An Intelligent Server load balancing based on Multi-criteria decision-making in SDN

Original Scientific Paper

Vani K. A.
Department of Information Science and Engineering,
Dayananda Sagar College of Engineering, Visveshwaraya Technological University, Bangalore, India
vanika-ise@dayanandasagar.edu

RamaMohanBabu K. N.
Department of Information Science and Engineering,
Dayananda Sagar College of Engineering, Visveshwaraya Technological University, Bangalore, India
ramamohanbabu-ise@dayanandasagar.edu

Abstract *–In an environment of rising internet usage, it is difficult to manage network traffic while maintaining a high quality of service. In highly trafficked networks, load balancers are crucial for ensuring the quality of service. Although different approaches to load-balancing have been proposed in traditional networks, some of them require manual reconfiguration of the device to accommodate new services due to a lack of programmability. These problems can be solved through the use of software-defined networks. This research paper presents a dynamic load-balancing algorithm for software-defined networks based on server response time and content mapping. The proposed technique dispatches requests to servers based on real-time server loads. This technique comprises three different modules, such as a request classification module, a server monitoring module, and an optimized dynamic load-balancing module using content-based routing. There are a variety of robust mathematical tools to address complex problems that have multiple objectives. Multi-Criteria Decision-Making is one of them. The performance of the proposed scheme has been validated by applying the Weighted Sum Method of the multi-criteria decision-making technique. The proposed method Server load balancing based on Multi-criteria Decision Making[SDLB-MCDM] is compared with different load-balancing schemes such as round robin, random, load-balancing scheme based on server response time [LBBSRT], and An SDN-aided mechanism for web load-balancing based on server statistics [SD-WLB]. The experimental results of SDLB-MCDM show a significant improvement of 58% when weights are equal and 50% when unequal weights are assigned to various QoS parameters in comparison with the ROUND ROBIN, RANDOM, LBBSRT and SD-WLB techniques.*

Keywords*: Quality of Service. Software-Defined Networks, Load-Balancing, Open Flow*

1. INTRODUCTION

Over the past few years, there has been a remarkable increase in services residing in modern data centers. Some of the critical components of data centres include different types of servers, storage systems, switches, routers, and application delivery controllers. The applications of data centres range from social networking, video streaming, web search, data storage, data processing and many more. With these growing applications, the frequency of communication between the nodes has increased to a greater extent.

Further, the users who access these applications expect greater QoS with a minimum response time from the application servers. The response time is the amount of time a service provider takes to respond to a request.

However, data centre operators must deal with the complexity of managing traffic both within and across data centers. This includes providing the necessary resources and establishing a connection, regardless of how they are hosted. On the other hand, network management and dynamic configuration using traditional networks impose a challenging task. The configuration of the network components in traditional networks is very laborious and time-consuming for the network operators. This is due to fixed functionalities of network components, vendor dependency and structural complexities and many more [1]. This architecture consists of a control plane, a data plane, and a management plane, as shown in Fig. 1. The control plane and the data plane are decoupled. The entire global view will be present in the controller, which acts as the brain of the network. The data plane is regarded as the forwarding plane that governs the flow rules laid out by the controller. The communication between SDN controllers and data plane elements is carried out via the Open Flow

protocol. This protocol enables flow-level programmability in software-defined networking, which may be used to program the network according to application QoS needs as well as network traffic conditions [2,3].

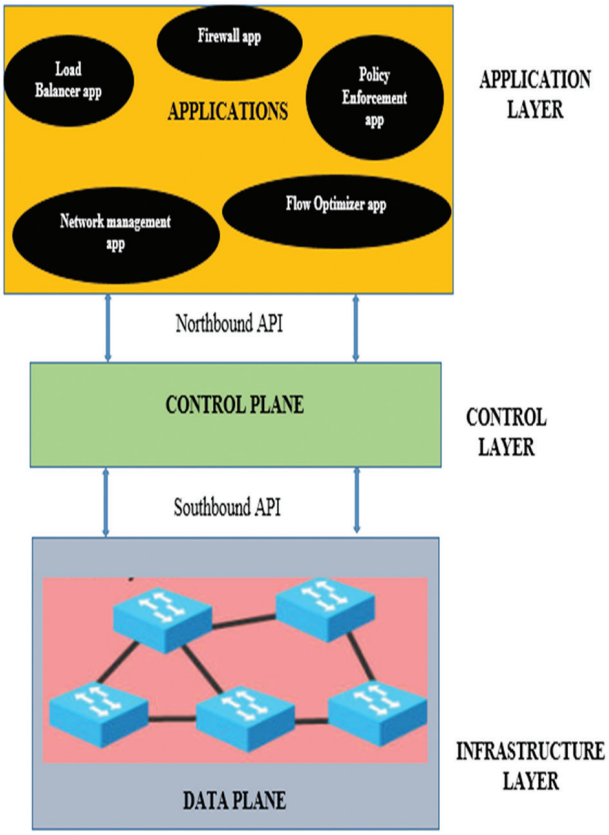

Fig. 1. SDN Architecture

During heavy traffic scenarios, deploying a dynamic load-balancing technique can aid in managing the network traffic more effectively. However, optimizing the response time while handling heavy network traffic and mapping the content is another challenging task. Though much research is carried out to address load-balancing in SDN, as discussed in [4], most of them perform load-balancing either based on the server's response time or content mapping. But this proposed research work takes both response time and content mapping into account while performing load-balancing in the server pool.

