March- April 2020
ISSN: 0193-4120 Page No. 911 - 919

SDN-CIFE: SDN-Controller with Instant Flow Entries to Improve First Packet Processing Period

Ramesh Chand Meena
Research Scholar, SET,Poornima University,Jaipur–303905, INDIA
rameshrmz@yahoo.com
0000-0002-2649-1384

Mahesh Bundele
Principal &Director,Poornima College of Engineering,Jaipur-302022, INDIA
maheshbundele@poornima.org

Meenakshi Nawal
Associate Professor, SET,Poornima University,
Jaipur–303905, INDIA meenakshi.nawal@poornima.edu.in

Article Info
Volume 83
Page Number: 911 - 919
Publication Issue:
March - April 2020

Article History
Article Received*: 24 July 2019*
Revised*: 12 September 2019*
Accepted*: 15 February 2020*
Publication*: 13 March 2020*

Abstract

Control and data planes have been separated in SDN technology and OpenFlow protocol supported routers/switches works as packet forwarding equipment in such network. Security monitoring, controlling and flow of data in network are the responsibilities of SDN controller. Initially, OpenFlow router/switch does not have any control & security polices and knowledge to deal with data packet generated by host for forwarding to its destination. In this condition, the first data packet of host is sent to SDN Controller by OpenFlow router/switch for checking, decision, generation of control packets for data packet and making flow entries in OpenFlow/SDN switch flow table for subsequent action on such type of data packets received from a host. These processes at SDN controller and SDN switch level are time intense and first data packet of a host always takes longer time to reach its destination. Here, we have proposed an SDN Controller with Instant Flow Entries (SDN-CIFE) to reduce the forwarding time period of first data packet of the host. This approach makes necessary flow entries in flow table of SDN switch before generation of actual traffic by the host. The approach is designed in python and experimented using mininet network emulator and RYU controller. SDN-CIFE test results have shown that first data packet processing time of a host is reduced more than 83%.

Keywords- *Controller, Instant, Flow, Entry, SDN-CIFE, First Packet, Processing, Period*

1. INTRODUCTION

SDN network handles incoming host packets using flow entries in SDN switch flow table and many fields of packet header are used to form flow entries. SDN switches also keep track of network traffic statistics. SDN technology features like programmability, simplicity &elasticity to network managersare growing its usages in the various enterprise networks and data centers solutions to take its ad advantages. Based on target IP address, conventional systems were forwarding incoming host traffic [15].

Initially, OpenFlow router/switch does not have any control & security polices and knowledge to deal with data packet generated by host for forwarding to its destination. In this condition, SDN router/switch forwards the first packet of a host to controller for checking, generation of control packets for data packet and setting up flow entries intoSDN switchflow table for subsequent action on such type of data packets received from a host. These processes at SDN controller and SDN switch level are time intense and first data packet of a host always takes longer time to reach its destination

March- April 2020
ISSN: 0193-4120 Page No. 911 - 919

Other issue is occupancy of all resources of network in case network has heavy traffic and all such packets need generation and transmission of control packets between SDN controller and switch. This situation may create more delays in generation control packets and in reaching data packet to its destination. SDN switch needs flow rules/entries into its flow table for forwarding data packet to its destination otherwise they will require help of SDN controller to generate control packets. The controller may generate control packets to setup match entries with action into flow table of SDN switch or to forward data packet to outport or for above both works. Some research suggested approaches to reduce the overload of forwarding data packets to controller and generation of control packets.

In [33] authors described that number of control packets create overhead for controller and reduction in control message will reduce the work load for controller. Authors In [35] also suggested a scheme to classify data packets as important or unimportant and it suggested dropping the unimportant data packet for minimizing load in OpenFlow switch and controller. It may need higher processing resources and drop more packets if network has heavy packet traffic. SwitchReduce [36] approach claimed that number of match entries in first SDN switch should be less than dealing packet actions for in packet load and installs flow entries only in first hop SDN switch but this approach requires setting up of wild card flow entries in all in-between hop SDN switches including the last hop SDN switch.

During study of related studies we found that no one approach provides an effective solution for setting up flow entries before transmission of actual data packets of host this could not be accomplished without detection of host and its details such as MAC, IP address & port of connected switch. Most of approaches do not know these details before host starts transmitting packets. In this research, we have proposed an SDN Controller with Instant Flow Entries (SDN-CIFE) to detect host instantly, to setup required flow entries into SDN switch flow table and to reduce forwarding time period of first data packet of a host. The approach is designed in python and experimented using mininet network emulator and RYU controller.

