ISSN No. 0976-5697

Volume 8, No. 3, March – April 2017

International Journal of Advanced Research in Computer Science

REVIEW ATICLE

Available Online at www.ijarcs.info

A Review Paper on Software Defined Networking

Sumit badotra
Department of computer Science and Engineering
Shaheed Bhagat Singh State Technical Campus
Ferozepur Punjab, India

Japinder Singh
Department of computer Science and Engineering
Shaheed Bhagat Singh State Technical Campus
Ferozepur Punjab, India

Abstract: The comprehensive concept of SDN(Software Defined Networking) has introduced the extensive change to the traditional networks with the integration of the network by decoupling the forwarding hardware (data plane) from the control logic of the network (control plane).Software Defined Networking (SDN) is a new way of technology that is helping to overcome the limitations of traditional networks. With the help of this emerging technology paradigm has created a great future, an expectation to overcome the need for reliable, secure, flexible and well management of next generation networks. The basic idea behind it is the separation of control plane from data plane. SDN has made the Networks more programmable. SDN is a three layered architecture. OpenFlow Protocol is the most common protocol used for providing the interface called as Southbound APIs between control layer and physical layer. On the other hand interface provided in between application layer and control layer is called as Northbound APIs. All the intelligence is lying in control layer. Controller acts as a brain of the network which is configuring and managing the flow of network through network devices present in physical layer. This paper illustrates the introduction about Traditional networks, Software Defined Networking its architecture, interfaces popular controllers, mininet emulation tool used in SDN, its topologies and security threats in SDN.

Keywords: Software Defined Networking, Data plane, Control plane, Network Function Virtualization, Open Network Foundation, Type-Length-Value

I.INTRODUCTION

Today computer networks are very complex as more and more devices are increasing day by day along with the content they access. The kind of equipment used in networks like Intrusion Detection system,switches,firewalls,Load balancers are typically very hard to manage by network administrator individually, the solution for this is Software Defined Networking. It has changed the way we used to manage the networks. The two main basic principles of Software Defined Networking (SDN) are 1) It separates the control plane from data plane (control plane contains the intelligence, control logic while data plane contains the physical infrastructure or low level network elements that are used for packet forwarding and switching) 2) Control plane acts as a brain of the network which has a direct control over the Data plane, all the elements in the Data plane can be manipulated as per the needs, there is no need to configure each and every element of data plane individually. Network Function Virtualization (NFV) and Software Defined Networking (SDN) complement each other, although they do not depend upon each other [1].For designing, managing and decoupling networking services Network Function Virtualization (NFV) is used. The network functions such as Domain Name Services (DNS), Network Address Translation (NAT), and Intrusion Detection System (IDS) etc. are decoupled by the NFV from propriety hardware appliances as they can run in software [1].For providing the multiple services to the customers, service provider has to implement the multiple virtual network function (VNF) instead of single virtual network function (VNF) ,the new concept came where multiple virtual Network Function(VNF) are providing services through concatenation is called as service chaining after this Quality of Service (QOS) is the issue to handle in for [1]. NFV standardization committee is ETSI NFV group. On the other hand SDN is an emerging technology which separates central logic from its infrastructure. SDN can be centrally managed and dynamic changes can be made, it is cost effective and easy to use. It is founded by Open Networking Foundation (ONF) group.

II.BACKGROUND

In the traditional networks as shown in fig.1 both control plane and data plane are coupled inside the proprietary hardware. In a dedicated appliance network functionality is mainly implemented, 'dedicated appliance' refers to one or multiple switches, routers and/or application delivery controllers. Within this appliance Most of the functionality is implemented in dedicated hardware only and for this purpose, Application Specific Integrated Circuit (or: ASIC) is often used [2].

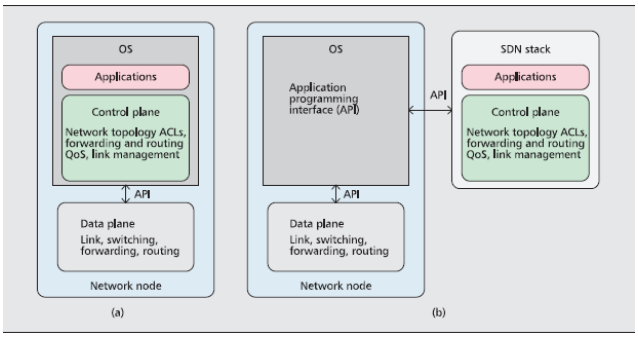

Fig.1 a) Traditional networks b) SDN approach is implemented

A. Limitations of Traditional Networks:

- **Network configuration was time consuming and Fickle:** Whenever an IT administrator needs to add or remove a single device in a traditional network many steps are needed. Firstly the manual configuration of multiple devices used in the network like switches,

routers, firewalls etc.The next step which he has to follow is to update numerous configuration settings, such as ACLs, VLANs and Quality of Service using device-level management tools. This configuration approach makes it that much more complex for an administrator to deploy a set of policies which are consistent [2].