The major contribution of this research work is server load-balancing based on response time and content mapping, as well as optimization of routing rules and mathematical analysis using Weighted Sum Method [WSM] of the Multi-Criteria Decision-Making [MCDM] technique to select the best server in the server pool.

The paper is organized as follows: Section 2 discusses the related work and introduces an overview of software-defined networks along with strategies for load-balancing. Section 3 reviews the proposed model. Section 4 covers experimentation Section 5 covers evaluation and results. Section 6 concludes the work with a future scope.

2. RELATED WORK

There have been a number of studies on load- balancing in software-defined networks. Nevertheless, this research work focuses on providing a dynamic load-balancing solution in data center networks that is based on response time in SDN. The response time is one of the crucial aspects when we are evaluating the QoS of any network. Some of the research related to controller response times and server response times is discussed in this section.

2.1. LOAD BALANCING BASED ON THE CONTROLLER'S RESPONSE TIME

This section provides some of the research work related to load-balancing based on a controller's response time.

The authors in [5] proposed an SDN framework for load-balancing based on the controller's response time that makes use of network heterogeneity and context-aware vertical mobility concepts. This scheme designed a mechanism for load dissemination between controllers called reducing the overhead. The scheme studies the bandwidth requirement based on ongoing traffic, not the type of service requirement. The study by Senthil et al. aims to compare the performance of two load-balancing algorithms, flow-based load-balancing and traffic pattern-based load-balancing, using distributed controller architecture [6]. Authors in [7] provided a mathematical analysis of existing techniques in SDN and proposed the Response Surface Methodology to reduce the response time of a controller. While adding a new QoS policy to this scheme requires repetition and analysis to determine the QoS-related outcome.

To reduce the response time during load-balancing among the controllers, a two-phase dynamic controller clustering is proposed in [8]. According to the scheme, the optimal cluster size was not taken into consideration. The majority of research studies achieved load-balancing during heavy loads but did not achieve continuous load-balancing among the controllers. To address this issue, a new scheme named multiple threshold load-balancing (MTLB) switch migration scheme is proposed in [9]. Most of the research focused on the static assignment of controllers and switches. Due to this, some of the controller's response time was high. In order to reduce the controller's response time, a two-phase algorithm is proposed in [10].

2.2. LOAD-BALANCING BASED ON SERVERS RESPONSE TIME

In conventional networks, it was extremely challenging to take advantage of server reaction time due to hardware restrictions. Many academics have suggested load-balancing plans based on server response time in SDN to fill this need. In this section, several of these methods are covered. The authors of [11] recom-

mend load-balancing based on server response time. This scheme supports the same kind of data traffic as the other scheme.. The research work discussed in [12] performs server load-balancing based on switch port statistics. [13] Investigated how to maximize server utilization while minimizing response time in a cloud environment using SDN-based load- balancing. This scheme makes use of an application module and server pool. The type of service provided is classified as compute request or data request in this case. The authors of [14] discussed multiple server tests in demonstrating the quality of service with a limited number of servers to demonstrate the benefits of SDN in accessing servers. To assess performance, the scheme compared round-robin, random, and least-bandwidth algorithms. In order to exploit the dynamic performance of servers using SDN and to showcase the limitations of traditional networks, the authors in [15] have designed server load-balancing based on round-robin and weighted round-robin techniques using POX controller [16]. However, this technique attempts to address server load-balancing using the POX Controller.

For the efficient distribution of load among multiple servers based on bandwidth and round-robin fashion, the authors in [17] have proposed server load-balancing using SDN. This scheme compared the results of bandwidth-based and round-robin-based load- balancing and proved that the former yields better results in comparison with the round-robin technique. Based on the concept of server clustering that is widely used to provide availability and achieve high performance and scalability, the study in [18] proposed a novel dynamic weighted random selection load- balancing algorithm. This technique considers real-time server loads when assigning requests among the servers. This method works well in a single-controller architecture. The authors in [19] proposed a multiple regression-based search algorithm for selecting an optimal server with an optimal routing path. The scheme distributes the traffic to the server with the fewest connections and the lowest path cost from the floodlight controller. Further utilizing the concept of correlation analysis, this scheme predicts the response time based on the load and bandwidth.

This proposed method considers diverting the incoming requests to the appropriate server based on the type of traffic with optimized routing rules.

3. PROPOSED METHOD

The proposed system model is depicted in Fig. 2. The system is composed of clients and servers connected to a Ryu controller, along with a load balancer module.

