Build SDN with OpenFlow Controller

Pratiksha Kallure[1], Arati Kaljate[2], Ashwini Kumbhar[3], Kajal Mane[4]

[1,2,3,4]*Department of Computer Engineering, VIIT, Pune, India*

---***---

Abstract — Network usage and demands square measure growing at a really quick rate and to fulfill this necessities, there's a requirement for automatic infrastructure scaling. Computer that guarantees to dramatically change network-control and therefore the management plane is achieved through innovative network programmability. OpenFlow, one among techniques of SDN technology, may be a new approach to networking thanks to several limitations, the event of the standard load equalization technology has encountered bottlenecks. This has forced corporations to search out new load equalization methodology. Software Defined Network (SDN) provides an honest methodology to resolve the load equalization drawback. We have a tendency to square measure proposing Firewall System for SDN atmosphere that identifies malicious behavior or attacks and report to network directors as intrusion events. During this paper, we have a tendency to enforced load equalization rule that supported the newest SDN specification. We have a tendency to square measure victimization Dijkstra's rule to search out multiple methods of same length that permits North American country to scale back the search to a little region within the fat tree topology. The technologies we have a tendency to describe alter network operators to implement a large vary of network policies during a high-level policy language that simply confirm sources of performance issues additionally to the system themselves .We have a to describe paradigm deployments in field network that demonstrates however SDN will improve common network management task.

Keywords: Software Defined Network, Floodlight Controller, Firewall, Load balancer.

I. INTRODUCTION

NEtwork management is difficult to operate, maintain and secure communication network. Therefore, the network operators should use tool to low-level vendor specific configuration to implement advanced high level network policies. Despite several previous proposals to form network management easier, several solutions to network management issues quantity to get rid of gap solutions thanks to the issue fixing the underlying infrastructure.

Todays, network operator should implement more and more intelligent policies and complicated tasks with a restricted and extremely affected set of low-level device configuration commands during a statement interface atmosphere. Not solely square measure network policies low-level, they are conjointly not well equipped to react to repeatedly ever-changing network conditions. Progressive network configuration ways will implement a network policy that deals with one photo of the network state. However, network state changes frequently, and a result configuration changes square measure frequent and unwieldy, resulting in frequent change configuration.

Network operators want higher ways that configure and manage their networks. Sadly, today's network typically involve integration and interconnection of the many proprietary, vertically integrated devices. This vertical combination make it improbably tough for operators to specify high-level network-wide policies victimization current technologies. Innovation in network management has so been restricted to stop-gap techniques and measures, like tools that analyze low-level configuration to find errors or otherwise reply to network events. Proprietary computer code and closed development in network devices by a couple of vendors create it very tough to introduce and deploy new protocols. Progressive "updates" to configuration ways and commands square measure usually determined unilaterally by vendors. Meanwhile, operator's necessities for additional practicality and more and more advanced network policies still expand.

In Software defined Network, administrators can kind traffic by programing the management at management plane whereas not interrupting networking devices (Switches) at information plane. Management (Control) plane includes a centralized controller, that sets forwarding rules in switch to route traffic from source to destination. Information (Data) plane, that consist of networking devices handles all packets according to flow entries set by controller. With this ability, SDN technique introduces to handle the cloud service suppliers challenges like energizing load traffic, further metric demand, and security and quantifiability issues. Enterprise and organization use openFlow primarily based SDN to balance the traffic load, direct the traffic, manage on demand, metric demand and execute policies to scale the network. The Software Defined Networking (SDN) approaches has been earlier adopted by many corporation for his or her business considerably cloud and telecommunication services. The traditional technologies not capable to drive their current business challenges, thus SDN network technique supported code.

The reminder of this paper is structured as follows:

In section II, we have a tendency to provide an outline of the firewall system, load balancing technology and openFlow technology; In short summarizes that analysis standing of openFlow based load balancing. In section V, introduced the implementation of our 2 algorithms. In additional detail one rule for firewall and different one for load balancing. The results concerning our experiments square measure delineated in section VI. We have a tendency to summarize the most contribution of this paper in section VIII.

