Implementation of Software Defined Network for Telecommunication Network

Ramaswamy. T, SPV. Subba Rao, Y. Gita Madhuri

Abstract*— *This paper fundamentally centers around the formation of the Software Defined Network (SDN). SDN is rising system engineering. In the conventional system, every last gadget is controlled and checked separately, and the significant downside is it is seller subordinate and very costly. We utilize virtual switches and virtual switches in the SDN engineering. SDN Technology and Evaluate SDN by making an exploratory system utilizing an Open Day Light Controller being in the control plane and the Virtual Box is utilized for making Virtual Routers and switches being in the Data Plan. The work likewise controls the Instructions in making a SDN based Network. The connection between the Controller and Network gadgets will be setup in the south bound API by utilizing a convention called Open Flow.

Keywords: Software Defined Network; Virtualization; Open Day Controller; Open switch; Open Flow Protocol

1. INTRODUCTION

A PC arranges is a mind boggling innovation that is utilized to empower the end gadget to speak with one another. When all is said in done the system more often than not comprises of the Routers, Switches, Web Servers, Firewalls, Load Balancers, interruption preventions frameworks and different gadgets. While preparing and overseeing colossal measure of information over a system we need to think about the productivity, unwavering quality, adaptability and robustness [1]. In Traditional system, each gadget must be controlled and overseen exclusively. The real Drawback is the Traditional Networks are Vendor subordinate. The control Plane and the Date Plane are coupled. Hence we can infer that the Traditional Networks are Inflexible and complex to manage.The above Drawbacks of the Traditional systems can be overwhelmed by utilizing the SDN. SDN will bolster numerous test system and emulator devices. SDN additionally underpins many Open Source and Commercial instruments, for example, Virtual Box, Network Simulators [NS3], EstiNet, MiniNet[8]. The above devices will assist us with implementing the test arrange before it is executed continuously. These instruments are more cost effective for Testing, Designing and Optimizing. This endeavor made its exploratory framework in perspective of Wireless framework thought and the Virtual box is picked as the suitable gadget. Virtual Box can be introduced on any OS machine, for example, Linux, MACOS, Windows, and Solarises. Different Guest OS's under a Single OS can be stacked and every visitor can be taken care of separately.

Revised Version Manuscript Received on 05 April, 2019.

Ramaswamy T. Associate Professor, Departmentt of ECE, Sreenidhi Institute of Science and Technology, Telangana, India

SPV. Subba Rao, Professor, Department of ECE, Sreenidhi Institute of Science and Technology, Telangana, India

Y. Gita Madhuri, M.Tech (DSCE), Department of ECE, Sreenidhi Institute of Science and Technology, Telangana, India.

Table I Comparing Different Simulators.

Simulators	EstiNet	NS3	Mininet
OpenFlow Version	1.3.1	0.8.9	1.3.1
Simulation(S)/Emulation(E)	S/E	S/E	E
Wireless Functionality	YES	NO	NO
Controller Compatibility	YES	NO	YES
Extendibility	NO	YES	YES

To test the SDN exhibitions progressively there are numerous business test beds or apparatuses are utilized, one of them are GEANT Open Flow Facility (GOFF) from GEANT Network[14]. These proving grounds are costly and can't be gotten to effortlessly. In this way, we will utilize the Open Source instrument for the experimentation and it is the correct answer for analysts. The Outline of the work is: We will examine about making a virtual system by utilizing the SDN Technology.

2. RELATED WORKS

A.Software-Defined Networking (SDN)

The possibility of SDN, its engineering, segments are to be talked about here. In extra the Data plan and Control plan will be secured independently for better understanding[7]. The correspondence between South Bound, Controller and North Bound Application Programming Interface will be depicted. The execution of Open Day Light (ODL) Controller is given:

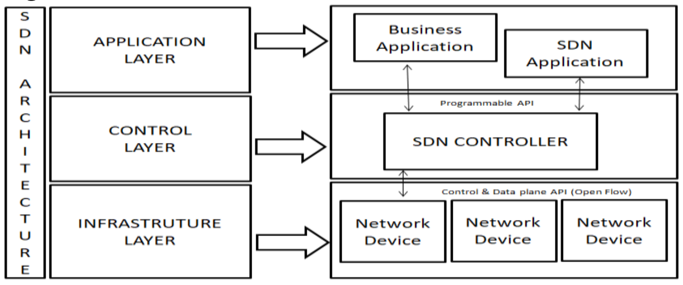


Fig: 1 Software Defined Network Architecture

*Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication*

1) SDN Architecture and components:

The SDN development depend on 4 major standards, for example, (i)Data Plane or Forwarding plan and control plan are decoupled.(ii)The Control plan choice is stream base as opposed to choice based.(iii)The control plan and Data plane rationale is preoccupied from equipment to programmable programming layer.(iv) The Controller is acquainted with co-ordinate arrange wide sending decisions[10]. The SDN design depends on partition of the sending plan and the control plan though the control plan is coherently finding the concentrated control plan.