- **Multiple vendors:** As there includes multiple physical devices in traditional networks so it implies for multiple vendor environment which ultimately needs high level of expertise and extensive knowledge of all the devices present in the network.

- **Distributed control plane:** The intelligence of the network resides in the control plane in case of traditional networks it is residing in multiple places because of coupling of both data plane and control plane in network devices. It becomes very difficult to manage the network for a network administrator as configuration was a bit complex [2].

Table 1 comparison of Traditional networking and SDN

Traditional Networking	**Software Defined Networking**
Inflexible and static networks.	They posses flexibility and agility.
These networks are logically distributed.	These are centralized.
Protocols are used for their working.	Different APIs (Application programming interface) are used such as Northbound APIs, Southbound APIs etc.)
Custom ASICs and FPGAs are used.	Merchant silicon is used.
Packet forwarding and high-level routing is done on the same device.	In this OpenFlow switch separates the control path and data path.
Configuration of switches used in it is done through command line.	SDN controller uses OpenFlow and gives you an interface to program the switched.

III.SDN ARCHITECTURE

As shown in Fig.2. SDN is a three layered architecture; the main layer is control layer because controller resides in it, and controller acts as a brain to the network because it manages the flow of traffic from switches using flow tables.

A. Features of SDN architecture are as follows:

Programmability is Direct: Because it is decoupled from forwarding functions network control is directly programmable [3].

Agility: Dynamically adjusting network wide traffic flow to meet network changing needs is easily achieved.

Management of network is central: Network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch.

Configuration is programmable: SDN lets network administrators to secure, configure, manage and optimize network resources very fast via dynamic, automated SDN programs, which they can be written by themselves because there is no more dependency on proprietary software [3].

Open standards-based and no more vendor-dependency : Through open standards when SDN is implemented ,it makes the network design and operations performed in a very simple manner because most of the instructions instead of multiple vendor-specific devices protocols ,are provided by SDN controllers(like POX,Ryu,OpenDaylight etc.)[4].

All three layers are dependent to each other and communicate with one another through some interfaces. The best advantage of SDN architecture is that it provides abstraction view of entire network for the applications it provides; this makes the network even more “Smarter”.

B.SDN Architecture contains the following three layers

- **Application Layer**: It is composed of the applications which are communicating with controller in control layer through some interfaces called as Northbound APIs.The commonly used API in providing Northbound API is REST (Representation State Transfer) API. Applications in SDN can be like Firewall, Load balancer etc [5].

- **Control Layer**: It is the middle layer of the SDN architecture and constitutes the SDN controller which acts as a brain of the network and has a global view over the network also known as Control plane.

- **Physical Layer**: It contains the infrastructure used in the network like switches, also known as Data plane. They provide packet forwarding and packet switching. Switches only perform the actions according to the controller. The interface they use to communicate with controller situated in control layer is called as Southbound APIs. The most common protocol used in providing Southbound APIs is OpenFlow Protocol [5].

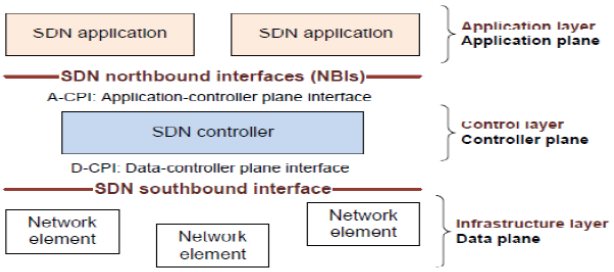

Fig.2 SDN Architecture

C. Network Interfaces used in SDN:

As discussed earlier also SDN is a 3-layered architecture top layer includes the high level instructions, controller resides in middle layer and the third layer constitutes all the physical & Virtual switches used in the network. Within a network each control device is equipped with some interfaces (one or more), every control device is able to communicate with other components through these interfaces. A network interface is a software or protocol which provides the communication medium through its interface between two equipment's or computer networks. The two types of API's used in SDN are as follows:

