Journal of Science and Technology
ISSN: 2362-1443 (Print) 3059-9490 (Online) 2026, Vol. 5, No. 1

Open Flow based Dynamic Traffic Distribution among Servers in Software Defined Networks

Duryodhan Chaulagain
Backbone Transmission Directorate
Nepal Telecom, Nepal
duryodhan.chaulagain@ntc.net.np

Kumar Pudashine
Nepal College of Information Technology
Pokhara University, Nepal
kumar.pudashine@gmail.com

***Abstract*—Software defined Networking (SDN) has been a major focus of research works, a solution for handling today's growing network architecture.For handling large number of web requests, network entities like Routers, Switches, Servers, Firewalls, Load Balancers,etc are deployed at different locations of globe by Web Service providers.To address issues such as network congestion and overloading,service providers use multiple replicas in the server cluster to provide the same services where network virtualization and effective load balancing is very important.This work proposes the implementation of real-time traffic management strategy based on Fuzzy Logic.The Fuzzy membership characteristic that affects performance parameters of server load has been analyzed and load state of virtual servers in real-time is evaluated through Fuzzy Logic.With three major parameters of web server (CPU,RAM and Bandwidth Utilization) as an input of fuzzy system,the clients requests are forwarded to the server having minimum load at real time.The proposed load balance algorithm written in Python is Simulated on Mininet and Performance of each web servers are compared with existing Round Robbin(RR) and Least Connections(LC)load balancing schemes.The results demonstrate that the proposed scheme has improved response time and higher throughput as compared to the other load balance solutions.**

Keywords:*SDN,OpenFlow,fuzzy Logic,Load Balancer*

I. Introduction

The expansion growth of cloud ecosystem has influenced service providers to develop their software on cloud platform and hosts all services on various server Virtual instances.It results in an increase in the number of servers that hosted various types of the services.In turn,the servers can be accessed by multi-tenant users from different locations.This raises the issue of balanced distribution,control and utilization of the available resources over the system's environment.To address this issue,Load Balance (LB) emerged as a technique used for distributing the incoming traffic among multiple web servers in order to achieve minimum response time and maximum utilization of the servers.The control plane of SDN network is a logical entity,which computes and take all routing decisions of OpenFlow packets that are generated in the data forwarding plane.

In this study,we proposed a dynamic load distribution algorithm,where input parameters (CPU,Memory and Bandwidth utilization) of the web servers has been considered for analysis of performance of server at real time,which is used for the redirection of client requests for the lightly loaded server node within the server clusters.The Proposed dynamic traffic management algorithm follows the layered architecture of SDN consisting control plane and data forwarding plane. To practically evaluate and compare the performance of proposed Algorithm with Round Robin and Least Connections Algorithm,we use Mininet with Docker Containers to setup virtual environment for simulation.Experimental results shows proposed algorithm has improved performance than Round Robin and Least Connections in heterogeneous environment with large number of concurrent users.

This paper is structured as follows.Section II covers the SDN related load balancing architectures that are relevant to the proposed work.Section III explains the main framework development and implementation of proposed algorithm.Section IV shows the results obtained from different experiment scenarios.Section V aims to draw the final remarks and conclusions of the proposed work and describes our future recommendations.

II. Related Works

The SDN architecture consists of three major components: SDN controller,a network brain,SDN switches,routers,etc. in data layer and application layer for defining and running network applications. as shown in Fig.1.The main feature of the SDN architecture is control plane and data plane are completely isolated.In addition,customized applications can be developed at the application layer using northbound interface.These applications can be used to instruct the switches using OpenFlow Protocol via secure channel [1]. In the SDN network, when a new request is sent by the client,the OpenFlow switch checks its flow table and for entry rules matched,the switch carries out the action and forwards the flows to the corresponding server.If no match is found,only the first packet of the flow is then sent as Packet-In to the controller.First packets in messages causes delays for the first time to responds the user,but after the flow entry setup traffic will flow normally between a server and the client.

Most of the popular and widely used algorithms are Round Robin algorithm, weighted round robin algorithm, Random

DOI : https://doi.org/10.3126/jost.v5i1.78945
Article History: Received: 28 Nov. 2025, Revised: 1 Jan. 2026, Accepted: 29 Feb. 2026.

