MODIFICATION OF L3 LEARNING SWITCH CODE FOR FIREWALL FUNCTIONALITY IN POX CONTROLLER (WORKING ON SDN WITH MININET)

Chaitra N. Shivayogimath[1], N.V. Uma Reddy[2]

[1]*PG Student, Dept. of ECE, AMC Engineering College, Bangalore, Karnataka, India*
[2]*Professor, Dept. of ECE, AMC Engineering College, Bangalore, Karnataka, India*

Abstract

Software-Defined Networking (SDN) is the new trend in the networking field. The separation of the control plane from the forwarding plane has enabled the complete programmability of the network, since the control plane and the forwarding plane are decoupled. An API for POX controller is firewall. A modification of the Learning layer 3 switch code for POX controller is done for a tree topology of depth 3 by using mininet network emulator and the packet flow between the hosts is controlled according to the rules inserted in the Learning switch using OpenFlow controller.

Keywords:*-POX, SDN, Controller, rules, topology, Learning switch, Firewall*

----------***----------

1. INTRODUCTION

The network is growing day by day, many new devices get added up in the network. Every time a new machine is added in the network, it can take hours, in some cases days, for IT person to reconFigure ACLs across the entire network. The complexity of today's network makes it very difficult for IT to apply a consistent set of access, security, QoS, and other policies to increasing mobile users, which leaves the enterprise vulnerable to security breaches, non- compliance with regulations, and other negative consequences.

To overcome the vendor dependence for the equipment and to build a standard of standards Software Defined Network (SDN) has been created and is being developed.

All the routing decisions are undertaken by the component of the control plane, the Controller. The forwarding function is performed by a dumb device like the Switch.

The physical test beds are very costly to implement for research and for study purposes of SDN, hence there is a need for virtualization. One such virtualization technique for SDN is the network emulation software, Mininet. Mininet allows the launching of a virtual network with switches, hosts and an SDN controller using the limited resources on a laptop all with a single command. Mininet has all the three components of SDN, namely, Controller, Openflow protocol (Southbound API) and the Controller Applications (Northbound API). The architecture of SDN is shown in Fig. 1.

The most popular controller implementation is POX, which has been developed by NICIRA. POX core and its components are written in Python. Mininet also uses the POX controller. The earlier version of POX was NOX, which was written in Python and C++. Many such controllers have been researched and developed by various groups of developers, such as MUL (Kulcloud), Maestro (Rice University), Trema (NEC), Beacon (Stanford), Flowvisor (NICIRA/Stanford) and RYU (NIT, OSRG Group) [2]. The controller is chosen on the basis of programming knowledge and suitability of the controller for applications.

The forwarding action is done by the switch, which is specifically an Openflow that supports openflow protocol to communicate with the controller and hence the name "Openflow Controller". A softswitch like OpenVswitch can also be used. The openVswitch is a production-quality, multilayer virtual switch. OpenVswitch can act as a soft switch running on a hypervisor or on a physical device, which can be installed on a wide range of platforms [3]. The default TCP port used by the POX controller to communicate with switch is 6633. The following three type of messages are supported by openflow [4][6]:

Table 1: Controller startup flow messages

Sl.. No.	Message	Type Name	Description
1	Hello Controller ->Switch	OFPT_HELLO	Following the TCP handshake, the controller sends its version number to the switch.
2	Hello Switch-> Controller	OFPT_HELLO	The switch replies with its supported version number.

3	Features Request Controller ->Switch	OFPT_FEATURES_REQUEST	The controller asks to see which ports are available.
4	Features Reply Switch-> Controller	OFPT_FEATURES_REPLY	The switch replies with a list of ports, port speeds, and supported tables and actions
5	Set Config Controller ->Switch	OFPT_SET_CONFIG	The controller asks the switch to send flow expirations.

Table 2: OpenFlow messages exchanged between controller and switch for ping.

Sl.. No.	Message	Type Name	Description
1	Packet_In Switch-> Controller	OFPT_PACKET_IN	This message is sent when a packet was received and it didn't match any entry in the switch's flow table, causing the packet to be sent to the controller.
2	Packet_Out Controller->Switch	OFPT_PACKET_OUT	Controller sends a packet out to one or more switch ports.
3	Flow-Mod Controller->Switch	OFPT_FLOW_MOD	Instructs a switch to add or delete a particular flow to its flow table.