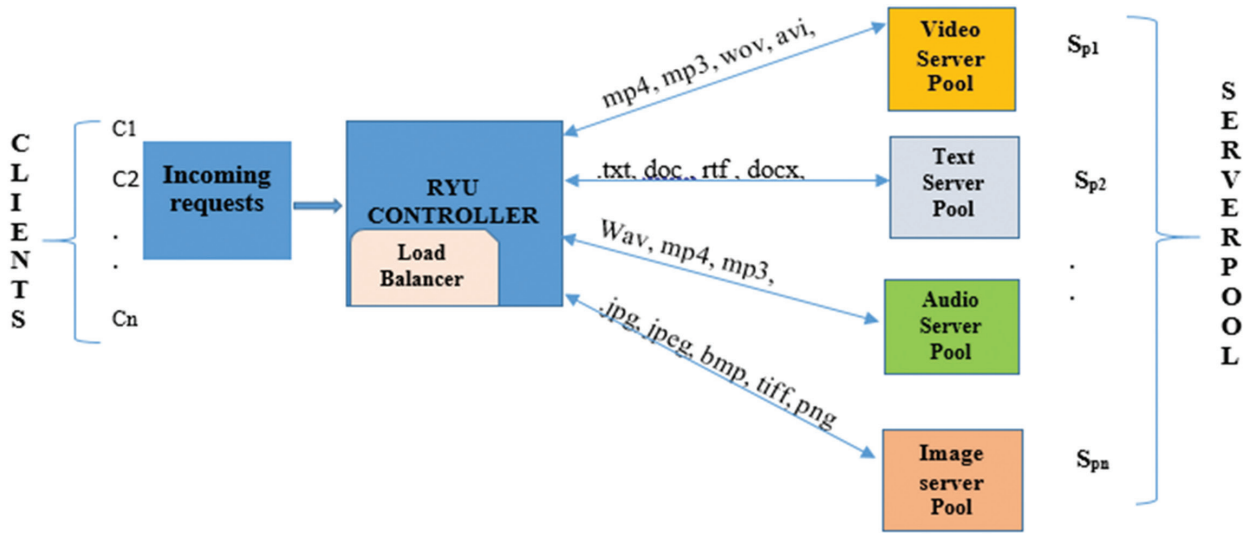

Fig. 2. System Architecture

The proposed model works on the principle of optimised routing rules laid out by the controller. This model is designed to support web services at different server pools. At each level, the controller directs the requests to the respective servers in the server pool based on the load balancer result for the required content type and response time. The different server pools are classified as video server pool, audio server pool, image server pool, and text server pool, respectively, as depicted in Fig. 2. This architecture consists of mainly three modules, namely: the request classification module, the server monitoring module, and the optimized dynamic load-balancing module. These three modules are discussed in detail below.

3.1 REQUEST CLASSIFICATION MODULE

The main idea behind creating this module is to classify the type of request based on its content. The model makes use of URL mapping instead of regular IP map-

ping. The request classification module is depicted in Fig. 3 below. Let us consider a scenario where the client requests a video by specifying it in the URL. e.g., myapp/switch/app/video. The request is sent to the controller via the OpenFlow Zodiac switch. The classification module determines the type of request, such as video, image, text, or audio. Once this information is extracted, it is sent to the load-balancing module.

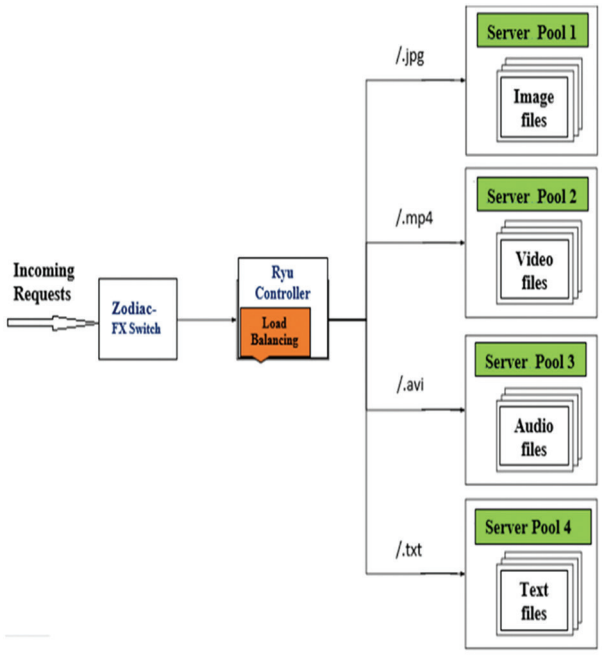

Fig. 3. Request classification module

The algorithm for request classification module is described in algorithm 1.

Algorithm 1: Request classification module

Input : Server Metrics

Output: Request classification

While*(true)*

Read (*Content-Type ='img')*

If *(Content-Type ='img')*

Send image data

If *(Content-Type ='video')*

Send video data

If *(Content-Type ='txt')*

Send text data

If *(Content-Type ='audio')*

Send audio data

End

3.2 SERVER MONITORING MODULE

This module is implemented in such a way that the servers in the server pool keep sending the load information to the controller every 5 milliseconds [ms]. The algorithm for this module is described in Algorithm 2. The severity metrics, like CPU utilization, memory utilization, requests per second, time per request, transfer rate, waiting time, and many more, are sent to the controller. The most interesting part of this module is the response time of the server. The response time of each server in different server pools is collected via this module based on real statistics.