Remaining paper is ordered as follows. Section IIin brief explained SDN background. Section III highlights on topology detection in SDN. Section IV discussed related works. Section V describes proposed SDN-CIFE. Section VI describes implementation of SDN-CIFE. Section VII investigates the performance of SDN-CIFE. Finally, Section VIII concludes work.

II. SDN Background

SDN technology is cost-effective, dynamic, adaptable and manageable making it suitable for active applications & high bandwidth requirements. SDN decouplesnetwork forwarding and control actions and it makes network control& applications directly configurable through programs &underlying infrastructure abstracts network services. The main & essential element of SDN network is OpenFlow protocol and it is promoted by Open Network Foundation to provide southbound interface between controller &SDN switches.

Handshake messages are forwarded to startan OpenFlow connection between controller and switches. An encrypted TCP connection is established between switches and controller to exchange configuration information. OFPT_FEATURE_REQUEST is sent by controller to OpenFlow protocol enabled switch and an OFPT_FEATURE_REPLY message is generated by switch to establish OpenFlow connection. The switch forwards its unique identifiers details like MAC addresses of active switch ports & datapath_id of switch. Presence of switch is discovered during handshaking but interconnections in forwarding devices are not revealed in this process. Network environment details are needed to process various network management & control ling tasks. Detection and setting up of right paths are most significant jobs to allow switch for forwarding of network traffic.

OpenFlow protocol enabled switches have group tables and flow entry tables.Controller makes required changes such as addition, deletion and alteration in flow tables through OFPT_FLOW_MOD messages. Packet header fields structure, set of counters and actions are part of flow entry. On arrival of a packet, its header fields are matched with field values of flow entries and on packet matching with any flow entry, relevant counter sets are increased and related actions are preformed. When no flow entry matches with packet header fields, a table miss messages is generated with instruction to switch for packet to forward to controller or forward to other table or drop it. In occasion of packet forwarding to controller, an OFPT_PACKET_IN event to controller is raised by switch and controller inspects to arriving packet and generates required control packets and returns original packet to switch with required actions to be performed. The controller setups required flow entries into OpenFlow switch flow table by posting OFPT_FLOW_MOD messages to deal with such potential packets.

March- April 2020
ISSN: 0193-4120 Page No. 911 - 919

III. Topology detection in SDN

Forwarding devices use a single-hop Link Layer Discovery Protocol(LLDP) to publicize their occurrence, neighbors and properties in wired LAN. Devices frame LLDP messages with ether type field value as hex 88cc value for multicast MAC address such as 01:80:c2:00:00:0E of bridge and sends at a predefined time period [27]from their active ports. Since, forwarding devices do not generate LLDP packets itself in SDN network.

Network topology was detected using procedure mentioned in [6]. This discovery procedure detects only connected switches and their interconnections. It does not reveal any connected host details. Open Flow controllers have fundamental host recognition process through table miss flow entry of SDN forwarding devices and process instructs SDN device to forward packet to controller. Address Resolution Protocol(ARP) or Internet Protocol (IP) traffic is started generating by host & forwarded to SDN forwarding device, it does not have flow rules for the traffic and in this situation, connected host's first packetis sent to controller. Host detail from the first packet is extracted by controller and completed host discovery process.

We can use LLDP for discovery of connected host and for this; the protocol has to be implemented at host level. Since host are monitored and managed by different entities therefore implementation of LLDP at host level is very tough.

V. Related Works

Dependency of host discovery activity is on ARP and DHCP packets details as described in [21] using Packet. In event at L2 protocol. Study described that OpenFlow Discovery Protocol is limited to learning of controller using LLDP frame format about the presence of forwarding devices in the network. Authors have not suggested any host discovery technique prior to connected host traffic generation.

NMAP utility inspired researchers in [16] to propose a host discovery approach by creating ARP-Request messages from controller side for ascertaining host details as soon as connection established between OpenFlow switch and controller. OFPT_PACKET_OUT control messages were used by authors to generate ARP-request messages using random destination IP and broadcast MAC to a switch port. ARP-request messages were broadcasted by OpenFlow switch. In this process, the connected live host which is having IP address will respond with ARP reply message to switch. The OpenFlow switch will forward ARP-reply messages to controller to extract required host details from messages. Using this host discovery module and LLDP, now the controller will have whole network environment details such as host IP & MAC, presence of switches & switch ports, interconnectivity between them. This approach worked properly when an OpenFlow switch was connected with operational controller. Approach did not performed well in to conditions such as first, while switch started prior to controller and second, switch detached from controller, some changes carried out in status of switch port and switch again connected to controller. OFPT_PORT_STATUS message was used in implementation by the approach and it works on changes such as deletion, addition, &status change in switch port status.