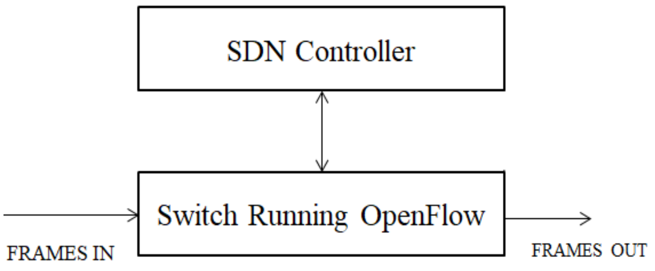


Fig: 2 Software Defined Network

In view of design the SDN can be sensibly spoken to in 3 Layers.

(a)Infrastructure Layer: This layer is generally alluded as the sending (or) Data plan. The switches and switches are executed. In the exploration we utilize the virtual switches and switches which are actualizing by Open Source programming "VM Machine".

(b)Control Layer: It is additionally called as Control Plan. The Network is an arranged so that it is consistently unified and programming constructed SDN controller is introduced with respect to any UNIX constructed OS keep running in light of any equipment and introduce's sending rules.

(c)Application Layer: Application and directors take authority over the Control plan and on the Infrastructure layer through illustrative state exchange API (Application Programming Interface). The SDN ideas normally help's designers to effectively build up the application that performs organizing assignments. The application is typically sent to isolate PC (or) cloud. There are 2 API in SDN engineering Northbound and Southbound. Southbound interface builds up the associations between systems administration gadgets and the controller's utilizing an Open Flow convention. Northbound is association between controller the application.

B.Software-Defined Network Controllers:

The controller assumes a noteworthy job in the SDN arrange. The controller dwells on the control plan. There are different SDN controllers available in the market, for instance Open Contrail, Flow Visor, Floodlight, ONOS and Open Day Light (ODL) Controller. In this test we have pick the ODL controller since it underpins the LINUX Foundation. The platinum patrons which support and develop the endeavor that makes it tried and true for use[1]. The ODL will permit the NON-Open Flow convention to be utilized and gives such capacities as grouping, Load Balancer, multi-Layer organize enhancement and others. The ODL controller has all around organized documentation what's more, describes API for making distinctive SDN application

There are different number of programming switches, for example, OpenvSwitch(OVS), CPqD programming Switch, LINC switch, Pantou. In this endeavor we pick the OpenvSwitch as the product switch since it is Open Source programming switch and executed by the LINUX foundation[12]. It is perfect with Open Flow 1.0 and 1.3. The OVS can be utilized with Flexible controller.

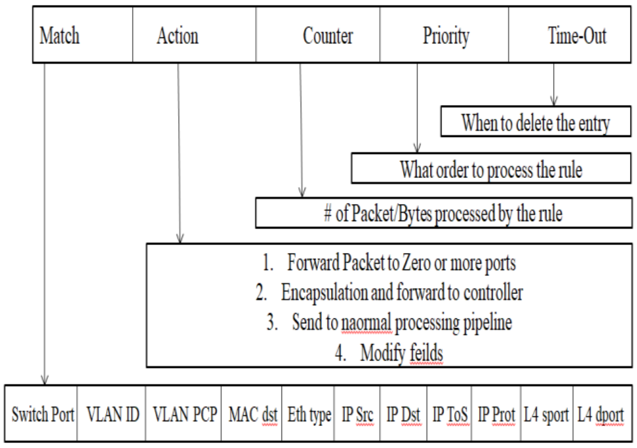

Match
Action
Counter
Priority
Time-Out
When to delete the entry
What order to process the rule
# of Packet/Bytes processed by the rule
1. Forward Packet to Zero or more ports
2. Encapsulation and forward to controller
3. Send to naormal processing pipeline
4. Modify feilds
Switch Port
VLAN ID
VLAN PCP
MAC dst
Eth type
IP Src
IP Dst
IP ToS
IP Prot
L4 sport
L4 dport

Fig: 3 Open V switch.

C. Open Flow Protocol

This convention is actualized, while the switch is speaking with the controller that is, while the information plane is speaking with the controller. It is an institutionalized correspondence convention that provides for the sending plane of a system. It more often than not directs us the best approach to control the practices of the switches all through our system progressively and programmability.

D.Virtualization

The virtualization is normally alluded as making a virtual adaptation of a gadget that is for our situation a Virtual Switch and Virtual Router as opposed to making a real gadget. The virtualization enables single gadget to carry out the activity of various gadgets, by sharing the asset of single equipment over different. Virtualization is generally utilized for the server union, Testing and Development, Dynamic Load Balancing, Disaster Recovery, Improved System Reliability and System Security.