Table 1 and Table 2 gives the OpenFlow meassages list. Table 1 gives all the OpenFlow messages shared between POX Controller and the OpenFlow Switch.

A different set of messages are exchanged between the controller and the switch for Ping. Ping utility is majorly used in our test. Table 2 summarizes all the messages exchanged during a ping activity.

In this paper, the L3 Learning switch code of the POX controller is modified to check for the source MAC address in the packet by inserting rules in all the switches and allow only specific source MAC addresses for a tree topology of depth 3, thus providing a firewall kind of functionality to the POX controller.

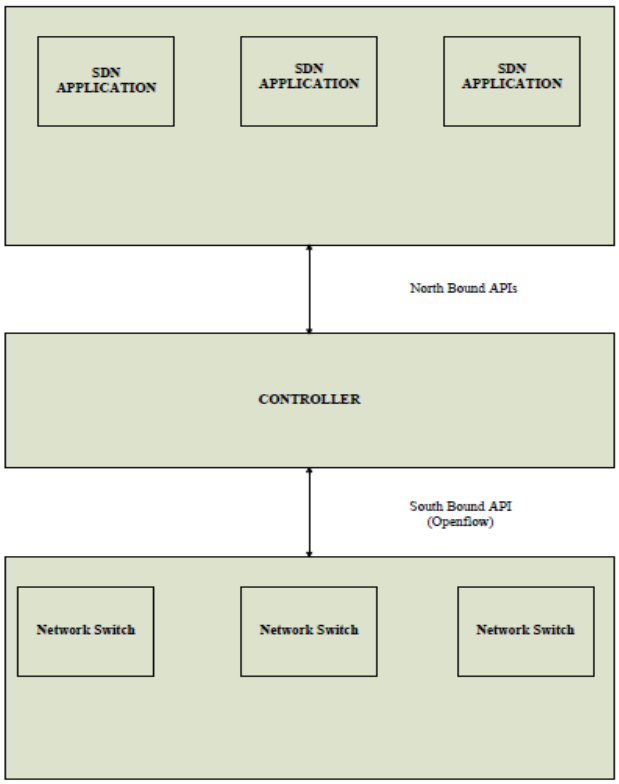

Fig. 1 SDN Architecture

2. NETWORK TOPOLOGY CREATION IN MININET

The topology created for the work is a tree topology with depth 3, as shown in Fig. 2. Mininet network emulator is used to create this topology [5].

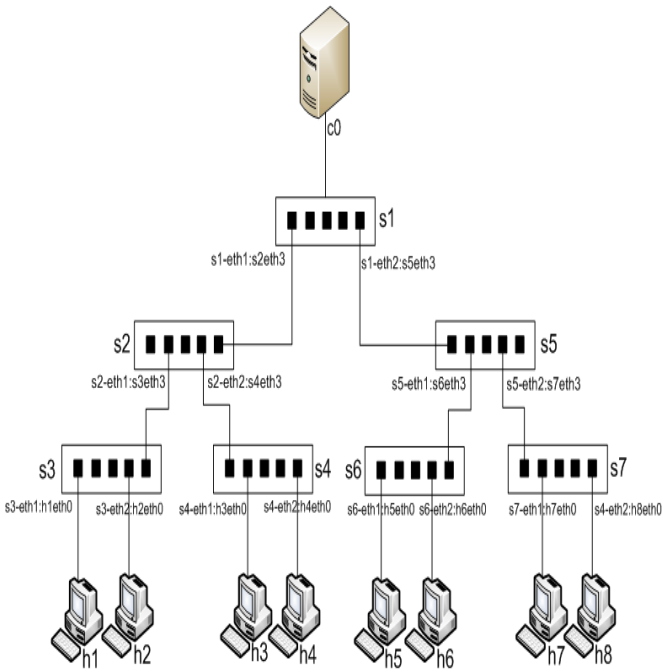

Fig 2 A Tree topology created in Mininet

The controller is named as c0. The openflow switches are named as s1, s2, s3, s4, s5, s6 and s7, and the terminal virtual hosts are named as h1, h2, h3, h4, h5, h6, h7 and h8. Mininet VM is started in Oracle VM Virtual Box. The virtual topology shown in Fig. 2 is a tree topology with depth 3 with only one controller c0. All devices are connected via virtual Ethernet links, as labeled in Fig.2. The topology creation in mininet was studied and implemented as shown in Fig. 3.