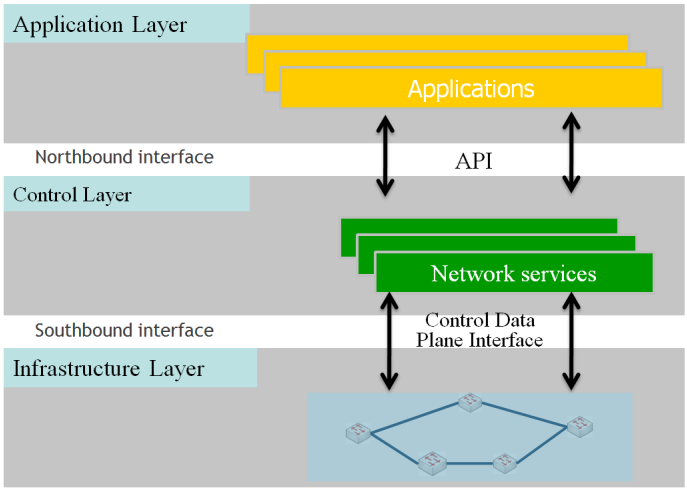

Fig. 1. SDN Architecture.

allocation and so on.The Round-Robin (RR) load balancing is popular and simpler load scheduling algorithm that distributes the request from the clients starting from first server to last server and restarts the process again in a cyclic order [2].Most of the researches predicts that RR algorithm performs efficiently where servers configuration are homogeneous.
Similarly,In Weighted Round robin (WRR) algorithm [2]load balancer distribute clients requests based on the priority given to weights of the server cluster.The server having higher weight are assigned with more web requests than web servers with less weightage.This algorithm performs better than Round Robbin for Heterogeneous server clusters. Sharma et al. [3]proposed a Random Load balancing Strategy where resource allocation among server clusters are on random basis.In dynamic load balancing,the web requests are distributed on the basis of the network real time status.In distributed load management, all servers within cluster share status information with each other.While non distributed load balancing is based on standalone server or groups of servers for transferring requests.Yi Liu et.al [4]implement Least-Loading (LL) load-balancing strategy that distributes requests to back end servers with monitoring the current usage of servers CPU,memory and disk I/O in the server cluster.The server load are calculated using linear function by assigning the administrative setting values as the weight for all attributes as per their influences. Saifullah et.al [5]proposed a server load balancing approach in SDN networks that uses current server health status monitoring.The CPU and Memory load factors are taken in consideration as a servers health parameters.These scheduling algorithm provides a improvement on overall throughput of web servers compared to existing static algorithms. Butt et.al [6] have presented a fuzzy logic control system to enhance the processor resource scheduling mechanism in distributed environment.Mason et.al [7] presented evolutionary Neural Networks (NN) model where the neural networks are used to predict the usage of CPU of the servers in advance.Datrois et.al [8]presented a machine learning approach for prediction of resource availability at the server node level.The prediction is based on the quantile regression method estimate unused resources.

III. Design and Implementation

This research work is to manage traffic efficiently among Server clusters which uses current utilization of each servers such as CPU usage,Bandwidth Usage and Memory Usage as an input parameters to find the best candidate server with least loading.This research purposes a fuzzy based algorithm to calculate the server load status at real time.Based on the predefined cutoff threshold,the overloaded servers are not participated for further load handling and only the servers within the normal limits are used for serving the client requests.With continuous monitoring, as soon as the overloaded servers have minimum load then they again participate for serving requests.

A. System Architecture

The Proposed dynamic traffic management algorithm follows the layered architecture of SDN consisting control plane and data forwarding plane and has been presented in the Fig.2 consisting different Python modules.
Whenever any user send a request for an access to a particular services,the DNS server resolves the load balancer IP address (i.e.Virtual IP address) which is not responsible for accessing web pages.Upon receiving requests from the client, load balancer then sends an acknowledgement of redirection page with IP address of the least loaded web server from the server cluster to the requesting client.

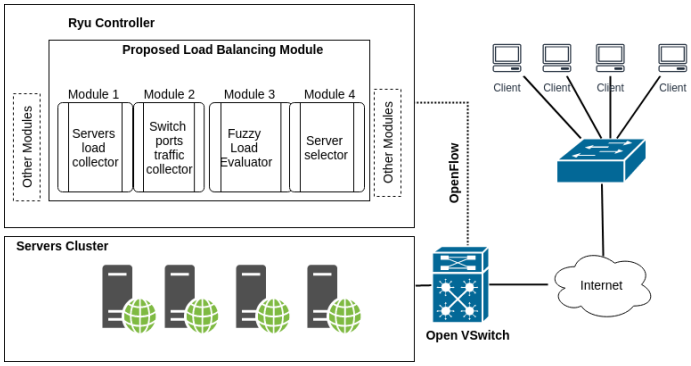

Fig. 2. Proposed Architecture of Dynamic Traffic Management.