In [34] researchers have suggested that delivery of first packet through sub-domain cluster model by selecting the high priority controller checking the load performance index. Approach selects a reasonably less busy controller for forwarding of data packets. The implementation of this approach requires more than one controller and controllers load record has to be maintained. This may take longer time to process the data packets in case all controllers have same processing load and approach has to find low load controller. Numbers of controller also required more processing power, it increases the cost of network and makes network more complex.

V. Proposed SDN-CIFE

Objectives of SDN-CIFE

- Detection of connected host information such as MAC address and OpenFlow switch ID &switch port ID at handshaking between OpenFlow switch and controller and at any modification occurs on switch port/host.
- Managing HostLink table at Controller level to store Host MAC, SDN switch ID & Port ID.
- Loading essential flow entries into OpenFlow switch flow table before actual data packet generation by host.

SDN-CIFE Architecture

Our SDN-CIFE structure has different type of network elements such as OpenFlow switches, connected hosts, controller. Hosts are connected with OpenFlow switch using a switch port. OpenFlow switches are linked to controller and have interconnections among them. SDN-CIFE architectural outline is shown in figure 1. Open Southbound API is used to establish connection between RYU Controller and OpenFlow switches. In proposed approach, SDN-CIFE function works on top of RYU Controller. Details of different modules of SDN-CIFE with specific functions are described in subsequent sub-section.

March- April 2020
ISSN: 0193-4120 Page No. 911 - 919

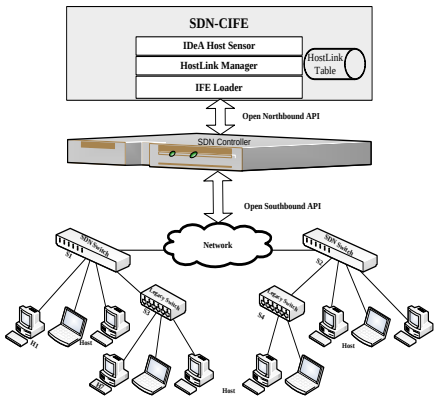

Fig.1: Architecture of SDN-CIFE

Modules of SDN-CIFE

IDeA Host Sensor:

Controller needs host information to create filtering rules, generate control packets and apply security policies [30]. OpnFlow protocol has partial network topology detection ability and it is restricted to detection of forwarding devices, their interconnectivity and presence of controller. Revealing existence of hosts before generation of traffic in network is not in scope of topology detection. Initially, filtering & forwarding rules are not available in packet forwarding devices for the host data packet generated first time. The host packet is sent by forwarding device to controller due to flow miss event. Incoming packet is inspected at controller level, host details are extracted from packet, define policies & rules for packet and setup necessary flow rules into forwarding device flow table to enable device to work with upcoming such traffic from host. Above processes are time consuming, have to be completed at controller level and the forwarding device has to wait till above processes are completed. The first packet of host requires longer reach time to its destination due to above waiting period.

Instant Detection of Host [6] in Software Defined Networks technologywas proposed to identify connected hosts at handshaking between SDN switch and controller.IDH-SDN [6] produced ARP-Request packets and forwarded in network and ARP-Reply packet generated by connected host was checked by SDN controller and connected host details were taken from ARP reply packet& savedinto a table.

This section proposed an Instant Detection Approach (IDeA) for Host in RYU SDN Controller and as and when a switch/switch port/host is introduced into network, it identifies connected hosts. IDeA algorithm and flow data are shown in figure 2.

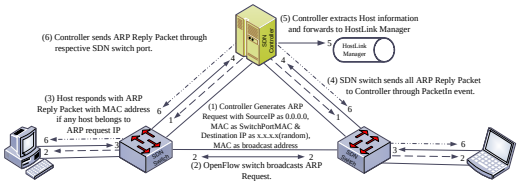

Fig.2: Algorithm and Flow Diagram for IDeA

HostLink Manager:

IDeA forwards host details after extracting from ARP Reply packet to HostLink Manager to setup needed entry into HostLink Table. HostLink Manager verifies host identity from HostLink Table and acts as per verification result. Incase host information is not available in table than it adds host information into table. Similar to HostLink Table, a table in [29] has been proposed to record details of connected host MAC &IP. Format of HostLink Table is mentioned in figure 3. It stores host identity details in fields like Host MAC, SDN Switch Port ID and SDN Switch ID.

Host MAC	SDN Switch Port ID	SDNSwitch ID

Fig.3: HostLink format

After recording required host information into HostLink Table, the module passes HostLink Table to IFE Loader for further action.