LITERATURE SURVEY

1] Paper Title: Brocade SDN in campus Networks.

Abstract:

Traditional network environment need application specific policies like security and access control, Virtual Local Area Network (VLAN) traffic isolation and Quality of Service (QoS) to be provisioned across the network one switch at time. This consumes a big amount of resources and results in a static network that can't be simply updated as business needs evolve or new applications needs to deployed. In contrast, SDN-enabled networks will dynamically allocate network resources in real time to satisfy the need of running applications. Custom-build or pre-packaged SDN applications running on the SDN controller will use input from several sources together with predefined applications-specific security and QoS needs, Physical network statistics, user activity, security threat analysis, then on to allocate and protect network resources, set access control rules, and prioritize traffic in a fully dynamic fashion.

Introduction:

Legacy architectures and networking protocols have reached a breaking point as they struggle to meet agility and flexibility requirements. New architecture and new protocols are required to confirm the secure, free flow of information on the campus. The Brocade hyper Edge design (discussed in separate white paper) allows a new networking topology of the Hyper Edge design. OpenFlow, running on Brocade switches, will either have an effect on all traffic on a given link or work in tandem with traditional protocols that utilize the Brocade hybrid per-flow mode. This enables legacy architectures and networking protocols have reached a breaking point as they try to meet agility and flexibility requirements.

Conclusion:

SDN in transforming the network to control in a very 21st-century applications. The change is similar to the transformative nature of moving from DOS to windows: moving from knowing far too much about how hardware worked to utilizing software without an awareness of or need to know about the underlying hardware. In this transformation, Brocade understands the necessity to exist and permit for a migration from the previous to the new.

2] Paper Title: Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design choices.

Abstract:

They justify the notion of software defined networking (SDN), whose south-bound interface could also be implemented by OpenFlow protocols. They describe the operation of OpenFlow and summarize the features of specification versions 1.0-1.4. We provide an overview of existing SDN-based applications sorted by topic areas. Finally point out architectural design choices for SDN using openflow and discuss their performance implications.

Introduction:

In this paper, introduce the notion of SDN using the definition from the Open Networking Foundation (ONF). OpenFlow is simply option for control protocol in SDN, however it's the predominant one. As OpenFlow presently evolves, many versions of its specification exist; newer releases are more powerful, as their feature sets support IPv6, Multi Protocol Label Switching (MPLS), rerouting, metering, policing and additional scalable control. However more existing applications are supported version 1.0, whose feature set is quite restricted. They will highlight however SDN contributes to network innovation by various networking applications that were implemented and analyzed in recent years. These applications fall within the areas of network management and traffic engineering, security, network virtualization and routing improvement.

Conclusion:

Review the concept of software defined networking (SDN) and explained the features of various specifications of the OpeFlow protocol, that may be suggests that to implement SDN-based networks. We illustrated the various SDN applications in various areas that improve network management and operation, enterprise networks and middle box routing, security problems and inter-domain routing.

3] Paper Title: SDN-based Enterprise and Campus Networks: A case of VLAN Management

Abstract:

Software Defined Networking (SDN) features the centralized network management and network programmability, it is a promising resolution for handling the said challenges in VLAN management. In this paper, first introduce a new architecture for campus and enterprise networks by investing SDN and OpenFlow. Next, they need designed and implemented an applications for simply managing and flexibly troubleshooting the VLANs during this design. This application supports each static VLAN and dynamic VLAN configurations. Additionally, they discuss the hybrid mode operation wherever the packet process is concerned by each the OpenFlow control plane and also the traditional control plane. By deploying a true model, they illustrate however our system works and so judge the network latency in dynamic VLAN operation.