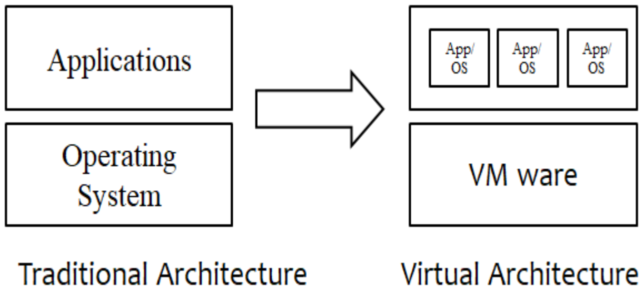


Fig: 4Difference Between Traditional and Virtual Networks

3. OPENDAYLIGHT CONTROLLER

The OpenDaylight controller is an open-source controller underneath the Linux Establishment people group adventures. It is an item application and has a worked in Java Virtual Machine (JVM) it very well may be continued running on any working framework that sponsorships Java. It involves a couple of squares: Northbound REST APIs, System Administration Capacities, System Coordination Capacities, Administration Reflection Layer and Southbound APIs.

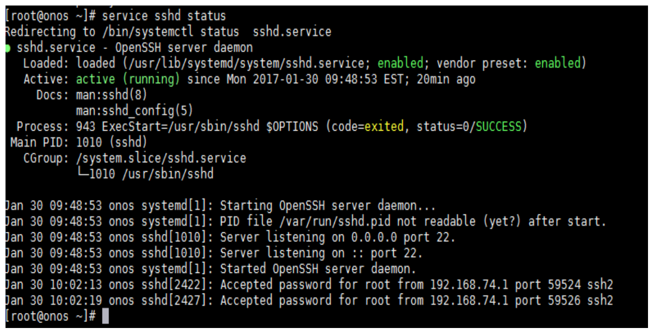
The most ebb and flow controller interpretation (Lithium at the period of forming) is plot using Model Driven Programming Designing (MDSE) by Protest Administration Gathering (OMG). MDSE depicts a structure in light of solid associations between (different) models, organized mappings and precedents that enable show age and, by enlargement, code/Programming interface age from models[3]. This hypothesis can overlay a specific modeming lingo. Despite the way that OMG focus their answer on Bound together Displaying Dialect (UML), YANG has created as the data showing vernacular for the frameworks organization space.There are open northbound REST APIs accessible for application use. Open Administration Portal Interface (OSGi) structure is in addition accessible for applications. This structure is utilized for running the applications in a similar location space as the controller, while the REST APIs are utilized by web applications that don't remain in a relative location space (or hardware) as the controller.

Organization Deliberation Layer was picked as estimated blueprint of the OpenDaylight controller. This layer enables the virtual gadget to help distinctive traditions on the southbound and furthermore gives sensible organizations to modules and applications.While the OSGi structure allows a dynamic associating for different southbound traditions, organizations, for instance, Gadget Revelation being used by modules like Topology Chief are given by the SAL. For whatever time allotment that organizations are fabricated using the features revealed by the modules, the SAL can manual for the fitting module and use the most reasonable southbound tradition for frameworks organization device correspondence in perspective of the organization inquire. Addresses the past determined module interfacing.

Installation Of Software

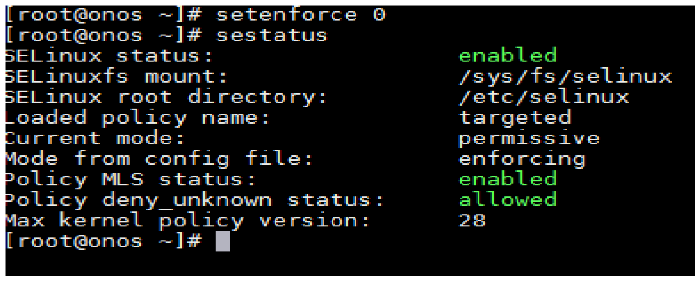
I.Mininet Installation:

Step 1: Install SSHD
yum –y install openssh-server
chkconfigsshd on
servicesshd start



```
[root@onos ~]# service sshd status
Redirecting to /bin/systemctl status  sshd.service
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2017-01-30 09:48:53 EST; 20min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 943 ExecStart=/usr/sbin/sshd $OPTIONS (code=exited, status=0/SUCCESS)
 Main PID: 1010 (sshd)
   CGroup: /system.slice/sshd.service
           └─1010 /usr/sbin/sshd

Jan 30 09:48:53 onos systemd[1]: Starting OpenSSH server daemon...
Jan 30 09:48:53 onos systemd[1]: PID file /var/run/sshd.pid not readable (yet?) after start.
Jan 30 09:48:53 onos sshd[1010]: Server listening on 0.0.0.0 port 22.
Jan 30 09:48:53 onos sshd[1010]: Server listening on :: port 22.
Jan 30 09:48:53 onos systemd[1]: Started OpenSSH server daemon.
Jan 30 10:02:13 onos sshd[2422]: Accepted password for root from 192.168.74.1 port 59524 ssh2
Jan 30 10:02:19 onos sshd[2427]: Accepted password for root from 192.168.74.1 port 59526 ssh2
[root@onos ~]#
```