Algorithm 2: Server Monitoring

Input: Server metrics

Output: Server monitoring

Start

While *(true)*

start the servers

if*(time='T' ms)*

start the server script for sending metrics

for each*(T=5 ms)*

Send metrics like CPU utilization, Memory Utilization, requests_per_second, time_per_request, transfer_rate, waiting_time to the controller

time.sleep (*INTERVAL_SECONDS*)

end

3.3 OPTIMIZED DYNAMIC LOAD-BALANCING MODULE USING CONTENT-BASED ROUTING

This module implements dynamic load-balancing using content-based routing. Upon the arrival of the client's request, the content is parsed by the load-balancing module in the controller, which runs algorithm 3 to find the server with the least response time in each server pool, and the controller installs the flow based on the requested content and the server with the least response time. Based on the content, for example, if the request pertains to images, it will be forwarded to the image server pool; similarly, if the request is to retrieve video, it will be forwarded to the server that handles video; the same holds true for text and audio files.

Algorithm 3: Optimized Load-Balancing module based on the requested content

Input: server metrics

Output: Best server $[B_S]$ with fast response time

Start the RYU controller

while *(true)*

Initially B_s*=null*

if (*time='T' ms*)

Collect server metrics and run the optimized load balancer module

Initialize Load-balancing module to Read the content of the request

if*(Content Type= 'Img")*

send the request to image server queue

if*(Content Type= 'video")*
send the request to video server queue
if*(Content Type= 'text")*
send the request to text server queue
if*(Content Type= 'audio")*
Send the request to audio server queue
Calculate the server with least response time[Rt]
Forward the requested content to the server with minimum response time [Rt], according to equation 2
end

The response time [Rt_s] and the average response time [$ARTs$] of each server are calculated as given in equations [1] and [2], respectively.

$$Rt_s = \sum_{i=1}^{n} X_i \quad (1)$$

Where

$$Xi = (N_{s1} * R_{s1}) + (N_{s2} * R_{s2}) + \cdots + (N_{sn} * R_{sn})$$
$$ART_S = \sum_{i=1}^{n} \left(\frac{X_i}{T_{nr}}\right) \quad (2)$$

Here, 'X_i' represents the response time of each server serving 'n' number of requests. 'T_{nr}' represents the total number of requests. 'N_{s1}' is the number of requests served by server 1, and 'R_{s1}' is the response time of server 1 serving the required content. Similarly, 'N_{s2}' is the number of requests served by server 2, and 'R_{s2}' is the response time of server 2. The requests served by the nth server are represented by Nsn, and the response time of the nth server is represented by R_{sn}.

4. EXPERIMENT AND RESULTS

The experiment setup consists of a controller, an OpenFlow switch, a pool of web servers, and various client machines. The experimental testbed is as shown in Fig. 4, the experiment is carried out in data centre network where a number of clients and various web servers, such as Apache 2, Ngnix, and SimpleHTTPServer, are connected to the RYU controller via a real-time Zodiac-fx switch. The load balancer module is placed within the RYU controller.

Initially, the experiment was carried out to perform load-balancing based on various techniques such as round robin, random, LBSSRT, SD-WLB, and SDLB-MCDM. The single-objective optimization and analysis approach is no longer widely used due to the increasing complexity and multiplicity of the load-balancing problem. Due to the fact that perfect load-balancing is driven by multiple dimensions, a good decision-maker may look into various parameters, such as non-economical or economical, that can be compromised in certain situations. The experiment is formulated using the multi-criteria decision-making [MCDM] mathematical model to find a suitable solution for the load-balancing problems involving multiple and conflicting objectives. This model works on the basic principle of the weighted sum method [WSM], i.e., the rank of the best load-balancing technique is evaluated based on the WSM of the MCDM technique [20-25].

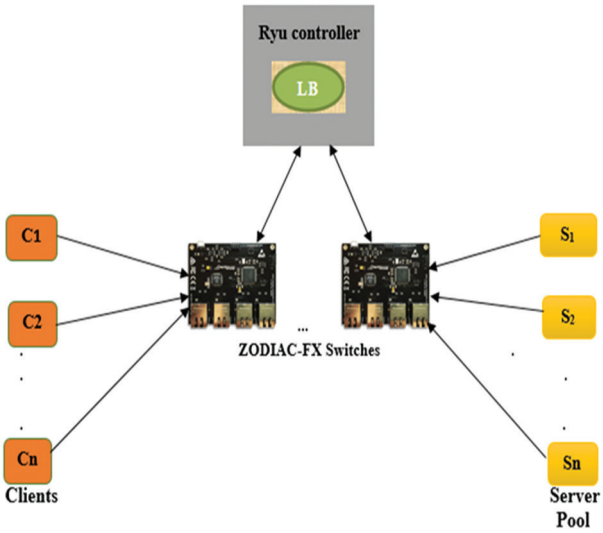

Fig. 4. Experimental setup

This technique takes into account various parameters and values, along with criteria.The criteria column represents the various methods used for evaluation, such as round robin, random, LBBSRT, SD-WLB and the proposed method SDLB-MCDM. The parameters to be considered are outlined in Table 1 below:

Table 1. Parameters used

Criteria	Average Response time	Transfer rate	Time per request	Request per second	Waiting time
Random	0.875 s	2496.68 kbps	3.357ms	297.85/s	4
Round robin	0.888 s	2246.57 kbps	3.223 ms	290.06/s	3
LBBSRT	0.723 s	3445.85 kbps	2.452 ms	312.14/s	2
SD-WLB	0.678 s	3876.45 kbps	2.126 ms	366.31/s	2
SDLB-MCOM	0.065 s	6687.23 kbps	0.157 ms	543.67/s	1

When we look at the measuring units of each of these parameters, they are different. In order to resolve this issue, the weighted sum method is used. The steps of experimentation using the WSM-MCDM technique are as follows:

Step 1: Construct a conversion scale that ranges from low to excellent as shown below in Table 2.

