Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

Telecommunication Engineering Graduate Program

Thesis

On

Performance Evaluation of Server Cluster Load balancing Algorithms Using SDN Open Flow Model

By:

Wubshet Abebe

Advisor

Dr. Yalemzewud Negash

A research Submitted to the School of Electrical and Computer Engineering in Partial Fulfillment of the Requirements for the Degree of Masters of Science in Telecommunication Engineering

October, 2018 Addis Ababa, Ethiopia

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

Telecommunication Engineering Graduate Program

Thesis

On

Performance Evaluation of Server Cluster Load balancing Algorithms Using SDN Open Flow Model

By: Wubshet Abebe

Dr. Yalemzewud Negash	_____
Advisor	**Signature**
_____	_____
Evaluator	**Signature**
_____	_____
Evaluator	**Signature**

Declaration

I, the undersigned, declare that this thesis is my original work, has not been presented for a degree in this or any other university, and all sources of materials used for the thesis have been fully acknowledged.

Wubshet Abebe Tedla _____
Name Signature

Place: Addis Ababa

Date of Submission: _____

This thesis has been submitted for examination with my approval as a university advisor.

Dr. Yalemzewud Negash _____
Advisor's Name Signature

Abstract

Due to the growth in the network usage, there is huge traffic pressure to server clusters in the datacenters. To manage incoming traffic, operators choose a load balancing technology to assign the incoming traffic to server clusters. However, this classical load balancing technology has complexity and challenge in scalability, flexibility and manageability. But, the emerging software defined networking having application, control and infrastructure layers offers an advantage of low cost, scalable, programmable and easy management to server clusters load balancing activity by abstracting the low level functionality of the network.

This study evaluates the performance of round robin, weighted round robin and least load server cluster load balancing algorithms using software defined networking open Flow model by providing ethio telecom's enterprise shop customers relation Management system users' data as input in order to answer a research question of which load balancing algorithms is better in terms of network performance parameters such as response time, transaction per second, throughput in SDN platform.

Simulation has been used as a methodology to evaluate server load balancing algorithms using Open Flow model by creating a virtual environment with oracle virtual box. In addition, mininet simulation tool has used to create the network topology and POX controller used in the control layer of the open Flow model to do the performance evaluation of the load balancing algorithms. The simulation result shows that round robin algorithm is better than weighted round robin and least load server load balancing algorithms interms of response time (sec), transaction rate (trans/sec), throughput (MB/sec).

Key words: Datacenter, server cluster algorithms, Mininet, SDN

Table of Content

List of Acronyms

ACM	Association for Computing Machinery
API	Application Program Interface
ARP	Automatic Resolution Protocol
BWBLB	Band Width Based Load Balancing
CAPEX	Capital Expenditure
CPU-	Central Processing Unit
CRM	Customer Relation Management
CORD	Central Office Rearchitectured as Datacenter
DFSM	Data Flow Entry Saving Multipath
DCN	Data Center Network
DNS	Domain Name Systems
DFSM	Dynamic Flow Entry Saving Multipath
ECMP	Equal Cost Multipath Protocol
EHLBOF	Extended Health Load Balance Open Flow
HTTP	Hyper Text Transfer Protocol
ICT	Information Commination Technology
IP	Internet Protocol
IaaS	Infrastructure as a Service
IT	Information Technology
IEEE	International Electronic and Electrical Engineers
LACP	Link Aggregation Control Protocol
MAC	Media Access Control
MB	Mega Byte
No C	Network on Chip
OF	Open Flow
OPEX	Operation Expenditure
OS	Operating System

OSPF	Open Shortest Path First
OVS	Open Virtual Switch
PC	Personal Computer
RSTP	Rapid Spanning Tree Protocol
Saas	Software as a service
SDN	Software Defined Networking
SDDC	Software Defined Datacenters
SHASDN	Server Health Admission Control Software Defined Network
SLA	Service Level Agreement
SME	Small Medium Enterprise
SOHO	Small Office Home Office
TCP	Transmission Controlled Protocol
TLS	Transport Layer Security
ToR	Top of Rack
UDP	User Datagram Protocol
WAN	Wide Area Network
WRR	Weighted Round Robin
WWW	World Wide Web
VIP	Virtual Internet Protocol
VXLAN	Virtual eXtensible Local Area Network
QoS	Quality of Service

List of Figures

List of tables

ACKNOWLEDGEMENTS

I would like to thank God who gave time and effort to make the completion of this research work. I am deeply grateful to my advisor, Yalemzewud Negash (PHD), who gave to me the basic knowledge and motivation for the research work and his great support throughout. Finally, my gratitude thanks also directly goes to the School of Electrical and Computer Engineering in AAiT that provides me the opportunity to upgrade my knowledge to a professional level.

Chapter 1: Introduction

Due to the fast growth in usage of network based services such as customer relation management system, domain name system, email, etc., there is a huge demand on server clusters. For effective network traffic management load balancing algorithms are used in order to distribute the incoming traffic load to a number of servers to get a better response from servers.

As the number of users grow, servers need to handle a large number of access request in short time. If the server cannot timely process the user request, waiting time to access increases which affect user experience, greatly reduce the quality of service. Questions like these make servers become the bottleneck in a network and researchers began to study how to improve server performance [1].

Enterprises have adopted a number of measures such as improve CPUs process speed, increase server cache capacity, use high speed disk arrays as well build server cluster. But, by establishing the server cluster, enterprises can forward access request to multiple servers. This method improves the performance of the server in a certain extent. However, such solution also has a new problem that when server cluster receives an access request, which server should respond if the control system cannot assign access request reasonably, the load imbalance situation between these servers will appear. Based on this, researchers have proposed the load balancing technology.

There are two dimensions that drive the need for load balancing: servers and networks. With the advent of the internet and intranet, networks connecting the servers to computers of employees, customers or suppliers have become mission critical. It is unacceptable for a network to go down or exhibit poor performance as it virtually shuts down a business in the internet economy. To build a website for e-

commerce, for example, there are several components that must be looked at: edge routers, switches, firewalls, caches, web servers and database servers [2].

The complexity and challenges in scalability, manageability, of server farms is one of the driving factor behind the need for intelligent switching for components starting from the edge routers that connect to the Internet to all the way to the database servers in the end. Therefore, load balancers have emerged as a powerful new weapon to solve many of these issues. In[1]Load balancing technology can effectively solve the problem of load imbalance between servers, thereby reducing the response time of access request and improve system throughput and fault tolerance .

The classical load balancing techniques have to use expensive hardware devices and cannot achieve precise control of the traffic load. Due to this limitations, the traditional load balancing technology is not suitable for large scale applications. Therefore, the emerging software defined networking offers an advantage of low cost, flexible, programmable and easy management and allows administrators to manage their network service through the abstraction of lower level functionality by separating the control plane and the forwarding plane of the traditional network.

Load balancer is a device that distributes load among several machines. It has the effect of making several machines appear as one. There are several components of server load balancing devices [3]:

- VIPs: is the load balancing instance where the world points its browsers to get to the site. A VIP has an IP address, which must be publicly available to be useable usually a TCP or UDP port number is associated with the VIP, such as TCP port 80 for web traffic. A VIP will have at least one real server assigned to it, to which it will dispense traffic usually there are multiple real servers, and the VIP will spread traffic among servers using metrics and methods. In [4] it is also explained that VIP identifies the service

supported by the load balancer and called virtual because of no server is associated with it and it exists only in the load balancer.

- **Server**: is a device running a service that shares the load among other services. A server is typically refers to an HTTP server, although other or even multiple services would also be relevant. A server has an IP address and usually a TCP/UDP port associated with it and does not have to be publicly addressable. In [5], a cluster refers to a group of servers and other resources that are connected through hardware, networks and software to behave as if they were a single system.

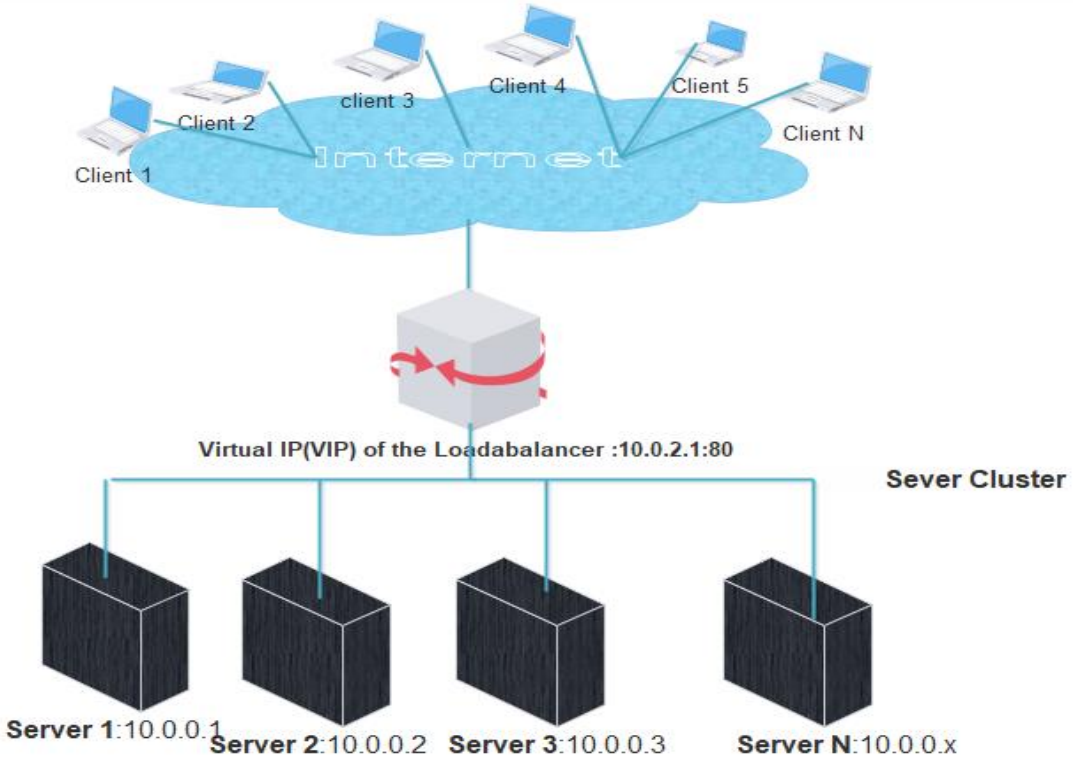

Figure 1. Server cluster load balancing set up [2]

1.1 Data Center Network Topology overview

Datacenter topology is the overall construction of the datacenter and in [6], it has also defined that it is a facility used to house computer systems and associated components, such as telecommunications and storage systems. They are key enabler for cloud computing to provide Software-as-a-service (SaaS), Infrastructure-as-a-service (IaaS) for online web services, big data computing, large simulations [6].

Today's data center network (DCN) contains thousands of compute nodes with significant network bandwidth requirements. Companies like Amazon, Google and Microsoft are building large data centers for cloud computing to keep up with application demands. Recent trends show companies like Dropbox,Apple embracing the move to build their own private cloud in order to gain better control, security and higher efficiency. The popularity of cloud, scale of data centers and desire to achieve highest level of application performance requires careful planning of compute, storage and the interconnection network or topology [6].

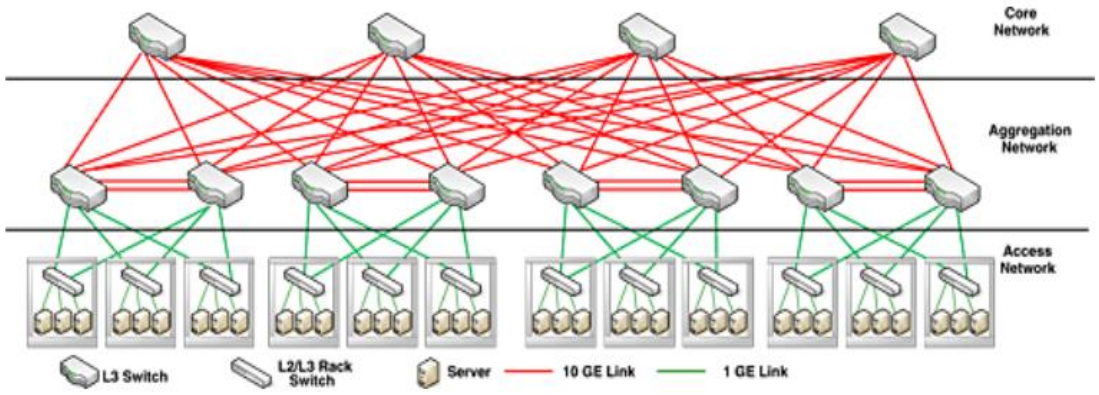

Figure 2.Legacy DCN Architecture [6]

Data Center Networks today, are typically based on the three tier, hierarchical, tree based architecture as shown in figure 2. It has a core tier at the root of the tree, an aggregation layer in the middle and edge tier at the leaves of tree. A group of hosts (mostly 16/32) is connected to one switch, called ToR switch (Top of Rack), building the edge (access) layer of the hierarchical tree structure.Core layer and

aggregation layer uses high end switches aggregating the traffic coming from the lower layers.

Unlike Network on Chip (NoCs), data center hosts and switches implement much more complex protocols to ensure reliability of communication.

- **Core layer:** provides the high-speed packet switching backplane for all flows going in and out of the data center. The core layer provides connectivity to multiple aggregation modules and provides a resilient Layer 3 routed fabric with no single point of failure [7].
- **Aggregation layer:** Provide important functions, such as service module integration, Layer 2 domain definitions, spanning tree processing, and default gateway redundancy. Server-to-server multi-tier traffic flows through the aggregation layer and can use services, such as firewall and server load balancing, to optimize and secure applications [7].
- **Access layer:** a place where the servers physically attach to the network. The server components consist of 1RU servers, blade servers with integral switches, blade servers with pass-through cabling, clustered servers, and mainframes with OSA adapters [7].