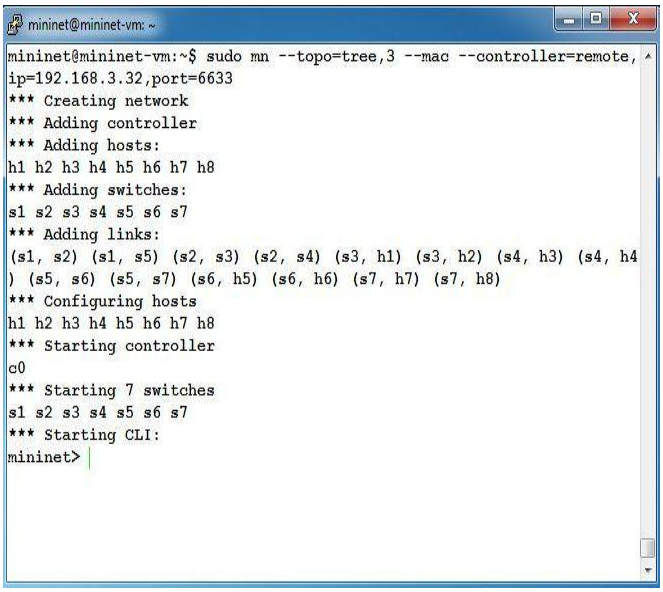

Fig. 3 Creation of Mininet topology through CLI

The net command output is also shown along with the topology for the link and port information in Fig. 4.

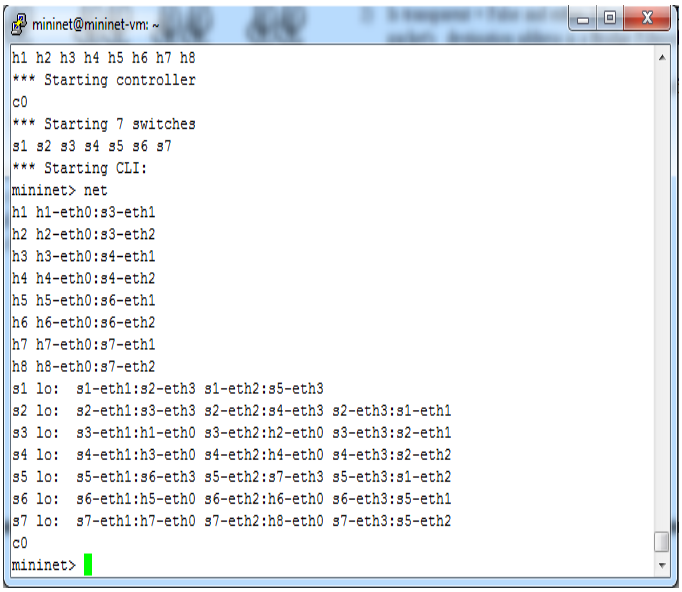

Fig 4 The net links between the switches and the hosts are obtained by issuing net command in mininet prompt.

The POX controller was started on a remote windows machine hosted in the network. The mininet VM was started in an Oracle VM virtual box. The mininet command used to create the tree topology is "mininet@mininet-vm:~$ sudo mn --topo=tree,3 --mac --controller=remote,ip=192.168.3.32,port=6633". The --mac option in the above command sets the virtual host MAC and IP addresses to small, unique and easy-to-read IDs. The controller is set to remote IP address on 192.168.3.32 and the TCP port used is 6633. The port need not be mentioned because it takes the default of 6633. The port needs to be mentioned only if a port other than the default has to be used.