Introduction:

The concept of Virtual Local Area Network (VLAN) was used for the primary time around 30 years ago. Even though it has a long history, it is still used as the most popular network virtualization technologies in today's enterprise and campus networks. The VLAN in enterprise and campus network is usually used to group in administrative domains and disregards their physical locations in the network topology. Example groups are engineers, sales, student clusters, managers, school clusters etc. For Example, VLAN will facilitate network directors produce several smaller broadcast domains from a large on instead of using expensive routers. As surveyed in, the authors declared that VLAN may be a helpful mechanism for limiting the scope of broadcast traffic, enforcing security and privacy policies, simplifying access control, decentralizing network management, and enabling host mobility.

Conclusion:

Leveraged SDN and OpenFlow into the campus and enterprise network environment. Our main contribution is that we designed and implemented a system for configuring a VLAN network in both a static and dynamic manner. It addresses the complexity of configuring VLANs, which network administrators must deal with when managing VLANs in campus and enterprise network.

II. PROPOSED ARCHITECTURE

The proposed architecture has been implemented using implementations such as Floodlight SDN controllers.

Data Usage: Operators sometimes specify policies whereby the behavior of network depends on the amount of data usage (download/upload) or data transfer rate over a particular time interval.

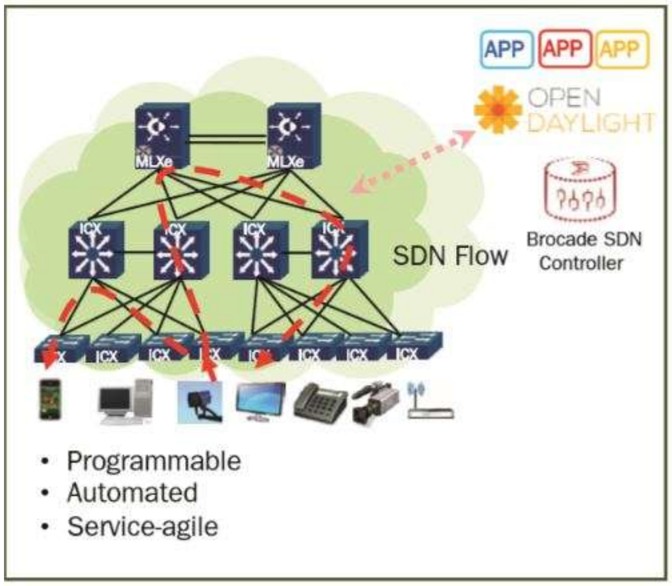


Fig3.1. Proposed architecture for network

Time: Network operators typically got to implement policies wherever network behavior depends on the date or time of day. In a home network, users may need to use the time of day because the basis for parental management.

Status: An operator may wish to specify privileges for different users or moreover, a user's privilege or standing typically changes thanks to varied reasons. A device's privilege ought to modification per the user who is presently exploitation the device.

Flow: Network operators wish to specify totally different network behaviors supported varied field values in multiple layers, specified in a packet or flow. A flow could be a 12-tuple management domain that already exist within the Open Flow specification. This network contains a remote controller, six switches; each with a host. This topology is utilized to test different principles and strategies characterized in the firewall design. The firewall application in a SDN to block particular host-to-host correspondence based on the time of the day. It recognizes the IP addresses to be blocked. The practical virtual systems will create, running genuine portion, switch and application code on a solitary machine utilizing Mininet Firewall policies were defined and rules added to the firewall table which changes progressively taking into account necessity and time of the day. Selective obstructing of packets in view of source and destination IP address will carried out. Application processing logic implemented using the floodlight Open Flow Controller. Network will tailored to treat the packet flows for video, audio.

III. TECHNICAL BACKGROUND

A] SDN Controllers

An SDN management is Associate in nursing application in computer code outlined networking that manages flow control to alter programmable networking. SDN controllers square measure supported protocols like OpenFlow, that permit servers to inform switches wherever to send packets.

