., Rothenberg C., Azodolmolky S., Uhlig S.: Software-defined networking: A comprehensive survey. Proc. IEEE, vol. 103, no. 1, pp. 14–76 (2015) [CrossRef]
5. Agarwal S., Kodialam M., Lakshman.: Traffic engineering in software defined networks", Proc. IEEE INFOCOM, Italy,pp. 2211–2219 (2013) [CrossRef]
6. Reza Mohammadi and Javidan R.: An intelligent traffic engineering method over software defined networks for video surveillance systems based on artificial bee colony. Int. J. Intell. Inf. Technol., vol. 12, no. 4, pp. 45–62 (2016) [CrossRef]
7. Online document, Open vSwitch, http://www.openvswitch.org, Accessed: Dec 2019
8. Lim S., Ha J., Kim H., Kim Y, Yang S.: A SDN-oriented DDoS blocking scheme for botnet-based attacks. IEEE International Conference on Ubiquitous and Future Networks (2014). [CrossRef]
9. García de la, Villa A.: Distributed denial of service attacks defenses and OpenFlow: implementing denial-of-service defense mechanisms with software defined networking (2014)
10. Dong P., Du X., Zhang H., Xu T.: A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows. IEEE International Conference on Communications, pp. 1–6 (2016) [CrossRef]
11. Mingxin, W., Huachun, Z., Jia, C., Hongke Z.: An entropy based anomaly traffic detection approach in SDN. Telecommun. Sci., vol. 31, issue 9 (2015)
12. Mousavi, S.M., Hilaire, M.: Early detection of DDoS attacks against SDN controllers. IEEE International Conference on Computing, Networking and Communications (2016) [CrossRef]
13. Shu, Y., Mei, M., Huang, H.: Research on DDoS attack detection based on conditional entropy in SDN environment. Wirel. Internet Technol. 5, pp. 75–76 (2016)
14. Jantila, S., Chaipah, K.: A security analysis of a hybrid mechanism to defend DDoS attacks in SDN. Proc. Comput. Sci. 86, pp. 437–440 (2016) [CrossRef]
15. Trung T. V., Huong T. T., Tuyen D. V., Duc D. M., Marshall A.: A multi-criteria-based DDoS-attack prevention solution using software defined networking. IEEE International Conference on Advanced Technologies for Communications, pp. 308–313 (2015)
16. Online document, Simple HTTP Server Documentation, http://docs.python.org (accessed April 2020)
17. Wang H., Xu L., Gu G.: Floodguard: A DoS attack prevention extension in software-defined networks. Proc. 45th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw., Rio de Janeiro, Brazil, pp. 239–250 (2015). [CrossRef]
18. Mehdi S. A., Khalid J., Khayam S. A.: Revisiting traffic anomaly detection using software defined networking. Proc. Int. Workshop Recent Adv. Intrusion Detection, Menlo Park, CA, USA, pp. 161–180 (2011) [CrossRef]
19. Cejka T., Krejci R.: Configuration of open vSwitch using OF-CONFIG. IEEE/IFIP Network Operations and Management Symposium, Istanbul, Turkey (2016) [CrossRef]
20. McKeown N., Anderson T. E., Balakrishnan H., Parulkar G., Peterson L. L., Rexford J., Shenker S. J., Turner J.: OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74 (2008) [CrossRef]
21. Pfaff B., Lantz B., and Heller B.: Openflow Switch Specification. Version 1.3. 0, Open Networking Foundation (2012)
22. Ali S. T., Sivaraman V., Radford A., Jha S.: A survey of securing networks using software defined networking. IEEE Trans. Rel., vol. 64, no. 3, pp. 1086–1097 (2015) [CrossRef]
23. Wei L., Fung C.: Flowranger: A request prioritizing algorithm for controller dos attacks in software defined networks. Proc. IEEE Int. Conf. Commun. (ICC), London, U.K., pp. 5254–5259 (2015) [CrossRef]