3. MODIFICATION OF THE LEARNING SWITCH ALGORITHM

3.1 Learning Switch Algorithm Present in POX Controller

The learning switch algorithm present in the POX controller is given below:

1) Keep a table that maps IP addresses to MAC addresses and switch ports. Stock this table using information from ARP and IP packets.
2) When you see an ARP query, try to answer it using information in the table. from step 1. If the info in the table is old, just flood the query.
3) Flood all other ARPs.
4) When you see an IP packet, if you know the destination port (because it's in the table from step 1), install a flow for it.

3.2 Firewall Algorithm

This algorithm for learning switch is modified to perform the action of a firewall, allowing only particular IP addresses to communicate over the network through all the Openflow switches in the network.

The algorithm for the firewall functionality is given below:

1) A hash table is required to store the key:value pair of (switch, sourceMAC).
2) Rules are added with MAC addresses only, since MAC address is used for the ARP replies.
3) Check the source IP address against the Rule added.
4) This IP address is checked in the table for its corresponding MAC and table updation by ARP resolution is done if there is no entry.
5) The table maps the (switch, source MAC) to True or False
6) The Controller decides to drop the traffic in either of the following two conditions:
   - If there is a firewall rule matching "False".
   - If there is no firewall rule entry.
7) The packet is forwarded if the rule matches "True".

Here the Firewall is called an L3 since the mapping for IP to MAC is done in a table along with the port number. The ARP replies are done using MAC addresses only.

3.3 Modification of Learning Switch Code of POX Controller to Perform the Firewall Functionality

The POX controller consists of 3 parts:

1) Listener
2) Control Logic
3) Messenger

The Listener has "Packet in" and "Connection Up" messages [7]. The logic in the "Packet in" module is modified to check for the added firewall rules and then perform the usual learning switch functionality if the rule matches "True" (i.e., sending a message to switch to add a new rule in the Openflow Flow Table), else the packet is dropped.

The existing Learning Switch algorithm is modified at step 2 to check the rule.

4. VERIFICATION OF MODIFIED ALGORITHM AND CODE

Tthe rules are added in all the 7 switches.

When a pingall test was conducted after all modifications, only h1 and h8 were reachable, as seen in Fig. 6. The debug messages for the pingall test are shown in Fig. 7.

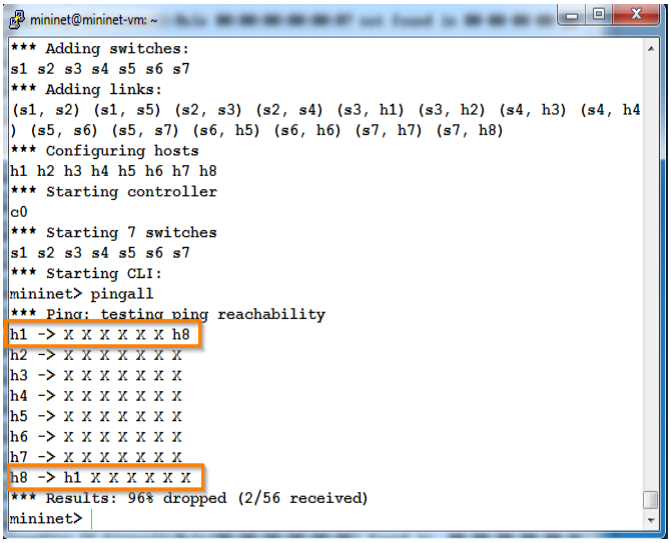

Fig. 6 Pingall test showing the successful modification of Learning Switch code to act as a firewall and allowing communication between h1 and h8 only.