Five uppermost open supply controllers in terms of their usage are analyzed here: POX, Trema, Ryu, Floodlight and OpenDaylight to grasp and understand the SDN thought.

Here we are using Floodlight Controller. The Floodlight open SDN controller is enterprise-class, Apache-licensed, Java-based OpenFlow Controller. It is supported by a community of developers and the variety of engineers from Huge Switch Networks are also enclosed. OpenFlow is an open standard managed by Open Networking Foundation. It specifies a protocol through switch an overseas controller will amendment the behavior of networking devices through a well-defined

"Forwarding Instruction Set". The Controller are designed to work with switches, virtual switches, routers and access points that support the OpenFlow standard. We are using Floodlight Controller v1.2 version.

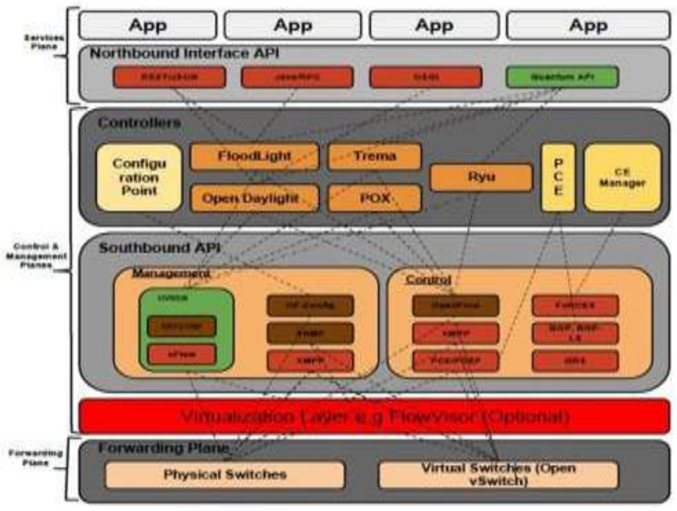


Fig.4.1 SDN Controllers

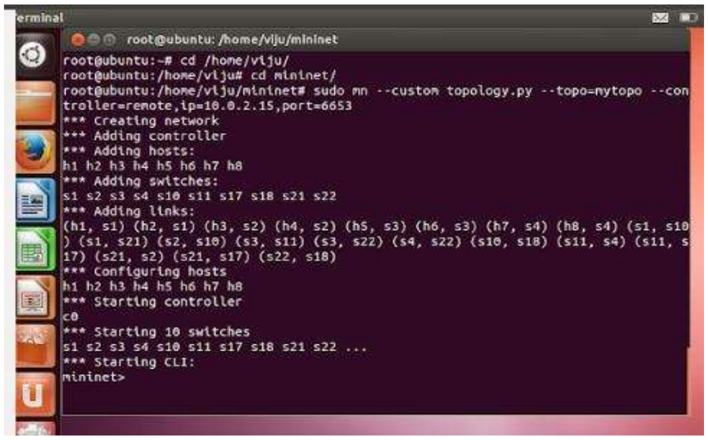
B] OpenFlow Protocol

OpenFlow provides a mechanism for SDN. Whereas OpenFlow was 1st planned as the simplest way to alter researchers to conduct experiments in field of networks, its advantages leads to its use beyond that. OpenFlow's central-control model will avoid the necessity to construct international policies from switch-by-switch configuration, and might conjointly support near-optimal traffic management. The behavior of forwarding devices may be summarized in 2 steps: (1) Once it receive a packet that doesn't match a precise entry in its routing table, it contacts the controller that defines however the packet to be forwarded or discarded; (2) Once the received packet matches a rule in routing table, the corresponding action within the forwarding table is performed.

C] Mininet

Mininet is a network emulator which creates a virtual network of virtual hosts, switches, controllers, and links. Mininet hosts run commonplace Linux network computer node, and its switches support OpenFlow for highly flexible custom routing and Software-Defined Networking. It supports custom topology. But cojointly provides an easy and protrusible python API for network creation and experimentation.

