

Traffic-aware adaptive server load balancing for software defined networks

C. Fancy¹, M. Pushpalatha²

¹Department of Information Technology, SRM Institute of Science and Technology, Kattankulathur, India

²Department of Computer Science and Engineering, SRM Institute of Science and Technology, Kattankulathur, India

Article Info

Article history:

Received Mar 25, 2020

Revised Sep 15, 2020

Accepted Oct 9, 2020

Keywords:

Data center networks

Floodlight controller

SDN

Server load balancing

ABSTRACT

Servers in data center networks handle heterogeneous bulk loads. Load balancing, therefore, plays an important role in optimizing network bandwidth and minimizing response time. A complete knowledge of the current network status is needed to provide a stable load in the network. The process of network status catalog in a traditional network needs additional processing which increases complexity, whereas, in software defined networking, the control plane monitors the overall working of the network continuously. Hence it is decided to propose an efficient load balancing algorithm that adapts SDN. This paper proposes an efficient algorithm TA-ASLB-traffic-aware adaptive server load balancing to balance the flows to the servers in a data center network. It works based on two parameters, residual bandwidth, and server capacity. It detects the elephant flows and forwards them towards the optimal server where it can be processed quickly. It has been tested with the Mininet simulator and gave considerably better results compared to the existing server load balancing algorithms in the floodlight controller. After experimentation and analysis, it is understood that the method provides comparatively better results than the existing load balancing algorithms.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

C. Fancy

Department of Information Technology

SRM Institute of Science and Technology

Potheri, Kancheepuram, Tamilnadu, 603203-India

Email: fancyc@srmist.edu.in

1. INTRODUCTION

Data center networks (DCN) faces different types of workload especially in 3 aspects like business, consumer, and entertainment. DCNs usually support multiple paths between any end hosts. DCNs should handle both the categories of flows-elephant and mice flows [1]. The main aim of load balancing is to improve the throughput avoiding processing delays in optimal path selection to balance the load. The various factors influencing load balancing in DCN include the energy of nodes, residual bandwidth, scalability of the network, types of flows. With the increasing number of devices in the network, managing the network traffic is becoming very difficult [2]. Avoiding buffer overflows is another major concern. Also, the most important node may be chosen repeatedly, which leads to its quicker degradation.

Balancing the incoming load among the servers in data center networks can be seen in various aspects like link load balancing, server load balancing. Few of the delay-sensitive applications include real-time video and voice. Delay sensitive applications need to maintain the QoS parameters also. The load balancing algorithm must also improve the metrics such as scalability, robustness, energy efficiency, etc.

The link utilization varies in the DCN, also the congestion increases in the bottom layers. The type of flow is another major component that varies dynamically [3, 4]. There are two possibilities of flow, i.e. mice flow and elephant flow. Usually, there are a lot of mice flows e.g. hello messages, meta-data requests, etc. The elephant flow takes huge resources but it happens not as frequently as the mice flows. A statistical report says that the rate of video traffic in a DCN is increasing in a heavily. An efficient way of controlling network devices is given in paper [5]. The network is partitioned into various sub-domains. Each sub-domain has an SDN enabled border router that helps in routing to other domains. Here the network management is adaptive. Because, if the domain size is large, inter-domain management functions were very less compared to a smaller domain size. With this scenario, effectiveness is achieved in the scalability of bandwidth allocation, route determination based on current load, and recovery from failures.

The challenges of wireless network virtualization [6] are highly varying traffic, multidimensional heterogeneity. Besides these drawbacks, efficiency can be achieved in QoS provisioning, resource sharing, and verification of new techniques before it can be widely deployed. The resource allocation in inter virtual network must be dynamic. This is due to the changes in service requirements and new requests for the virtual network.

Mujiono [7] explains the issues in creating a load-balanced aggregation tree for a wireless sensor network. He analyzed three related problems and proposed an optimal solution. They have prepared a traffic matrix that helped in analysing the parameters of the topology. The important parameter discussed in their model is the transmission success ratio for every link in the network. This ratio defines the successful delivery of the packet. Another two metrics which are used in the proposed model include potential load and actual load. The techniques used to solve the identified problem are linear relaxation and random rounding. Upon analysis, it is found that the proposed idea increases the network lifetime.

