Efficient Packet Forwarding Using Static Routing in SDN

M. Krithika[1], M. Nithya[2]
[1]*PG Scholar, Department of CSE, Anna University Regional Campus, Coimbatore*
[2]*PG Scholar, Department of CSE, Anna University Regional Campus, Coimbatore*
E-mail: krithikalilly2@gmail.com,nithisri92@gmail.com

Abstract: **Software Defined Networks (SDN) is a new networking paradigm in which the control plane is decoupled from switch. SDN is a radical new idea in networking promises to dramatically simplify network management and enable innovation and evolution. The network intelligence of a SDN is logically centralized (at the control plane) in software based controllers, and network nodes become simple packet forwarding devices (the data plane) that can be programmed via an open interface (Open Flow). This paper explains the system forwarding the packets between client and web server using static routing in software defined networks. A SDN router will receive raw Ethernet frames. It will process the packets, and then it will forward them to the correct outgoing interface. The system is simulated using Mininet simulator. The results show that packet forwarding using static routing in SDN minimizes the cost and time varies depends upon the usage when compared with traditional networking.**

Keywords: **SDN, OpenFlow, Mininet, NOX, POX.**

I. INTRODUCTION

Software Defined Networks (SDN) promises to simplify network management and enable innovation through network program ability. SDN has two defining characteristics. SDN separates the control plane from the data plane, and then the control plane is consolidated by SDN, so that the multiple data-plane elements are controlled using single software control program. The SDN control plane directs the control over the network's data-plane nodes such as switches, routers, and middle boxes, which are connected through Application Programming Interface. OpenFlow is a prominent example of such an API.

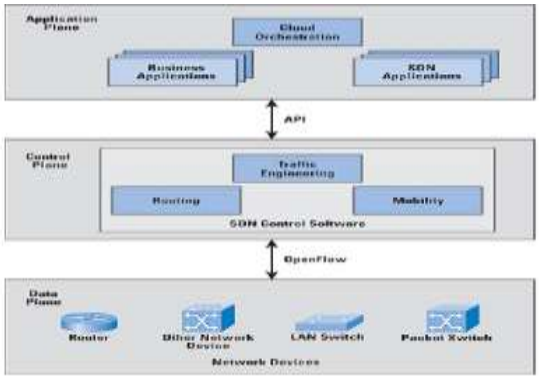

Fig 1: SDN data plane and Control plane

A. SDN Network

The SDN architecture is composed by separating the control plane from the data plane devices which provides a programmable interface for the separated control plane. The forwarding rules are received to the data-plane devices from the separated control plane and those rules are applied to the hardware ASICs.

B. Software Defined Networks (SDN) Architecture

SDN architecture encompasses the complete network platform. The bottom layer of SDN (e.g. Fig 2) involves the network nodes such as Ethernet switches and routers which enables the data plane. The middle layer consists of the controllers that facilitate the paths in the network. The controllers use capacity and demand information which obtained from the networking nodes via the traffic flows. An application programming interface (API) used to link the middle layer with the bottom layer. East and westbound APIs operates based on the Connections between controllers. The northbound API is defined through controller application interface.

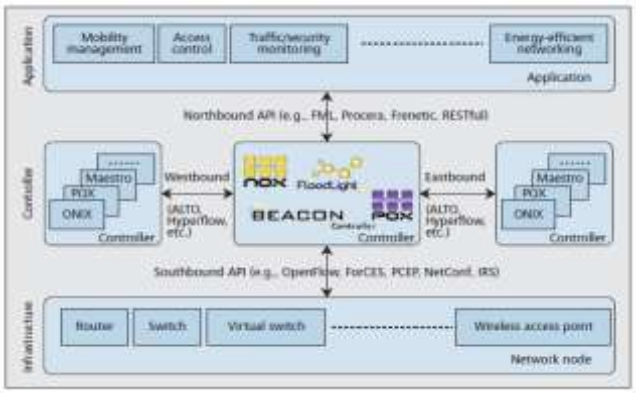

Fig 2: SDN Architecture

The operation and management of the network are represented using Functional applications such as energy-efficient networking, security monitoring, and access control.

C. Working Principle of SDN

Working of SDN (e.g. Fig 3) explains that, (Step 1) the packet of a new flow arrives at the switch from the sender, (Step 2) the switch checks for a flow rule in the SDN cache. If a packet is matched with flow table it will update counter, execute packet/match fields, action set, and metadata. (Step 5) Packets are forwarded to the receiver. If the packet does not match with the flow table, the packet will be forwarded to the controller over a secure channel (Step 3). Using the southbound API (e.g., OpenFlow, ForCES, PCEP), the SDN controller can add, drop, and update the data flow entries. The

© 2015 IJAIR. All Rights Reserved

controller executes the routing algorithm, and adds a new forwarding entry to the flow table in the switch and to each of the relevant switches along the flow path (Step 4).The switch then forwards the packet to the corresponding port to send the packet to the receiver (Step 5).