Step 2: Disable SELinux to get OVSDB to start
setenforce 0



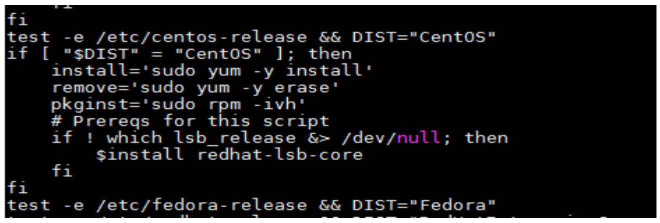
```
[root@onos ~]# setenforce 0
[root@onos ~]# sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   permissive
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      28
[root@onos ~]#
```

Stage 3: Install Git
yum – y introduce git
Stage 4: Get Mininet from the git store
git clone git://github.com/mininet/mininet.git

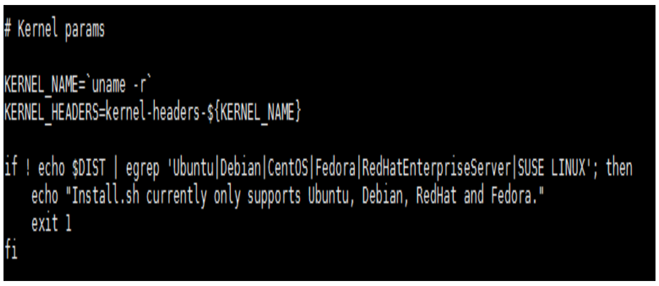
II.Mininet Installation On Centos:

test - e/and so on/centos-discharge && DIST="CentOS"

in the event that ["$DIST" = "CentOS"]; at that point
install='sudo yum - y introduce'
remove='sudo yum - y delete'
pkginst='sudo rpm - ivh'
# Prereqs for this content
in the event that !whichlsb_release&>/dev/invalid; at that point
$install redhat-lsb-center
fi
***EDIT
in the event that !resound $DIST | egrep 'Ubuntu|Debian|Fedora|CentOS'; at that point
resound "Install.sh as of now just backings Ubuntu, Debian and Fedora."
leave 1
fi



```
fi
test -e /etc/centos-release && DIST="CentOS"
if [ "$DIST" = "CentOS" ]; then
    install='sudo yum -y install'
    remove='sudo yum -y erase'
    pkginst='sudo rpm -ivh'
    # Prereqs for this script
    if ! which lsb_release &> /dev/null; then
        $install redhat-lsb-core
    fi
fi
test -e /etc/fedora-release && DIST="Fedora"
```



```
# Kernel params

KERNEL_NAME=`uname -r`
KERNEL_HEADERS=kernel-headers-${KERNEL_NAME}

if ! echo $DIST | egrep 'Ubuntu|Debian|CentOS|Fedora|RedHatEnterpriseServer|SUSE LINUX'; then
    echo "Install.sh currently only supports Ubuntu, Debian, RedHat and Fedora."
    exit 1
fi
```

Introduce Mininet and OpenFlow reference yet not OVS by giving a direction:

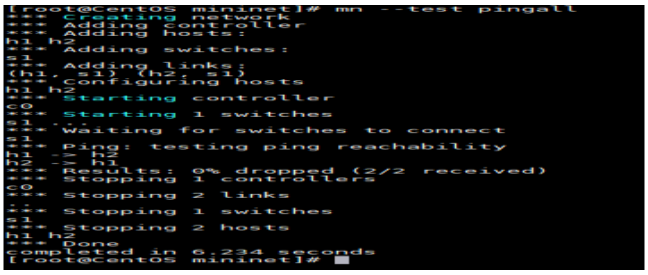
/root/mininet/util/install.sh– nf or ./install.sh – nf

III.Mininet Custom Topology:

Check Default topology:

sudo mn – test pingall

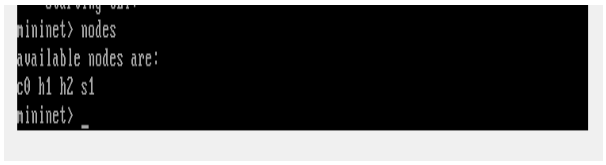
The above topology can make the system of two hosts, one switch with two connections.



```
[root@CentOS mininet]# mn --test pingall
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 6.234 seconds
[root@CentOS mininet]# 
```