- **Southbound Application programming interface (API**): The communication between control layer and physical layer is done through this interface. For this many protocols are used like OVSDB, NETCONF, SNMP etc. but mainly OpenFlow protocol is used, it provides the programmatic control of forwarding rules from the data path given by network elements present in the physical layer. OpenFlow protocol is used in providing the interface between control layer and physical layer its first specification was created in 2008 by Stanford University. The first version of OpenFlow, managed by ONF (Open Network Foundation)was 1.1 then they released another version 1.2 in February 2011,most recent version of this is 1.4,as a southbound API it has to be implemented in both sides(control layer and physical layer) called as OpenFlow Switch. OpenFlow Switch is a switch which is actually managed by SDN controller. It consists of 1) Group Table. 2) OpenFlow Table. Switches are also of two types1) OpenFlow only (operations used in this are OpenFlow only, all packets are processed by the OpenFlow pipeline.2) OpenFlow-Hybrid (it uses both Ethernet switching operations and OpenFlow operations). Fig.2 shows the overall workflow in OpenFlow Switch used in communicating and providing interface between control layer and physical layer [6].OpenFlow switch are implemented in both the layers as well. Easy deploying, switching protocols new routing technique can be implemented using OpenFlow switch. In high security networks it can be used.

- **Northbound Application programming interface (API**): The communication between control layer and application layer is done through this interface. Applications using OpenAPI's information about the network state could be asked by the applications and further this Network control measures can be applied by the applications if they find it necessary. For example REST API (Representational State Transfer) is a software based architecture style for building scalable web services. Northbound API provides routing, security, data path computation & other basic network functions.

- **Westbound Application programming interface (API):** This interface acts as a channel for providing the interface between SDN control plane and different network domains [6]. Network state information and routing decisions for each controller is exchanged through this. Inter domain routing protocols like BGP are used.

- **Eastbound Application Programming interface (API):** communication is done from control plane to non SDN domains. Depends upon the technology used in non SDN domains its implementation is proportional [6].

The OpenFlow protocol is the most commonly used protocol for the southbound interface SDN, which separates the data plane from the control plane. OpenFlow was initially proposed by Stanford University, and it is now standardized by the ONF. OpenFlow is an open interface for remotely controlling the forwarding tables in network switches, routers, and access points.

OpenFlow architecture constitutes the three basic concepts.

1. With the help of OpenFlow-compliant switches (that compose the data plane.) network is built.
2. More than one OpenFlow controller is constituted in control plane of SDN network.
3. A secure control channel connects the switches with the control plane [7].

D. OpenFlow Switch:

An OpenFlow-compliant switch is a network's basic forwarding device. According to its flow table it forwards the packet to its destination. This flow table includes a set of flow table entries, each of which consists of match fields, counters and instructions, as illustrated in Figure 3.1. Flow table entries are also called flow rule or flow entries.

Header Fields	Counters	Actions

Fig.2

The **header fields** in a flow table are the first entry which describes that which packets this entry is applicable in for. Constitution of a wildcard-capable match over specified header fields of packets is included in it [7]. For faster packet forwarding with the help of OpenFlow, ternary content addressable memory (TCAM) is required by the switch that allows the fast lookup of wildcard matches. Depending on the OpenFlow specification, e.g., Ethernet, IPv4, IPv6 or MPLS. The header fields can match different the above stated protocols.

Collection of statistics about flows is reserved by the counters. The number of received packets and bytes, as well as the duration of the flow is stored by these **counters** only.

How the packets of that flow are handled is decided by the **actions** [7]. Some common actions are —forward drop, —modify field, etc.

E. OpenFlow Controller

A software program in SDN, called the controller, it is responsible for manipulating and populating the flow tables of

the switches and act as a brain of the network. By, modification, insertion and removal of flow entries from the flow table of switches the controller can modify the behavior of the switches with regard to forwarding the packets. The protocol that enables the controller to instruct the switches is OpenFlow. For the purpose of communication the controller uses a secure control channel.

The three classes of communication exist in the OpenFlow protocol those are followings:

- **Communication between controller-to-switch**: The controller-to-switch communication is responsible for, switch programming, information retrieval feature detection and its configuration.

- **Communication is asymmetric:** without any request from the controller asynchronous or asymmetric communication is get started by the OpenFlow compliant switch [8].It is used for the transfer of information to the controller about state changes packet arrivals, at the switch and errors occurred in between.

- **Communication is symmetric**: Symmetric messages are sent without any request from either side of switch and controller both controller and switch are free to start the communication without the permission from the other side. To check whether the control channel of network is still live or not a simple message of hello or some other echo messages are used to help in such cases are the examples of symmetric communication [8].