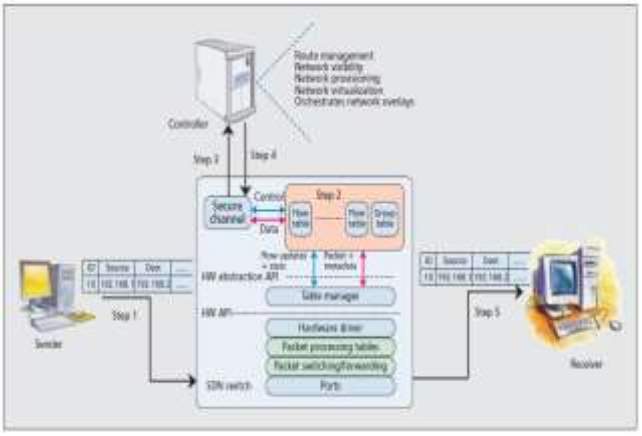

Fig 3: Working Principle of SDN

II. OPENFLOW

OpenFlow is the fundamental element for building SDN solutions. In Traditional switch/router, the packet forwarding done at the data plane and the routing decisions are done at the control plane occur on the same device. An OpenFlow Switch decouples the data plane and control plane functions. The data plane parts still reside on the switch, while routing decisions are placed to a separate controller which is a standard server. The OpenFlow protocol is used for communication between the OpenFlow Switch and Controller, which show the messages such as send-packet out, packet-received, Modify-forwarding-table. The data plane of an OpenFlow Switch gives flow entry which contains a packet fields to match and a defined action. When an OpenFlow Switch receives a packet, if it has no matching flow entries, it forwards this packet to the controller. The controller takes a decision on packets to handle it effectively such as it can drop the packet, or it can add/delete a flow entry directing the switch and then it forward similar packets in the future.

A. *OpenFlow Network*

OpenFlow technology is a set of networks elements and an own protocol to configure the behavior and the actions of a network.

.

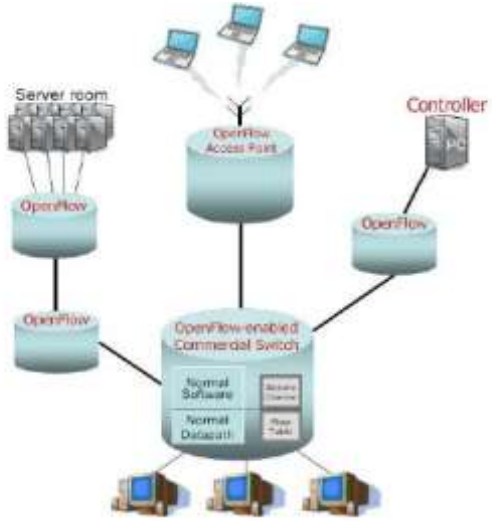

Fig 4: OpenFlow Network

An OpenFlow network is formed by OpenFlow-switches which contain a Flow-table which consists of flow entries of the packets and what are the actions to be performed, the controller is used for adding and deleting flow entries. The secure channel interconnects the switch, controller and OpenFlow protocol for signaling the whole architecture.

B. OpenFlow Switch

OpenFlow switch consists of 2 types of switches. One is the hardware based commercial switches that use the TCAM and the OS of the switch/router for implementing the Flow-table and the OpenFlow protocol. The another type is the software-based switches that use UNIX/LINUX OS system to implement the entire OpenFlow switch functions.

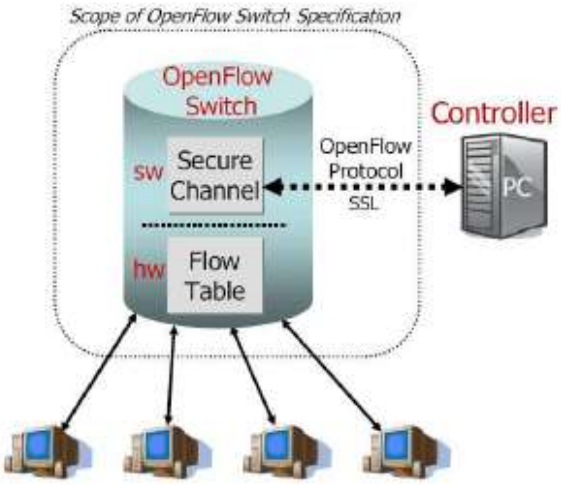

Fig 5: OpenFlow switch Architecture

© 2015 IJAIR. All Rights Reserved

An Open Flow Switch mainly acted based on the following 3 parts.