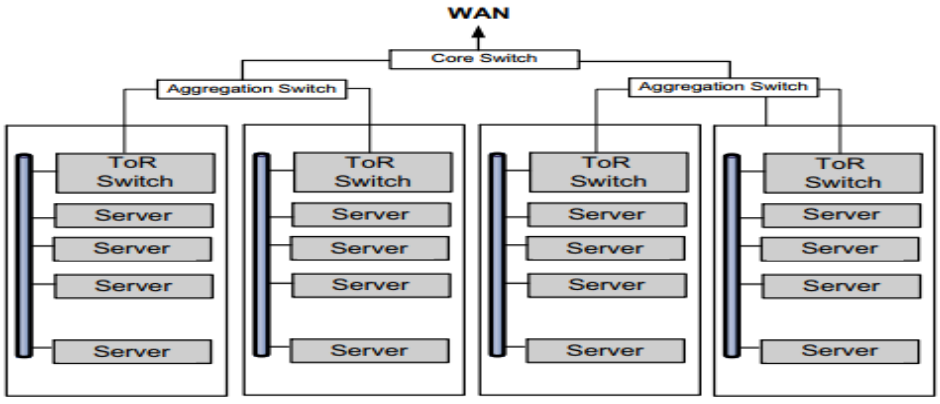

Figure 3. Typical datacenter network topology [8]

1.2 Background

1.2.1 Emergence of software defined datacenter

As the data center continues to evolve, there is an emerging need for flexibility, agility, and control. With web scale comes challenges that aren't found in legacy or smaller infrastructures, and require new ways of approaching the data center. The current approach to address these issues is the software defined approach which refers to the idea of abstracting a physical data center resource from the underlying hardware and managing it with software. As an example most IT professionals would be familiar with virtualization of computing resources. No longer allowing physical servers to be the container for data center systems, while providing and manipulating their resources with software, is the new normal [9].

The ability to create a new "server" with a few clicks or migrate a running workload between physical servers is the essence of the software defined approach. The software defined approach took hold with compute, but is now starting to encompass all areas of the data center, which has led to the term software defined data center [9].

The SDDC isn't any one thing specifically, but rather a way of describing a data center where many pieces as possible are abstracted into software.SDDC is characterized by automation, orchestration, and abstraction of resources into software and code. By nature, code is more reliable than humans, which means that compared to a legacy data center, the SDDC is more secure, more agile, and moves more rapidly. The fallout of abstracting physical resources across the data center is that all of a sudden, the hardware is substantially less important to the big picture [9].

1.2.2 Shift to software defined networking datacenter

It wouldn't be a data center without networking! Networking is the backbone of everything that happens within and outside the data center. Because connectivity is so integral to the function of a data center, abstracting network functions and topologies into software will substantially change the way that data centers are designed and operated. Network abstraction also unlocks new levels of scale and flexibility with technologies like VXLAN.

Software defined networking decouples the control plane from the data plane and allows programmatic interaction with the control plane. This change allows the network to be redesigned on the fly or scaled to meet demand. Traditional networking has relied heavily on link oversubscription, and as virtualization has changed the game, SDN allows network architects to create dynamic networks that can be adapted to meet changing requirements [9].

1.2.3 SDN architecture Overview

Software defined network has emerged as the new network framework which is programmable and vendor neutral. It conquers the shortcomings of long-established network architecture by abstracting the main intelligence of network i.e. control plane from forwarding plane (data plane).SDN architecture consists of application plane, control plane and data plane. In the application layer, applications like traffic engineering, network virtualization, QoS, monitoring, routing and any business application. In the control layer, either open source or commercial SDN controllers can used as a controlling device and in the infrastructure layer, network device such as open flow switch could be used as a forwarding element[10].

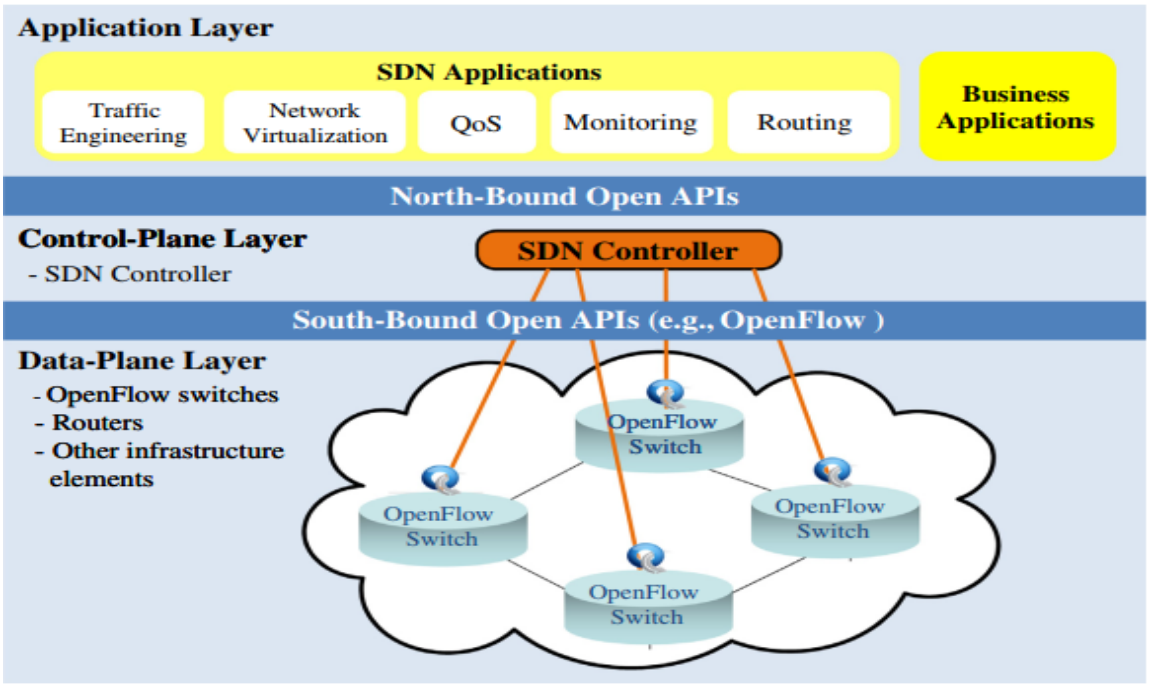

Figure 4.Overview of SDN architecture [11]

The recently emerged Software Defined Networking paradigm separates the network control plane from the data forwarding plane, and provides user applications with a centralized view of the distributed network states.It includes three layers and interactions between layers as shown in Fig 4.The details of the SDN architecture overview are explained as follows: There may be more than one SDN controller if the network is large-scale or a wide-area region network [11].

The control layer globally regulates the network states via network policies in either a centralized or distributed manner. Due to the unrestricted access to global network elements and resources, such network policies can be updated timely to react to the current flow activities. Furthermore, SDN applications exist in the application layer of the SDN architecture [11].

A set of application programming interfaces (such as North-bound Open APIs) are supported to communicate between the application layer and the control layer in order to enable common network services, such as routing, traffic engineering, multicasting, security, access control, bandwidth management, quality of service

(QoS),energy usage, and many other forms of the network management. In other words, these interfaces facilitate various business objectives in the network management [11].

On the other hand, the data forwarding layer can employ programmable Open Flow switches through Open Flow controller, and the switches communicate with the controller via South-bound Open APIs. The Open Flow protocol provides access to the forwarding plane of a network switch over the network and enables software programs running on OF switches to perform packet lookups and forwarding the packets among the network of switches or routers. These programmable switches follow the policies of the SDN/OF controller and forward packets accordingly in order to determine what path the packets will take through the network or switches or routers. In short, through the interactions among these layers, the SDN paradigm allows a unified and global view of complicated networks, and thus provides a powerful control platform for the network management over traffic flows [11].

1.2.3.1 Southbound interface

South bound application program interface is the link between controller and forwarding devices. Forwarding devices are controlled by application program interface. Open Flow protocol is the best example for south bound interface, it shows execution of the switch to controller communication and vice versa. The Open Flow provides information sources to controllers such as, when port change or link is triggered messages are sent by forwarding elements to the controller, forwarding elements generate flow statistics and controllers collect [12].

When the new incoming packet having action “send to controller” in matching entry of the forwarding table, then forwarding devices send this packet to controllers. This gives Interoperability between Open Flow equipment to different

vendors. Open Flow defines set of instructions exchanged between the forwarding device and a controller through secure channel. By this remote controller can update, delete or add flow entries from flow tables. This can be done in two ways reactively and proactively. Reactive means each time a decision is made, forwarding element should consult to controller. In a proactive way controller push policies to switches [12].

1.2.3.2 Northbound interface

A northbound interface allows any lower level component to communicate with a higher level component. This is the communication interface between the application layer and the control layer in the architecture of SDN. In data center network northbound application program interface manages orchestration and automation, and actively shares data between the systems. North bound application program interface can support for network functions like loop avoidance, security, routing, load balancing, computation and many other network functions [12].

1.2.3.3 Network application layer

Management applications execute the control logic to install commands in the data plane and dictate the operations of the forwarding elements. SDN is deployed in the data center for management applications to perform network functions such as load balancing, routing, reducing power consumption and security enforcement. The main goal of management application is to maximize network utilization, optimization of load balancing, engineer traffic to minimize power consumption and optimization of traffic [12]

1.2.4 Standard Bodies SDN

For new technologies to become widely adopted standards are required. The following are standardization bodies of SDN and NFV related efforts in the world.

No	Organization	Mission	SDN- and NFV-Related Effort
1	Open Networking Foundation (ONF)	An industry consortium dedicated to the promotion and adoption of SDN through open standard development.	Open Flow
2	Internet Engineering Task Force (IETF)	The Internet's technical standards body. Produces RFCs and Internet standards.	Interface to routing systems (I2RS)
			Service function chaining
3	European Telecommunications Standards Institute (ETSI)	An EU-sponsored standards organization that produces globally applicable standards for information and communications technologies.	NFV architecture
4	Open Daylight	A collaborative project under the auspices of the Linux Foundation.	Open Daylight
5	International Telecommunication Union — Telecommunication Standardization Sector (ITU-T)	United Nations agency that produces Recommendations with a view to standardizing telecommunications on a worldwide basis.	SDN functional requirements and architecture
6	Internet Research Task Force (IRTF) Software Defined Networking Research Group (SDNRG)	Research group within IRTF. Produces SDN-related RFCs.	SDN architecture

Table 1.SDN standardization bodies [13]

1.2.5 SDN implementation in real world datacenters

Currently, there are three ways of implementing software defined based datacenter in real world enviroments.These are SDN via over relays and open SDN and hybrid of SDN via overlay and open SDN.The below sections will discuss about SDN over relays and Open SDN.

1.2.5.1 SDN via Overlays

SDN via Overlays is designed for solving data center issues. It utilizes contemporary data center technologies, specifically VXLAN and NVGRE and it also creates a virtual overlay network and thus does a very good job of Overcoming Networking limitations and improving agility.SDN via Overlays does not address any issues related to improving the behavior of the physical infrastructure. However, this has the advantage of incurring little or no costs related to new networking equipment. It is sometimes difficult to diagnose problems and determine whether they relate to the virtual network or the physical one [8].

1.2.5.2 Open SDN

Open SDN is designed for solving networking issues in a wide range of domains, not just data centers only. It can be implemented using networking devices designed for Open SDN. Legacy switching hardware can often be enhanced by adding Open Flow to function as Open SDN devices. Open SDN used in conjunction with virtual networks (e.g., VXLAN and NVGRE) and does a very good job of addressing all the networking issues. It is more disruptive and may pose more transitional issues than does SDN via overlays [8].

The below table presents some list of enterprises and early adopters of SDN in their datacenters. These adopters and implementers are those enterprise which are the current

leaders of ICT industry. The table presents in three columns where the first column is the enterprise that implements SDN.The second column is the type of SDN that the corresponding enterprise and the third column is the description of the way how the implementer adopts and get success in the SDN implementation.

Enterprise	SDN Type	Description
Google	Open SDN	Has implemented lightweight Open Flow switches, an Open Flow controller, and SDN applications for managing the WAN connections between their data centers. Google is moving that Open SDN technology into the data centers
Microsoft Azure	Overlays	Implementing overlay technology with NVGRE in vSwitches, communicating via enhanced open Flow, creating tens of thousands of virtual networks
eBay	Overlays	Creating public cloud virtual networks using VMware's Nicira solution
Goldman Sachs	Open SDN	Using Floodlight-based application and Open Flow-supporting commodity switches. Also replacing firewalls by augmenting Open Flow-supporting switches for firewalling functionality
Rackspace	Overlays	Creating large multitenant public clouds using VMware's Nicira solution

Table 2.Datacenter SDN Implementation [8]

1.3 Load balancing Algorithms

Load balancing is the process of reassigning the total load to the individual servers in the cluster to make the resource utilization effective and improve the response time for the request, simultaneously it removes a condition in which some of the servers are overloaded while the others are under loaded. There are

numerous algorithms that can be used to intelligently load balance client requests across server pools.

The algorithm chosen will depend on the type of service or application being served and the status of the network and service at the time of requests. The current level of requests to the load balancers often determines which method is used. Some of the popular server cluster load balancing algorithms are: round robin algorithm, weighted round robin algorithm, least load algorithm, ratio algorithm, priority algorithm and predictive algorithm. The following are some of the benefits of server load balancing:-

- **Increase Server Reliability**: The first thing server load balancing can do is to increase the reliability servers. If there is a single problem with one node, an unexpected traffic spike or another issue, then the incoming traffic can be distributed across multiple nodes so that resources are shared [14].
- **Improve the Customer Experience**: servers are flooded with new visitors in response to a promotion or marketing strategy, then the load could potentially slow down service. It could even cause the servers to crash. This can leave consumers with a poor first impression of company. Load balancing will ensure customers have a good experience every time [14].

1.3.1 Round Robin Load balancing algorithm

Round robin algorithm is one of the popular algorithms used for server cluster load balancing in datacenter. It passes new requests to the next server in line and distributes user requests evenly across an array of servers being load balanced. The main advantages of round robin algorithm is it is simple to implement, cheap, very predictable, fair and works best when all servers have equal capacity. The disadvantage of this algorithm is it has no priority means that it doesn't give any

special priority to more important tasks. In SDN open flow model, when the controller is initialized, and first statistics of all servers' information (IP address, MAC address, ID, port) in the cluster are collected and stored it in a Hash Map. When the VIP module is calling Open Flow enabled Round-Robin algorithm to assign the incoming traffic, the algorithm will determine which server to use according to the last selected server's ID. This method ensures that all servers will be visited in a loop.

Pseudo code [15]:

Supposing that there are n servers in the set S = {S0, S1... Sn-1}, where n > 0;

i indicates the server selected last time, and i is initialized with -1;

```
j = i;
do {
j = (j + 1) mod n;
if  (Available (Sj )) {
i = j;
return Si;
}
} while (j != i);
return NULL;
```

1.3.2 Weighted Round Robin Load Balancing algorithm

Weighted round robin algorithm is one of the methods used in server cluster load balancing. It assigns user requests first by checking the weight of each server in the cluster. The advantage of weighted round robin algorithms is for those datacenters having different servers in speed and memory. In [21], it has explained that weighted round robin algorithm usually specify weights in proportion to actual capacities. So, for example, if Server 1'scapacity is 5 times more than Server 2's, then we can assign a weight of 5 to server 1 and weight of 1 to server 2.