B. Load Balancing Algorithm

For calculating best candidate server,three parameters such as CPU Usage, RAM Usage and Bandwidth Usage are considered.As soon as any request arrive from the client to Load Balancer,the background process running inside the Load Balancer will allocate the appropriate least loaded server and send flow rules to the OpenFlow switches.Each request generated simultaneously from the client are put in a queue and executed on First Come First Served(FCFS)basis.

The main load computation algorithm is based on the fuzzy decision making system.The fuzzy logic system takes crisp input and produce the crisp output with the help of knowledge base.The detail working of proposed algorithm consists of different phases as described below.

1) Servers Data Collection Module: The daemon application residing inside the servers will periodically update the load status of each server to the central controller from open flow switches.This information is sent to the controller every 5 seconds to avoid the controller itself from overloading.As shown in the figure the Controller has a "Servers Load Collector" module,which collects the current CPU,Memory usages of individual web server and stores the data in JSON format.

2) Switch Port Traffic Collector: The Open Flow switches in SDN network has a built-in counters to record statistics of each and every flow of packets that passes via switch ports.The Received counters collected at two time intervals t_1 time and t_2 are Bt_1 and Bt_2.Bandwidth Utilization(BU) is calculated as:

$$\text{BU} = \frac{(Bt_2 - Bt_1)}{P} * [8bits] \tag{1}$$

The SDN controller set a thread for collection of port statistics periodically.

3) Load Evaluation Phase: The current load status of the servers are calculated using the input parameters(CPU,MEM and Bandwidth Usage) with help of fuzzy-logic theory.The Northbound server selection application written in Python computes the load of servers using Mamdani [9] Fuzzy Inference System model(FIS).The fuzzy logic system is divided into three parts for analysis: Fuzzifier,Fuzzy Inference Engine and Defuzzifier.

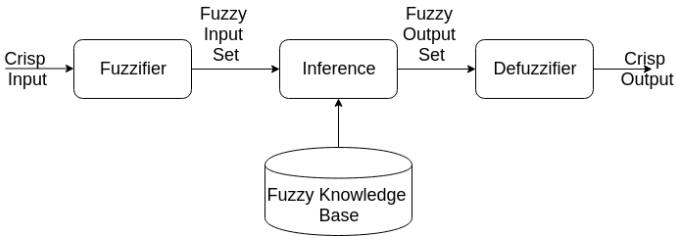

Fig. 3. Proposed Fuzzy Inference System

Fuzzifier

Let X be a universe of discourse.A fuzzy set A in X is denoted with the help of membership function$\mu_A(x)$ that associates with each point x,a real number in the interval[0,1],representing the degree of membership of X in A.

$$\text{A} = (x, \mu_A(x)); x \in X)$$

where,

$$\mu_A(x) : X \implies [0, 1]$$

The value of $\mu_A(x)$ closer to unity represents higher grade of membership of x in A.If $\mu_A(x) = 1$ then x fully belongs to A and If $\mu_A(x) = 0$,then x fully not belong to A.
In a fuzzy-logic based system,any parameter can take a whole range of values within an interval.The parameter modeled as a fuzzy variable can belong to different classes in the interval with different probabilities.Typically Sigmoid,Gaussian and pi functions are used to represent fuzzy set.However,these functions increase the time of computation.Therefore,linear fit functions are used in general.Input and output variables are defined with help of triangular and trapezoidal membership function as shown in Fig 4 and 5.