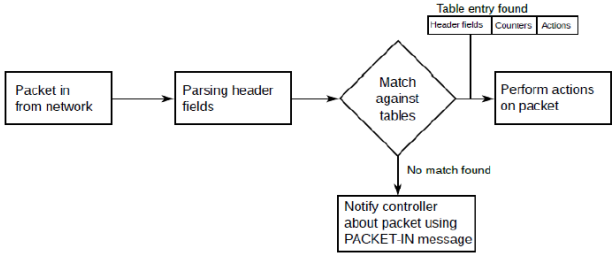

Fig.3 OpenFlow work flow

F.OpenFlow protocol:

Various OpenFlow protocol versions are discussed and these are as following:

- **OpenFlow 1.0:** The OpenFlow 1.0 specification was released in December, 2009. The most commonly deployed version of OpenFlow is this version of it. Based on the source and destination address Ethernet and IP packets can be matched. Specification of the packet parsing and matching algorithm is done by it [9]. The algorithm of packet matching starts with a comparison in between the Ethernet and VLAN fields and continues if necessary with IP header fields.

- **OpenFlow 1.1**: OpenFlow 1.1 was released in February, 2011. A drastic changes compared to OpenFlow 1.0 were contained by this version of OpenFlow. In multiple flow tables a packets that contain the information are now processed by a pipeline. Two major alterations are introduced in this and these are as follows:
  - A group table
  - A pipeline of multiple flow tables [9].

- **OpenFlow 1.2**: OpenFlow 1.2 was released in December, 2011. It has come to include the advanced support for protocol in particularly for IPv6. OpenFlow version 1.2 can now match IPv6 source as well as destination addresses, protocol number, flow label number, various ICMPv6 fields and traffic class. In additional matching capabilities now vendors have new possibilities to offer or make available OpenFlow by them to support in for.

- **OpenFlow 1.3**: OpenFlow 1.3 introduces new features for monitoring and operations and management (OAM). To a flow table entry a meter is attached directly and the rate of packets assigned to it is measured by its meter identifier. If a given rate is exceeded with the help of meter band instead of dropping all those packets from the network it may optionally choose to recolor all those packets by modifying their differentiated services (DS) field [9] .Thus on OpenFlow 1.3 and later specifications a simple or complex QoS frame works can be implemented.

- **OpenFlow 1.4**: This was released in October 2013. The support for the OpenFlow Extensible Match (OXM) was enhanced by ONF. To the protocol TLV structures for ports, tables and queues are added .Previously defined hard-coded parts from earlier versions and specifications are now replaced by the newly defined TLV structures. Now it is possible to have the configuration of optical ports. In addition to it, now the control messages in a single message bundle to all the switches can be sent by the controller. There is a minor improvement in group tables, flow eviction on full tables and feature of monitoring the traffic flow is also included [9].

IV. CONTROLLERS IN SDN

The vital element of SDN network is considered to be its controller. It is a platform which manages the flow of control to the routers and switches via Southbound OpenFlow protocol and applications via Northbound APIs. A collection of Pluggable modules is contained by controller which performs different network tasks.

Five most important commonly used controllers which are opensource.POX[10],Ryu[11],Trema[12],Floodlight[13],Open Daylight[14] apart from these above mentioned controllers there are many others controller like Jaxon,NOX, Beacon, Maestro etc. because of less usage and poorly documented these controllers are not used.

- **POX**: It is developed and inherited from NOX controller. POX is python based SDN controller.
  - Pythonic OpenFlow interface.

- Runs anywhere – Can bundle with installing-free Py runtime for easy distribution.
- The similar visualization tools and GUI as NOX are used. [10].

- **RYU:** It gives the component based platform for SDN, for managing the network flow and applications it uses different APIs. Ryu provides software components with well defined API that make it easy for developers to create new network management and control applications. For managing different network devices, such as OpenFlow, Netconf, OF-config, etc. Ryu supports plenty of protocols. About OpenFlow, versions 1.0, 1.2, 1.3, 1.4, 1.5 and Nicira Extensions all are supported by RYU. Under the Apache 2.0 license all of these codes are freely available.

- **Trema:** For developing different controllers which use OpenFlow protocol for configuring and connection to the network devices (switches, routers) through Southbound APIs called as OpenFlow Controller , Trema provides a framework (open source) to them in the programming language like c and ruby.