Pseudo code [16]:

Supposing that there is a server set S = {S0, S1… Sn-1};

W (Si) indicates the weight of Si;

i indicates the server selected last time, and i is initialized with -1;

cw is the current weight in scheduling, and cw is initialized with zero;

max(S) is the maximum weight of all the servers in S;

gcd(S) is the greatest common divisor of all server weights in S;

```
while (true) {
i = (i + 1) mod n;
if (i == 0) {
cw = cw - gcd(S);
if (cw <= 0) {
cw = max(S);
if (cw == 0)
return NULL;
}
}
if (W(Si) >= cw); return Si;}
```

1.3.3 Least Load Balancing algorithm

The least load server load balancing algorithm is one of the load balancing methods applied for server cluster in a datacenter. It assigns user requests to the server which has lowest workload of the cluster.

Pseudo code [17]:

```
begin procedure least_load_algo

set array servers = {s1,s2,s3……sn};

set array server_load = {l1,l2,l3……..ln};

while ( request) do

integer pos = find_min( server_load);

goto server[pos];

end while

end procedure

begin procedure find_min (server_load [1 to n])

set integer pos = 0;

for i = 1 to n-1 do

if server_load [pos] > server_load[i] then

pos = i;

end if

return pos;

end procedure
```

1.4 Related works

According to [18], server load balancing policy plays a critical role to achieve scalability of services offered by cluster of web servers. It has presented a new policy which is called Server Health monitoring and Admission control using SDN and called Server Health Admission Software Defined Networking(SHASDN) which is capable of prioritizing different client priorities such as premium, default and honored customers Service Level Agreements(SLAs).Using SHASDN,web servers could maintain service level agreements without the need of a priori over dimensioning of server resources.SHASDN server load balancing strategy has compared with weighted round robin and Extended Health Monitoring for load balancing in open flow based networks. The parameters used for comparison are throughput and response time. Even through SHASDN strategy takes a little more processing resources than weighted round robin and Extended Health monitoring algorithms. It is also capable of providing better service to premium customers.

As [19], explained that software defined networking offers a cost effective and flexible approach in implementing a load balancer. The new SDN approach for load balancing of server clusters reduces cost and provides flexibility in configuration that reduces time to deploy, automation, and facilitates building a network without requiring the knowledge of any vendor specific software or hardware. Round robin algorithms has successfully implemented in POX controller and software based nature of SDN based load balancer helped to reduce cost of implementation for users. Therefore, SDN based server load balancing significantly reduces the period of time required to deploy new services compared to a traditional hardware based approach.

Referring [20], Bandwidth based load lancing has developed to distribute the network requests among multiple users based on servers bandwidth consumption. The performance of the proposed approach has evaluated and compared with different load balancing schemes such as round robin and connection based under mininet emulation and Raspberry pi-based implementation. It has also pointed out a new open flow based solution for server's headache problem using SDN that could replace the traditional load balancer which is more dedicated and expensive hardware.

The evaluation of badwithbased load balancing scheme compared and evaluated with round robin and connection based algorithms in both mininet emulation and Raspberry pi test bed environments. Httperf and open load benchmarking tools showed that the proposed bandwidth based approach using SDN has improved the throughput and response time of web servers when compared with round robin and connection based strategies.

In [21], discussed that traditional load balancers are exorbitant and not flexible to change. As traditional load balancer are vendor locked and due to its non-programmable nature network administrators does not have capability to build their algorithms by own. But in case of SDN load balancers, it provide the facility to the programmers to build and design our own load balancing strategy thus makes the SDN load balancer programmable and flexible.

Another strong point of using Software Defined Networking load balancer is that it does not require any separate hardware that behaves as a load balancer. In comparison to traditional load balancers SDN load balancers solves many problems. Weighted round robin load balancing using Open Flow switch in SDN is implemented in the paper and results are co related with round robin load balancer in SDN. Results reveal that weighted round robin strategy is more prominent than round robin strategy.

[22] Has proposed a novel algorithm is proposed for purpose of load balancing for SDN-based datacenters. Mininet emulator was utilized for the purpose of emulating the proposed system, the suggested algorithm was added to the POX controller. To evaluate their algorithm, they have simulated a datacenter with a Fat-Tree topology (k=4). The algorithm was proposed to dynamically balance the load by means of re-routing utilizing the information at the SDN controller. The network performance was tested in term of throughput, loss, and received data size with and without applying the proposed algorithm. Results showed that the proposed algorithm outperforms the traditional load balancing scheme as follows; improves the throughput by a minimum of 21.9%, reduce the loss by 88.2%, increase the received data size by 20.8 %.

[23] Has proposed, a fabric topology to provide a flexible data-center network. With SDN controllers, and has the potential to address critical issues such as bandwidth utilization and network capacity in datacenter networks. In addition, the proposed network fabric consists of several key components: SDN controller, commoditized switches, and host machines. All switches are connected to form a switch pool. Each switch can connect to a certain number of switches based on performance requirements.

As a centralized manager, an SDN controller connects to the switch pool and collect network statistics at run-time. Host machines connect to switches via their Ethernet interfaces. The fabric topology is a critical part of proposed networks. Majority physical devices in such networks are low-cost and commoditized switches. There is no explicit hierarchy in the proposed network. The flat architecture gives data-center networks more flexibility in bandwidth utilization and packet re-routing. In the network fabric, each switch shares a fair amount of workload.

In [24], new adaptive load balancing scheme for data center networks is proposed. The proposed algorithm presented a loss-free performance compared to 15% to 31% loss rate in the traditional scheme for the first scenario and showed up to 81% improvement in the loss rate in the second scenario. In term of throughput, the proposed scheme maintained the same level of throughput in the first scenario compared to an average of 5Mbps reduction in the throughput when using the traditional scheme. While in the second scenario, the new scheme outperformed the traditional scheme by showing an improvement of up to 16.6% in the throughput value.

[25] Discussed that Central Office Re-Architected as a Data Center is a new design of a telco central office that replaces closed and proprietary hardware with software running on commodity servers, switches, and access devices. It allows network operators to benefit from both, the economies of scale (infrastructure constructed from a few commodity building blocks) and agility (the ability to rapidly deploy and elastically scale services) that commodity cloud providers enjoy today.

Central Office Re-Architected as a Data Center is a revolutionary effort to transform legacy central offices in the telco network. In the re-architected central office, closed and proprietary hardware is replaced with software running on commodity servers and switches. This software, in turn, is managed and orchestrated as a collection of scalable services. In doing so, Central Office Re-Architected as a Data Center goal is to demonstrate the feasibility of a central office that enjoys both the CAPEX and OPEX benefits of commodity infrastructure, and the agility of modern cloud providers.

[26] Has proposed that, an SDN-based Dynamic Flow Entry-Saving Multipath (DFSM) mechanism for inter-DC WAN traffic forwarding. DFSM adopts source destination-based multipath forwarding and latency-aware flow based traffic

splitting to save flow entries and achieve better load balancing. DFSM saves 15% to 30% system flow entries in different topologies compared to label-based tunneling, and also reduces average latency by 10% to 48% by consuming 8% to 20% more flow entries than Equal Cost Multipath (ECMP) in less-interconnected topologies. In addition, compared to even traffic splitting, DFSM reduces the standard deviation of path latencies from 14% to 7%.

[27] Has implemented and evaluated an alternative load balancing architecture using Open Flow switches connected to a controller, which gains high flexibility without additional equipment, and has the potential to be more robust than traditional load balancing approach. Their system could measure network and server status in real-time and dynamic set weights of server according to the server's processing capability and load balancer installs wildcard rules in the switches proactively to direct requests of large groups of clients without involving the controller which effectively saves the flow table space and reduces the delay of the network.

Their implementation has used the Open Flow controller Floodlight and network emulator Mininet to verify the validity of this algorithm. The preliminary evaluation results demonstrated that their dynamic load balancing scheme is superior to not only the random load balancing algorithm but also the round robin load balancing algorithm.

[28] Has proposed load-balancing algorithm on the basis of the Extended Dijkstra's shortest path algorithm for Software Defined Networking (SDN). The Extended Dijkstra's algorithm considers not only the edge weights, but also the node weights to find the nearest server for a requesting client. The proposed algorithm also considers the link load in order to avoid congestion. They have used Pyretic to implement the proposed algorithm and compared it with related ones under the Abilene network topology with the Mininet emulation tool.

[29], discussed that traditional vendor specific Load Balancer is a device that is much more expensive and Internet user cannot change the functionality of the device means traditional Load Balancer are non-programmable. In the network single Load Balancer become a single point of failure or sometimes in large network it become bottleneck. Software defined networking is a promising solution to all these disadvantages. In Software Defined Networking architecture, control and data plane are to be separated. They have converted a dumb open flow switch into Load Balancer by writing a program or application and run application at control plane and Mininet emulation tool is used.

[30] Has proposed an SDN-based dynamic load management algorithm for optimizing link utilization in DCNs while considering the flow priority. The algorithm finds the shortest paths from each host to others and calculates every link's cost. When congestion occurs in a certain path, it replaces the old path with the alternative best route that has the minimum link cost and lower traffic flow. Performance of the algorithm is evaluated by measuring throughput, delay and packets loss in a fat-tree DCN.

Simulation results show improved performance in load balancing over time as the algorithm keeps on running. Dijkstra's algorithm is implemented to find multiple paths of same length and reduce the search to a small region in the topology. The traffic flows are ordered according to their priority. Among selected paths, the path with least cost and load is selected and traffic flow is forwarded on that route. The new flows rules are then pushed to Open Virtual switches (OVSs) to update switch forwarding tables.

The performance of the algorithm is evaluated in a Fat-Tree DCN topology by collecting operational data of the links and switches. It has concluded that Performance of data centers rely on proper resources utilization for timely services to the end users. Traditional load management employed in DCNs suffer as few

links experience congestions while majority of the links are underutilized. The algorithm finds the shortest paths from each host to others and calculates every link's cost while considering flow priority when congestion occurs, it replaces the old path with the alternate best route with minimum link cost that has shortest distance.

[31] Implemented that the Bellman – Ford algorithm to find the shortest path between two nodes in a network using SDN environment. POX controller and API's has been used to implement the Bellman – Ford Algorithm. The Bellman – Ford Algorithm has been used as each node only needs to know its own number and be able to derive the number of neighbors it has. The calculation of the shortest path to a given destination is done by hoping neighbor by neighbor from each source to destination. Mininet is a network emulator tool that accurately emulates any type of forwarding element, in terms of function and performance. SDN network can be created as per required specifications and testing can be on different network configurations. Once the testing is done in the SDN solution on Mininet we can move it to a real physical network.

1.5 Motivation

Today, operators need their networks to be capability of programmability, centralized intelligence and abstraction. The programmability nature encourages innovation, accelerates service introduction and the centralized intelligence behavior of their network helps to simplify control, optimize performance and smart management. The abstraction of the network infrastructure decouples the control with the data plane of the traditional network.

To achieve the operators' need, software defined networking has the potential and also a key to transform from traditional datacenter network to flexible datacenter network by reducing CAPEX and OPEX and improving their traffic management to obtain good quality of service. Moreover, traditional Internet and telecom datacenters are facing the rapid development of ICT market, which include emerging services applications such as 3G,4G,M2M,Iaas,Saas,Pass.In this context, next generation datacenters are required to provide more powerful IT capabilities, more bandwidth, more storage space, slower time to market for new services to be deployed, and most important lower cost.

The trends of future datacenters are towards resource utilization and cloud computing, with converged IT and network resources management to design and implement practicable and easily maintainable datacenters which fully meet the requirement. The motivation for thesis is to introduce SDN based datacenter and evaluate the performance of SDN based server cluster load balancing algorithms.

1.6 Statement of the Problem

Network traffic management in a network is vital to achieve quality of service. In ethio telecom ,for enterprise customers' a web based CRM (Customer Relation management) servers are used to manage sales activities such as creation of service orders and follow up by respective sales employees in eight sales shops of Addis Ababa.

To manage these CRM users, load balancing technology is used to ensure better traffic management in server clusters by assigning servers using various server cluster load balancing algorithms such as round robin algorithm, Weighted round robin algorithm and least loaded algorithm to provide reliable quality of service and effective resource utilization.But,these algorithms could be used to achieve dynamic traffic management by using an SDN open Flow model to become programmable, flexible and easy to manage. Therefore, this study evaluates the performance of these server cluster load balancing algorithms using SDN open Flow model by providing ethio telecom shop users' as input data in order to answer a research question of which load balancing algorithms is better or best in terms of network performance parameters such as response time, transaction per second, throughput and data transferred in SDN platform.

Software Defined Networking is an emerging technology which is used for the Softwarization of an infrastructure, specifically in telecommunication. It changes the static way of managing network resources in to dynamically for efficient resource utilization of datacenters. Moreover, software defined networking could bring holistic network management, adaptive load balancing, agility, less downtimes, increase security and cloud abstraction for datacenters.

1.7 Methodology

The research methodology that has been used in this study in order to achieve the objectives of the research are Literature review, SDN open flow model as a system model, simulation and analysis of results.

- **Literature Review**: The primary research method for this study are intensive literature review about software defined networking and its application in datacenters from various sources such as IEEE papers,ACM papers and different sites and books.
- **System model:** Software defined network open flow model has been used as a system model in the study. Open flow protocol is used as a southbound interface between the infrastructure layer and the control layer of the SDN open flow model and python programming language Application Programming Interface (API).
- **Simulation :**The simulation environment that has been used with PC have Intel processor Core i5-240M CPU@2.5GHZ,8.0GB RAM which has 64 bit operating system installed and oracle virtual box is used as a hypervisor to run Ubuntu 16.04 Linux operating system. In addition, mininet and SDN POX controller have installed inside Ubuntu 16.04 Linux operating system and siege load test tool also used.
- **Analysis of results**: The results and observations from the simulation are analyzed and the performance of the server cluster load balancing algorithms are evaluated and reported. The performance parameters that are used for the evaluation are response time (sec), transaction rate (trans/sec), throughput (MB/sec).