IFE Loader:

Instant Flow Entry (IFE) Loader is proposed to define and supervise flow entries into Openflow switch flow table whenever a new host identified by IDeA Host Sensor and host details added into HostTable by HostLink Manager. The HostLink Manager handovers hostlink table to Instant Flow Entry (IFE) Loader after host detection completed. IFE Loader setups flow entries into device flow table based on host information available in HostLink Table for direct sending of packets to their target address. These required flow entries are created at time of handshaking between OpenFlow switch and controller or any change noticed at switch Port.

Host starts transmission of data packets after completion of SDN/OpenFlow switch handshake with controller or change on switch port. Our module loads all required flow entries into SDN switch flow table at handshaking or change noticed for SDN switch Port. Therefore, now host's packets will be forwarded to its target address through matching with flow entries of respective SDN switch. There will not be any flow miss event and no packet will be forwarded to controller. We can say that there will not be any flow miss event because all required flow matches are already present

March- April 2020
ISSN: 0193-4120 Page No. 911 - 919

into flow table of SDN switch and all actual traffic of connected hosts will be forwarded directly to the destination.

In this situation, controller will not receive the first packet of host for inspection of packet, taking decision, creating control packets for data packet and installing flow entries & actions into SDN switch flow table for succeeding such kind of data packets received from a host. With this approach, we have reduced generation of control packets and decreased forwarding time period of host's first data packet by instant detecting connected hosts, storing host link details into HostLink Table and making necessary flow entries proactively at the time of SDN switch handshaking.

I. SDN-CIFE IMPLEMENTATION

To reduce first packet processing time and generation of control packets between controller and SDN switch, we setup flow entries well before host starts generating traffic. This process has to be completed at the time of handshaking between forwarding device &controller and at notification of switch ports status change. Our proposed IDeA generates ARP-Request packet at controller level with source IP=0.0.0.0, SrcMAC=SwitchPortMAC, DstMAC=broadcast &DstIP=random IP address. It is implemented in controller at SDN switch handshaking and at notification switch ports status change. HostLink Table Manager and IFE Loader are mentioned as Algorithm 1 & 2 respectively.

Algorithm 1: HostLink Manager

Level: PacketIn event // implementation level
Input: env PacketIn massage // input value for event
Output: HostDetails for HostLink Table //

1. Extract DstMAC, SDNSwitchID &SDNSwitchPortID, PacketType from evn // extraction of host information from ARP-request
2. If SrcIP is "0.0.0.0" and DstMAC is broadcast than goto 9 // incase ARP-Request messages
3. If DstIP is "0.0.0.0" than // ARP-reply
4. If SrcMAC not exists in HostTable and DstIP is "0.0.0.0" than
5. Forward evn and HostLink Table to IFE Loader
6. Add SrcMAC, SDNSwichID, SDNSwitchPortID into HostLink Table // save host details into HostLink Table
7. End if
8. End if
9. Forward packet to outport
10. Return

Algorithm 2: IFE Loader

Implementation as: IFE Loader
Input: env PacketIn and HostLink Table from HostLink Manager
Output: Flow Entries for SDN Switch

1. Extract DstMAC, SwitchID & SourcePortID, PacketType from evn
2. For x in HostLink Table (Switch ID) do
// load flow entries for forward traffic
3. FlowEntry = SourceMAC=SrcMAC, In_port = SwitchPortID DstMAC = (HostLink Table(x).DstMAC) Outport = (HostLink Table(x).SDNSwitchPortID)
4. Set FlowEntry// install flow entry
// load flow entries for reverse traffic
5. FlowEnrty = SourceMAC=(HostLink Table(x).DstMAC), In_port = (HostLink Table(x).SDNSwitchPortID) DstMAC = SrcMAC Outport = SwitchPortID
6. Set FlowEntry// install flow entry
// load ARP flooding packets
7. FlowEnrty = SourceMAC= SrcMAC, //load ARP flooding packets In_port = SwitchPortID DstMAC = "ff:ff:ff:ff:ff:ff" //broadcast address Outport = Flood
8. Set FlowEntry// install flow entry
9. End for
10. Return

SDN-CIFE PERFORMANCE:

This section of paper has examined the performance of our technique based on bandwidth test with &without SDN-CIFE, No of flow entries generatedand First Packet Processing Time comparisons. The aims of this research paper are detection of connected hosts, taking details like SwitchID, SwitchPortID& hostMAC,managing HostLink Table at controller and load needed flow entries into forwarding device at the time of handshaking SDN switch with controller or any change reported on switch port before actual transmission of host packet. These objectives are also reducing process time for generation of control packets and setup flow entries into SDN switch on transmission of first packet of host. Proposed approach loads needed flow entries into OpenFlow forwarding device well before transmission of host first packet and first & subsequent packets are forwarded directly by forwarding device to their targeted address through matching with flow entries in device flow

March- April 2020
ISSN: 0193-4120 Page No. 911 - 919

table. It also removes the overhead from controller by reducing generation of control packet requirements and increase packet transmission efficiency of network.