- **Floodlight**: It is a java based OpenFlow controller, managed by ONF (Open Networking Foundation) and licensed by Apache. It specifies a "Forwarding instruction set" in which a remote controller can make changes in network behavior through some defined protocols through switch.

- **OpenDaylight:** It the largest open source SDN controller, managed ONF (Open Networking Foundation).A flexible common platform is provided by OpenDaylight which serves many purposes like Automated Service Delivery, NFV and cloud, Network Visibility and control, Network Resource Optimization [15]. Model-driven service abstraction platform that allows users to write applications that easily work across a wide variety of hardware and south-bound protocols is provided by OpenDaylight. The OpenDaylight Controller is able to deploy in a variety of production network environments. Upcoming protocols and other SDN standards are supported by this modular controller. The OpenDaylight Controller exposes open northbound APIs, which are used by applications. The Controller is used by the applications to collect information about the network and then algorithms are run to conduct analytics, and then again make use of OpenDaylight Controller to create new rules throughout the network. Within its own Java Virtual Machine (JVM) OpenDaylight is kept and implemented singly in software [15].

	POX	Ryu	Trema	FloodLight	OpenDaylight
Interfaces	SB (OpenFlow)	SB (OpenFlow) +SB Management (OVSDB JSON)	SB (OpenFlow)	SB (OpenFlow) NB (Java & REST)	SB (OpenFlow & Others SB Protocols) NB (REST & Java RPC)
Virtualization	Mininet & Open vSwitch	Mininet & Open vSwitch	Built-in Emulation Virtual Tool	Mininet & Open vSwitch	Mininet & Open vSwitch
GUI	Yes	Yes (Initial Phase)	No	Web UI (Using REST)	Yes
REST API	No	Yes (For SB Interface only)	No	Yes	Yes
Productivity	Medium	Medium	High	Medium	Medium
Open Source	Yes	Yes	Yes	Yes	Yes
Documentation	Poor	Medium	Medium	Good	Medium
Language Support	Python	Python-Specific + Message Passing Reference	C/Ruby	Java + Any language that uses REST	Java
Modularity	Medium	Medium	Medium	High	High
Platform Support	Linux, Mac OS, and Windows	Most Supported on Linux	Linux Only	Linux, Mac & Windows	Linux
TLS Support	Yes	Yes	Yes	Yes	Yes
Age	1 year	1 year	2 years	2 years	2 Month
OpenFlow Support	OF v1.0	OF v1.0 v2.0 v3.0 & Nicira Extensions	OF v1.0	OF v1.0	OF v1.0
OpenStack Networking (Quantum)	NO	Strong	Weak	Medium	Medium

Fig.4 SDN Different Controllers

V.SOFTWARE DEFINED NETWORKING WITH MININET

For the creation of a network of virtual hosts, switches, controllers, and links in between them a network emulator is used called a Mininet emulator. Standard Linux network software is run by Mininet hosts, and mininet switches support OpenFlow protocol for the purpose of communication and highly flexible custom routing and ultimately creating Software-Defined Networking. Mininet not only supports research and development but it also aids in learning, prototyping, testing, debugging and any other tasks .Instead of having a real hardware with the help of mininet a complete experimental network can be made on a laptop or other PC.

A. Working with Mininet:

Within a single machine emulation is done by mininet with an OpenFlow network and end-hosts are connected within the network. The built-in support to create several common topologies (default and custom) is a great feature of it. The construction of custom topologies is done using a python script. The comfort, advantage and authenticity is provided by the mininet at very low cost. The substitute to Mininet is hardware test beds which are agile, exact but due expensive and shared nature cannot be used for all experiments. Mininet is freely available open source software that emulates OpenFlow devices and SDN controllers. Simulation of SDN networks, a controller (POX, Ryu, OpenDaylight etc.) for experiment is run by mininet. Emulation of real world network scenarios is provided by it and few of SDN controllers are by default included with in Mininet VM,for using some other controllers (as per the need) there installation is required. With the help of Mininet ease usability, scalability in the network and accuracy in performance can be achieved.

These following steps are to be followed for installing Mininet emulator in your PC:

- Download the Mininet VM image.

- Downloading and installation of virtualization system like VMware or virtual box is required at this stage.

- A Sign up is required for the Mininet-discuss mailing list. Mininet support and consultation with the

Mininet community, friendly is present there in case if you need any type of help.

- VM Setup Notes can be go through at this stage if required and then log in to the VM and customization is done as per the wish.

- To accelerate for additional tutorials go to Mininet [16].