The paper [8] proposed a myopic algorithm that uses the cost of each link to decide the path. When a flow arrives, it sends through the path having the minimum cost. They also tested with varying versions of myopic algorithm and obtained better network efficiency. Benlalia [9] makes use of the greedy round-robin algorithm.

The challenges of vehicular ad hoc networks (VANETs) [10] such as mobility and scalability affect the performance of routing protocol. With the help of SDN, a solution for these issues was found. The proposed system includes the SDN based connectivity aware geographical routing protocol. In the simulation, the proposed model provided an optimized routing path while the following parameters are evaluated; i) Determining the traffic density, ii) Tracking the distance, and iii) Estimating the link lifetime.

Thus, load balancing is an important task in any data center. There will be multiple servers in every DCN, computing continuously for providing seamless connectivity for applications such as WhatsApp, Facebook, web search, live programs, etc. Hence it is needed to focus on the server load balancing algorithms.

This paper aims to focus on the server load balancing in a heterogeneous server environment. It also focuses on how the controller can adaptively choose the algorithms. This adaptivity is included to avoid additional complexity to the load balancing module. The proposed paper is designed according to the following protocol. In section 2, we explain the important similar research aspects, section 3 explains the proposed model, section and section 4 analyses the experimentation done by the authors.

2. RELATED WORKS

A lot of researchers have contributed to the server load balancing research. The selection of the best routing path is always an expensive work in (DCN) data center networks [11-13]. The major parameters to be considered while selecting an optimal path are storage resource, bandwidth consumption, and delay in packet transmission. To improve the cloud gaming experience, the hierarchy process could be used. The process assured better results, once the routing path for a game session is chosen based on the game type. DCNs are multilayered topology. The collaboration of SDN based multipath TCP and segment routing [14, 15] improves the efficient usage of memory resources. Virtualization supports the existence of several networks in one substrate network. This work could be done by a centralized controller. By this, the resources which are utilized can be reduced and the service to several clients can be increased.

This method provides an online approach to serve the cloud client requests. The paper [16] proposed a myopic algorithm that uses the cost of each link to decide the path. When a flow arrives, it sends through the path having the minimum cost. They also tested with varying versions of myopic algorithm and obtained better network efficiency. The authors in [17-19] make use of the greedy round-robin algorithm. The proposed load balancing is based on the flow size. The algorithm is used only for the long flows, hence reduce the controller's workload. The performance of a switch can be analysed with the help of queuing models [20, 21]. The challenges of wireless network virtualization [22] are highly varying traffic,

multidimensional heterogeneity. Besides these drawbacks, efficiency can be achieved in QoS provisioning, resource sharing, and verification of new techniques before it can be widely deployed. The resource allocation in inter virtual network must be dynamic. This is due to the changes in service requirements and new requests for the virtual networks.

In the floodlight controller [23, 24], there are few traditional load balancing algorithms available. They are RR approach, weight induced RR method, and statistical method. In the weighted round-robin method, the possible paths to reach the destination are found. Then based on the weights for each of these paths, the one having the highest weight will be chosen. The statistical method is based on the remaining bandwidth used. If the remaining bandwidth of a node is higher, then it will be chosen for the transmission.

The main aim of load balancing [25-28] is to improve the throughput avoiding processing delays in optimal path selection to balance the load. The various factors influencing load balancing in DCN include the energy of nodes, residual bandwidth, scalability of the network, types of flows. With the increasing number of devices in the network, managing the network traffic is becoming very difficult. In this paper, we have devised a traffic-aware server load balancing that works adaptively. It checks the incoming flow type and categorizes it as short flow and long flow. Differentiating short flow and long flows is a separate research topic but we have taken few references for differentiating the flows that helped in focusing on our research.

The proposed load balancing is working based on the flow size. The algorithm is used only for the long flows. The first step is to distinguish the incoming flows into short and long flows, then the adaptive algorithm. Hence reducing the controller's computational overhead.