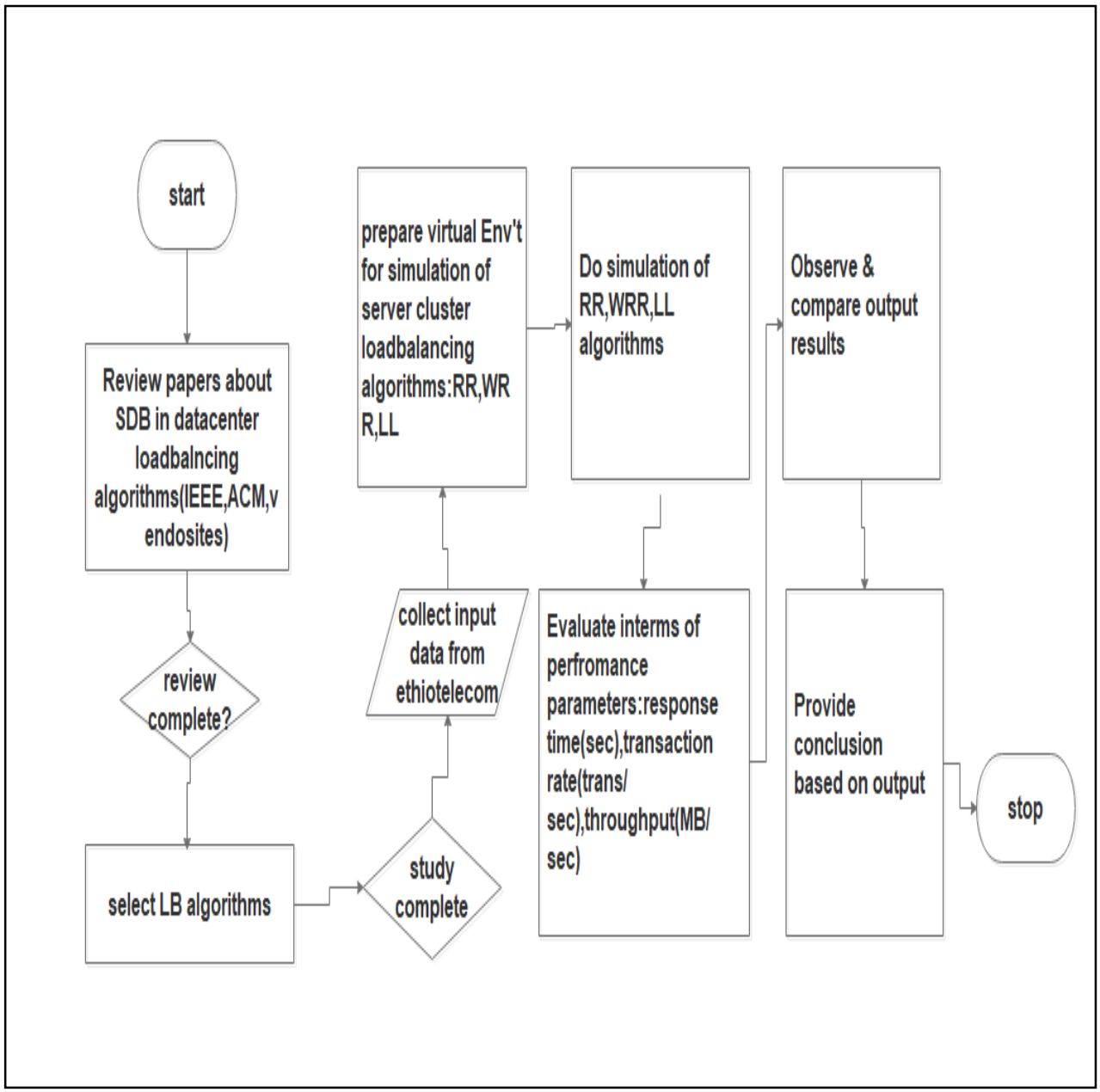

Figure 5.Methodology flow chart

1.8 Objective

1.8.1 General Objective

The main objective of this study is to do performance evaluation of server cluster load balancing algorithms using Software Defined Networking open Flow model in terms of network performance parameters by providing ethio telecom enterprise shops customer relation management system users' data as input.

1.8.2 Specific Objectives

- Study server cluster load balancing algorithms such as round robin, weighted round robin and least loaded algorithms.
- Understand mininet Simulation tool for Software Defined Networking for server cluster load balancing.
- Know SDN POX controller for server cluster load balancing.
- Do performance analysis of server cluster load balancing algorithms response time, transaction rate and throughput network performance parameters
- Compare the simulated results interms of network performance parameters response time, transaction rate and throughput.
- Provide conclusions based on results.

Chapter 2: SDN Open Flow Model

Software defined networking has three models. These are SDN open flow model, SDN over relay model and hybrid model. From the three mentioned models SDN open flow model is the one. It Separates the control and data planes, uses a standardized protocol between controller and an agent on the network element for instantiating state, and provides network programmability from centralized view via a modern, extensible API and also uses Open Flow protocol that provides a great deal of flow/traffic control.[32] Has discussed that Open Flow model was originally imagined and implemented as part of network at Stanford University. The Key components of the open flow model is shown below.

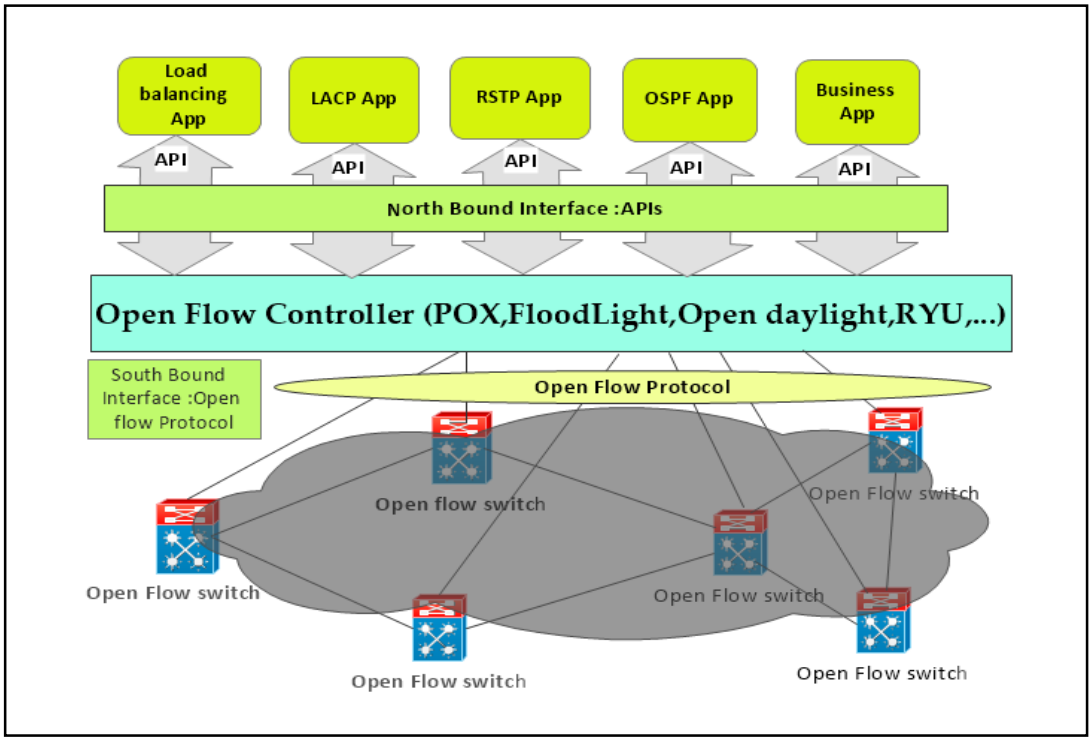

Figure 6.SDN Open Flow Model Architecture [32]

2.1 Components of SDN Open Flow model

The SDN Open flow model has three components: These are the network devices such as open flow switch in the infrastructure Layer, open flow controller in the controller layer and the network applications in the application layer. In addition, it has an open flow protocol as a south bound interface and APIs as a north bound interface.

2.1.1 Open Flow Switch

The SDN Open flow switch, the SDN controller and interface present on the controller for communication with forwarding device, generally southbound interface(Open flow) and network application interface(northbound interface) are the fundamental building blocks of an SDN deployment.SDN switches are often represented as a basic forwarding hardware accessible via an open interface, as the control logic and the algorithms are offloaded to the controller[33].

The open flow switch consists of a flow table, which performs packet lookup and forwarding. Each flow table in the switch holds a set of flow entries that consists of:

- **Header fields or match fields**, with information found in packet header, ingress port and metadata used to match incoming packets.
- **Counters**, used to collect statistics for the particular flow, such as number of received packets, number of bytes, duration of the flow.
- **A set of instructions or actions** to be applied after a match that dictates how to handle a matching packet. For instance, the action might be to forward a packet out to a specified port.

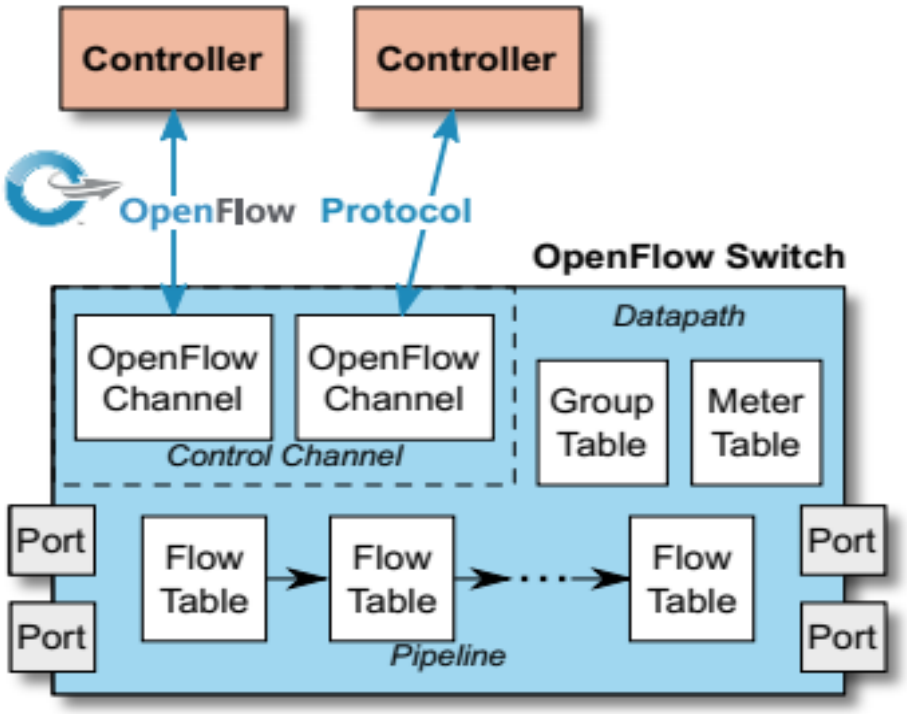

Figure 7.Main components of Open Flow Switch [34]

Open flow switch contains one or more flow tables and a group table, which performs packet lookups and forwarding, and an open flow channel to external controller. It communicates with the controller and the controller manages the switch via the open flow controller and using the open flow protocol, the controller can add, update and delete flow entries in flow tables both reactively and proactively[34].

Each flow table in the switch contains a set of flow entries and each flow entry consists of match fields, counters and a set of instructions to apply the matching packet. Matching starts at the first flow table and may continue to additional flow tables. Flow entries match packet in priority order, with the first matching entry in each table being used. If a matching entry found in a flow table, the outcome depends on the configuration of the table- miss flow entry. Instructions associated with each flow entry either contains actions or modify pipeline processing actions

included in instructions describe packet forwarding, packet modification and group table processing [34].

Pipeline processing instructions allow information in the form of metadata, to be communicated between tables. Table pipeline processing stops when the instructions set associated with the matching flow entry doesn't specify a next table; at this point the packet is usually modified and forwarded[34].

1. **Open Flow channel**: The open flow channel is the interface that connects each open flow logical switch to an open flow controller. Through this interfaces the controller configures and manages the switch, receives events from the switch and send packets out the switch.
2. **Group table**: A group table consists of group entries. The ability for a flow entry to point to a group enables open flow to represent additional methods of forwarding and table 3 below shows the four fields of a group table.

Group identifier	Group type	counter	Action buckets

Table 3.Main components of a group entry [34]

As it shown in table 3, each group entry is identified by its group identifier and contains:

- **Group identifier**: a 32 bit unsigned integer uniquely identifying the group on the open flow switch.
- **Group type**: to determine group semantics
- **Counters**: Updated when packets are processed by group
- **Action buckets**: an ordered list of actions buckets, where each action bucket contains a set of actions to execute and associated parameters. The actions in a bucket are always applied as an action set.

3. **Meter Table**: consists of meter entries, defining per-flow meters. Per-flow meters enable open flow to implement rate – limiting, a sample QoS operation constraining a set of flows to a chosen bandwidth. The components of a meter table have shown in the table 4 below.

Meter identifier	Meter bands	Counters

Table 4.Meter entry components

Each meter entry is identified by its meter identifier and contains:

- **Meter identifier**: a 32 bit unsigned integer uniquely identifying the meter.
- **Meter bands**: is unordered list of meter bands, where each meter band specifies the rate of the band and the way to process the packet.
- **Counters:** Updated when packets are processed by meter.

4. **Flow table and flow entries**: a flow table consists of flow entries:

Match field	priority	counters	instructions	timeouts	cookies	flags

Table 5.Main components of flow entry [34]

- **Match fields**: to match against packets. These consists of the ingress port and packet headers, and optionally other pipeline fields such as metadata specified by previous table.
- **Priority**: Matching precedence of the flow entry.
- **Counters**: updated when packets are matched
- **Instructions:** to modify the action set or pipeline processing
- **Timeouts**: Maximum amount of time or idle time before flow is expired by the switch
- **Cookies**: opaque data value chosen by the controller.
- **Flags**: Flags offer the way flow entries are managed.

Open Flow switch is a basic forwarding element, which is accessible via Open Flow protocol and interface. In an Open Flow network, switches come in two flavors, hybrid (Open Flow enabled) and pure (Open Flow only). Hybrid switches support Open Flow in addition to traditional operation and protocols [33].

Pure Open Flow switches have no legacy features or onboard control, and completely rely on a controller for forwarding decisions. Most of the currently available and commercial switches are hybrids. Since Open Flow switches are controlled by an open interface (Over TCP-based TLS session), it is important that this link remains available and secure. The Open Flow protocol can be viewed as one possible implementation of SDN-based controller-switch interactions (which is a messaging protocol), as it defines the communication between the Open Flow switch and an Open Flow controller [33].