- Flow table
- Secure channel
- Open flow protocol

Where,

Flow Table with its action for each every flow entry, which tells the switch, how to process the flow.

Secure channel allows the commands and packets which to be sent between a controller and the switch using the open flow protocol.

The OpenFlow Protocol, which provides a standard way for a controller to communicate with a switch.

C. Flow-Table

The Flow-table of the OpenFlow switch defined by three field table.

- **A packet header:** It defines the flow.
- **The action:** It defines how the packet is processed.
- **Statistics:** It will maintain the buffer which contains the number of packets and bytes for each every flow, and the time in the view of last packet matched the flow.

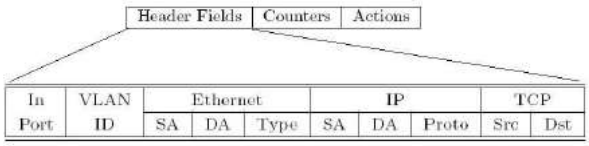

Fig 6: OpenFlow Table Fields

D. Controller

The Controller is the OpenFlow network element that is the responsible for managing the OpenFlow switches. The controller can be a device that only adds and removes flow-entries statically or a sophisticated device that can dynamically add and remove flow-entries depending on different pre-configured conditions.

The OpenFlow provides a simple controller to manage switches. Sometimes it uses controller like NOX controller, which is compatible with the OpenFlow protocol.

E. OpenFlow Secure Channel

OpenFlow switch connects the controller through secure channel and controller configures and manages the switch, then the controller collects actions from the switch, and moves the packet to the switch. OpenFlow protocol formats the secure channel messages and Secure Channel is encrypted using SSL. An OpenFlow Switch consists of one or more new tables and a group table. The table performs the packet lookups and forwarding. The controller manages the switch via the OpenFlow protocol. OpenFlow allows switches to segment traffic into flows using rules Configured in a particular switching table called flow table.

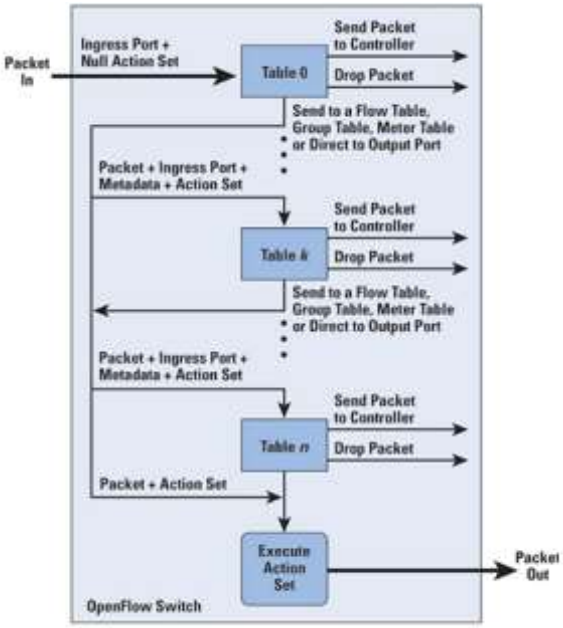

Fig 7: OpenFlow switch packet forwarding

III.PACKET FORWARDING

A. Switching and Routing

Switching and Routing are two methods to forward packets. The difference between them is the OSI level at which they perform the forwarding. Switching/ Bridging is done at layer 2 of OSI while Routing is performed at layer 3 of OSI.

B. Switching

Ethernet switching is performed at Data Link Layer of the OSI stack. This means that link layer controls the data flow, handles errors at transmissions, and manages the physical addressing. Media Access Control (MAC) sub layer and Logical Link Control (LLC) sub layer are the 2 sub layers of the OSI which is classified via Ethernet. The MAC sub layer permits and orchestrates media access, during the LLC sub layer handles flow control, framing, error control, and MAC layer addressing.

C. Routing

The Network Layer of OSI performs the IP routing. The Network Layer forwards the packet between the source and destination (i.e.,) end- to-end delivery. The packet delivery of the network layer allows the routing through

© 2015 IJAIR. All Rights Reserved

intermediate hosts. Data Link Layer of OSI does the node-to-node frame delivery, (i.e.,) hop-to-hop frame delivery on the same link. The Network Layer gives the functional variable length data sequences between the source and the destination host using networks, during the QOS maintenance and error control functions. The function of Network Layers such as Logical Addressing, Datagram Encapsulation, Error Handling, Routing, Fragmentation and Reassembly.

IV. POX CONTROLLER

POX is a python based SDN OpenFlow controller. POX is rapidly used than NOX because POX provides huge development in prototyping. OpenFlow switch is interacted through POX framework. POX is used to build the emerging applications on Software Defined Networking, which is used for prototype distribution, controller design, network virtualization, SDN debugging, and programming models. The goal of POX is to develop a modern SDN controller.