The figure 4.2 shows the creation of custom topology.



```
root@ubuntu:~# cd /home/viju/
root@ubuntu:/home/viju# cd mininet/
root@ubuntu:/home/viju/mininet# sudo mn --custom topology.py --topo=mytopo --con
troller=remote,ip=10.0.2.15,port=6653
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s10 s11 s17 s18 s21 s22
*** Adding links:
(h1, s1) (h2, s1) (h3, s2) (h4, s2) (h5, s3) (h6, s3) (h7, s4) (h8, s4) (s1, s10
) (s1, s21) (s2, s10) (s3, s11) (s3, s22) (s4, s22) (s10, s18) (s11, s4) (s11, s
17) (s21, s2) (s21, s17) (s22, s18)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 10 switches
s1 s2 s3 s4 s10 s11 s17 s18 s21 s22 ...
*** Starting CLI:
mininet>
```

Fig. 4.2 Mininet Fat Tree Topology Creation

D] Firewall System

In OpenFlow switch the ingress packet matched against the flow table and sent to the controller if no match is found. The controller chooses what to try and do with the packet and sends it back to the switch. The switch then executes the activity that controller is characterized. In the event that the packet within the flow table matches the table then the activity is execute. If the packet doesn't match against list then it will block that address.

E] Load Balancing Technology

The main goal is to perform load balancing but at the same time we would like to confirm that latency is minimum. Here are using Dijkstra's algorithm to help us to find multiple paths of same length which enables us to reduce the search to a small region in the fat tree topology. In the program simply finds the path with least load. It will forwards traffic on that path.

IV. IMPLEMENTATION AND EVALUATION

A. Experimental environment for firewall system:

It is a firewall application implemented in a Software Defined Network, using Mininet and python based on custom topology as shown in the figure 4.2. In figure 4.2, a small network is ready to setup on virtual machine, with Mininet installed. This network contains remote controller, 10 switches; every with a host. This topology is used to check completely different principles and methods categorized within the firewall design. Algorithm - firewall.py: The code actualizes a firewall application during a SDN to block specific host-host correspondence supported the time of the day. It recognizes the ip address to be blocked.

B. Experimental environment for load balancing system:

It is load balancer application implemented in a Software Defined Network, using Mininet and python based on custom topology as shown in the figure 4.2. In figure 4.2, a small network is ready to setup on virtual machine, with Mininet installed. This network contains remote controller, 10 switches; every with a host. This topology is used to check completely different principles and methods categorized within the firewall design. Algorithm- loadbalancer.py: The code actualizes a load balancing application in a SDN to balance network.