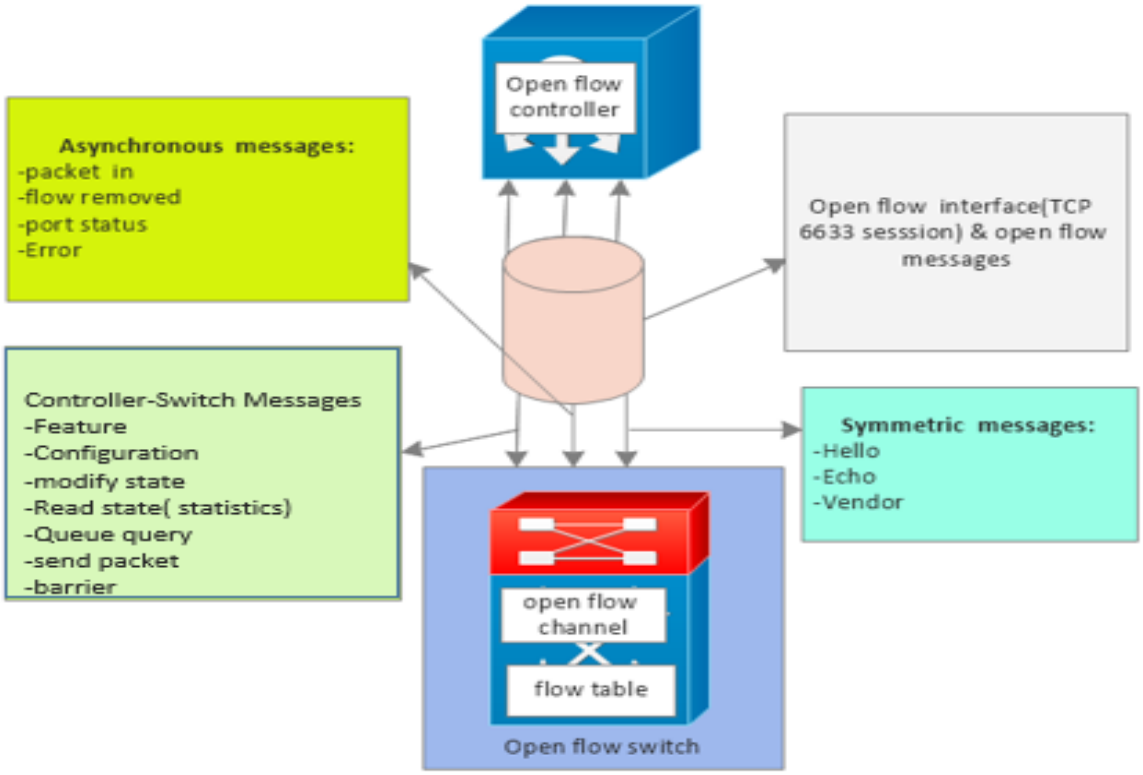

Figure 8.Open Flow interface and messaging protocol [33]

2.1.2 Controller-Switch messages

Messages are used to directly manage or inspect the state of the switch:

- **Features**: Upon the establishment of the TLS session (for example, TCP TLS session on port 6633),the controller sends an OFPT_FEATURES_REQUEST message to the switch and the Open Flow switch reports back (via OFPT_FEATURES_REPLY message) the features and capabilities that it has and supports. The data path identifier (data path_id), number of supported flow tables by data path (Open Flow switch), switch capabilities, supported actions, and definition of ports are the important features that are reported to the controller. The datapath_id field uniquely identifies an Open Flow switch (data path). It is a 64-bit entity and the lower 48 bits are intended for the switch MAC address, while the top 16 bits are up to the manufacturers [33].

- **Configuration**: The controller is able to set and query configuration parameters in the switch with the OFPT_SET_CONFIG and OFPT_GET_CONFIG_REPLY messages, respectively. The switch responds to a configuration request with an OFPT_GET_CONFIG_REPLY message; it does not reply to a request to set the configuration [33].

- **Modify state**: Modifications to the flow table from the controller are done with the OFPT_FLOW_MOD message and the controller uses the OFPT_PORT_MOD .Message to modify the behavior of the physical ports. The flow modification Commands are ADD, MODIFY, MODIFY_STRICT, DELETE, and DELETE_STRICT.The port configuration bits indicate whether a port has been administratively brought down, options for handling 802.1D spanning tree protocol (STP) packets, and how to handle incoming and outgoing packets. The controller may set OFPPFL_NO_STP to 0 to enable

STP on a port, or to 1 to disable STP on a port. The Open Flow reference implementation sets this bit to 0 (enabling STP) by default [33].

• **Read State (Statistics**): The controller can query the status of the switch using OFPT_STAT_REQUEST message. The switch responds with one or more OFPT_STATS_REPLY messages. There is a type field in these message exchanges, which specifies the kind of information that is being exchanged (Open Flow switch description, individual flow statistics, aggregate flow statistics, flow table statistics, physical port statistics, queue statistics for a port, and vendor-specific messages) and determines how the body field should be interpreted[33].

- **Queue query**: An Open Flow switch provides limited Quality of Service (QoS) support through a simple queuing mechanism. One (or more) queue(s) can be attached to a port and could be used to map flows on it (them). The flows, which are mapped to a specific queue, will be treated according to the configuration of that queue (for example, minimum rate control). Note that queue configuration takes place outside the Open Flow protocol (for example, through command-line interface) or an external dedicated configuration protocol. The controller can query the switch for configured queues on a port using queue query message [16].

• **Send packet**: Using this message (that is, OFPT_PACKET_OUT), the controller is able to send packets out of a specified port of the Open Flow switch [33].

• **Barrier**: This message is sent whenever the controller wants to ensure message. Dependencies have been met or wants to receive notifications for completed operations. The message is OFPT_BARRIER_REQUEST and has no message body. Upon receipt, the Open Flow switch must finish processing all previously-received messages before executing any message

beyond the barrier request. When current processing is completed, the switch must send an OFPT_BARRIER_REPLY message the transaction ID (xid) of the original request [33].

2.1.3 Asynchronous messages

Asynchronous messages are initiated by the switch, used to update the controller of network events, and changes to the switch state. Switches send asynchronous messages to the controller to denote a packet arrival, flow removal, port status change, or an error. When packets are received by the switch (data path), they are sent to the controller using the OFPT_PACKET_IN message [33].

When a packet is buffered in the switch, some number of bytes from the message will be included in the data portion of the message. If the packet is sent because of a send-to-controller action, then the max_len bytes are sent and if the packet is sent due to a flow table miss, then at least the miss_send_len bytes are sent. If the packet is not buffered inside the switch, then the entire packet is included in the data portion of the message. Switches that implement buffering are expected to expose the amount of buffering and the length of time before buffers may be reused. An Open Flow switch must gracefully handle the case where a buffered packet in message gets no response from the controller [33].

2.1.4 Hardware Implementations

Open Flow reference standard (Open Flow 1.0.0, Wire Protocol 0x01) is the main and early SDN enabling technologies being currently implemented in the commodity-networking hardware. Below is a brief list of a few options that are available in the market [33].

No	Manufacturer	Switch models	Open Flow version
1	Brocade	NetIron CES 2000 Series,CER 2000	1.0
2	Hewlett Packard	3500/3500yl, 5400zl,6200zl,6600, 8200zl,RackSwitch G8264, G8264T	1.0
3	IBM	Rack Switch G8264, G8264T	1.0
4	Juniper	EX9200 Programmable switch	1.0
5	NEC	PF5240, PF5820	1.0
6	Pica8	P-3290, P-3295, P-3780, P3920	1.0
7	Pronto	3290 and 3780	1.0
8	Broadcom	BCM56846	1.0
9	Extreme Networks	Black Diamond 8K, Summit X440, X460, X480	1.0
10	Net gear	GSM7352Sv2	1.0
11	Arista	7150, 7500, 7050 series	1.0

Table 6.Lists of commercial switches currently available [33]

2.1.5 Open flow controller

The three most resonant concepts of SDN are programmability, the separation of control and data planes and the management of ephemeral network state in a centralized control model, regardless of the degree of centralization. The general description of an SDN controller is a software system or collection of systems that together provides [32]:

- Management of network state, and in some cases, the management and distribution of this state, may involve a database. These databases serve as a repository for information derived from the controlled network elements and related software as well as information controlled by SDN

applications including network state, some ephemeral configuration information, learned topology, and control session information). In some cases, the controller may have multiple, purpose-driven data management processes (e.g., relational and non-relational databases). In other cases, other in-memory database strategies can be employed, too [32].

- A high-level data model that captures the relationships between managed resources, policies and other services provided by the controller. In many cases, these data models are built using the Yang modeling language [32].
- A modern, often Restful (representational state transfer) application programming interface (API) is provided that exposes the controller services to an application. This facilitates most of the controller-to-application interaction [32].
- This interface is ideally rendered from the data model that describes the services and features of the controller. In some cases, the controller and its API are part of a development environment that generates the API code from the model [32].
- Some systems go further and provide robust development environments that allow expansion of core capabilities and subsequent publishing of APIs for new modules, including those that support dynamic expansion of controller capabilities [32].

No	Open flow controller	Description
1	NOX	One of open source controllers developed by Nicira in 2008 and provides a C++ API
2	RYU	component based open source frame work entirely implemented in python
3	Floodlight	is enterprise class Apache-licensed Java based open flow controller
4	Open daylight	Open source controller that supports open flow
5	POX	python based open flow controller used for prototyping

Table 7.Some Open flow controller

2.1.6 Open flow protocol

Protocols are currently undergoing numerous mutations, owing to the advent of the Cloud and of SDN architectures. The legacy protocols based on routing of IP packets and MPLS (Multiprotocol Label Switching) are still in place for the connection of users or data transport, offered by the major operator networks. However, an entirely new generation of networks is emerging and fighting to earn a place in the new arsenal of operators and Cloud providers [35].

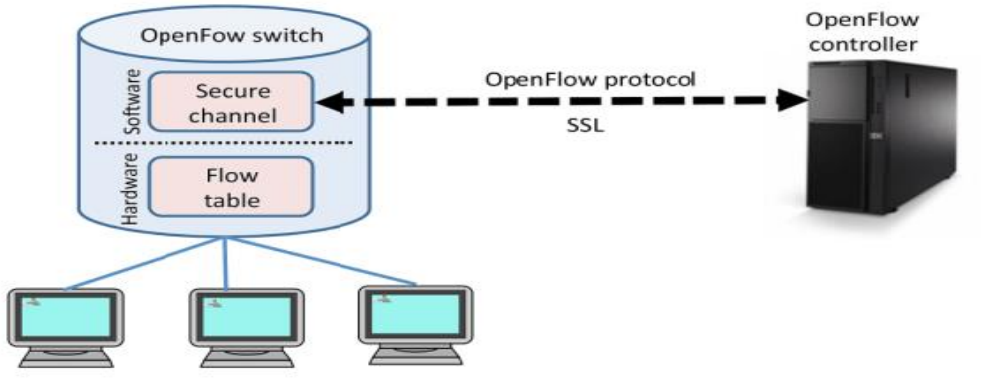

Figure 9.Open flow protocol [35]

Open flow protocol defines the communication between an open flow controller and an open flow switch. The protocol consists of a set of messages that are sent from the controller to the switch and a corresponding set of messages that are sent in the opposite direction. [36] Has explained that open flow protocol is implemented on both sides of the interface between network infrastructure devices and the SDN controller. It uses the concept of flows to identify network traffic based on predefined match rules that can be statically or dynamically programmed by SDN controller [8].

Chapter 3: Evaluation

This chapter will discuss about the input data taken from ethiotelecom which is enterprise shops customer relation management system users', simulation set up environment, network simulation test bed, software defined Network controller, load test tool, simulation network topology, results and observations will be presented.Inaddition, performance evaluation results interms of response time(sec),transaction rate(trans/sec),throughput(MB) by increasing ethiotelecom enterprise shops concurrent users' data as input including tables and graphs have been explained.

3.1 Simulation environment set up

The environment for simulation have prepared with PC:

- Processor: Intel(R) Core™ i5-240M CPU @2.50GHZ
- Installed RAM:8.0GB
- System type:64-bit Operating System X-64 based processor
- Oracle Virtual box: an open source hypervisor to run Ubuntu 16.04 OS.
- Ubuntu 16.04 OS
- Mininet: installed inside Ubuntu 16.04 OS
- POX: installed Ubuntu 16.04 OS

3.2 Network Emulation test bed: Mininet

Mininet is a popular open source network emulator capable of creating networks of virtual hosts, Switches and software defined network controllers. It was created in python programming language and provides a python application programming interface(API) for user customization and can run in a virtual machine based on Linux versions such as 16.04.It uses process based virtualization

that can create hundreds or thousands of virtual hosts and network instances on a single operating system[37].

3.3 Software defined controller: POX

POX is a python based open source Open Flow controller. It is used for faster development and protyping of a new network applications. POX controller pre-installed with the mininet virtual machine. Using POX controller, it is possible to dumb Open flow devices in to hub,switches,loadbalancer,firewall devices. The pox controller allows easy way to run SDN open Flow experiments. POX is an open source controller for developing SDN applications. It provides an efficient way to implement the open flow protocol which is the defacto communication protocol between the controllers and switches. Using POX controller it is possible to run different applications like hub, switch, and load balancer and firewall [38].

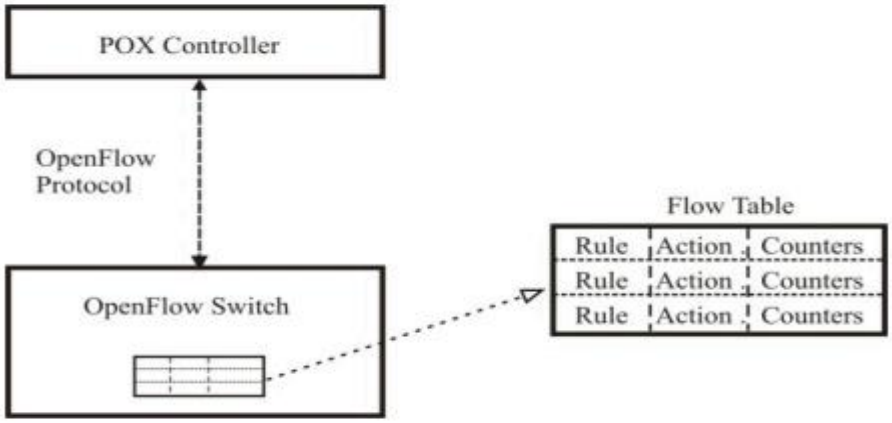

Figure 10.POX controller [38]

When a switch powered on, it will immediately connect to an open flow controller. Initially, the flow tables of the switches is empty. When a packet arrives a switch, it doesn't know, how this packet is to be handled, then it send a packet-

in message to the controller. To handle the packets, the controller inserts a flow entries in flow table of switch, flow entries in flow table contains three parts rule (match field), actions and counters. For, each packet that has to pass through a switch, a flow entry will have to be installed so that the switch can be forward this traffic without further intervention of the controller. Flow modification messages are sent to the switches to install the flow entries in flow table [38].