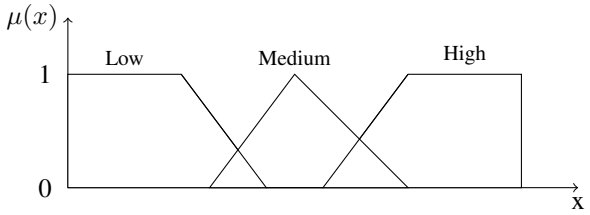

Fig. 4. Membership Function for Input Variables

CPU, Bandwidth and Memory Utilization

The current servers CPU ,Bandwidth and Memory utilization are used as domain.The fuzzy memberships are defined as a fuzzy subset of server CPU, Bandwidth and Memory load indicating membership as "high", "medium" and "low" and are computed as following:

$$\mu_l(X) = \begin{cases} 1 & \text{if } X \in [0, 25], \\ 1.5 - 2X & \text{if } X \in [25, 75], \\ 0 & \text{else} \end{cases} \tag{2}$$

$$\mu_m(X) = \begin{cases} 0 & \text{if } X \in [0, 25], \\ 4X - 1 & \text{if } X \in [25, 50], \\ 3 - 4X & \text{if } X \in [50, 75], \\ 0 & \text{else} \end{cases} \tag{3}$$

$$\mu_h(X) = \begin{cases} 0 & \text{if } X \in [0, 50], \\ 2X - 0.5 & \text{if } X \in [50, 75], \\ 1 & \text{else} \end{cases} \tag{4}$$

Server Load status

The current servers status is used as domain for output variable. The fuzzy memberships are defined as parts of fuzzy subset of current server load status indicating membership as "extreme", "strong","normal"and "light" and are computed as following:

$$\mu_l(L) = \begin{cases} 1 & \text{if } L \in [0, 10], \\ 1.5 - 5L & \text{if } L \in [10, 30], \\ 0 & \text{else} \end{cases} \tag{5}$$

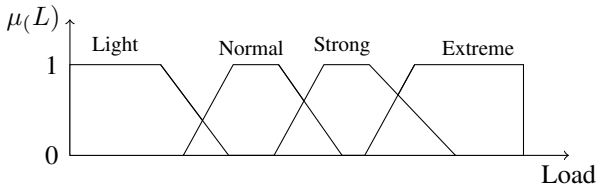

Fig. 5. Membership Function for Output Variable

$$\mu_n(L) = \begin{cases} 0 & \text{if } L \in [0,10], \\ 5L-0.5 & \text{if } L \in [10,30], \\ 1 & \text{if } L \in [30,40], \\ 3-5L & \text{if } L \in [40,60], \\ 0 & \text{else} \end{cases} \quad (6)$$

$$\mu_s(L) = \begin{cases} 0 & \text{if } L \in [0,40], \\ 5L-2 & \text{if } L \in [40,60], \\ 1 & \text{if } L \in [60,70], \\ 4.5-5L & \text{if } L \in [70,90], \\ 0 & \text{else} \end{cases} \quad (7)$$

$$\mu_e(L) = \begin{cases} 0 & \text{if } L \in [0,70], \\ 5L-3.5 & \text{if } L \in [70,90], \\ 1 & \text{else} \end{cases} \quad (8)$$

Fuzzy Inference Engine

For the implementation of above membership functions,we create the fuzzy sets based on heuristics by dividing the domain of discourse into trapezoidal and triangular fuzzy sets with 10% to 20% overlap typically.In this research,For easy deployment and understating we have considered combinations of two input variable t_1 and t_2 to be considered as memory,bandwidth or CPU usage in a cyclic order.These two input parameters along with the Server load status are mapped and illustrated in Fuzzy Associative Memory with appropriate linguistic notation as shown in Fig.6 Here,we have taken Mamdani method which captures experts

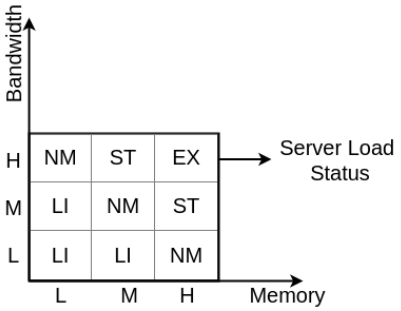

Fig. 6. Fuzzy Associative Memory Rule

knowledge and is widely used in developing Decision Support systems.For three crisp input CPU,memory and Bandwidth there are 27 possible rules are defined.