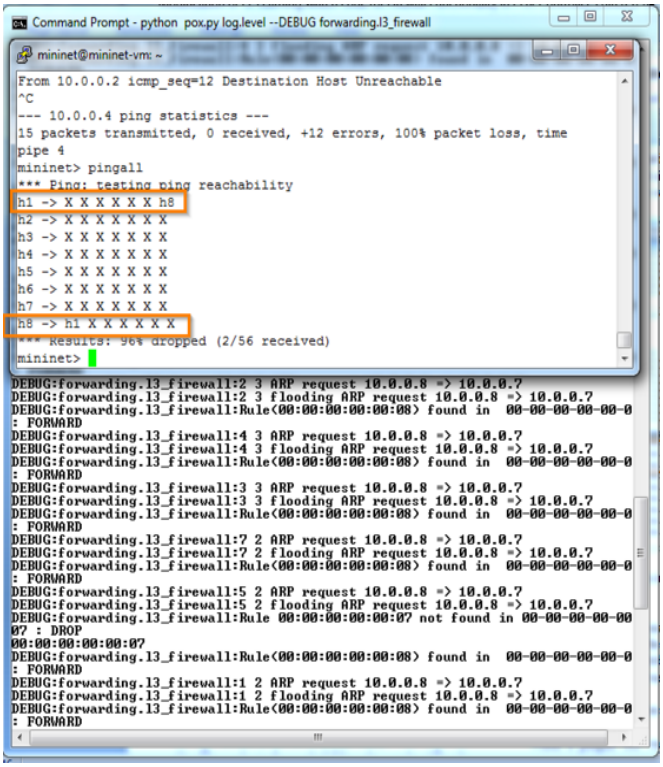

Fig. 7 Debug Messages for the pingall test demonstrating the Flow rule installation and action on the packet based on the rules

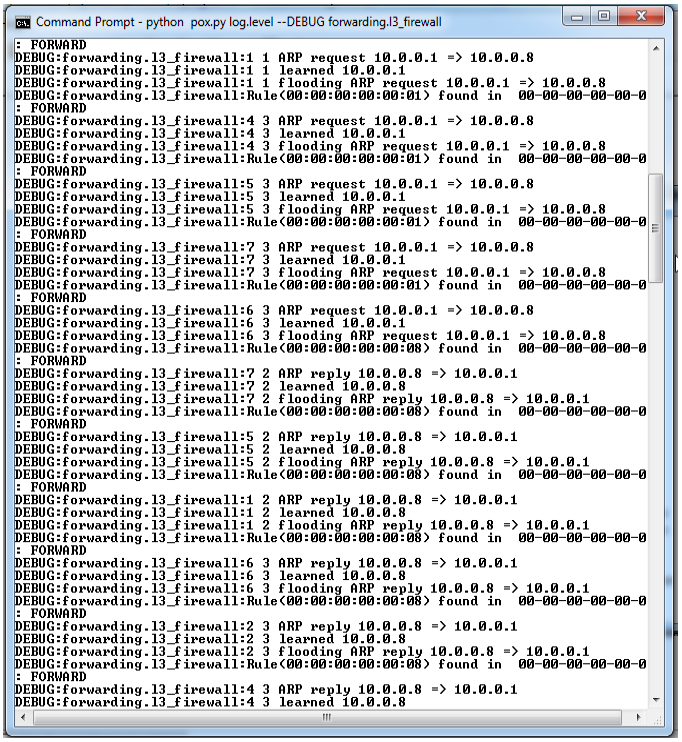

Fig. 8 Debug messages for ping from h1 to h8 demonstarting ARP flooding and the reply.

Fig 8. demonstrates the ARP flooding when there is a request from 10.0.0.1(i.e h1) to 10.0.0.8 (i.e h8). The rules are added in the POX controller when the controller starts up, as shown in Fig,9. The MAC address mapping of IP address of h1 is found by the ARP request. The ARP

requests flow to all the switches. The rules are checked in all the switches. If there is a rule existing in the switch for that particular IP address, then MAC address of that host, here its is h1 with IP address 10.0.0.1 is learned as 00:00:00:00:00:01. Once the source is learned, the destination h8 is also learned in the similar way. The rules are checked for h8 on all the switches, and then the ARP flooding is done if there is a rule allowing traffic from h8. This is demonstrated as shown in Fig 8.

When the test was conducted for communication between h1 and h7, the rules for source are checked on all switches first. Since there is a rule present for h1 on all switches, the packet from h1 is forwarded to the destination i.e. h7 (10.0.0.7) through all the switches. But, there is no rule present for h7 in any of the switches. Starting from the switch s7, which is directly connected to h7. The rule for forwarding the response for h7 is checked on directly connecting switch s7. Since there is no rule present, the packet is dropped, as shown in Fig. 9.