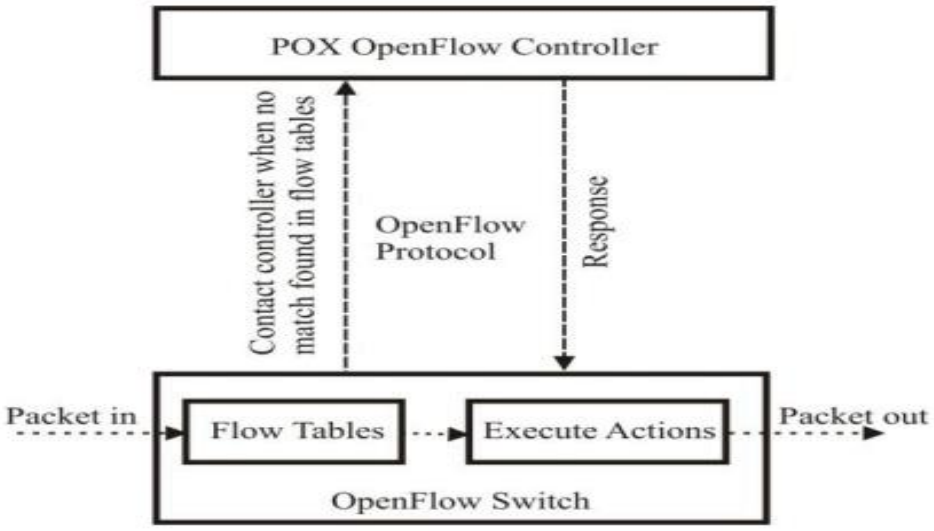

Figure 11.POX to Open Flow switch Communication in SDN [38}

3.4 Load test tool: Siege

Siege is an HTTP load testing and benchmarking utility that can be used to measure the performance of a web server when under stress. It evaluates the amount of data transferred, response time of the server, transaction rate, throughput, concurrency, and times the program returned okay. Siege offers three modes of operation: Regression, internet simulation, and brute force [39].For this

evaluation work internet simulation of siege tool has been used for concurrent users.

3.5 Simulation network topology

3.5.1 Simulation in POX controller with mininet

In this part, SDN based server cluster load balancing simulated topology using mininet in the infrastructure layer with the open flow switch, software defined POX controller connected to the open flow switch using open flow protocol in the south bound interface of the SDN open flow model. Moreover, clients send their request through internet and the servers are connected to the open flow switch.

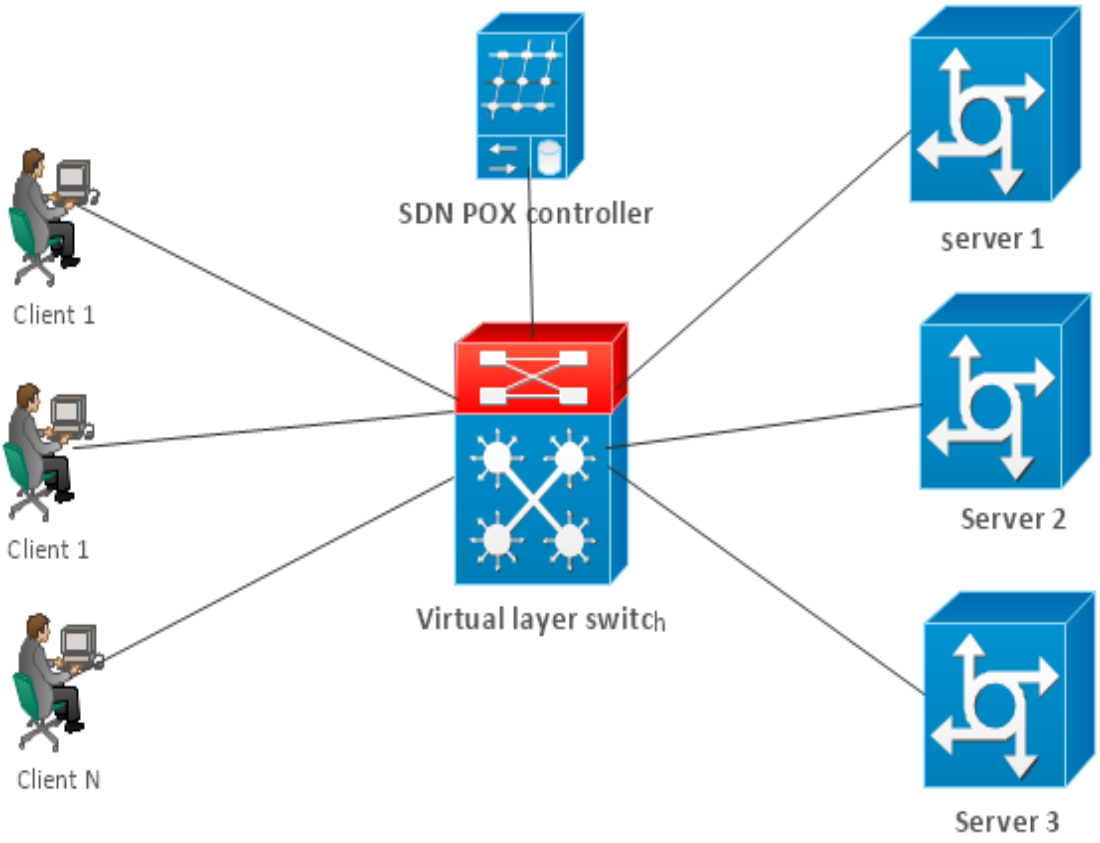

Figure 12.SDN based server cluster load balancing topology

3.6 Results and observation

The results have presented in tabular and graphical form to make easy for understanding and also to do a comparison between round robin, weighted round robin and least load server cluster load balancing algorithms.

3.6.1 CRM sales shop users Input data

In ethiotelecom, the sales activities in enterprise division have accomplished using a CRM (Customer Relation Management system) through the web internet in order to create and terminate sales orders. As it has shown in the below table, there are eight shops to serve the enterprise customers which have used only as input data for this performance evaluation of load balancing algorithms. These telecom shops are:-TPO (Transformation Project office) located around Churchill road, CAAZ (central Addis Ababa zone) located around bamibis supermarket, NAAZ (North Addis Ababa zone) located around yared music school, SAAZ (south Addis Ababa zone) located around saries area, EAAZ Bole (East Addis Ababa zone) around bole,WAAZ (West Addis Ababa Zone),SWAAZ (South west Addis Ababa Zone) and EAAZ GS (East Addis Ababa Zone GS).The D-level-level ,C-level, Sup & coor are user levels .

Ethio Telecom Shops	Zone	Number of CRM users				Total
		D -level	C- level	Corners	Sup & coor	
	Section staffs	1			1	2
shop 1	TPO	5	11	0	2	18
shop 2	CAAZ	4	3	1	1	9
shop 3	NAAZ	3	5	3	1	12
shop 4	SAAZ	3	5	3	1	12
shop 5	EAAZ Bole	4	6	3	1	14
shop 6	WAAZ	2	5	3	1	11
shop 7	SWAAZ	3	5	3	1	12
shop 8	EAAZ GS	2	6	1	1	10
	Total	26	46	17	9	100

Table 8.Distribution of ethio telecom enterprise shops CRM users in Addis Ababa

3.6.2 Response time (sec)

The response time in (sec) of round robin, weighted round robin and least load server cluster load balancing algorithms by increasing the number of concurrent users starting from 10 concurrent user to 70 concurrent users with gradual increase of 10 concurrent users in between has presented below.

ethio telecom enterprise shops concurrent users	load balancing algorithms		
	Response time(sec)		
	Round Robin	Weighted Round Robin	Least Load
10	0.09	0.14	0.11
20	0.18	0.32	0.33
30	0.27	0.45	0.37
40	0.32	0.61	0.73
50	0.46	0.77	0.8
60	0.44	0.92	0.76
70	0.54	1.1	0.76

Table 9.Results in response time (sec) by increasing the number of users

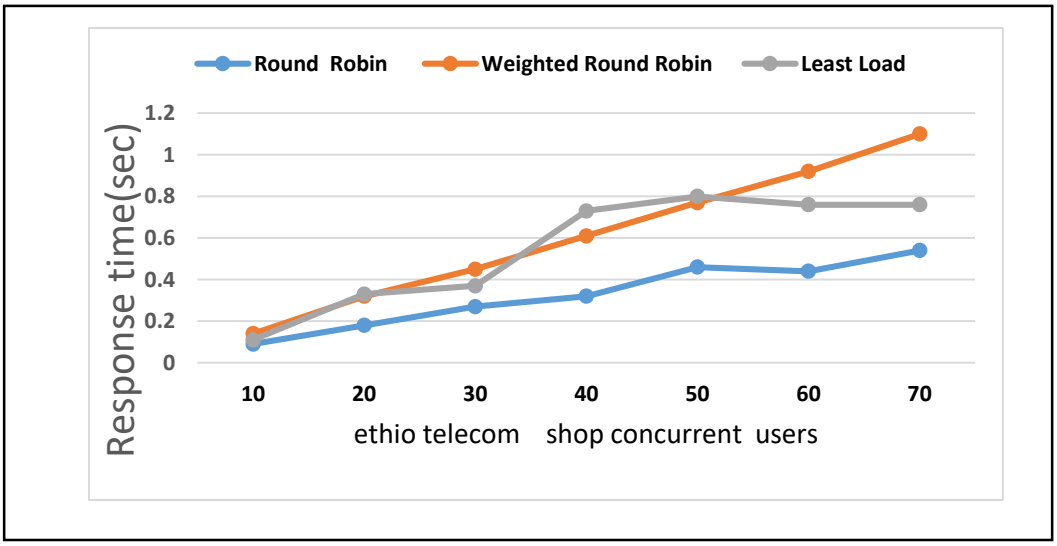

Figure 13.Response time graph

- As we see in the response time (sec) graph in figure 13,the response time(sec) of round robin, weighted round robin and least load server cluster load balancing algorithms increases as the number of concurrent users increases, but the response time(sec) of round robin server cluster load balancing algorithm is less than when compared to weighted round robin and least load server cluster load balancing algorithm. This is because of the round robin server cluster algorithm doesn't check the current performance of server cluster. It simply assigns requests in a predefined pattern in cyclic fashion.
- However, the response time in (sec) of weighted round robin server cluster algorithm increases linearly as the number of concurrent users' increases. This is because of the weighted round robin server cluster algorithm checks the weight of each server before it assigns the incoming user request and also the response time in (sec) of least load server cluster load balancing algorithm increases as the number of concurrent users increases this is because of it finds the minimum server load before it assigns the incoming request from users.
- At 60 concurrent users, least load server cluster load balancing algorithm has less response time (sec) 0.76 sec than weighted round robin having 0.92 sec.
- As referred in figure 13,when 10 concurrent users have sent their request to the load balancer, the response time(sec) for round robin is 0.09sec,the response time(sec) for weighted round robin 0.14 sec and also the response time for least load server load balancing algorithm is 0.11 sec.
- When 70 concurrent users have requested to the load balancer, the response time (sec) for round robin, weighted round robin, and least load server load balancing algorithms are 0.54 sec, 1.1 sec and 0.76 sec respectively.

3.6.3 **Transaction rate (trans/sec)**

The transaction rate in (trans/sec) of round robin, weighted round robin and least load server cluster load balancing algorithms by increasing the number of concurrent users starting from 10 concurrent user to 70 concurrent users with gradual increase of 10 concurrent users in between has presented .

Ethio telecom shops concurrent users	Server cluster load balancing algorithms		
	Transaction rate(trans\sec)		
	Round Robin	Weighted Round Robin	Least Load
10	116.91	70.87	89.9
20	110.85	63.14	60.5
30	112.34	66.26	80.75
40	124.2	64.94	54.03
50	107.96	64.53	61.55
60	136.7	64.58	78.33
70	130.14	62.81	91.12

Table 10.Results of transaction rate (trans/sec) for each algorithm

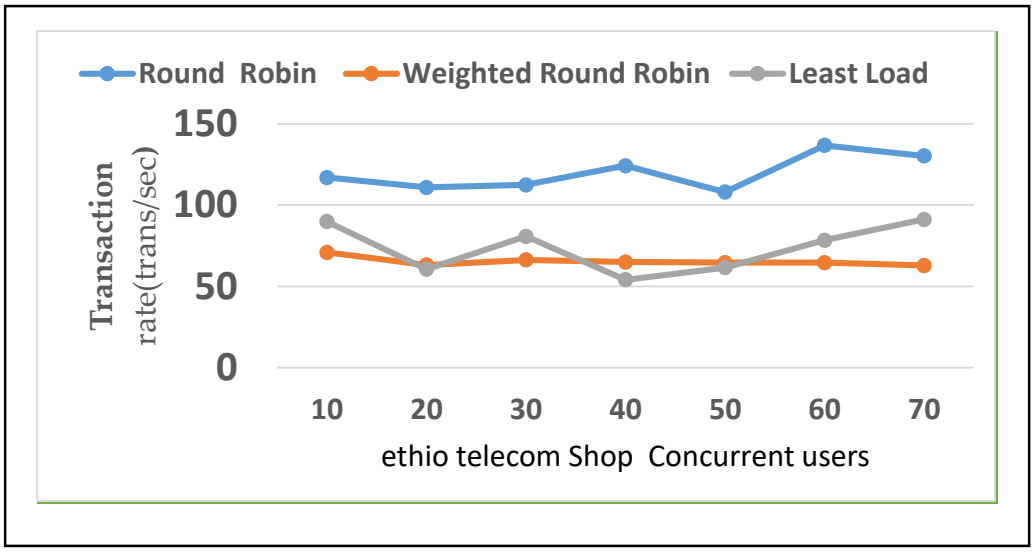

Figure 14.Transaction rate graph

- When we see transaction rate (trans/sec) graph in figure 14, round robin server cluster load balancing algorithm has a better transaction rate (trans/sec) as the number of concurrent users increases when compared to weighted round robin and least load server cluster load balancing algorithms. The reason for this is that the response time(sec) of round robin server cluster load balancing algorithm was better than weighted round robin and least load server cluster load balancing algorithms as shown in figure 13, and if round robin server cluster load balancing has less response time(sec) ,it can handle more transaction per second.
- I t has shown in figure 13, that the response time(sec) for weighted round robin algorithm increases linearly as the number of concurrent users increases from 10 to 70.If the response time (sec) of weighted round robin server cluster load balancing algorithm increases as the number of requests increases, it cannot handle more transaction because the response to a concurrent request takes more time. Therefore, it has less transaction rate than round robin server cluster load balancing algorithm.
- Again by referring figure 13, of the response time(sec) graph, the response time (sec) of least load server cluster load balancing algorithm increases as the number of concurrent users increases. It cannot handle more transactions since it has less response time (sec) than round robin server cluster load balancing algorithm. When the concurrent users has reached 40, the response time (sec) of least load server cluster load balancing algorithms increases to 0.73 sec, then the transaction rate at 40 concurrent users fluctuates and decreases to 54.03 transactions.