Simulation Environment:

SDN-CIFE proposed system was tested, implemented and outcomesgained for achievement of its objectives. The experimental environment was created by including Intel processori5-6200U@2.30Ghz2.40Ghz with RAM 4GB and SSD 500GB, OS Ubuntu 18.04 LTS. RYU-4.28 SDN controller &python version 2.7.12 programming language were installed and configured for different SDN network scenarios. Python programs required during experiment were generated using Gedit text editor.We also used Wireshark-2.2.6 for capturing & inspection of network packets forwarded amonghosts and switches.

Network Scenarios:

System was implemented and tested using different SDN network scenarios during simulation of system. The first network topology as Single SDN Switch Scenario (4S) with one SDN Controller, one SDN Switch and four hosts was used. The second scenario named as Multiple SDN Switch Scenario (M3S) with one SDN Controller, threeSDN switches and seven hosts was used. The third scenario was known as Hybrid SDN Switch Scenario (H3S) with one SDN Controller, two SDN Switches, twoNon-SDN (legacy) switches and eleven hosts was used.Figure 4, 5 & 6 show the diagrams of aforesaid network scenarios:

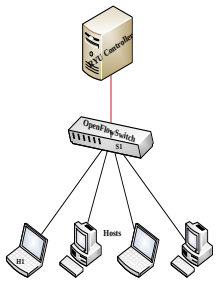

Fig.4: Single SDN Switch Scenario (4S)

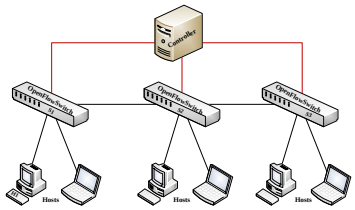

Fig.5: Multiple SDN Switches Scenario (M3S)

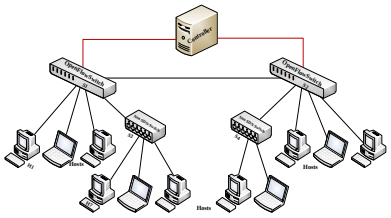

Fig.6: Hybrid SDN Switches Scenario (H3S)

Scenario-wise HostLink Table Data:

Proposed system used 4S, M3S&H3Snetworksfor implementation and experimentation. It revealed linked host in network topology at handshakingbetween SDN switch & controller and saved extracted host information in the HostLink Table. Details of connected hosts were verified using Wireshark network traffic analyzer. The results given table 1 shows that system performedas indented and discovered the networkedhosts effectively.

TABLE 1: HOSTLINK TABLE STATUS

Host MAC	SDN Switch Port ID	SDN Switch ID
Network Scenario: H3S		
00:00:00:00:00:03	1	1
00:00:00:00:00:07	2	
00:00:00:00:00:09		
00:00:00:00:00:08		
00:00:00:00:00:06	3	
00:00:00:00:00:05		
00:00:00:00:00:04		
00:00:00:00:00:10		
00:00:00:00:00:11		
00:00:00:00:00:01	4	
00:00:00:00:00:02	5	
00:00:00:00:00:04	1	2
00:00:00:00:00:05	2	
00:00:00:00:00:06	3	
00:00:00:00:00:10	4	
00:00:00:00:00:11		
00:00:00:00:00:03	5	
00:00:00:00:00:02		
00:00:00:00:00:01		
00:00:00:00:00:09		
00:00:00:00:00:08		
00:00:00:00:00:07		
Network Scenario: M3S		
00:00:00:00:00:03	1	1
00:00:00:00:00:06		
00:00:00:00:00:05		
00:00:00:00:00:04		
00:00:00:00:00:02	2	
00:00:00:00:00:01	3	
00:00:00:00:00:02	1	2
00:00:00:00:00:01		
00:00:00:00:00:03	2	
00:00:00:00:00:04	3	
00:00:00:00:00:06	4	
00:00:00:00:00:05		
00:00:00:00:00:05	1	3
00:00:00:00:00:06	2	
00:00:00:00:00:03	3	
00:00:00:00:00:02		
00:00:00:00:00:01		
00:00:00:00:00:04		
Network Scenario: 4S		
00:00:00:00:00:02	1	1
00:00:00:00:00:01	2	
00:00:00:00:00:03	3	

March- April 2020
ISSN: 0193-4120 Page No. 911 - 919

00:00:00:00:00:04	4	

No of Flow Entries:

4S, M3S&H3S network scenarios used various arrangements like number of SDN switches, hosts, controller and other for implementation and instant detection of networked hosts, host details mining and installing flow entries into SDN Switch flow table. No of flow entries installed in forwarding devices are given in table 2. Flow entries installed during testingmay be retrieved with**ovs-ofclt dump-flows**instruction. We found that system has defined and setup flow entries into SDN switchesas per approach requirement.