3. RESEARCH METHOD

The proposed method consists of three phases traffic monitoring phase, Adaptive decision phase, server selection phase. In the traffic monitoring phase, the controller monitors the entire topology. The proposed method takes three important details which are the size of each flow, residual bandwidth in all the links to the servers, and computational capacity of all the available servers. The second phase is the Adaptive decision phase. If the flow size is less than a threshold of 10 kB, then it is decided not to use any complex load balancing algorithm, hence round-robin load balancing is recommended. If the flow size exceeds the threshold value, then the TA-ASLB method will be followed. The final phase is the server selection phase with the help of the TA-ASLB method.

The important parameters needed for server load balancing include residual bandwidth and server capacity. This is identified with the help of a literature survey among various papers. It aims at finding the nodes having a higher residual bandwidth. The algorithm is explained in Figure 1.

| Algorithm 1: TA-ASLB algorithm |
|--|
| <p>Traffic monitoring phase: Data: Flow in the Switches $F = \{f_1, f_2, \dots, f_n\}$ Data: Capacity of all possible links to reach server $L = \{L_1, L_2, \dots, L_n\}$ Data: Capacity of the servers in the server pool $S = \{S_1, S_2, \dots, S_n\}$</p> <p>Adaptive decision phase: begin while packet(i) generated by clients do for each $f_i \in F$ do if Flow size (f_i) > Threshold T_s follow Traffic aware load balancing else: follow Round robin load balancing</p> <p>Server selection phase: $[\alpha] = \max(L_i)$ find the servers S_i which are present in this best links $\beta_i = \max(S_i)$ Optimal Server = β_i</p> |

Figure 1. Traffic aware-adaptive server load balancing [TA-ASLB] method

The TAALB method works with the combination of two parameters, link capacity towards the server and server characteristics. It aims at finding the nodes having a higher residual bandwidth. The

residual bandwidth of a link is calculated based on the load that is currently handled by the link. The links which are carrying a load at any time will be utilizing bandwidths, so they will be having lesser residual bandwidth.

Then, among them, the nodes will be ordered in the descending order of the server's weight. It means, a node having higher residual bandwidth and higher weight then. Thus, it shares the load among nodes based on two parameters, bandwidth, and weight. Thus, the load will be optimally shared among the nodes.

4. RESULTS AND DISCUSSIONS

The algorithm is implemented with the help of a Mininet simulator. Data center networks are usually maintained by following the fat-tree topology structure. Hence for this testing, a fat-tree topology of depth 3 is created to depict the data center network. The network has eight client nodes termed from h1 to h8 and seven distribution layer switches s1 to s7. The experimentation is done in virtual having Ubuntu 14.10 operating system. The network is assumed to have heterogeneous servers. The topology is depicted in the below diagram mentioned in Figure 2.

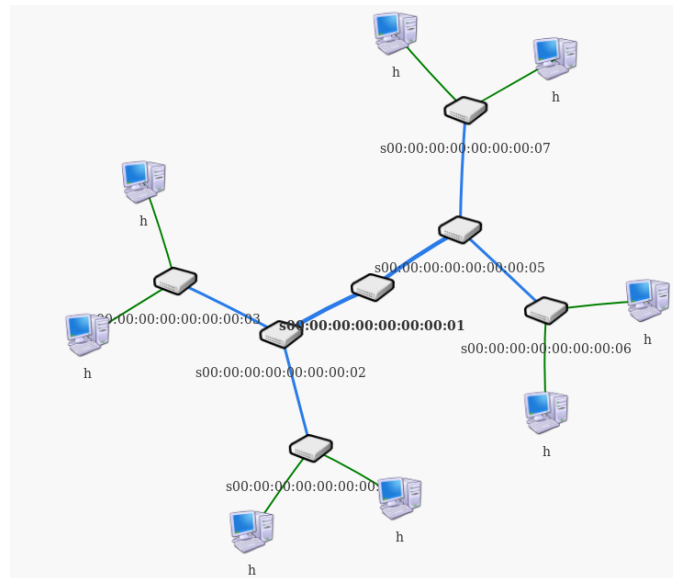


Figure 2. DCN sample topology in Mininet

Here the assumption is host 1 and host 8 are clients. Hosts 2, 3, 4, 5, 6, 7 is defined as servers. Servers are assumed to have different processing capabilities such as 20 core, 16 core, 8 core, and dual-core. Numerous trials are made on a fat-tree topology, using the existing load balancing methods in floodlight controller and the proposed algorithm traffic aware-adaptive server load balancing [TA-ASLB] method. After the various experimentation, the conclusion is derived.