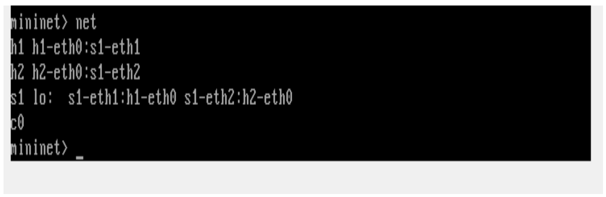
IV. Basic Mininet Commands:

➢ Nodes



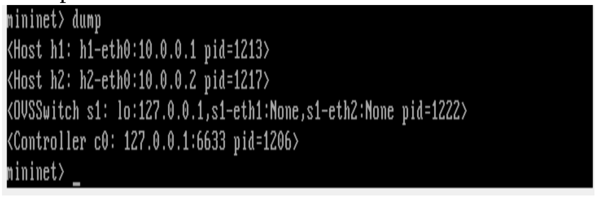
```
mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet> _
```

➢ Net



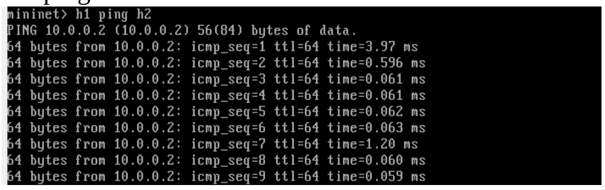
```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet> _
```

➢ Dump



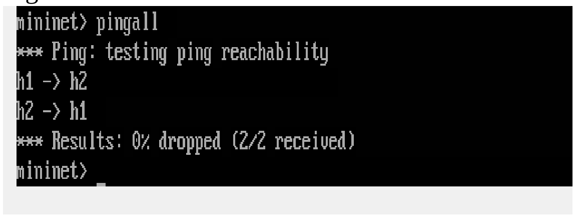
```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=1213>
<Host h2: h2-eth0:10.0.0.2 pid=1217>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=1222>
<Controller c0: 127.0.0.1:6633 pid=1206>
mininet> _
```

➢ h1 ping h2



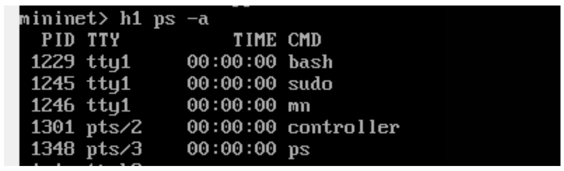
```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=3.97 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.596 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.061 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.061 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.062 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.063 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=1.20 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.060 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.059 ms
```

➢ Pingall



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

➢ h1 ps– a

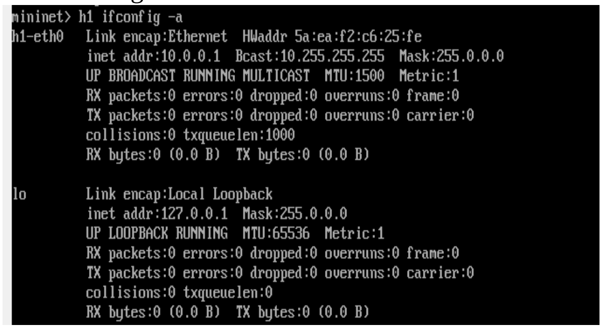


```
mininet> h1 ps -a
  PID TTY          TIME CMD
 1229 tty1     00:00:00 bash
 1245 tty1     00:00:00 sudo
 1246 tty1     00:00:00 mn
 1301 pts/2    00:00:00 controller
 1348 pts/3    00:00:00 ps
```

➢ · ifconfig

h1 ifconfig -a

s1 ifconfig -a



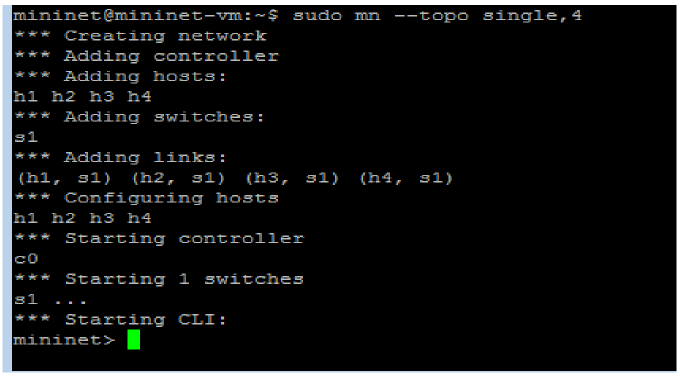
```
mininet> h1 ifconfig -a
h1-eth0   Link encap:Ethernet  HWaddr 5a:ea:f2:c6:25:fe
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

V. Diverse Mininet Topologies:

➢ Single Topology

$ sudo mn - topo single,4:

By utilizing this order, system of four hosts , one switch and one controller can be made.

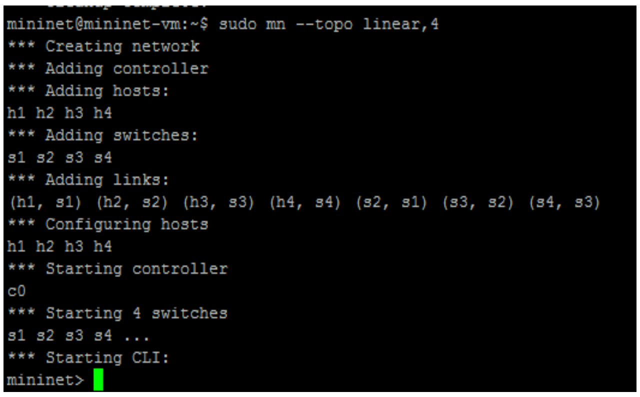