24. Scott-Hayward S., Natarajan S., Sezer, S.: A survey of security in software defined networks. IEEE Commun. Surveys Tuts., vol. 18, no. 1, pp. 623–654 (2015) [CrossRef]
25. Online document, Hping, http://www.hping.org/, Accessed March 2020
26. Online document, Scapy, http://www.secdev.org/, Accessed March 2020
27. Online document, Mininet, http://www.mininet.org, Accessed March 2020
28. Mota E., Passito A., Braga R.: Lightweight DDoS flooding attack detection usingNOX/Openflow. IEEE 35th conference on Local Computer Networks, pp. 408-415 (2010)
29. Ostermann S., Tjaden B., Ramadas M.: Detecting anamalous network traffic with self-organizing maps. Recent Advances in Intrusion Detection, pp. 36-54 (2003) [CrossRef]
30. Shin S., Yegneswaran V., Porras P., Gu G.: AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks. Proc. ACM SIGSAC Conf. Comput. Commun. Security, Berlin, Germany, pp. 413–424 (2013) [CrossRef]
31. Nugraha M., Paramita I., Musa A., Choi D., Cho B.: Utilizing OpenFlow and sFlow to detect and mitigate SYN flooding attack. J.Korea Multimedia Soc., vol. 17, no. 8, pp. 988-994 (2014) [CrossRef]
32. Ambrosin M., Conti M., Gaspari F. De, Poovendran R.: Lineswitch: Efficiently managing switch flow in software-defined networking while effectively tackling dos attacks. Proc. 10th ACM Symp. Inf. Comput. Commun. Security, Singapore, pp. 639–644 (2015) [CrossRef]
33. Dhawan M., Poddar R., Mahajan K., Mann V.: Sphinx: Detecting security attacks in software-defined networks. In: Proc. NDSS, San Diego, CA, USA (2015) [CrossRef]
34. Online document,POX, https://openflow.stanford.edu/display/ONL/POX , Accessed April 2020
35. Chin T., Mountrouidou X., Li X., Xiong K.: Selective packet inspection to detect DoS flooding using software defined networking. Proc. IEEE 35th Int. Conf. Distrib. Comput. Syst. Workshops, Columbus, OH, USA, pp. 95–99 (2015) [CrossRef]
36. Fichera S., Galluccio L., Grancagnolo S. C., Morabito G., Palazzo S.: OPERETTA: An Openflow-based Remedy to mitigate TCP SYN FLOOD attacks against Web servers. Comput. Netw., vol. 92, no. 1,pp. 89–100 (2015) [CrossRef]
37. Mohammadi R., Javidan R., Conti M.: SLICOTS: An SDN-Based Light weight Counter measure for TCP SYN Flooding Attacks. IEEE Trans., vol. 14, No. 2, pp. 487 – 497 (2017) [CrossRef]
38. Paxson M., Allman M., Chu J., Sargent M.: Computing TCP's retransmission timer. IETF, Fremont, CA, USA, Tech. Rep. RFC 6298 (2011) [CrossRef]
39. Kumar P., Tripathi M., Nehra A., Conti M., Lal C.: SAFETY: Early Detection and Mitigation of TCPSYN Flood Utilizing Entropy in SDN. IEEE Trans., Vol. 15, No. 4, pp. 1545 – 1559 (2018) [CrossRef]
40. Vinayakumar R., Alazab M., Soman K.P., Poornachandran P., AlNemrat A.: Deep Learning Approach for Intelligent Intrusion Detection System. IEEE Access vol. 7, pp 41525-41550, DOI: 10.1109/ACCESS.2019.2895334 (2019) [CrossRef]
41. Vinayakumar R., Alazab M., Srinivasan S., Pham Q.; Soman K., Simran K.: A Visualized Botnet Detection System Based Deep Learning for the Internet of Things Networks of Smart Cities. IEEE Transactions on Industry Applications vol. 56(4), pp 4436-4456, DOI: 10.1109/TIA.2020.2971952 (2020) [CrossRef]
42. Jafari L., Mostafavi S., Mizanian K., Jafari E.: An intelligent botnet blocking approach in software defined networks using honeypots. Journal of Ambient Intelligence and Humanized Computing DOI: 10.1007/s12652-020-02461-6 (2020) [CrossRef]
43. Ahuja N., Singal G., Mukhopadhyay D., Kumar N.: Automated DDOS attack detection in software defined networking. Journal of Network and Computer Applications, Vol. 187 (2021) [CrossRef]
44. Book, Oswald, Azodolmolky: Setting Up the Environment in Software-Defined Networking with OpenFlow. IInd Ed., Packt Publishing. Peters, Birmingham, UK: Packt, ch. 5, pp. 1667–1823 (2017).

## AUTHORS PROFILE

**Jitendra Patil** received M.E degree in Computer Science from Devi Ahilya Vishwavidyala, Indore, India in 2013. He is currently working in Computer Division of Raja Ramanna, Centre for Advanced Technology, Indore, India and also pursuing his Ph.D. work. His research interests include Security issues in SDN, Machine learning and High-performance email systems.

**Vrinda Tokekar** received M.E. and Ph.D. degrees in Computer Engineering from Devi Ahilya University, Indore, M.P., India in 1992 and 2007 respectively, and B.E. (Hons.) Electrical and Electronics Engineering from BITS, Pilani in 1984. She is currently Professor and Head of Department of Information Technology, Institute of Engineering and Technology at Devi Ahilya University. She is also Head of University's IT Centre, which maintains University Campus Wide Network and IT services. Her research interests are in Wireless Communication and Mobile Computing Networks, Network Security and QoS issues. She is senior member of Association for Computing Machinery (ACM) and serves as reviewer for many International Journals and IEEE conferences.

**Alpana Rajan** received M.S degree in Software System from Birla Institute of Technology & Science, Pilani, India and Ph.D. in Computer Science from Devi Ahilya Vishwavidyala, Indore, India in the years 2005 and 2014 respectively. She is currently leading the Computer Division in Raja Ramanna, Centre for Advanced Technology, Indore, India. Her research interests include High performance scientific computing cluster, high bandwidth infiniband and Network Security.