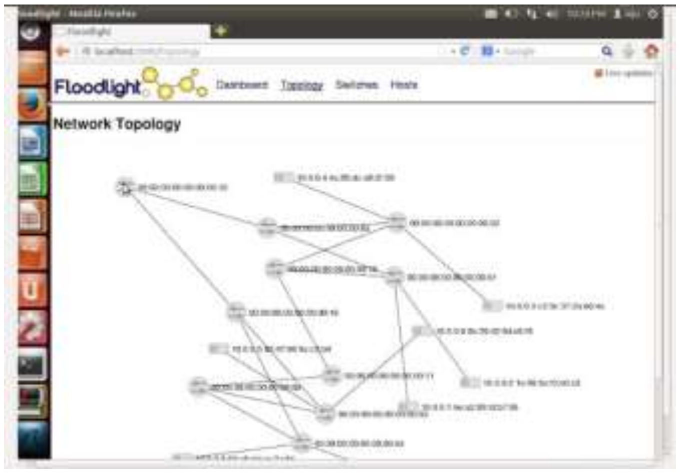


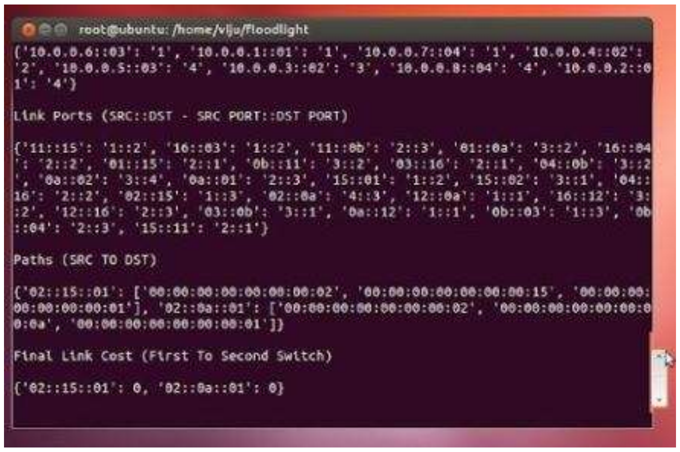
Fig. 5.1 Topology for Load Balancer System

C .EXPERIMENTAL PROGRAM FOR FIREWALL SYSTEM:

In the system run the FloodLight Controller. Run the Fat Tree custom topology topology.py. Type the following command in Mininet xterm h4 h7. Now ping h4 to h7 and h7 to h4. Here H7 is allowed host. Go to terminal open the new tab Ctrl + Shift + T and type sudo wireshark. In wireshark, click on Capture -> Interfaces and Select s1-eth4 and begin the capture. You will see the packets are transferring. Now go to your terminal open a new tab and run the firewall.py script and it will display device information and it will block the h7. In Wireshark, click on Capture -> Interfaces and select s1-eth4 and begin the capture. No packets are present.

D. EXPERIMENTAL PROGRAM FOR LOAD BALANCER SYSTEM:

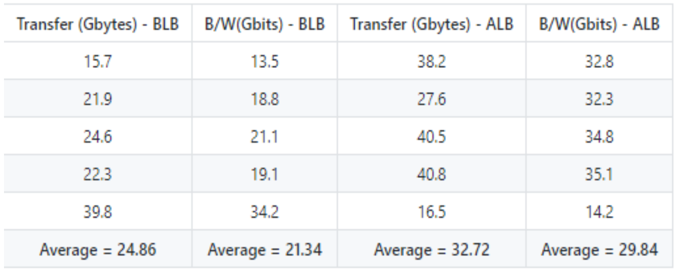
In the system run FloodLight Controller. Run the Fat Tree custom topology topology.py. Type the following command in Mininet xterm h1 h1. In 1st console of h1 type, ping 10.0.0.3. In 2nd console of h1 type, ping 10.0.0.4. Go to Terminal open the new tab Ctrl + Shift + T and type sudo wireshark. In Wireshark, click on Capture -> Interfaces and choose s1-eth4 and begin the capture. In filter section of wireshark type ip.addr == 10.0.0.3 and check if you are receiving packets for h1->h3. Do the same thing for h1->h4. Once you see packets, you will figure that this can be the most effective path. However to verify it is repeat the above to steps for s1-eth3 and you will notice that no packets are transmitted to this port. Only packets it will receive are going to be broadcast and multicast. Ignore them. Now in the 2nd console of xterm of h1, stop pinging h4. Our goal is to make congestion on the most effective path of h1->h3, h1->h4 and vice versa and h1 pinging h3 in enough for that go to your terminal and open new tab and run the loadbalancer.py program by providing input arguments like 1, 4, 3 wherever one is host 1, four is host 2, and three is host 2's neighbor. Look at the topology above and you will notice that this hosts are nothing but h1, h4 and respectively. The loadbalancer.py performs rest requests, thus initially the link cost will be 0. Re-run the script few times. This may vary range from 1-10 times. Here the most effective root is for h1->h4 and vice versa. Now on 2nd console of h1 type, ping 10.0.0.4. Go to wireshark and monitor interface s1-eth4 with the filter ip.addr==10.0.0.x wherever x is 3 and 4. You will notice 10.0.0.3 packet however no 10.0.0.4 packets stop the above capture. And now do the capture on s1-eth3, s21-eth1, s21-eth2, s2-eth3 with filter ip.addr==10.0.0.x where x is 3 and 4. You will notice 10.0.0.4 packets but no 10.0.0.3 packets.



```
root@ubuntu: /home/vlju/floodlight
{'10.0.0.6::03': '1', '10.0.0.1::01': '1', '10.0.0.7::04': '1', '10.0.0.4::02':
'2', '10.0.0.5::03': '4', '10.0.0.3::02': '3', '10.0.0.8::04': '4', '10.0.0.2::0
1': '4'}

Link Ports (SRC::DST - SRC PORT::DST PORT)

{'11::15': '1::2', '16::03': '1::2', '11::0b': '2::3', '01::0a': '3::2', '16::04
': '2::2', '01::15': '2::1', '0b::11': '3::2', '03::16': '2::1', '04::0b': '3::2
', '0a::02': '3::4', '0a::01': '2::3', '15::01': '1::2', '15::02': '3::1', '04::
16': '2::2', '02::15': '1::3', '02::0a': '4::3', '12::0a': '1::1', '16::12': '3:
:2', '12::16': '2::3', '03::0b': '3::1', '0a::12': '1::1', '0b::03': '1::3', '0b
::04': '2::3', '15::11': '2::1'}

Paths (SRC TO DST)

{'02::15::01': ['00:00:00:00:00:00:00:02', '00:00:00:00:00:00:00:15', '00:00:00:
00:00:00:00:01'], '02::0a::01': ['00:00:00:00:00:00:00:02', '00:00:00:00:00:00:0
0:0a', '00:00:00:00:00:00:00:01']}

Final Link Cost (First To Second Switch)

{'02::15::01': 0, '02::0a::01': 0}
```