The average waiting time in the queue of a server is defined as the formula in (1).

$$w_t = \frac{r}{2u(1-r)} \quad (1)$$

Here, the variable w_t refers to the waiting time of flow in the queue, r refers to the utilization rate, u refers to the utilization rate.

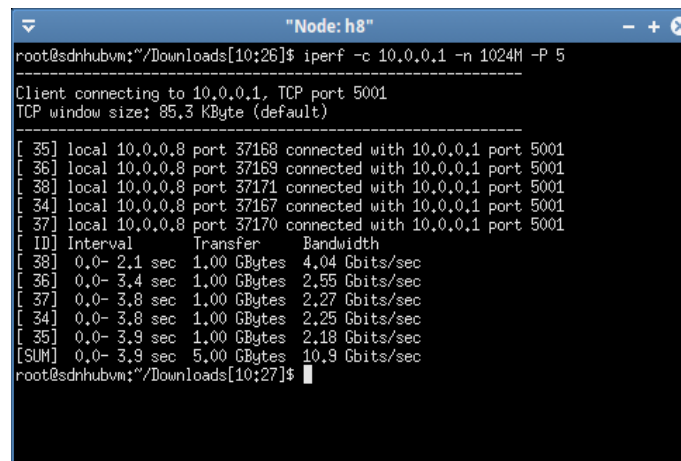
The number of entries in the queue can be defined by the following (2).

$$n_q = \frac{1}{2} \left(\frac{r^2}{1-r} \right) \quad (2)$$

In the above equation, n_q refers to the average count of flows present in the queue, r refers to the rate at which the flow is been utilized.

There is another important part of the algorithm, which is the flow classification. For this experiment, there is a constant threshold defined for the classification of incoming flows. A predefined value of 10 Mbps is been set. If the flow comes with a size above this value, should be treated by the proposed algorithm and they are called as the elephant flows. The incoming flows below this value will be termed as mice flows and they can be sent to servers in round-robin fashion for processing. Round Robin method aims at allotting flow requests to each of the servers one by one.

The topology is tried under various working conditions such as TCP packets transmission, UDP packets transmission. With that, it is possible to measure the parameters such as throughput, packet loss, jitter, response time (latency), etc. Figures 3 and 4 are working examples for the UDP and TCP packet transmissions done. Thus, the above figure tells clearly that the Mininet emulator provides TCP packets and its features. It includes the window size, the port number in which it is listening, the throughput value, and the time series.

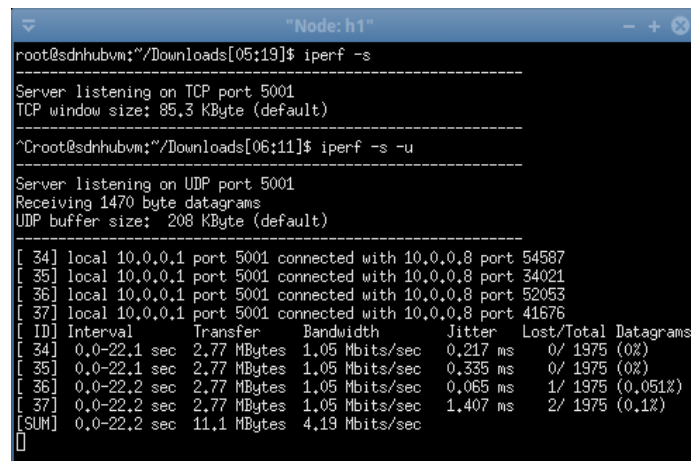


```

root@sdnhubvm:~/Downloads[10:26]$ iperf -c 10.0.0.1 -n 1024M -P 5
-----
Client connecting to 10.0.0.1, TCP port 5001
TCP window size: 85,3 KByte (default)
-----
[ 35] local 10.0.0.8 port 37168 connected with 10.0.0.1 port 5001
[ 36] local 10.0.0.8 port 37169 connected with 10.0.0.1 port 5001
[ 38] local 10.0.0.8 port 37171 connected with 10.0.0.1 port 5001
[ 34] local 10.0.0.8 port 37167 connected with 10.0.0.1 port 5001
[ 37] local 10.0.0.8 port 37170 connected with 10.0.0.1 port 5001
[ ID] Interval      Transfer      Bandwidth
[ 38] 0.0- 2.1 sec  1.00 GBytes  4.04 Gbits/sec
[ 36] 0.0- 3.4 sec  1.00 GBytes  2.55 Gbits/sec
[ 37] 0.0- 3.8 sec  1.00 GBytes  2.27 Gbits/sec
[ 34] 0.0- 3.8 sec  1.00 GBytes  2.25 Gbits/sec
[ 35] 0.0- 3.9 sec  1.00 GBytes  2.18 Gbits/sec
[SUM] 0.0- 3.9 sec  5.00 GBytes  10.9 Gbits/sec
root@sdnhubvm:~/Downloads[10:27]$