3.6.4 Throughput (MB/sec)

The throughput rate in (MB/sec) of round robin, weighted round robin and least load server cluster load balancing algorithms by increasing the number of concurrent users starting from 10 concurrent user to 70 concurrent users with gradual increase of 10 concurrent users in between has described below.

Ethio telecom shops concurrent users	Server load balancing load balancing algorithms		
	Throughput(MB\sec)		
	Round Robin	Weighted Round Robin	Least Load
10	0.24	0.14	0.18
20	0.23	0.13	0.12
30	0.23	0.14	0.16
40	0.25	0.13	0.11
50	0.22	0.13	0.13
60	0.28	0.13	0.16
70	0.27	0.13	0.19

Table 11.Results of throughput (MB/sec) for each algorithm

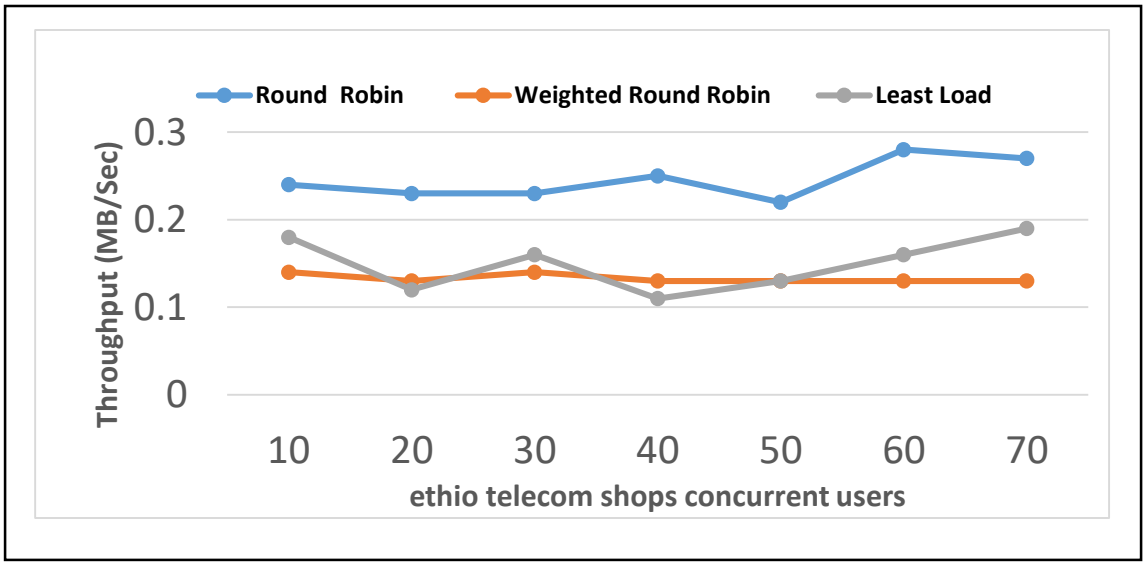

Figure 15. Throughput graph

- As we see in throughput (MB) graph in Figure 15, it has observed that the throughput(MB/sec) for round robin server cluster load balancing algorithm is better than that of weighted round robin and least load server cluster load balancing algorithms. The justification for having a better throughput(MB/sec) in round robin server cluster load balancing algorithm is that since it has a better response time(sec) and if it has a better response time(sec),it will have better transaction rate(trans/sec) and also if it has a better transaction rate(trans/sec),it will achieve better throughput(MB/sec).
- In throughput (MB/sec) in figure 15, when 20 concurrent users has sent their request to the load balancer, the throughput in (MB/sec) for weighted round robin server cluster load balancing algorithm is better because at 20 concurrent users request in figure 13, the response time (sec) of weighted round robin is 0.32 sec when the response time for least load server cluster load balancing algorithm has a response time (sec) of 0.33 sec.
- In 30 concurrent users, the throughput in(MB)graph shows that least load server cluster load balancing algorithm has better throughput because at this point least load server cluster load balancing algorithms has less response time(sec) which is 0.37sec when weighted round robin server cluster load balancing algorithms has 0.45 sec in response time.
- In 40 concurrent users, the transaction rate(trans/sec) of least load server cluster load balancing algorithm is 54.03 trans/sec, and the weighted round robin server cluster load balancing algorithms has 64.53 trans/sec, then the transaction rate of least load server cluster load balancing algorithm is less than with that of weighted round robin algorithm. It is observed that, when the response time is less, it can handle more transactions and also if it handles more transactions, it will have a better throughput (MB/sec).

Chapter 4: Conclusions

4.1 Conclusion

Round robin, weighted round robin and least load server cluster load balancing algorithms have been simulated using SDN Open Flow model in open source POX controller and Open Flow switches in mininet for topology creation on the principles of SDN.The obtained results have compared in terms of network performance parameters: response time(sec),transaction rate(trans/sec) and throughput(MB/sec) by increasing the number of ethio telecom enterprise shop customer relation management system users' as input data .Hence, Round robin algorithm is better than weighted round robin and least load server load balancing algorithms in terms of response time(sec),transaction rate(trans/sec) ,throughput(MB/sec) .

4.2 Contribution summary

- Performance evaluation of Round robin, weighted round robin and least load server cluster load balancing algorithms using SDN Open Flow model by providing ethio telecom's sales shop users' data in Addis Ababa as input.
- Since there is no any SDN deployment in the exiting server cluster load balancing implementation of ethio telecom's web based services such as CRM or DNS,this study provides an insight that round robin, weighted round robin and least load server load balancing algorithms could be implemented in SDN Open Flow model to achieve programmability and better operational efficiency.

4.3 Future work

- In this study, performance evaluation of Round robin, weighted round robin and least load server cluster load balancing algorithms using SDN Open Flow model in open source POX controller and mininet have done .As a future work, the evaluation can be done in any commercial SDN controllers.
- In SDN Open Flow model, Open Flow protocol is used to interface the control and infrastructure layers and Open Flow protocol has many versions. Therefore, a future study can be conducted on various versions of open Flow protocol for different applications.

5. References

[1] H. Zhang and X. Guo, "SDN-based load balancing strategy for server cluster," *2014 IEEE 3rd International Conference on Cloud Computing and Intelligence Systems*, 2014.

[2] C. Kopparapu, *Load balancing servers, firewalls, and caches*. New York: J. Wiley & Sons, 2002.

[3] T. Bourke, *Server load balancing*. Sebastopol, CA: OReilly, 2001.

[4] Data Center Fundamentals. *Mauricio Arregoces, CCIE No. 3285. Maurizio Portolani. Datacenter. Book, November 12, 2003.*

[5] Clustering: A basic 101 tutorial, http://www.ibm.com/legal/copytrade.shtml [Accessed ,Sept 10, 2018].

[6] B. Lebiednik, A. Mangal, N. Tiwari, *Survey and Evaluation of Data Center Network Topologies*, May 2016.

[7] Cisco Data Center Infrastructure 2.5 Design Guide, Cisco Validated Design— November 2, 2011.

[8] P. Gorasson and C. Black, *Software Defined Networks: a Comprehensive Approach*. Saint Louis: Elsevier Science, 2014

[9] Scott D. Lowe et.al, *Building a Modern Data Center Principles and Strategies of Design,* First Printing, 2016. [E-Book].

[10] P. Kumari et.al,"*Load Balancing in Software Defined Network*", Available Online at: www.ijcseonline.org [Accessed, August 10, 2018].

[11] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, W. Chou, "A roadmap for traffic engineering in SDN- Open Flow networks", *Comput. Netw,* vol. 71, pp. 1-30, Oct. 2014.

[12] S.Arlimatti, S.Hassan, A.Habbal, S.Arif,"software defined network and open flow: a critical review", VOL. 10, NO. 3, FEBRUARY 2015: Available [www.arpnjournals.com].

[13] http://www.informit.com/articles/article.aspx?p=2451956&seqNum=3[Accssed , June 2018].

[14] William Stallings ,"Foundations of Modern Networking: Background and Motivation of Software-Defined Networks (SDN), Dec 3, 2015 [Online].Available. "https://www.volico.com/what-server-load-balancing-can-do-for-your-company[Accessed, July 2018.]

[15] http://kb.linuxvirtualserver.org/wiki/Round-Robin_Scheduling[Accessed,July 2018].

[16] **http://kb.linuxvirtualserver.org/wiki/Weighted_Round-Robin_Scheduling[Accessed, July 2018.]**

[17] R. Saini , A.Bisht "A Hybrid Algorithm for Load Balancing", International Journal of Advanced Research in Computer Science and Software Engineering 5(7),July- 2015, pp. 1-6.

[18] A .Saifullah, M. Mohamed" SDN-based Server Load Balancing Using Improved Health Monitoring and Admission Control", Australian Journal of Basic and Applied Sciences, 9(27) August 2015, Pages: 132-138.

[19] G. Deep and J. Hong. 2016. Round Robin Load Balancer using Software Defined Networking (SDN). Capstone Team Research Project, Vol. 5, 1-9, 2016.

[20] M.I. Hamed, B.M. ElHalawany, M.M. Fouda, A.S. Tag Eldien, "A New Approach for Server-based Load Balancing Using Software-Defined Networking", *Proceedings of the 2017 IEEE International Conference on Intelligent Computing and Information Systems (ICICIS 2017)*, December 5–7, 2017.

[21] J. S. Sabiya, "Weighted round-robin load balancing using software defined networking", *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 6, no. 6, pp. 621-625, 2016.

[22] F. S. Fizi, S. Askar, "A novel load balancing algorithm for software defined network based datacenters", *Proc. IEEE Int. Conf. Broadband Commun. Next Generat. Netw. Multimedia Appl. (CoBCom)*, pp. 1-6, Sep. 2016.

[23] L. Chen, M. Qiu, J. Xiong, "An sdn-based fabric for flexible data-center networks" in Cyber Security and Cloud Computing (CSCloud) 2015 IEEE 2nd International Conference on, IEEE, pp. 121-126, 2015.

[24] S. K. Askar, "Adaptive load balancing scheme for data center networks using Software defined network".

[25] L. Peterson, "Central office re-architected as a data center," IEEE Commun. Mag., vol. 54, no. 10, pp. 96–101, Oct. 2016

[26] Y. Wang, Y. Dar, G. chang," SDN-Based Dynamic Multipath Forwarding for Inter-Data Center Networking", IEEE, 2017.

[27] Q.Du, Zhuang, "Open Flow-based dynamic server cluster load balancing with measurement support", *Journal of Communications*, vol. 10, no. 8, August 2015.

[29] H.Kaur, N.Jyoti," Traffic Based Load Balancing in Software Defined Networking", International Journal on Computer Science and Engineering, Vol.9.

[30] U. Zakia, H.Yedder, "Dynamic Load Balancing in SDN-Based Data Center Networks", *IEEE Annual Information Technology Electronics and Mobile Communication Conference (IEMCON)*, 2017.

[31] A. Shivendu, D. Dhakal, D. Sharma, "Emulation of Shortest Path Algorithm in Software Defined Networking Environment", *International Journal of Computer Applications (0975-8887)*, vol. 116, no. 1, April 2015.

[32] Thomas Neadeau, Software *Defined Networks*, First Edition. August 2013[E-Book].

[33] S. Azodolmodky, Software Defined Networking with Open Flow, Birmingham, U.K.: Packt Publishing, Oct. 2013.

[34] Open Flow Switch Specification, version 1.5.0(wire protocol ox04, March 26, 2015, ONF TS-025.

[35] G.Pujolle, *Networks & Telecommunication_ Advanced Networks –Software Networks Virtualization, SDN, 5G, Security*, Wiley, Volume 1, 2015[E-Book].

[36] C. Fernandez, L. Muñoz, *Software Defined Networking (SDN)* OpenFlow1.3,https://upcommons.upc.edu/bitstream/handle/2117/77684/sdnbook.pdf.zip[Accessed, August 20, 2018].

[37] C. DeCusatis, A. Carranza, J. Delgado, "Modeling Software Defined Networks using Mininet", *Proceedings of the 2nd International Conference on Computer and Information Science and Technology (CISTI6)*, no. 133, May 11 12,2016

[38] S. Kaur, J. Singh, N. S. Ghumman, *Network Programmability Using POX. International Conference on Communications Computing and Systems*, pp. 134-138, 2014.

[39] "Load Testing Web Servers with Siege," *Linode Guides & Tutorials*, 18-Feb-2015. [Online].Available:https://www.linode.com/docs/tools-reference/tools/load-testing-with-siege. [Accessed: 12-Jun-2018].