```
mininet@mininet-vm:~$ sudo mn --topo single,4
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> 
```

➢ Linear Topology

$ sudo mn - topo linear,4

By utilizing this direction, system of four hosts , four switch and one controller can be made.

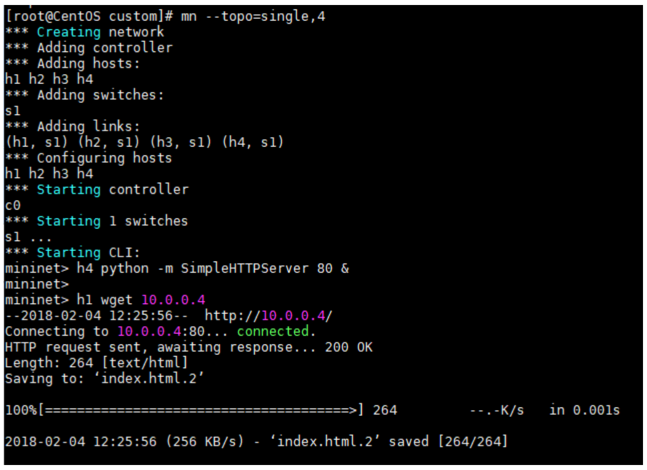


```
mininet@mininet-vm:~$ sudo mn --topo linear,4
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (s2, s1) (s3, s2) (s4, s3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
mininet>
```

VI. Mininet Custom Topologies:

➢ $sudo mn – topo=single,4

By utilizing this direction, system of four hosts , one switch and one controller can be made in which all the four hosts are connected to single switch and single controller.



```
[root@CentOS custom]# mn --topo=single,4
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h4 python -m SimpleHTTPServer 80 &
mininet>
mininet> h1 wget 10.0.0.4
--2018-02-04 12:25:56--  http://10.0.0.4/
Connecting to 10.0.0.4:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 264 [text/html]
Saving to: ‘index.html.2’