TABLE 2: FLOW ENTRIES GENERATED FOR NETWORK SCENARIOS

Network Scenario	Flow Entries Number			
	SDN Switch			Sum
	S1	S2	S3	
4S	16	-	-	16
M3S	21	30	21	72
H3S	89	83	-	172

Flow Entry with Varying Host numbers:

We carried outtests with 4Snetwork scenario using different number of hosts to verifyflow entries of SDN switch flow tables. The numbers of flow entry in S1 SDN switch flow table during testing using different host numbersare given in table 3. System result depicts that SDN-CIFE performed correctlywith different host numbers in network, defined &setup flow entries in SDN Switch as per system requirement.

TABLE 3: FLOW ENTRIESUSING DIFFERENT HOST NUMBERS

Test No	Flow Entries	Host Numbers
1	225	15
2	900	30
3	2025	45
4	3600	60
5	5625	75
6	8100	90
7	11025	105
8	14400	120
9	18225	135
10	22500	150

Bandwidth Test at Different Hops:

Systems bandwidth delivery analysis was carried out using H3Snetwork scenario with single, dual & multiple connections and different number of hops. The test was conducted without& with SDN-CIFE system and results depicts that after implementation of proposed system bandwidth delivery improved with three or more hops. The results given in figure 7 & table 4 are collected during test.

TABLE 4: BANDWIDTH DELIVERY TEST

Connection Scenario		Hops (B/w in Gbps)			
		1	2	3	4
With SDN-CIFE	Single	47.70	45.50	37.80	33.50
	Dual	25.00	24.10	17.90	17.70
	Multiple	9.00	8.90	7.20	7.20
Without SDN-CIFE	Single	50.00	41.00	42.30	35.30
	Dual	18.60	22.30	17.70	17.60
	Multiple	9.70	7.20	8.00	7.80

First Packet Processing Performance:

First packet processing time is tested issuing ICMP echo request and echo reply messages. The total time was calculated from issue of echo request to till receipt of echo reply message from the destination host. We used "**pingallfull**" command at mininet simulator command line for testing of RTT (round trip time) time.

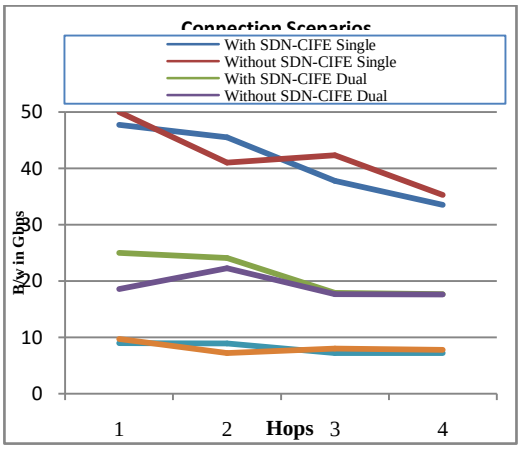

Fig.7: Bandwidth Delivery Test for H3S

TABLE 5: FIRST PACKET PROCESSING PERFORMANCE

Network Scenario	RTT (in ms)		Time Saved
	without SDN-CIFE	With SDN-CIFE	
SH3	505.623	43.432	91.41%
M3S	214.572	15.999	92.54%
4S	51.809	7.157	86.19%
4S with 150 hosts	130608.116	20911.016	83.99%

Table 5 data depicts that M3S network scenario has saved the highest 92.54% time in forwarding and receiving the packets of connected hosts and H3S has saved 91.41%. Our approach has achieved more than 83% times saving with 150

hosts in 4S network scenario. Experiment result shows that we have saved more than 83% time for forwarding and processing of first packet in every network scenario.

II. CONCLUSION

We have proposed an SDN-CIFE system for improvement of first packet processing and forwarding time period. During experiment, system installed all required flow rules in SDN switch and it processed and forwarded the packet as per flow rules. The flow rules guided to the SDN switch for proper forwarding route for destination and first packet of connected hosts were not forwarded to controller. The system identified connected hosts at time of handshaking between SDN switch and controller, prepared the HostLink Table and loaded required flow entries. The system was tested with various parameters such as flow entry generation, bandwidth performance and first packet processing and forwarding time duration. After implementation of SDN-CIFE,first packet processing time duration reduced to more than 83% and it took overall less than 17% time. This performance may vary due to experiment environment setup, network topologies & number of connected hosts. SDN-CIFE approach may be used for development of security systems, policies and applications in future.