For the proposed algorithm,Let t1 and t2 are inputs and L is output control variable then the fuzzy rules are generally designed as follows:
IF (antecedents) THEN (Conclusion)
The fuzzy inference rules are written in such a way that are equivalent to fuzzy conditional proposition of the form
IF (t1,t2) is $X \times Y$,THEN L is Z,
where,

$$[X \times Y](p,q) = min[X(p), Y(q)] \quad (9)$$

for all $p \in [0,a]$ and $q \in [0,b]$
The output variable Server load status L becomes problem of approximate reasoning with composite inference fuzzy proposition.The fuzzy rule base consists of n fuzzy inference values,then
Rule 1: IF (t1,t2) is X_1 x Y_1, THEN L is Z_1,
Rule 2: IF (t1,t2) is X_2 x Y_2, THEN L is Z_2,
...
Rule n: IF (t1,t2) is X_n x Y_n, THEN L is Z_n,
The Symbols X_i,Y_i and Z_i (i=1,2,...n) denote fuzzy sets that represent the linguistic states of variables t1,t2 and L. The rules are explained in terms of relation R_i and are disjunctive in nature.The output variable L is defined by the fuzzy set as C=$\bigcup_{i \in I}(f_{t_1}(p_0)$ x $f_{t_2}(q_0)),o^j R_i$ where o^j is the sup-j composition for a t-norm j.The choice of the t-norm is a mater similar to the choice of fuzzy sets for given linguistic labels. The pseudo code is mentioned on Algorithm 1.

Algorithm 1 Fuzzify Inference Engine Algorithm

1: **Input**
$k = 3$ Number of fuzzy Inputs
$N = 3$ Number of membership function per input
$R = N^k$ Number of fuzzy rules
$R = N^k$ Number divisions for centroid calculation
$\mu[R][1...k]$ Matrix of membership values for each MF
$\mu_o[R][G]$ Matrix of output membership function values.
2: **Variables**
$\mu_R[R]$ Array of membership values for each rule
$\mu_A[G]$ Array of activation values aggregated over all rules
3: **for** x:1 step 1 until R **do**
$\mu_R[x]$:= min$\{\mu[x][1], \mu[x][2], .., \mu[x][k]\}$
4: **end for**
5: **for** x:1 step 1 until G **do**
$\mu_A[x]$:=max$\{$min$\{\mu_R[1], \mu_o[1][x]\}$,..,min$\{\mu_R[R], \mu_o[R][x]\}\}$
6: **end for**

Defuzzifier

The defuzzification is a final process of fuzzy system to produce a quantifiable crisp value from the fuzzy input set.As single quantitative value is required,the crisp output of the system is defuzzified using Weighted Average Method defined as follows

$$X^* = \frac{\sum_{i=1}^{n} \mu(x_i).x_i}{\sum_{i=1}^{n} \mu(x_i)} \quad (10)$$

where $\mu(x_i)$ is the weighted strength and x_i is the centre point of the i^{th} output membership function and n is the total number of output membership functions.The above equation results a crisp output value regarding the load of the server.

4) Decision Phase: This python module will now fetch the information of Server load in JSON format and check whether the current load for each server is under a threshold(δ)or not and discard the server from the member list if load greater than 70% and sort the normal servers with "index 0" being a least loaded server.The pseudo code of algorithm is mentioned below. Load Balancer have a knowledge of least loaded

Algorithm 2 Server Selection Module

```
for Every-time System Startup do
    Fetch Server Load Data in JSON Format
    if load of S_i > δ then
        Remove server from active member list
    else
        Add and sort servers with least load at beginning
    end if
    Fetch Sid with index [0]
end for
```

server prior to the requests from clients as these monitoring and selection processes are already running inside the data center.As soon as the Load Balancer VIP get requests then it updates flow rules on OpenFlow Switches and clients requests are assigned to the server dynamically.

IV. Performance Evaluation

A. Experimental Environment

A suitable network topology for testing load balancing is simulated in Mininet [10] using python API.It consists of 2 OpenFlow switches,3 docker hosts [11] connected on edge switches which are configured as Apache web servers,clients are connected with core switch and a remote controller on port: 6633 as shown in the Fig.7 As Mininet provides network

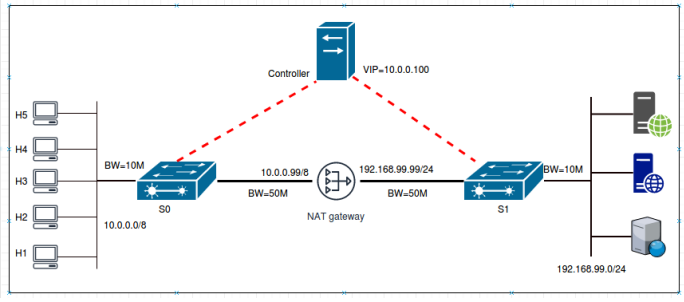

Fig. 7. Topology Used for Simulation

isolation between different hosts but the file system and the processes of the base machine are shared with each other.So for simulation purpose,we deployed Mininet environment with docker containers to ensure that the applications running on the servers are completely isolated from each other.