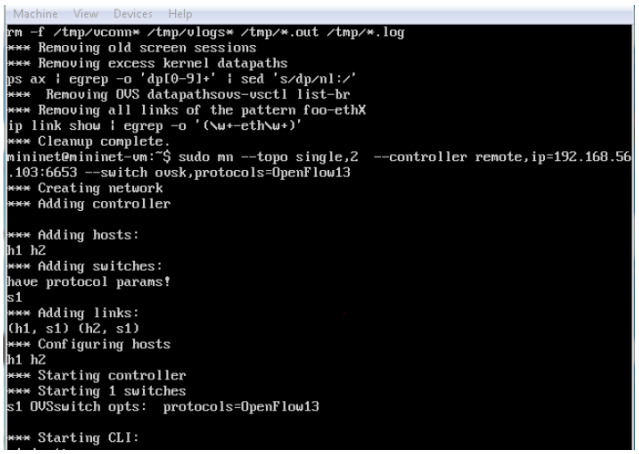

Fig.4.1 Mininet

B. Mininet Commands

- **$ sudo mn –h:** This is used for displaying a help message describing Mininet startup option where $ proceeds Linux command that should be typed in root shell prompt.

- **$ sudo mn:** The default topology is the minimal topology which is very simple topology that contains OpenFlow switch and 2 hosts. It also creates links between switch and two hosts. This show following results:

  ***creating controller
  ***adding controller
  ***adding hosts:
  h1 h2
  ***adding switches:
  s1
  ***adding links:
  (h1,s1) (h2,s1)
  ***configuring hosts
  h1 h2
  ***starting controller
  c0
  ***starting controller
  s1
  ***starting CLI
  Mininet>

- **Mininet>net** : This command shows following results:
  mininet> net
  h1 h1-eth0:s1-eth1
  h2 h2-eth0:s1-eth2
  s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0

- **Mininet>dump**
  This will show switches and hosts listed.
  mininet> dump
  <Host h1: h1-eth0:10.0.0.1 pid=8683>
  <Host h2: h2-eth0:10.0.0.2 pid=8684>

- **mininet> nodes**
  available nodes are:
  c0 h1 h2 s1

  mininet> pingall
  *** Ping: testing ping reachability
  h1 -> h2
  h2 -> h1
  *** Results: 0% dropped (2/2 received)

C. Mininet topologies

Default topologies such as minimal, single, reversed, linear and tree are contained by Mininet although apart from these default topologies custom topologies can also be created in mininet as per the need while creating or experimenting any application used in SDN environment [17].

- **Minimal:** Minimal is very simple topology that contains 1 OpenFlow switch and 2 hosts.Fig.4.2 shows the diagrammatic representation of minimal topology. The code used to create this topology in mininet is also given as follows:

  sudo mn --topo minimal
  mininet> net
  h1 h1-eth0:s1-eth1
  h2 h2-eth0:s1-eth2
  s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0

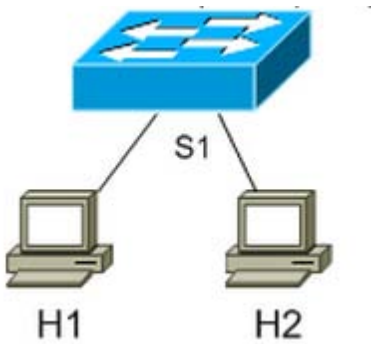

Figure 4.2: Minimal topology

- **Single**: This topology includes one OpenFlow switch with k hosts [17]. A link between switch and k hosts is also created as shown in figure 5.3.The code used in mininet for the single topology is as follows:

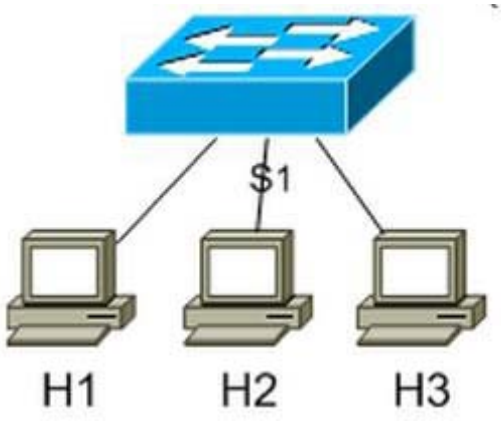

Figure 4.3: Single topology

```
sudo mn --topo single,3
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0
```

- **Reversed***:* Reversed topology is same as that of single topology. The only difference between Reversed topology and single topology is the order of connection [18]. In case of reversed the order is opposite to single topology.