REFERENCES

[1] A. Akhunzada, E. Ahmed, A. Gani, M. K. Khan, M. Imran and S. Guizani, "Securing Software Defined Networks: Taxonomy, Requirements and Open Issues," in *IEEE Communications Magazine*, vol. 53, no. 4, pp. 36-44, April 2015.

[2] A. Prakash and R. Priyadarshini, "An Intelligent Software Defined Network Controller for Preventing Distributed Denial of Service Attack," *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, Coimbatore, 2018, pp. 585-589.

[3] B. H. Lawal and A. T. Nuray, "Real-time Detection and Mitigation of Distributed Denial of Dervice (DDoS) Attacks in Software Defined Networking (SDN)," *2018 26th Signal Processing and Communications Applications Conference (SIU)*, Izmir, 2018, pp. 1-4.

[4] C. Zhang, G. Hu, G. Chen, A. K. Sangaiah, P. Zhang, X. Yan and W. Jiang, "Towards a SDN-Based Integrated Architecture for Mitigating IP Spoofing Attack," in *IEEE Access*, vol. 6, pp. 22764-22777, 2018

[5] D. Satasiya and Raviya Rupal D., "Analysis of Software Defined Network firewall (SDF)," *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, Chennai, 2016, pp. 228-231

[6] Ramesh Chand Meena, Meenakshi Nawal, Mahesh Bundele, "Instant Detection of Host in SDN (IDH-SDN)", International Journal of Recent Technology and Engineering, Vol-8:Issue-3, Sep-2019, pp.5603-5608

[7] Gian M. di Marzo and Francesco Benedetto, "Software Defined Networks for Data Center Optimization", Recent Patents on Computer Science (2014) 7: 24.

[8] Guolong Chen, Guangwu Hu, Yong Jiang and Chaoqin Zhang, "SAVSH: IP source address validation for SDN hybrid networks," *2016 IEEE Symposium on Computers and Communication (ISCC)*, Messina, 2016, pp. 409-414

[9] H. T. Nguyen Tri and K. Kim, "Assessing the Impact of Resource Attack in Software Defined Network," *2015 International Conference on Information Networking (ICOIN)*, Cambodia, 2015, pp. 420-425

[10] K. Guerra Pérez, X. Yang, S. Scott-Hayward and S. Sezer, "A Configurable Packet Classification Architecture for Software-Defined Networking," *2014 27th IEEE International System-on-Chip Conference (SOCC)*, Las Vegas, NV, 2014, pp. 353-358.

[11] K. K. Karmakar, V. Varadharajan and U. Tupakula, "Mitigating Attacks in Software Defined Network (SDN)," *2017 Fourth International Conference on Software Defined Systems (SDS)*, Valencia, 2017, pp. 112-117

[12] Kwon, Jonghoon, Dongwon Seo, Minjin Kwon, Heejo Lee, Adrian Perrig and Hyogon Kim. "An Incrementally Deployable Anti-spoofing Mechanism for Software-Defined Networks." *2015 Computer Communications 64*: pp.1-20.

[13] L. M. van Adrichem, Niels & Doerr, Christian & A. Kuipers, Fernando. (2014). "OpenNetMon: Network Monitoring in OpenFlow Software-Defined Networks." *IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World* 10.1109/NOMS.2014.6838228: pp. 1-8.

[14] M. Dabbagh, B. Hamdaoui, M. Guizani and A. Rayes, "Software-Defined Networking Security: Pros and Cons," in *IEEE Communications Magazine*, vol. 53, no. 6, pp. 73-79, June 2015.

[15] M. Z. Masoud, Y. Jaradat and I. Jannoud, "On preventing ARP poisoning attack utilizing Software Defined Network (SDN) paradigm," *2015 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, Amman, 2015, pp. 1-5.

[16] Manzanares-Lopez, Pilar & Muñoz-Gea, Juan & Manuel Delicado-Martinez, Francisco & Malgosa, Josemaria & Flores de la Cruz, Adrian. (2016). Host Discovery Solution: An Enhancement of Topology Discovery in OpenFlow based SDN Networks. 80-88.

[17] Michele Amoretti, Gianluigi Ferrari, Jean-Luc Richier and Andrzej Duda, "Patents on IPv6-Related Technologies", Recent Patents on Computer Science (2013) 6: 170.