```

Figure 3. Sample performance evaluation after generating TCP packets



```

root@sdnhubvm:~/Downloads[05:19]$ iperf -s
-----
Server listening on TCP port 5001
TCP window size: 85,3 KByte (default)
-----
^Croot@sdnhubvm:~/Downloads[06:11]$ iperf -s -u
-----
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
[ 34] local 10.0.0.1 port 5001 connected with 10.0.0.8 port 54587
[ 35] local 10.0.0.1 port 5001 connected with 10.0.0.8 port 34021
[ 36] local 10.0.0.1 port 5001 connected with 10.0.0.8 port 52053
[ 37] local 10.0.0.1 port 5001 connected with 10.0.0.8 port 41676
[ ID] Interval      Transfer      Bandwidth      Jitter    Lost/Total Datagrams
[ 34] 0.0-22.1 sec  2.77 MBytes  1.05 Mbits/sec  0.217 ms  0/ 1975 (0%)
[ 35] 0.0-22.1 sec  2.77 MBytes  1.05 Mbits/sec  0.335 ms  0/ 1975 (0%)
[ 36] 0.0-22.2 sec  2.77 MBytes  1.05 Mbits/sec  0.065 ms  1/ 1975 (0.051%)
[ 37] 0.0-22.2 sec  2.77 MBytes  1.05 Mbits/sec  1.407 ms  2/ 1975 (0.1%)
[SUM] 0.0-22.2 sec  11.1 MBytes  4.19 Mbits/sec

```

Figure 4. Sample performance evaluation after generating UDP packets

The above figure is a screenshot obtained during our experimentation on the simulated topology. The iperf is an efficient tool to explore the evaluation parameters of the given network. It even depicts the various analyses summary related to the packet transmission.

4.1. Throughput

Throughput refers to the rate of successful data transmitted between a source and a destination. Bandwidth is defined as the amount of possible data that can be transmitted in a link. For this experimentation, traffic is generated between host 1 and host 7. Host 1 is the client and host 8 is defined as a

server in our experiment. The throughput of the above-mentioned algorithms is noted for the given topology. The iperf command is used to obtain the throughput. TCP packets are sent carrying 1024 bytes. The experiment is repeated with a varying number of parallel requests such as 100, 200, 300, 400, and 500. The below is a sample screenshot that depicts our experiments. The results are depicted as a graph that is given below Figure 5.

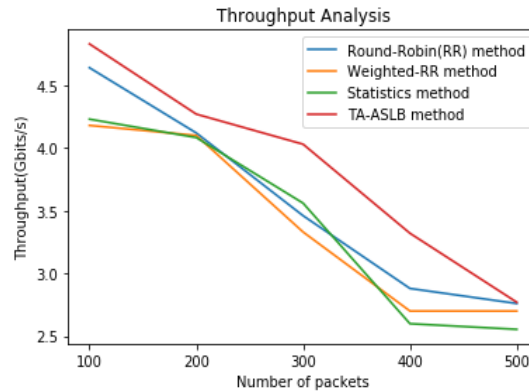


Figure 5. Throughput result analysis

By the results, it is inferred that the proposed method is giving higher throughput compared to the traditional algorithms. When number of packets increase, their throughput decreases. The proposed algorithm outperforms among the other existing load balancing algorithms.