B. Experimental Design

The Python script is written on each back-end server for sending the load status to the central controller periodically.Siege bench marking tool [12] is used to measure the performance of RR,LC and the proposed Dynamic Traffic Management algorithms.During Simulation,a large amount of requests through concurrent connections are sent to the server clusters using Virtual IP.The virtual machines has been restarted for every test carried to avoid the influence of caching on performance.For the validation purpose the simulation was carried out three times with concurrency of users as 2000,4000 and 8000 for 30 min time interval.

C. Experimental Results

The line graphs in Fig.8 and 9 compares the three load balancing algorithms in terms of their Throughput,Average response time and error rate for the Concurrent Users starting from 2000 to 20000.It has been observed that the Average Throughput of proposed load balance algorithm decreases as the number of generating requests increases,but with lower rates than RR and LC algorithms.Since,web servers are installed with minimal configuration,the throughput graph remains almost flat after reaching its maximum value. Similarly, average response time increases as the number

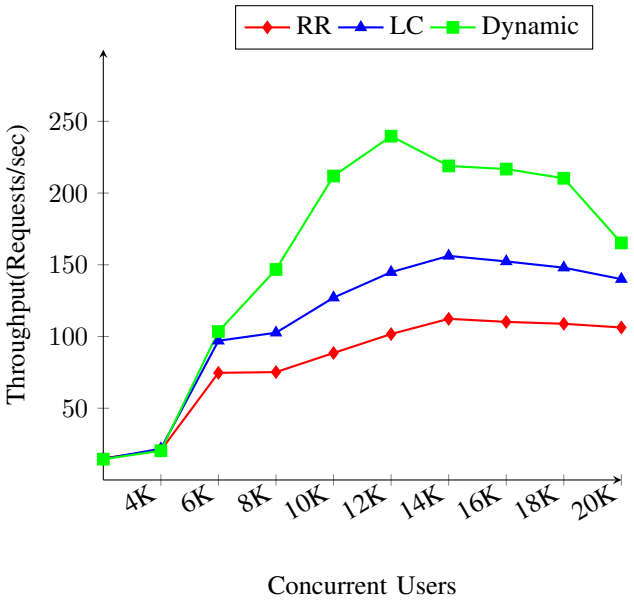

Fig. 8. Comparison of Throughput Tests Results

of generated requests increases but slower than RR and LC algorithms.In other words,proposed algorithm has better results for response time as this parameter should be kept as small as possible.