[18] O. Chippalkatti and S. U. Nimbhorkar, "An approach for detection of attacks in software defined networks," *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, Coimbatore, 2017, pp. 1-3.

[19] O. Strugaru, A. D. Potorac and A. Graur, "The impact of using Source Address Validation filtering on processing resources," *2014 10th International Conference on Communications (COMM)*, Bucharest, 2014, pp. 1-4

[20] P. B. Pawar and K. Kataoka, "Segmented proactive flow rule injection for service chaining using SDN," *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, Seoul, 2016, pp. 38-42

[21] P. Zanna, S. Hosseini, P. Radcliffe and B. O'Neill, "The challenges of deploying a software defined network," *2014 Australasian Telecommunication Networks and Applications Conference (ATNAC)*, Southbank, VIC, 2014, pp. 111-116.

March- April 2020
ISSN: 0193-4120 Page No. 911 - 919

[22] Ramesh Chand Meena, Meenakshi Naval and Mahesh Bundele, "SIPAV-SDN: Source Internet Protocol Address Validation for Software Defined Network", in *International Journal of Innovative Technology and Exploring Engineering (2019)*, Vol.8, Issue 12, pp.3386-3393

[23] S. Murtuza and K. Asawa, "Mitigation and Detection of DDoS Attacks in Software Defined Networks," *2018 Eleventh International Conference on Contemporary Computing (IC3)*, Noida, 2018, pp. 1-3.

[24] S. Nadar and S. Chaudhari, "Proactive-routing path update in Software Defined Networks(SDN)," *2017 International Conference on Intelligent Computing and Control (I2C2)*, Coimbatore, 2017, pp. 1-3.

[25] S. T. Ali, V. Sivaraman, A. Radford and S. Jha, "A Survey of Securing Networks Using Software Defined Networking," in *IEEE Transactions on Reliability*, vol. 64, no. 3, pp. 1086-1097, Sept. 2015.

[26] S.K. Agrawal and Kapil Sharma*, "Millimeter Wave Channel Capacity for 5th Generation Software Defined Radio Communication System in Vegetation Area", International Journal of Sensors, Wireless Communications and Control (2018) 8: 172.

[27] T. Alharbi, M. Portmann and F. Pakzad, "The (in)security of Topology Discovery in Software Defined Networks," *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, Clearwater Beach, FL, 2015, pp. 502-505.

[28] T. Chin, X. Mountrouidou, X. Li and K. Xiong, "Selectiv Packet Inspection to Detect DoS Flooding Using Softwar Defined Networking (SDN)," *2015 IEEE 35th Internationc Conference on Distributed Computing Systems Workshop*: Columbus, OH, 2015, pp. 95-99

[29] T. Javid, T. Riaz and A. Rasheed, "A layer2 firewall fc software defined network," *2014 Conference o Information Assurance and Cyber Security (CIACS)* Rawalpindi, 2014, pp. 39-42.

[30] T. Xu, D. Gao, P. Dong, C. H. Foh and H. Zhang "Mitigating the Table-Overflow Attack in Software Defined Networking," in *IEEE Transactions on Networ and Service Management*, vol. 14, no. 4, pp. 1086-109 Dec. 2017.

[31] Y. Jia, Y. Liu, G. Ren and L. He, "Revisiting inter-AS I spoofing let the protection drive source address validation, *2017 IEEE 36th International Performance Computing an Communications Conference (IPCCC)*, San Diego, CA 2017, pp. 1-10.

[32] Bingyang Liu, Jun Bi and Yu Zhou, "Source Addres Validation in Software Defined Networks," i *SIGCOMM'16* August 22-26 2016, p. 595.

[33] Alif Akbar Pranata, Tae Soo Jun, Dong Seong Kim "Overhead reduction scheme for SDN-based Data Cente Networks," *Computer Standards & Interfaces*, Volume 6 2019, Pages 1-15, ISSN 0920-5489,

[34] Mingyong Chen, Weimin Wu, "A first packet processin subdomain cluster model based on SDN", *AIP Conferenc Proceedings-1864*, 020042 (2017)

[35] D. Kotani and Y. Okabe, "Packet-In Message Control fc Reducing CPU Load and Control Traffic in OpenFlov Switches," *2012 European Workshop on Software Define Networking*, Darmstadt, 2012, pp. 42-47. do 10.1109/EWSDN.2012.23

[36] A. S. Iyer, V. Mann and N. R. Samineni, "SwitchReduce Reducing switch state and controller involvement i OpenFlow networks," *2013 IFIP Networking Conferenc* Brooklyn, NY, 2013, pp. 1-9.