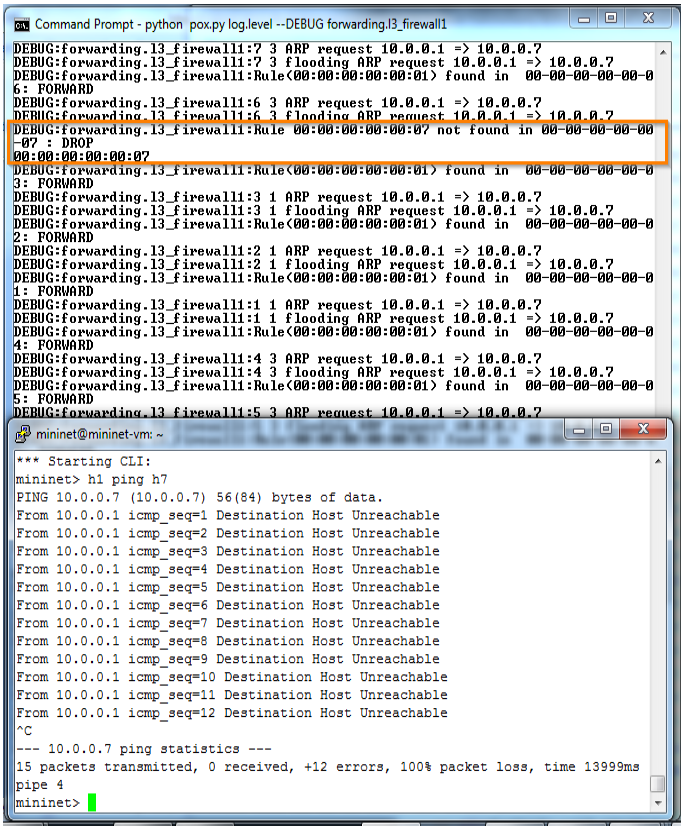

Fig. 9 Debug messages showing DROP of packet for h7 and FORWARD ing of packets for h1.

The POX controller when started with log level set to DEBUG, the first component of the POX controller, i.e. Listener (ConnectionUp), is executed and once the mininet topology is created, the switches get connected to the POX controller. The Info message "Connection {switch dpid} is connected" is displayed. The rules get added on all the switches, as shown in Fig. 10

.

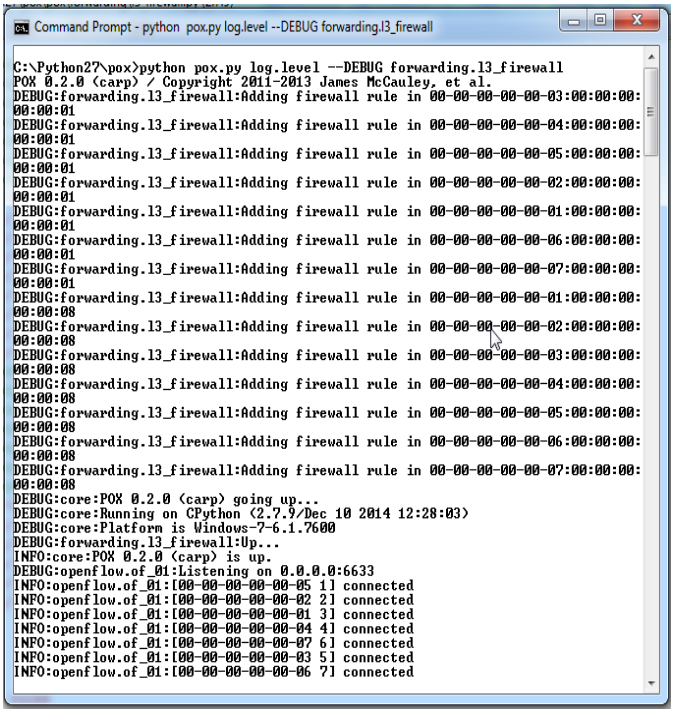

Fig. 10 The Info message showing "Connection Up message" and the Debug messages for "Addition of Firewall Rules".