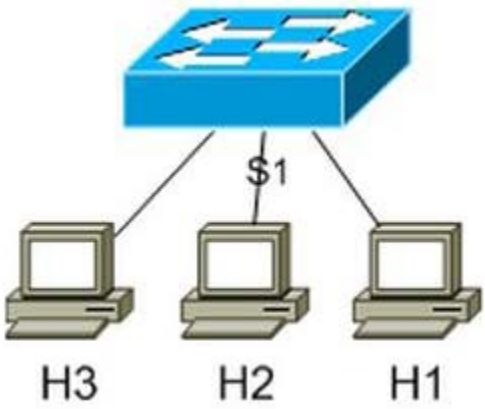

Figure 4.4: Single topology

- **Linear***:* Linear topology is a combination of k switches and k hosts. A link between every switch and every host and all the switches is created to create a topology as shown in the figure 4.5 [18] .The code used to create this topology in mininet is as follows:

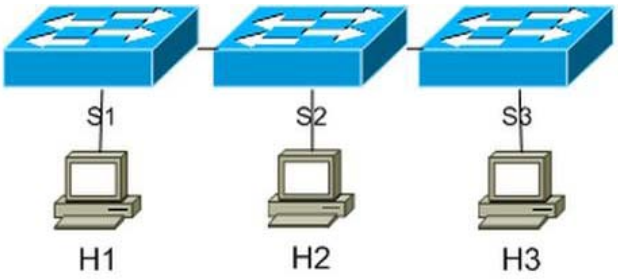

Figure 4.5: linear topology

```
sudo mn --topo linear,3
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
s1 lo: s1-eth1:h1-eth0 s1-eth2:s2-eth2
s2 lo: s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
s3 lo: s3-eth1:h3-eth0 s3-eth2:s2-eth3
```

- **Tree**: To per switch (s1, s2 ….s8) this topology contains *m* levels and 2 hosts (H1, H2….H8) are attached as shown in fig.4.6 [18].

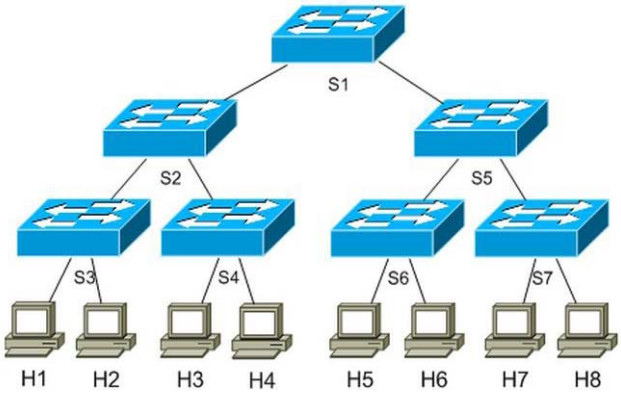

Figure 4.6: Tree topology

VI. SECURITY THREATS IN SDN

For building a SDN environment, the security of each and every component of SDN is to be ensured. Following is some of the essential security requirements for securing the SDN key components [19].

- ***Security of the controller*:** Top priority should be Securing the SDN controller. The responsibility for the overall management of the network is of SDN controller. The controller acts as a central decision points for whole the whole network, it can lead (centralized behavior of SDN controller) to the disaster of the entire network.

  SDN controller is a single point of failure and also acts as a target for attackers as well. For the whole network the availability of the SDN controller in such a way is of serious concern [19]. In reconfiguring the complete network the SDN controller, essentially supports a hacker in it .The address of an SDN controller to a fake controller through spoofing can be done and can be taken over the whole network. Defense mechanism in depth is needed for this and from physical and external threats the protection of the system containing the SDN controller should be included as well. From different attacks like DoS and distributed DoS (DDoS) attacks it should be prevented. Security of the operating system must be there, which includes no patches, back-door accounts

open doors at the same time, like vulnerable open ports, services, and protocols [20].

- ***Attack in denial of services*** — Take an example of a POX controller, for creating a particular virtual network placed on the end user side placement of the special controller is done. Knowledge of the network is contained by the controller and it is prone to many attacks such as DoS attacks. Generation of huge number of flows which ultimately rendering network breakdown or improper functioning can be done by the attacker during this [21, 22].

- ***Attack through spoofing*** —launch of a simple and easy spoofing attack can be done in SDN network [21] in which one person or program successfully masquerade the SDN controller as another by falsifying data, and hence gaining an illegitimate advantage of useful data inside the network.