Table 2. Conversion scale

Low	1
Below average	2
Average	3
Good	4
Excellent	5

Step 2: To obtain the decision matrix, assume that the decision maker has determined the importance (or measure of performance) of alternative Ai based on criterion Cj (for i = 1,2,3,..., M and j = 1,2,3,...,N), and W_i represents the weights assigned, as shown in Table 3.

Table 3. Decision matrix

Criteria

Alternative	C_1	C_2	C_3	...	C_n
	W_1	W_2	W_3	...	W_n
a_1	a_{11}	a_{12}	a_{13}	...	a_{1n}
a_2	a_{21}	a_{22}	a_{23}	...	a_{2n}
a_3	a_{31}	a_{32}	a_{33}	...	a_{3n}
⋮	⋮	⋮	⋮	⋮	⋮
a_m	a_{m1}	a_{m2}	a_{m3}	...	a_{mn}

Here, a decision-makers primary objective is to select the best alternative or rank all possible alternatives. After considering all of the decision criteria, Pi (for i = 1, 2, 3 …M) represents the final preference for alternative Ai. We can calculate the preference Pi for alternative A_i (i = 1, 2, 3…M) using the formula below [26-29].

$$Pi = \sum_{j=1}^{N} A_{ij} Wj \quad (3)$$

(for i=1, 2,3… M)

Step 3: The next step is to categorize the parameters as beneficial or costly. The beneficial parameters are the ones whose higher values are preferred, and the costly parameters are the ones whose lower values are preferred [30]. Accordingly, the table is categorized by parameters as shown below in Table 4.

Table 4. Parameter categorizing table

	Costly	**Beneficial**	**Beneficial**	**Beneficial**	**Costly**
Criteria	Average Response time	Transfer rate	Time per request	Request per second	Waiting time
Random	0.875 s	2496.68 kbps	3.357 ms	297.85 /s	4
Round robin	0888s	2246.57 kbps	3.223 ms	290.06 /s	3
LBBSRT	0.723 s	3445.85 kbps	2.452 ms	312.14 /s	2
SD-WLB	0.678 s	3876.45 kbps	2.126 ms	366.31 /s	2
SDLB-MCDM	0.065 s	6687.23 kbps	0.157 ms	543.67 /s	1

Step 4: Further, the table needs normalization. In order to normalize the following expressions are used.

$$Costly = \max(a_{ij}) / a_{ijj} \quad (4)$$

$$Costly = \max(a_{ij}) / a_{ijj} \quad (5)$$

Step 5: Applying the expression in equations (4) and (5), the table is normalized as shown below in Table 5 below.

Table 5. Normalized values

	Costly	**Beneficial**	**Beneficial**	**Beneficial**	**Costly**
Criteria	Average Response time	Transfer rate	Time per request	Request per second	Waiting time
Random	0.875	2496.68	3.357	297.85	4
Round robin	0.888	2246.57	3.223	290.06	3
LI313SRT	0.723	3445.85	2 452	312.14	2
SD-WLB	0.678	387645	2.126	366.31	2
SDLB-MCDM	0.065	6687.23	0.157	543.67	1

Step 6: The next step is to obtain a weighted normalized matrix by adding weights to all these criteria. Here the proposed technique is evaluated for both equal and unequal weights for all the criteria, as shown below in Table 6 below.

Weightage	**20%**	**20%**	**20%**	**20%**	**20%**
Normalization	Costly	Beneficial	Beneficial	Beneficial	Costly
Criteria	Average Response time	Transfer rate	Time per request	Request per second	Waiting time
Random	0.875	2496.68	3.357	297.85	4
Round robin	0.888	2246.57	3.223	290.06	3
LBBSRT	0.723	3445.85	2.452	312.14	2
SD-WLB	0.678	3876.45	2.126	366.31	2
SOLB-MCDM	0.065	6687.23	0.157	543.67	1

Step 7: The next step is to obtain the performance matrix to select the best among the given alternatives, as shown in Table 7 below.

Table 7. Performance Matrix for equal weights

Weightage	**20%**	**20%**	**20%**	**20%**	**20%**
Normalization	Costly	Beneficial	Beneficial	Beneficial	Costly
Criteria	Average Response time	Transfer rate	Time per request	Request per second	Waiting time
Random	0.014857	0.05	0.07467	0.2	0.10957
Round robin	0.01464	0.066667	0.06719	0.192017	0.106704
LBBSRT	0.017981	0.1	0.103058	0.146083	0.114827
SD-WLB	0.019174	0.1	0.115936	0.126661	0.134755
SDLB-MCDM	0.2	0.2	0.2	0.009354	0.2

Step 8: Obtain the performance ranking matrix as shown in Table 8.

From the final performance table, it is seen that the proposed method SDLB-MCDM stands out best among all the other techniques such as random, round robin, LBBSRT, and SD-WLB. The results are discussed in the next section.