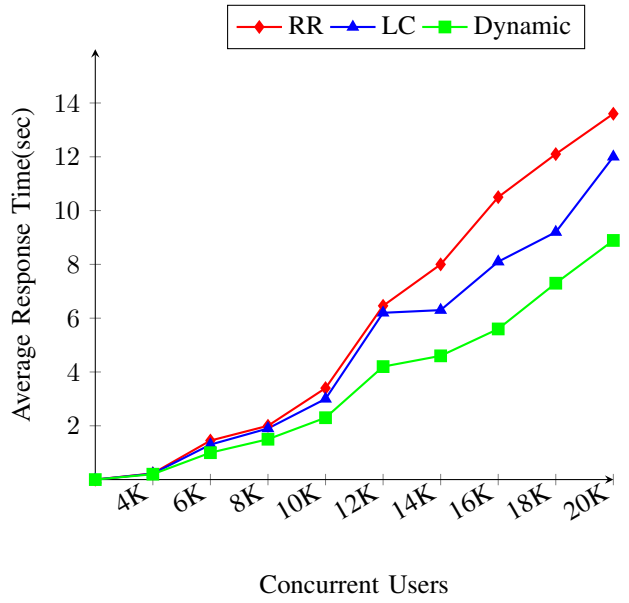

Fig. 9. Comparison of Response Time Tests Results

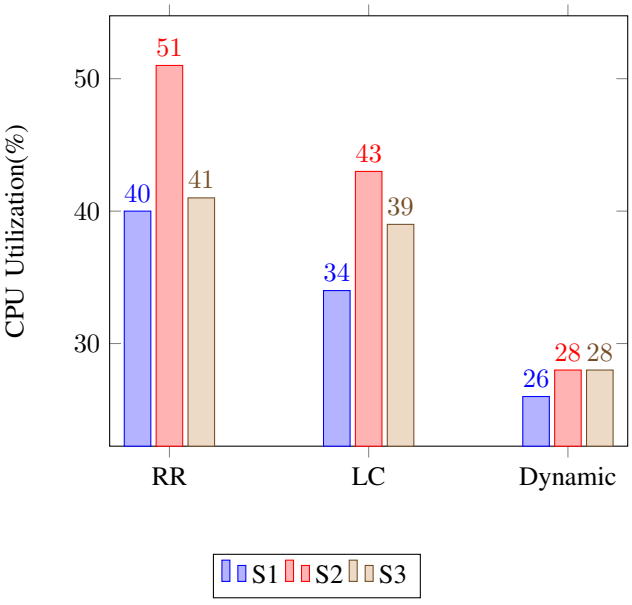

Fig. 10. Comparing Overall Server's CPU Utilization

From Fig.10,it is observed that CPU utilization are much closer for proposed algorithm whereas there is a gap in the servers CPU utilization in RR and LC algorithms.It means that proposed dynamic traffic management algorithm can allocate user requests better and make rational utilization of CPU for each server.

V. Conclusion and Future Work

An efficient dynamic traffic management algorithm based on Fuzzy Inference System has been proposed to distribute web traffic load to reduce overloading of the web servers considering CPU,RAM and Bandwidth resources. Further,Considering a heterogeneous environment of servers,a comparative analysis of the proposed load balancing technique has been performed with the existing algorithms(RR and LC).The simulation results shows that the proposed load balancing algorithm provides improved results for large number of users simultaneously accessing the web pages than the existing algorithms. In future the proposed work can be rerun in a real hardware environment and could simulate with large number of web servers.Another possible feature that may be added is to use multiple controllers in master slave configurations with the proposed mechanism to avoid a single point of the failure problem.

References

[1] Z. Bozakov and V. Sander, "Openflow: A perspective for building versatile networks," in Network-Embedded Management and Applications, pp. 217–245, Springer, 2013

[2] S. Kaur, K. Kumar, J. Singh, and N. S. Ghumman, "Round-robin based load balancing in software defined networking," in 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 2136–2139, IEEE, 2015.

[3] S. Sharma, S. Singh, and M. Sharma, "Performance analysis of load balancing algorithms," World Academy of Science, Engineering and Technology, vol. 38, no. 3, pp. 269–272, 2008.

[4] H.-Y. Liu, C.-Y. Chiang, H.-S. Cheng, and M.-L. Chiang, "Openflow-based server cluster with dynamic load balancing," in 2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pp. 99–104, IEEE, 2018.

[5] M. A. Saifullah and M. M. Mohamed, "Open flow-based server load balancing using improved server health reports," in 2016 2nd International Conference on Advances in Electrical, Electronics, Information, Communication and BioInformatics (AEEICB), pp. 649–651, IEEE, 2016.

[6] M. A. Butt and M. Akram, "A novel fuzzy decision-making system for cpu scheduling algorithm," Neural Computing and Applications, vol. 27, no. 7, pp. 1927–1939, 2016

[7] K. Mason, M. Duggan, E. Barrett, J. Duggan, and E. Howley, "Predicting host cpu utilization in the cloud using evolutionary neural networks," Future Generation Comp. Syst., vol. 86, pp. 162–173, 2018.

[8] J.-E. Dartois, A. Knefati, J. Boukhobza, and O. Barais, "Using quantile regression for reclaiming unused cloud resources while achieving sla," in 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pp. 89–98, IEEE, 2018.

[9] M. S. Khan, "Fuzzy time control modeling of discrete event systems," ICIAR- 51, WCECS, pp. 683–688, 2008.

[10] B. Lantz, N. Handigol, B. Heller, and V. Jeyakumar, "Introduction to mininet," Mininet Project,[Available: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet.[Accessed On: 12-12-2019], 2014

[11] M. Peuster, H. Karl, and S. van Rossem, "Medicine: Rapid prototyping of production-ready network services in multi-pop environments," in 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), pp. 148–153, Nov 2016.

[12] "Siege benchmarking tool," Available: [https://www.tecmint.com/load-testingweb-servers-with-siege-benchmarking-tool/