5. CONCLUSION

The algorithm and hence the L3 Learning switch code of the POX controller are successfully modified to insert rules and provide the firewall functionality for a bigger topology, i.e. a tree topology of depth 3. The modified code controls the traffic by adding rules. This implementation is not an intelligent firewall implementation. Further work would comprise adding the rules with source and destination IP addresses, rather than the dpid of the switch and the source MAC Address, into a CSV file format and importing the file in the code and then adding the rules to perform the firewall functionality. Further, instead of using MAC addresses for ARP replies, IP addresses should be directly used. A module to fetch the IP address from the DPID is to be written.

ACKNOWLEDGMENTS

The authors would like to thank Mr. Santhosh Sundarasamy, Sr.Architect & Engineering Manager, Cloud Managed Security, Paladion Networks Pvt. Ltd., for his complete support and help in carrying out the experiment at Paladion Networks Pvt. Ltd as part of the internship program.

REFERENCES

[1] ONF, "SDN architecture overview version 1.0," Open Networking Foundation, December 2013.

[2] Nunes B.A.A, Mendonca M, Xuan Nam Nguyen,Obraczka K, Turletti T, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," Commun. Surveys Tuts., 2014, 16 (3) IEEE, pp.1617 – 16343.

[3] B Pfaff, B.Davie, Ed "The Open vSwitch Database Management Protocol," RFC 7047, December 2013
[4] B. Heller, "Openflow switch specification, version 1.0.0,"Last accessed, Dec 2011. [Online]. Available: www.openflowswitch.org/documents/openflow-spec-v1.0.0.pdf
[5] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," Proc. of HotNets-IX. ACM, 2010, p. 19.
[6] W Stallings. (2013, March.). "Software Defined Networks and Openflow,"IPJ. [Online]. Available: http://williamstallings.com/Papers/
[7] S Kaur, J Singh andN S Ghumman, "Network Programmability Using POX Controller," [Online] Available at: http://sbsstc.ac.in/icccs2014/Papers/Paper28.pdf.
[8] Thomas D Nadeau & Ken Gray, "SDN Controllers," in SDN Software Defined Networks, 1st ed., O'rielly, 2013.
[9] Wolfgang Braun and Michael Menth (May, 2014), "Software Defined Networking Using OpenFlow Protocols, Applications and Architectural Design Choices", The Future Internet journal.
[10] B. Heller, "Openflow switch specification, version 1.0.0," Last accessed, Dec 2011. [Online]. Available: www. openflowswitch.org/documents/openflow-spec-v1.0.0.pdf
[11] OpenFlow tutorial online: Available at: http://archive.openflow.org/wk/index.php/OpenFlow_Tutorial
[12] de Oliveira, R.L.S. , Shinoda, A.A. ; Schweitzer, C.M. ; Rodrigues Prete, L., "Using Mininet for emulationand prototyping Software-Defined Networks", COLCOM, 2014, IEEE ,pp 1-6.
[13] Thomas D Nadeau & Ken Gray, "SDN Controllers," in SDN Software Defined Networks, 1st ed., O'rielly, 2013.
[14] Wolfgang Braun and Michael Menth (May, 2014), "Software Defined Networking Using OpenFlow Protocols, Applications and Architectural Design Choices", The Future Internet journal
[15] . N. Feamster. "SDN Lectures 2014", [Online] Available at:https://class.coursera.org/sdn-001

BIOGRAPHIES

Chaitra N.Shivayogimath completed her Bachelor of Engineering from BMS Institute of Technology, Karnataka India in 2011.She is Pursuing 4th Semester, Master in Technology, Dept. of E&C at AMC Engineering College, Bangalore, India. She has been certified CE|HV7, by the EC-Council. Her areas of interest are Software Defined Networks, Network Penetration testing, Vulnarability Assessment, Wireless Networks and Computer Networks.

Mrs. N. V. Uma Reddy received her BE (Electronics) degree from Kuvempu University in 1995 and M.Tech (Electronics) degree from Viveswaraya technological University Belgaum, India in 2004, Presently pursuing PhD in VTU, Bangalore, India.she has more than fifteen years of experience in teaching her area of interest include CMOS VLSI, HDL, CCN, AMS, presently she is Working as HOD, Professor in the Department of Electronics and Communication Engineering in AMC Engineering College, Bangalore.