Table 8. Ranking Matrix for equal weights

Weightage	20%	20%	20%	20%	20%	
Normalization	Costly	Beneficial	Beneficial	Beneficial	Costly	RANK
Criteria	Average Response time	Transfer rate	Time per request	Request per second	Waiting time	
Random			0.449097			4
Round robin			0.447217			5
LBBSRT			0.481948			3
SD-WLB			0.496525			2
SDLB-MCDM			0.809354			1

Step 9: The performance matrix for unequal weights are shown in Table 9 below.

Table 9. Performance Matrix for unequal weights

Weightage	25%	20%	10%	20%	25%
Normalization	Costly	Beneficial	Beneficial	Beneficial	Costly
Criteria	Average Response time	Transfer rate	Time per renuest	Request per second	Waiting time
Random	0.018571	0.0625	0.07467	0.1	0.10957
Round robin	0.0183	0.083333	0.06719	0.096008	0.106704
LBBSRT	0.022476	0.125	0.103058	0.073041	0.114827
SD-WLB	0.023968	0.125	0.115936	0.06333	0.134755
SDLB-MCDM	0.25	0.25	0.2	0.004677	0.2

Step 10: Obtain the final performance ranking matrix as shown below in Table 10

Table 10. Performance ranking

Weightage	20%	20%	20%	20%	20%	
Normalization	Costly	Beneficial	Beneficial	Beneficial	Costly	RANK
Criteria	Average Response time	Transfer rate	Time per request	Request per second	Waiting time	
Random			0.365312			4
Round robin			0.371536			5
LBBSRT			0.438402			3
SD-WLB			0.462988			2
SDLB-MCDM			0.904677			1

From the final performance table, it is seen that the proposed method SDLB-MCDM stands out best among all the other techniques such as random, round robin, LBBSRT, and SD-WLB. The results are discussed in the next section.

5. RESULTS

An analysis of the results obtained using the real experimental setup implemented using an OpenFlow environment is presented in this section. The experimental setup included a Ryu controller and Zodiac-FX switch, as well as web servers such as Apache 2, Ngnix, and SimpleHTTPServer, and a set of client machines installed with Ubuntu 20.0. The steps are configured as follows:

The hosts are configured to use services such as image data, video data, audio data, and text data. Apache Bench is used to generate the traffic. Here different metrics such as average response time, transfer rate, time-per-request, requests-per-second, and waiting time are considered for the performance evaluation of SDLB-MCDM. The comparison of SDLB-MCDM with different techniques like random, round robin, LBBSRT, and SD-WLB is considered. In this experiment, the SDLB-MCDM module runs on a Ryu controller that runs three different algorithms: the request classification module, the server monitoring module, and the optimized dynamic load-balancing module using content-based routing. Averaging ten experiments yielded the reported results. The proposed mechanism, SDLB-MCDM, shows better performance in comparison with other techniques, and this mechanism can be used in many data center environments.

The graphs shown in Fig. 5 illustrate the average response time of different schemes like round robin, random, LBBSRT, SD-WLB, and SDLB-MCDM. The proposed scheme (SDLB-MCDM) shows better performance in comparison with other techniques.

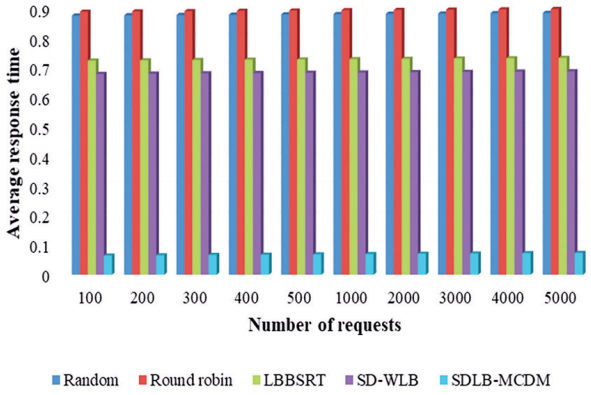

Fig. 5. Average Response Time

Fig. 6 depicts the transfer rate, which indicates that the proposed technique performs better at transferring more data in comparison with other techniques.

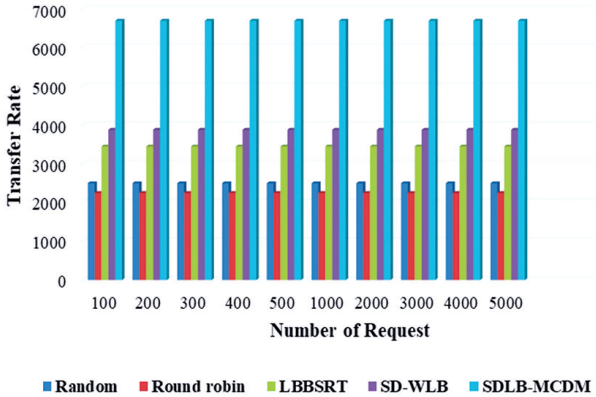

Fig. 6. Transfer Rate

The time-per-request is depicted in Fig. 7, which indicates the proposed [SDLB-MCDM] technique takes very little time to serve the request.