100%[======================================>] 264         --.-K/s   in 0.001s

2018-02-04 12:25:56 (256 KB/s) - ‘index.html.2’ saved [264/264]
```

4. EXPERIMENTAL RESULTS

The Experimental system has been setup as outlined in the Fig:2. In this, the experiments are illustrated in detail so that it will be useful for the future use. The experiment is conducted in the emulation part. For emulation part, Mininet has been utilized to produce a topology comprising of two hosts, one open flow switch and one controller. The open Flow change was set to be OVS switch. The ODL controller runs script for Open Flow Switch.

Table ii. Hardware Specification In Real Time Experiment

Device Name	Hardware	Operating System
Host 1 (h1)	Lenovo ideapad 100	Ubuntu 14.04.4 LTS
Host 2 (h2)	Lenovo ideapad 100	Ubuntu 12.04.4 LTS
Controller	MacBook Pro	Ubuntu 14.04.4 LTS

The Host 1 and Host 2 are created on one Laptop which is installed in the Lenovo ideapad 100 where as the Controller is installed on a different laptop which is MacBook Pro in our case.

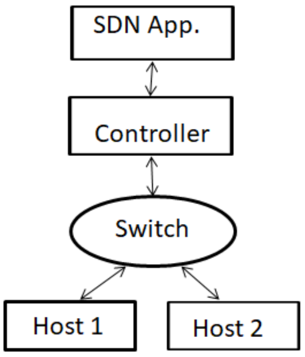


Fig: 5 SDN Network Testing Model

In the above Testing Model, it has been illustrated network which is being implemented in the experiment.

Table iii. Experimental Parameters

Parameter	Setting
Emulation Tool	Mininet 2.1
Link Speed	100Mbps
OpenFlow vSwitch	2.4.0
Controller	Open Day Light COntroller
TCP Window Size	85 KBytes

The Experimental framework has been setup as plot in the Fig:2. In this, the analyses are shown in detail with the goal that it will be helpful for the future utilize. The investigation is directed in the copying part. For copying part, Mininet has been used to create a topology containing two hosts, one open stream switch and one controller. The open Flow change was set to be OVS switch. The ODL controller runs content for Open Flow Switch.

We have utilized Mininet, on the grounds that we can make a system with the single order. For instance, we can experience the underneath directions.

The essential order used to make the system is:

>sudo mn

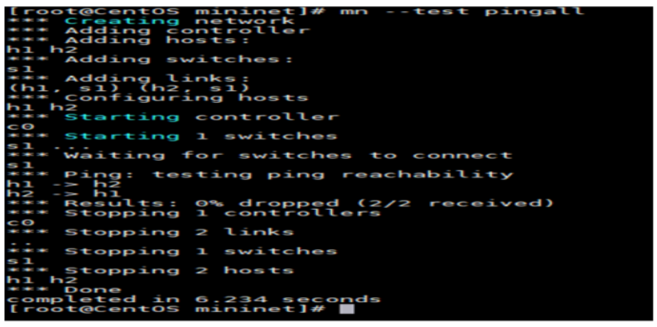
By utilizing the above order we make an essential virtual system with host and controller and virtual switch.

To build up an association among every single virtual gadget, the direction utilized is:

>sudo mn-test pingall

In our analysis, We will utilize two PCs. By utilizing the main Laptop we will introduce the MiniNet and making the virtual switch and two virtual hosts. This PC is put at a settled area that is the place we typically used to set our server framework. By utilizing the second workstation we will introduce the controller that is for our situation an Open Day light Controller, by utilizing the putty programming. This will be conveyed by the server administrator. The server administrator can work the system which was made in the primary PC from anyplace by utilizing the server. This server will be made while making the virtual system in first workstation. The yield of the virtual system made in the principal workstation is given beneath.

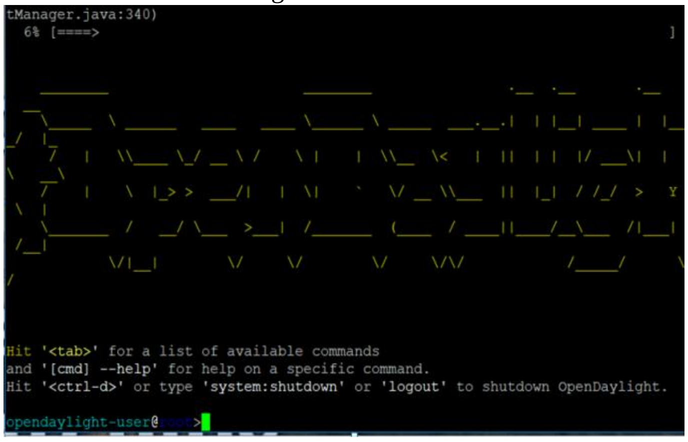
Its yield is given as:



```
[root@CentOS mininet]# mn --test pingall
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 6.234 seconds
[root@CentOS mininet]#
```

Fig: 3 Network Output

The output of the second laptop, that is by using which we can create a controller is given below:



```
tManager.java:340)
  6% [====>                                                                  ]

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.