Fig 8: POX Controller

A. POX Components

1) Forwarding.layer2_learning

OpenFlow switches act as a type of Layer2 learning switch, which operates like a NOX's "pyswitch", but the implementation is little different. This component learns Layer2 addresses, it flows the exact-matches on as many fields as possible. For instance, different TCP connections will result in different flows being installed.

2) Forwarding.layer2_pairs

Unlike layer2_learning, layer2_pairs installs rules based purely on MAC addresses.

3) Forwarding.layer3_learning

Layer3_learning does not care about IP stuff like subnets; it just learns where IP addresses. But hosts usually care about that stuff. Some special cases such as, if a host has a set of gateway, host communicates with that subnet using gateway.

4) OpenFlow.spanning_tree

Spanning tree uses the discovery component to build a network topology, which constructs a spanning tree, and it disables flooding on switch ports which are not on the tree. The topologies with loops no longer turn the network into useless hot packet soup. It does not have much of a relationship to Spanning Tree Protocol. The spanning tree component has two options which alter the start-up behavior:

- No-flood: This disables the flooding on all ports as soon as a switch connects.
- Hold-down: It prevents flood control alternates until a complete discovery cycle has completed.

B. Working of POX

Flow entry of a switch matches the incoming packet; the switch updates the counters and applies the corresponding actions. If the packet does not match with the flow entry in switch, it simply forwarded it to a POX controller. The POX controller chooses the packets which are received from certain (e.g., DNS) protocols.

NOX applications use these flow-initiations and forwarded traffic for

- Construct the network view
- Determine whether to forward and control traffic.

V. CONCLUSION

SDN with static routing forwarding enhances cost efficient solutions. Static routing is used to establish the connection between client and server with the help of POX controller.

REFERENCES

[1] Marc Mendonca, Bruno Astuto A. Nunes, Xuan-Nam Nguyen, Katia Obraczka, Thierry Turletti *"A Survey of Software-Defined Networking: Past, Present,and Future of Programmable Network"* The journal of vous consultez L'archieve, May 2013.

[2] Hyojoon Kim and Nick Feamster, *"Improving Network Management with Software Defined Networking",* IEEE Communications Magazine, 2013.

[3] HU Yan-nan, WANG Wen-dong, GONG Xiang-yang, QUE Xi-rong, CHENG Shi-duan, *"On the placement of controllers in software-defined networks"*, The Journal of China Universities of Posts and Telecommunications, July 2012

[4] Soheil Hassas Yeganeh, Amin Tootoonchian, and Yashar Ganjali, University of Toronto, *"On Scalability of Software-Defined Networking"* IEEE Communications Magazine, February 2013.

[5] Sakir Sezer, Sandra Scott-Hayward, and Pushpinder Kaur Chouhan, Barbara Fraser and David Lake, Jim Finnegan and Niel Viljoen, Marc Miller and Navneet Rao , *"Implementation*

© 2015 IJAIR. All Rights Reserved

Challenges for Software-Defined Networks" IEEE Communications Magazine, July 2013.
[6] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, Jonathan Turner, *"OpenFlow: Enabling Innovation in Campus Networks"* IEEE Communications Magazine ,March 2008.
[7] Arka Bhattacharya, Shaunak Chatterjee , *"Routing Algorithms for Rapidly Fluctuating Networks"* The journal of University of California, April 2010.
[8] Bob Lantz, Brandon Heller, Nick McKeown. *"A Network in a Laptop: Rapid Prototyping for Software-Defined Networks"* The journal of Stanford University, October 2010.
[9] Shuo Fang , YangYu , Chuan Heng Foh , and Khin Mi Mi Aung, *"A Loss-Free Multipathing Solution for Data Center Network Using Software-Defined Networking Approach"* IEEE Transactions on Magnetics, June 2013.
[10] Janos Tapolcai , Pin-Han Ho, and Anwar Haque *"A Novel Approximate Link-State Dissemination Framework For Dynamic Survivable Routing in MPLS Networks"* the journal of University of Waterloo, Canada, July 2013.
[11] Abhinava Sadasivarao, Sharfuddin Syed , Ping Pan, Chris Liou, Andrew Lake, Chin Guok, Inder Monga, *"A Software Defined Networking Architecture for Transport Networks"* the joural of Infinera Corporation, August 2013.
[12] Aaron Gember, Robert Grandl, Junaid Khalid, Aditya Akella , *"Design and Implementation of a Framework for Software-Defined Middlebox Networking"* the journal of University of Wisconsin-Madison, WI, USA, August 2013.

.

© 2015 IJAIR. All Rights Reserved