- ***Injecting Malicious things inside the network —*** Malicious injections can be made by simply following the existing Field Rewrite problem, to change the VLAN ID tag subject in particular circumstances is provided to end users. To inject packets into another slice .This situation simply creates an opportunity for a nasty controller [22].

VII. CONCLUSION

Due to the Dynamic management of traffic in networks provided by SDN technology, more bandwidth is available to the users. No more dependency is there on dedicated hardware which is a cost effective way too. An abstracted view of network is provided. SDN is considered to be the best solution for meeting the new demands in networking. As SDN is an emerging technology so, research is still going on in order to make it more efficient way of networking. It is hoped that introduction about SDN its architecture and Controllers discussed here will prove to be helpful for the researchers working in this area.

ACKNOWLEDGMENT

We would like to thank almighty for his constant blessings. Then we would like to dedicate our gratitude towards parents, teachers, family, friends, and in essence, all sentient one beings.

REFERENCES

[1] Jammal,Manar,Taranpreet Singh,Abdallah Shami,Rasool Asal,Yimming Li.”Software Defined Networking:State of the Art And Research Challenges”Elsevier Computer networks 72(2014)74-98

[2] Wickboldt , Juliano Araujo, Wanderson Paim de Jesus, Pedro Heleno Isolani, Cristiano Bonato Both, Juergen Rochol, and Lisandro Zambenedetti Granville “Software-Defined Networking: Management Requirements and Challenges “, IEEE Communications Magazine,January 2015

[3] N. Feamster, J. Rexford, and E. Zegura, “The Road to SDN: An Intellectual History of Programmable Networks,” SIGCOMM Comput. Commun. Rev., vol. 44, no.2, Apr. 2014, pp. 87–98.

[4] Kim, H. and N. Feamster, “Improving Network Management with Software Defined Networking,” IEEE Commun. Mag., vol. 51, no. 2, Feb. 2013, pp. 114–19.

[5] Wenfeng, Xia ,Yonggang Wen, Senior Member, IEEE, Chuan Heng Foh, Senior Member, IEEE,Dusit Niyato, Member, IEEE, and Haiyong Xie, Member, IEEE.” A Survey on Software-Defined Networking.”

[6] Jarschel, Michael, Thomas Zinner, Tobias Hoßfeld, Phuoc Tran-Gia, and Wolfgang Kellerer.”Interfaces, Attributes, and Use Cases:A Compass for SDN”. IEEE Communications Magazine, June 2014.

[7] Nick, McKeown, Tom Anderson, Hari Balakrishnan Guru Parulkar,Larry Peterson, Jennifer Rexford.Scott Shenker. Jonathan Turner “OpenFlow: Enabling Innovation in Campus Networks”, March 14, 2008.

[8] OpenFlow Switch Specification, March 26, 2015.

[9] OpenFlow, http://www.openflow.org/

[10] Python at https://www.python.org/

[11] Ryu at https://osrg.github.io/ryu/

[12] Trema at https://github.com/trema/trema

[13] Floodlight http://www.projectfloodlight.org/floodlight/

[14] OpenDaylight at http://www.opendaylight.org/

[15] Figuerola, A.;” OpenDaylight as a Controller for Software Defined Networking” Spring 2014/2015.

[16] Mininet at www.mininet.org

[17] Kaur, Karamjeet, Japinder Singh, and Navtej Singh Ghumman. "Mininet as Software Defined Networking Testing Platform.

[18] DeCusatis, Casimer, Aparicio Carranza and Jean Delgado-Caceres “Modeling Software Defined Networks using Mininet”, Proceedings of the 2nd International Conference on Computer and Information Science and Technology (CIST’16) Ottawa, Canada – May 11 – 12, 2016.

[19] A. Akhunzada et al., “Man-At-The-End Attacks: Analysis,Taxonomy, Human Aspects, Motivation and Future Directions,”J. Network and Computer Applications, 2014.

[20] L. Wei et al., “Security and Privacy for Storage and Computation in Cloud Computing,” Info. Sciences, vol. 258, 2014, pp. 371–86

[21] Q. Duan, Y. Yan, and A. V. Vasilakos, “A Survey on Service-Oriented Network Virtualization Toward Convergence Of Networking and cloud Computing,” IEEE Trans. Network and Service Management, vol. 9, no. 4, 2012, pp. 373–92.

[22] Akhunzada, Adnan, Ejaz Ahmed, Abdullah Gani, Muhammad Khurram Khan, Muhammad Imran, and Sghaier Guizani “Securing Software Defined Networks: Taxonomy, Requirements, and Open Issues”. IEEE Communications Magazine, April 2015.