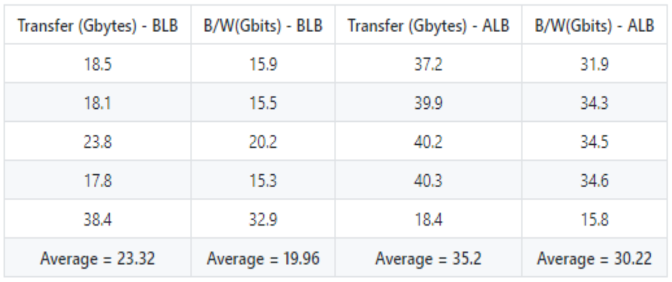
Fig. 5.2 Output of Load Balancer System

V. RESULT



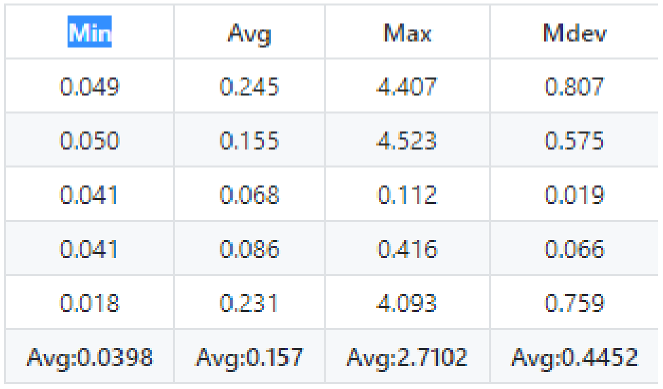
Transfer (Gbytes) - BLB	B/W(Gbits) - BLB	Transfer (Gbytes) - ALB	B/W(Gbits) - ALB
15.7	13.5	38.2	32.8
21.9	18.8	27.6	32.3
24.6	21.1	40.5	34.8
22.3	19.1	40.8	35.1
39.8	34.2	16.5	14.2
Average = 24.86	Average = 21.34	Average = 32.72	Average = 29.84