Appendix: Simulation in mininet

A.1. Creating topology in mininet

The below is the created topology to perform the evaluation.

```
wubsheta@wubsheta-VirtualBox:~$ sudo mn --topo single,6 --mac --arp  --controller=remote
--switch=ovs,protocols=OpenFlow10
```

Figure 16.Creation of the simulated topology in mininet

- **sudo**: a command to run as a root user of all privileges
- **mn**: a command to set up mininet emulator with sudo command
- **--topo single,6**: a command to create a linear topology with six nodes in mininet emulator.
- **--mac**: Auto set MAC addresses
- **--arp**: Populate static ARP entries of each host in each other
- **--controller** : software defined controller with remote options
- **--switch=ovs,protocols=OpenFlow10**: Open flow switch with Open flow protocol 1.0

A.2. Connecting remote controller, adding links, starting controller & switches in mininet

```
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4 h5 h6
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

Figure 17.Connecting remote controller and creating links

A.3. Running Software defined POX controller

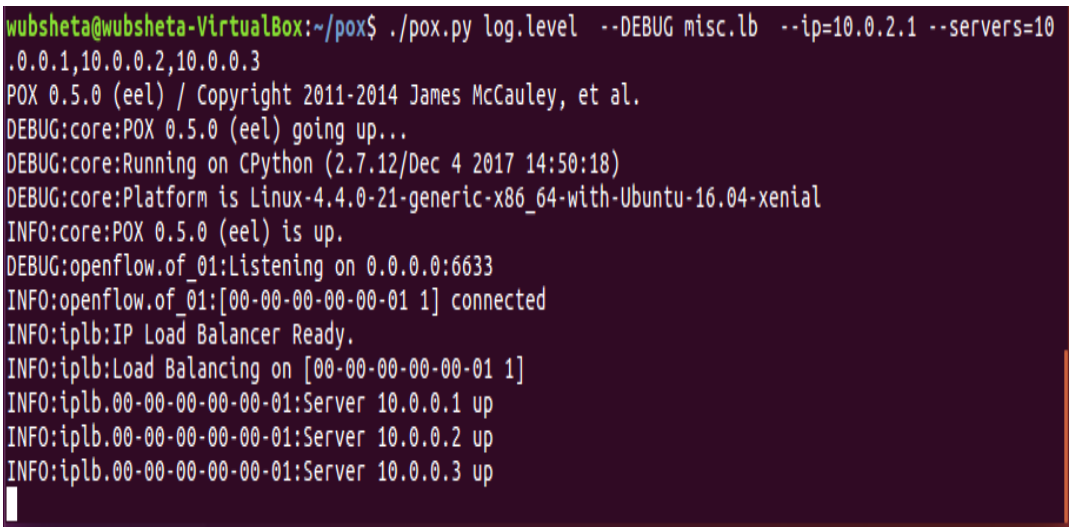

Figure 18.Running POX controller as load balancer

- ./POX.py : run POX controller
- **Log. Level --DEBUG** : log level as DEBUG open flow messages
- **Misc.lb** : is a load balancer component of POX controller
- **--ip** : create Virtual IP(VIP) for load balancer
- **--servers** : create servers

As explained in the above ,software defined network has three layers which are infrastructure, control and application layers.Therefore,in this simulation mininet has used as infrastructure layer to create the network nodes in the data plane of all the six nodes, POX controller as controller layer to manage the load balancing activity and in the application layer all the three load balancing algorithms such as round robin, weighted round robin and least load python programming language based applications have installed on the POX controller to perform in the load balancing activity in the SDN open flow model.

A.4 .Directing incoming request using round robin algorithm

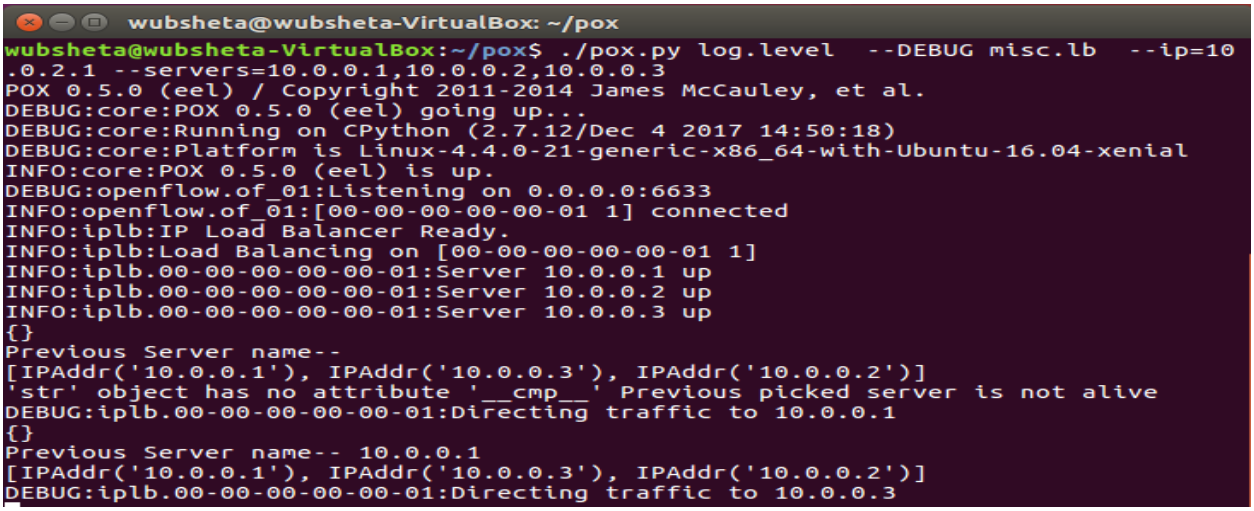

Figure 19.Directing traffic using round robin load balancing algorithm

A.5. Directing incoming request using weighted round robin algorithm

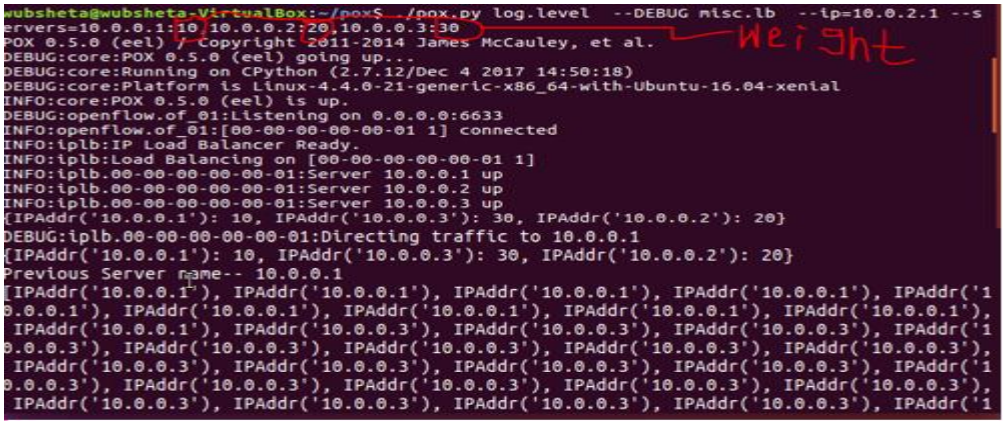

Figure 20.Directing traffic using weighted round robin load balancing algorithm

A.6.Directing incoming request using least load algorithm

```
wubsheta@wubsheta-VirtualBox:~/pox$ ./pox.py log.level  --DEBUG misc.LeastLoad_LB  --ip=10.0.2.1 --
servers=10.0.0.1,10.0.0.2,10.0.0.3
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
DEBUG:core:POX 0.5.0 (eel) going up...
DEBUG:core:Running on CPython (2.7.12/Dec 4 2017 14:50:18)
DEBUG:core:Platform is Linux-4.4.0-21-generic-x86_64-with-Ubuntu-16.04-xenial
INFO:core:POX 0.5.0 (eel) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
INFO:iplb:IP Load Balancer Ready.
INFO:iplb:Load Balancing on [00-00-00-00-00-01 1]
INFO:iplb.00-00-00-00-00-01:Server 10.0.0.1 up
INFO:iplb.00-00-00-00-00-01:Server 10.0.0.2 up
INFO:iplb.00-00-00-00-00-01:Server 10.0.0.3 up
('PORT_BYTES ARE:', {1: 77583391032, 2: 86473404472, 3: 98990761632})
{1: {1: 77583391032, 2: 86473404472, 3: 98990761632}}
('MIN bytes are:', 77583391032, 'PORT NO:', 1, 'IP of MIN port:', IPAddr('10.0.0.1'))
DEBUG:iplb.00-00-00-00-00-01:Directing traffic to 10.0.0.1
```

Figure 21.Directing round robin using least load server load balancing algorithm

A.7. Creating nodes for test

Figure 22 below shows that, using XTerm command the six nodes have created. In h1, h2, h3, h4, h5 and h6:

```
mininet> xterm h1 h2 h3 h4 h5 h6
```

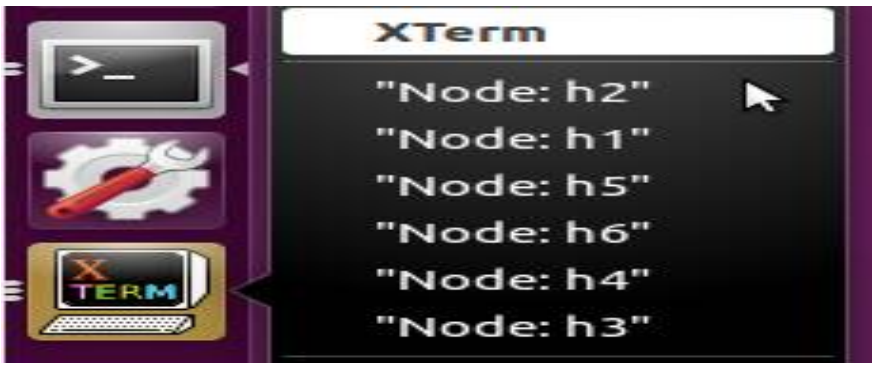

Figure **22**.Creating six nodes using XTerm

A.8.Creating http server in mininet using python –m SimpleHTTPServer 80 & command

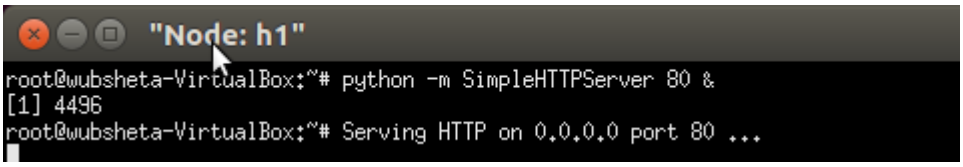

Figure 23.Creating h1 as HTTP server

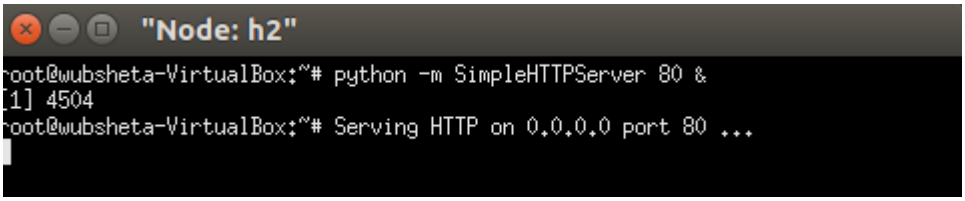

Figure 24.creating h2 as HTTP server

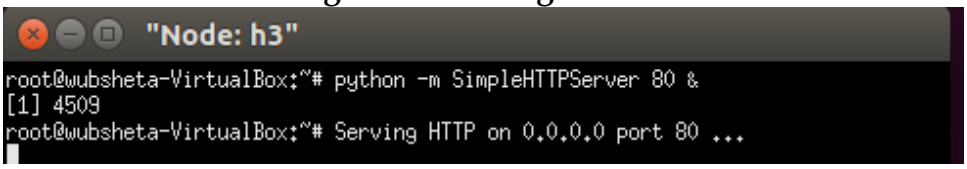

Figure 25.Creating h3 as HTTP server

A.9. Connecting client nodes with http server at port 80 using curl command

Among the created nodes,h4,h5 and h6 are used as client nodes which sends request to the created servers

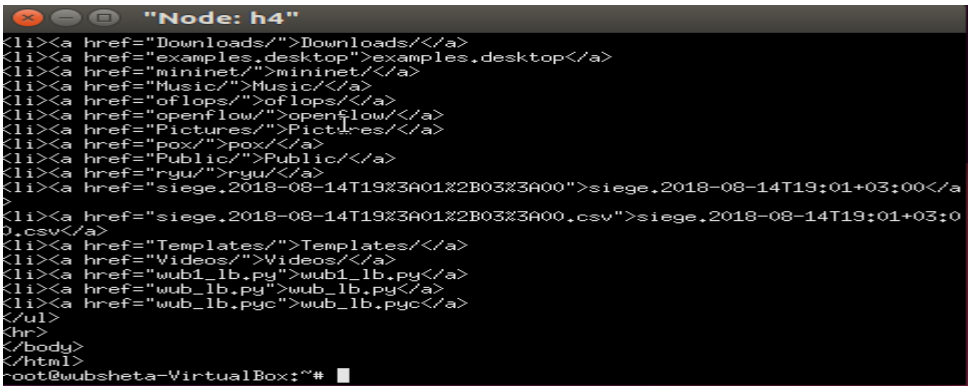

Figure 26.Connecting h4 as client node

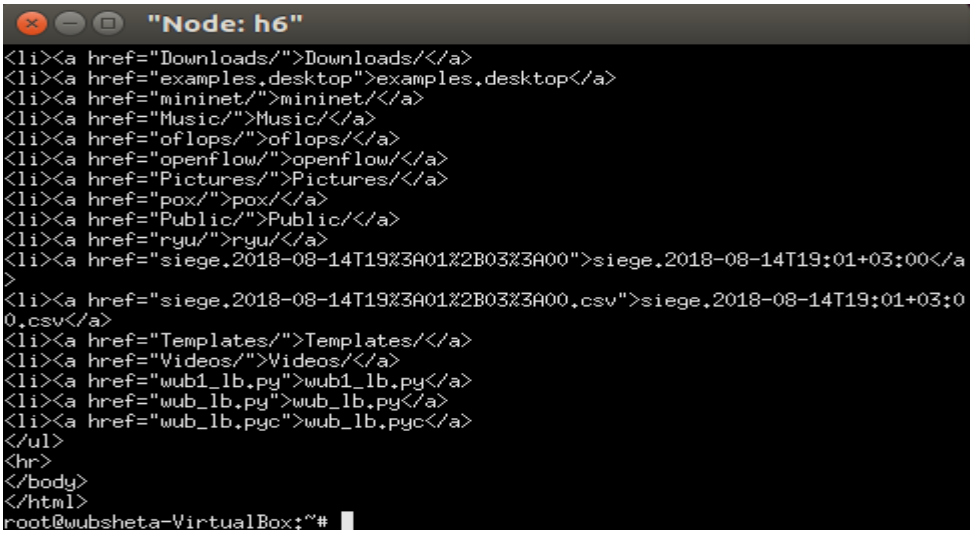

Figure 27.Connecting h5 as client node

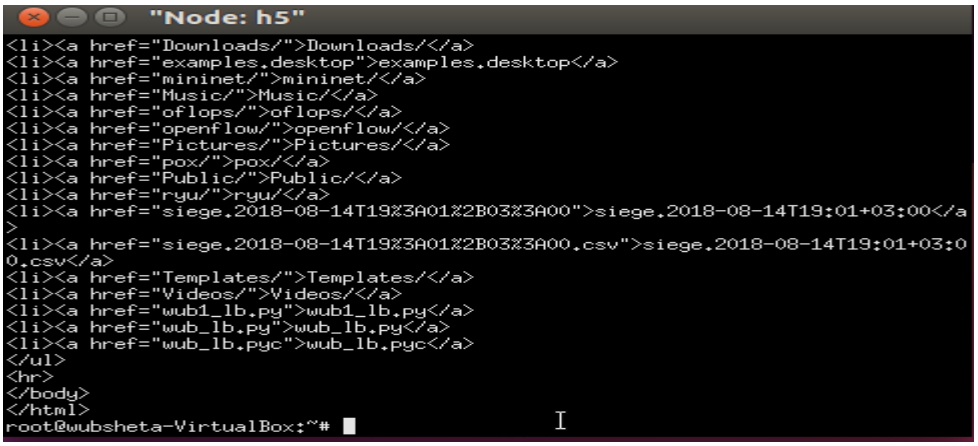

Figure 28.Connecting h6 as client node