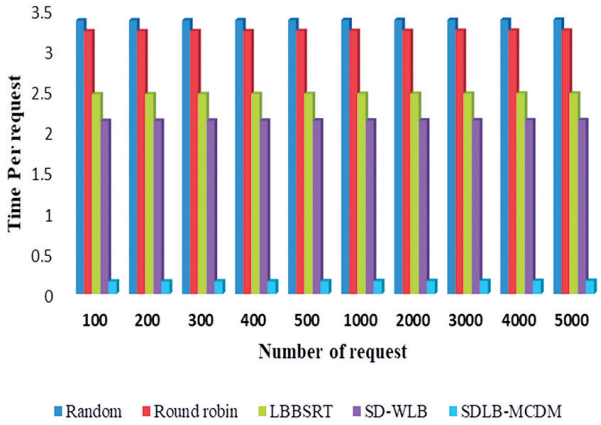

Fig. 7. Time per request

The results in Fig. 8 clearly indicate that the proposed method [SDLB-MCDM] serves a greater number of requests per second in comparison with other techniques.

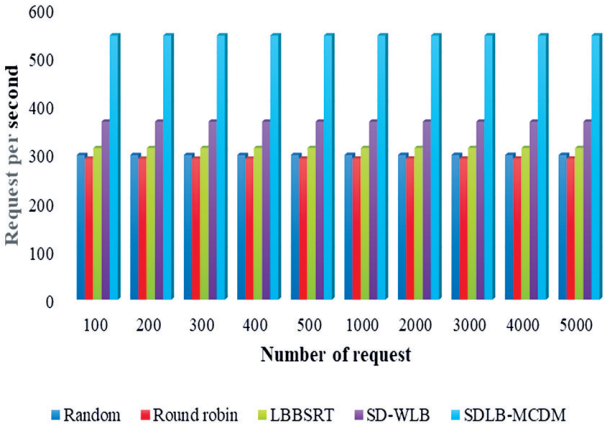

Fig. 8. Request per second

It is very important for any method to have a short waiting period that indicates a very small number of requests are waiting in the queue.

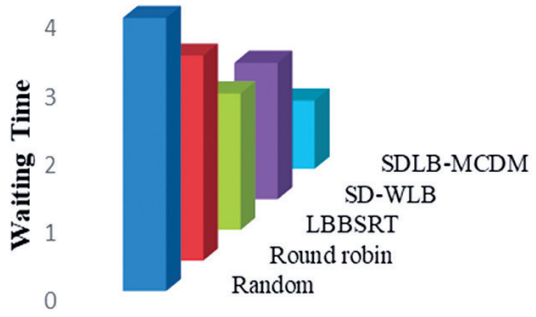

Fig. 9. Waiting Time

The results shown in Fig. 9 clearly indicate that the proposed method has a very low number of outstanding requests.

6. CONCLUSION

This proposed research work presents an optimized load-balancing in a software-defined network based on a multi-criteria decision-making technique [SDLB-MCDM]. The SDLB-MCDM method proposes three algorithms based on response time and content mapping to choose the best server among the pool of servers. In order to appreciate the efficacy and feasibility of the proposed technique, different parameters are considered for decision-making rather than a single parameter, which makes it more efficient in comparison with other techniques. The proposed technique makes use of WSM and the MCDM method to determine the load-balancing technique. The experimental results show a 58% improvement in the performance of the proposed method when equal weights are assigned. The results show a 50% improvement in the performance of SDLB-MCDM when unequal weights are assigned. The performance results under both equal and unequal weights show better performance in comparison with round robin, random, LBBSRT, and SD-WLB techniques.

The proposed SDLB-MCDM method can be adopted in data centre networks where load-balancing among many virtual machines is a major challenge. The future scope of this research work can be tested in a heterogeneous environment with different servers.

7. REFERENCES

[1] C. H. Benet, "Traffic Management in Software-Defined Data Center Networks", Faculty of Health, Science and Technology, Department of Mathematics and Computer Science, Karlstad University, PhD Thesis ,2021.

[2] M. M. Tajiki, B. Akbari, N. Mokari, "Optimal Qos-aware network reconfiguration in software defined cloud data centers", Computer. Networks, Vol. 120, 2021 pp. 71-86.

[3] K. Benzekki, A. El Fergougui, A. Elbelrhiti Elalaoui, "Software-defined networking (SDN): a survey", Secure Communication Networks, Vol. 9, No. 18, 2016, pp. 5803-5833.

[4] L. Li, Q. Xu, "'Load balancing researches in SDN: A survey", Proceedings of the 7th IEEE International Conference on Electronics Information and Emergency Communication, Shenzhen, China, 21-23 July 2017, pp. 403-408.

[5] M. Alotaibi, A. Nayak, "Linking handover delay to load balancing in SDN-based heterogeneous networks", Computer. Communication., Vol. 173, 2021, pp. 170-182.

[6] S. Prabakaran, R. Ramar, Software-defined network: Load balancing algorithm design and analysis", The International Arab Journal of Information Technology, Vol. 18, No. 3, 2021, pp. 312-318.

[7] A. Moravejosharieh, S. Ahmad, K. Ahmadi, "Shortening the response time in software-defined networking: A sensitivity analysis approach", Proceedings of the IEEE 7th International Conference on Communications and Electronics, Hue City, Vietnam 18-20 July 2018, pp. 90-95.