opendaylight-user@root>
```

Fig: 4 Open Day Light Controller Output

Thus by using the Software Defined Network we can create the virtual and cost efficient networks with high security.

5. DISCUSSION

According to the results, the endeavor accomplished its goals – the SDN system was worked in a virtualized area with some moved features presented. The OpenDaylight controller, which was presented in the midst of the endeavor, can be easily used using a virtualized organize. Additionally, a case application, which was presented on an alternate VM, could bring NFV features into the structure. Despite the way that the OpenDaylight controller (Lithium interpretation at the period of making) similarly gives abundance, its use has requirements. It is great just with a segment of the features open for foundation. Prior to utilizing them, it is recommended to check at the wiki-pages5 for more point by point data. I expected to make another copy of the controller remembering the ultimate objective to test the application.

6. CONCLUSION

The target of the undertaking is to build a SDN system using the OpenDaylight controller in a virtualized condition, was accomplished. Alongside the controller, a system including virtual systems administration gadget was presented. The OpenDaylight can be used as an essential open-source controller if a customer needs to make reliable and present-day systems giving that he or she can add to or impact to his or her own specific moves up to the source.

As the last result, the SDN innovation is included and answered to the endeavor with built up rules. That can be a purpose behind future SDN commitments in regards to a realistic learning system if understudies need to test with present-day SDN innovation. Future endeavor changes can be attempted after the most present version called Beryllium will be released. Those could be: VTN use with a group of controllers and presenting streams using DLUX GUI. Also, if Metropolia has hardware with OpenFlow support, a SDN structure could facilitate into its grounds establishment.

REFERENCES

1. Open Networking Foundation, "Programming Defined Networking: The New Norm for Networks," in ONF White Paper, April 2012.
2. C. Alaettinoglu, "Programming Defined Networking," in PACKET DESIGN, 2013.
3. NS3 OpenFlow switch bolster in adaptation 3.18. Accessible: http://www.nsnam.org/docs/discharge/3.18/models/html/openflowswitch.html
4. Gustavo J.A.M Carneiro. NS-3:Network Simulator 3. Accessible: https://www.nsnam.org/instructional exercises/NS-3-LABMEETING-1.pdf
5. S.Y. Wang, C.L. Chou, and C.M. Yang, "EstiNet 8.0 OpenFlow Network Simulator ans Emulator," in Vol.51, Issue 9, September 2013.
6. Mininet: An Instant Virtual Network on your Laptop (or other PC). Accessible: http://mininet.org
7. M. Chan, C. Chen, JX. Huang, T. Kuo, LH. Yen, and CC. Tseng, "OpenNet: A Simulator for Software-Defined Wireless Local Area Network," IEEE WCNC'14, April, 2014.
8. C. Argyropoulos, B. Arslan, J. Aznar, K. Baumann, K. Dombek, E. Escalona, D. Guija, E. Jacob, A. Juszczyk, J. Melnkov, A. Mendiola, S. Naegele-Jackson, T. Ogrodowczyk, D. Pajin, D. Pamiewicz, R. van der Pol, M. Przwecki, "Deliverable D13.1 (DJ2.1.1) Specialized Applications' Support Utilizing OpenFlow/SDN," GEANT, March, 2015.
9. Open Networking Foundation, "OpenFlow-empowered Transport SDN (ONF Solution Brief)," May 2014.
10. J. Postel, "Transmission Control Protocol," in RFC 793, September 1981.
11. Open Networking Foundation, "OpenrFlow Switch Specification variant 1.5.1 (Protocol form 0x06)," March 2015.
12. POX wiki. Accessible: https:// openflow. stanford. edu/ show/ONL/POX +Wiki
13. Ryu 3.18 documentation: WELCOME TO RYU THE NETWORK OPERATING SYSTEM (NOS). Accessible: http://ryu.readthedocs.org/en/most recent/index.html
14. Pyretic. Accessible: http://http://frantic lang.org/pyretic/
15. Trema. Accessible: https://trema.github.io/trema/
16. Floodlight.Available:http://www.projectfloodlight.org/floodlight/
17. ONOS. Accessible: http://onosproject.org/
18. OpenDayLightController.Available:http://www.opendaylight.org
19. A. Shalimov, D. Zuikov, D. Zimarina, V.Pashkov, and R.Smeliansky, "Propelled Study of SDN/OpenFlow Controllers," CEE-SECR '13, October 2013.
20. R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou, "Highlight based Comparison and Selection of Software Defined Networking (SDN) Controllers," WCCAIS 2014, October 2014.
21. K. Kaur, S. Kaur, and V. Gupta, "Perfomance Analysis of Phyton Based OpenFlow Controllers," EEECOS 2016, June 2016.
22. Open Networking Foundation, "OpenFlow Switch Specification Version 1.2 (Wire Protocol 0x03)," December 2011.
23. Open Networking Foundation, "OpenFlow Switch Specification Version 1.3.0 (Wire Protocol 0x04)," June 2012.
24. B. Pfaff, J. Pettit, T. Koponen, Ethan J. Jackson, A. Zhout, J.Rajahalme, J. Net, A.Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado, "The Design and Implementation of Open vSwitch," NSDI'15, May, 2015.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

25. CPqDOpenFlow1.3SoftwareSwitch.Available:http://cpqd.github.io/ofsoftswitch13/
26. LINC-OpenFlowsoftwareswitch.Available:https://github.com/FlowForwarding/LINC-Switch
27. Pantou:OpenFlow1.0forOpenWRT.Available:http://archive.openflow.org/wk/index.php/Pantou : OpenFlow 1.0 for OpenWRT
28. OpenFlow1.3forOpenWRT.Available:https://github.com/CPqD/ofsoftswitch13/wiki/OpenFlow-1.3-forOpenWRT
29. OpenFlow for OpenWRT OpenFlow Wireless Network. Accessible: https://github.com/Farzaneh1363/OpenFlow-for-OpenWRT-OpenFlowwireless-arrange/wiki
30. Turning TP-LINK WR1043NDv2.1 switch into OpenFlow-empowered switch. Accessible: http://ljdelight.com/transforming tp-interface wr1043ndv2-1router-into-openflow-empowered switch/.
31. OpenWrtWirelessFreedom.Available: https://downloads.openwrt.org/previews/trunk/ar71xx/conventional/
32. iPerf - The system data transfer capacity estimation instrument. Accessible: https://iperf.fr/
33. Wireshark.Accessible: https://www.wireshark.org/#learnWS
34. tshark - Dump and investigate arrange traffic. Accessible: https://www.wireshark.org/docs/man-pages/tshark.html