iPerf H1 to H3 Before Load Balancing (BLB) and After Load Balancing



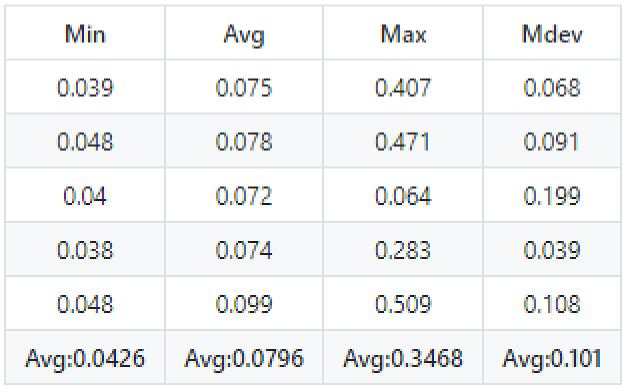
Transfer (Gbytes) - BLB	B/W(Gbits) - BLB	Transfer (Gbytes) - ALB	B/W(Gbits) - ALB
18.5	15.9	37.2	31.9
18.1	15.5	39.9	34.3
23.8	20.2	40.2	34.5
17.8	15.3	40.3	34.6
38.4	32.9	18.4	15.8
Average = 23.32	Average = 19.96	Average = 35.2	Average = 30.22

iPerf H1 to H4 Before Load Balancing (BLB) and After Load Balancing



Min	Avg	Max	Mdev
0.049	0.245	4.407	0.807
0.050	0.155	4.523	0.575
0.041	0.068	0.112	0.019
0.041	0.086	0.416	0.066
0.018	0.231	4.093	0.759
Avg:0.0398	Avg:0.157	Avg:2.7102	Avg:0.4452

Ping from H1 to H4 Before Load Balancing



Min	Avg	Max	Mdev
0.039	0.075	0.407	0.068
0.048	0.078	0.471	0.091
0.04	0.072	0.064	0.199
0.038	0.074	0.283	0.039
0.048	0.099	0.509	0.108
Avg:0.0426	Avg:0.0796	Avg:0.3468	Avg:0.101

Ping from H1 to H4 After Load Balancing

VI. CONCLUSION AND FUTURE WORK

The practical virtual systems is create, running real portion, switch and application code, on a solitary machine utilizing Mininet. Firewall policies are defined and rules added to firewall table which changes more and more taking into account. Selective obstructing of packets in view of source and destination IP address will carried out. Load balancer system is also working. Application process logic implemented using FloodLight OpenFlow Controller. The project will more extended work by work additional connected load balancing algorithms and multicast algorithms for SDN. Also in future, Load Balancer system may be evaluated by considering additional parameters.

VII. REFERENCES

[1] Dependra Dhakal, Bishal Pradhan, Sunil Dhimal "Campus Network using Software Defined Networking" in SMIT,CSE Department,Rangpo, Majitar, Sikkim ,International Journal of Computer Applications (0975 - 8887)Volume 138 - No.4, March 2016

[2] "The design and implementation of open vswitch," in 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI),USENIX, Ed., vol. 1, no. 1. Oakland, CA: USENIX, May 2015, pp.1–14.

[3] Hailong Zhang1, Xiao Guo2,"SDN-BASED LOAD BALANCING STRATEGY FORSERVER CLUSTER" in1School of Information Engineering, Communication University of China, Beijing 100024, China 2Computer and Network Center, Communication University of China, Beijing 100024, China zhlcuc@163.com, xguo@cuc.edu.cn 2014 IEEE, May 2014.

[4] Senthil Ganesh N,Ranjani S , "Dynamic Load Balancing using Software Defined Networks" in International Journal of Computer Applications (0975 – 8887),International Conference on Current Trends in Advanced Computing (ICCTAC-2015)