[8] H. Sufiev, Y. Haddad, L. Barenboim, J. Soler, "Dynamic SDN controller load balancing", Future Internet, Vol. 11, No. 3, 2019, pp. 1-21.

[9] H. Mokhtar, X. Di, Y. Zhou, A. Hassan, Z. Ma, S. Musa, "Multiple-level threshold load balancing in distributed SDN controllers", Computer Networks, Vol. 198, 2021, pp. 108-369.

[10] T. Wang, F. Liu, J. Guo and H. Xu, "Dynamic SDN controller assignment in data center networks: Stable matching with transfers", Proceedings of the 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, USA, 2016, pp. 1-9.

[11] H. Zhong, Y. Fang, J. Cui, "Reprint of 'LBBSRT: An efficient SDN load balancing scheme based on server response time", Future Generation Computer Systems., Vol. 80, 2018, pp. 409-416.

[12] K. Soleimanzadeh, M. Ahmadi, M. Nassiri, "SD-WLB: An SDN-aided mechanism for web load balancing based on server statistics", ETRI Journal, Vol. 41, No. 2, 2019, pp. 197-206.

[13] A. A. Abdelaziz, E. Ahmed, A. T. Fong, A. Gani, M. Imran, "SDN-Based load balancing service for cloud servers", IEEE Communication Magazine, Vol. 56, No. 8, 2018, pp. 106-111.

[14] J. V. O. Farias, E. F. Coutinho, C. I. M. Bezerra, "Applying load balancing algorithms for multiple access management on software-defined networking servers", Proceedings of the 10th Euro-American Conference on Telematics and Information Systems, New York, NY, USA, November 2020 ,pp. 1-5.

[15] A. S. Linn, S. H. Win, S. T. Win, "Server Load Balancing in Software Defined Networking", National Journal of Parallel and Soft Computing, Vol. 1, No. 1, 2019, pp. 261-265.

[16] S. Kaur, K. Kumar, J. Singh, N. S. Ghumman, "Round-robin based load balancing in Software Defined Networking," Proceedings of the 2nd International Conference on Computing for Sustainable Global Development, New Delhi, India, 2015, pp. 2136-2139.

[17] A. K. Arahunashi, G. G. Vaidya, K. V. Reddy, "Implementation of Server Load Balancing Techniques Using Software-Defined Networking", Proceedings of the 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions, Bengaluru, India, 2018, pp. 87-90.

[18] M. L. Chiang, H. S. Cheng, H. Y. Liu, C. Y. Chiang, "SDN-based server clusters with dynamic load balancing and performance improvement", Cluster Computing, Vol. 24, No. 1, 2021, pp. 537-558.

[19] G. S. Begam, M. Sangeetha, N. R. Shanker, "Load Balancing in DCN Servers through SDN Machine Learning Algorithm", Arabian Journal for Science and Engineering, Vol. 47, No. 2, 2022, pp. 1423-1434.

[20] M. Cinelli, M. Kadziński, G. Miebs, M. Gonzalez, R. Słowiński, "Recommending multiple criteria decision analysis methods with a new taxonomy-based decision support system", European Journal of Operational Research, Vol. 302, No. 2, 2022, pp. 633-651.

[21] B. Sarkar, "Fuzzy decision making and its applications in cotton fibre grading", Soft Computing in Textile Engineering, 2011, pp. 353-383..

[22] S. G. Ozcan, M. Sayit, "Improving the QoE of DASH over SDN: A MCDM Method with an Intelligent Approach", Proceedings of the 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops, Paris, France, 2019, pp. 100-105.

[23] S. Chakraborty, T. K. Jana, S. Paul, "On the application of multi criteria decision making technique for multi-response optimization of metal cutting process", Intelligent Decision Technologies, Vol. 13, No. 1, 2019, pp. 101-115.

[24] J. Gyani, A. Ahmed, M. A. Haq, "MCDM and Various Prioritization Methods in AHP for CSS: A Comprehensive Review", IEEE Access, Vol. 10, 2022, pp. 33492-33511.

[25] E. Triantaphyllou, A. Sánchez, "A sensitivity analysis approach for some deterministic multi-criteria

decision-making methods", Decision Sciences., Vol. 28, No. 1, 1997, pp. 151-194.

[26] P. C. Fishburn, "Letter to the Editor—Additive Utilities with Incomplete Product Sets: Application to Priorities and Assignments", Operations Research, Vol. 15, No. 3, 1967, pp. 537-542.

[27] S. Das, A. Ghosh, "A Fuzzy Multi-Criteria Decision-Making Approach for Grading of Raw Jute", Journal of Natural Fibers, Vol. 18, No. 5, 2021, pp. 685-693.

[28] A. Hussain, J. Chun, M. Khan, "A novel multicriteria decision making (MCDM) approach for precise decision making under a fuzzy environment", Soft Computing, Vol. 25, No. 7, 2021, pp. 5645-5661.

[29] J. Ali and B. H. Roh, "A Novel Scheme for Controller Selection in Software-Defined Internet-of-Things (SD-IoT)", Sensors, Vol. 22, No. 9, 2022, p. 3591.

[30] J. Ali, B. H. Roh, S. Lee, "QoS improvement with an optimum controller selection for software-defined networks", PLoS ONE, Vol. 14, No. 5, 2019.