4.1.1. Latency

Latency is defined as the time taken for the data to get transmitted from source to destination and the receiver processing it. It can also be called as round-trip time. The latency of the traditional, as well as algorithms, is noted for the given topology. The experiment is repeated with a varying number of packets such as 100, 200, 300, 400, and 500. This is obtained with the help of the ping command. By the results, it is inferred that the proposed method is showing lesser delay than the existing methods. The results are depicted as a graph that is given as Figure 6.

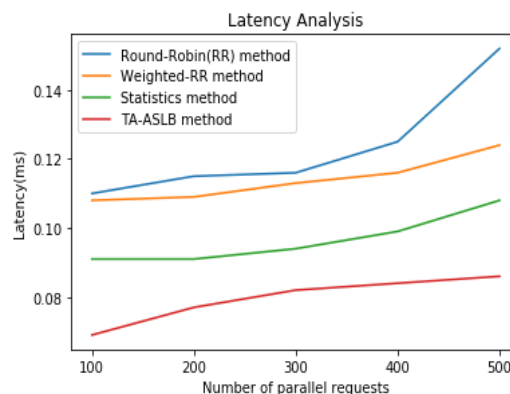


Figure 6. Latency result analysis

5. CONCLUSION AND FUTURE ENHANCEMENTS

The focus of this paper is to perform an efficient server load balancing. The proposed novel algorithm provided an optimal server selection whenever long flows arrive. The experimentation was done to understand the importance of the load regulating process in data centers. The first step was to identify the parameters that influence the delay in data transmission. Next, the type of flows is been considered, because there can be short-lived flows and long-lived flows in the DCNs. Based on the various types of flows i.e.

video traffic, message, or image transfer, the delay in data transmission will vary. It is observed from the distance between the nodes and the influence of a node in the path to destination plays a major role in the efficient load balancing of flows in an SDN environment. Hence the adaptive load balancing scheme is very good in reducing the delay and improving the throughput of the network. Also, the adaptive nature of the algorithm helps the controller to avoid computational complexity in case of short flows.

In the future, this server selection process can be made as an intelligent application working separately on top of the centralized controller. Because to propose the server selection activities quickly and also to overcome any failure, the neural network aspect proposes optimal and suboptimal solutions. This makes the neural network solution as a better option for the future. Another aspect is in the flow classification. In this paper, we have used it as a static way to split the flows. We have planned to include a convolutional neural network, to split the incoming flows into a five-scale meter. This can be used to arrive at a detailing of flows and it could help in the generic priority assignment. Also, dynamicity in threshold setting could be arrived based on the current situation in the network. With this, a precise model that can generate that focus on the current state of the machines like, number of flows currently waiting to be processed, the number of flows handled already, the server capacity, the residual bandwidth in the links connecting to the servers, the queue length, arrival rate, etc. Focus on these could, even more, improvise the efficiency of the current load balancing systems.

REFERENCES

- [1] K. Ramana and M. Ponnaivaikko, "AWSQ: an approximated web server queuing algorithm for heterogeneous webserver cluster," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 3, pp. 2083-2093, Jun. 2019, doi: 10.11591/ijece.v9i3.pp2083-2093.
- [2] K.S. Qaddoum, "Elastic neural network method for load prediction in cloud computing grid," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 2, pp. 1201-1208, 2019, doi: 10.11591 /ijece.v9i2.pp1201-1208.
- [3] T. Emad Ali, A. H. Morad, M. A. Abdala., "Load Balance in Data Center SDN Networks," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 5, pp. 3086-3092, Oct. 2018, doi: 10.11591/ijece.v8i5.pp3084-3091.
- [4] M. Tarahomi and M. Izadi, "A hybrid algorithm to reduce energy consumption management in cloud data centers," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 1, pp. 554-561, Feb. 2019, doi: 10.11591/ijece.v9i1.pp554-561.
- [5] D. Hartanti, "Optimization of smart traffic lights to prevent traffic congestion using fuzzy logic," *TELKOMNIKA Telecommunication Computing Electronics and Control*, vol. 17, no. 1, pp. 320-327, 2019, doi: 10.12928/TELKOMNIKA.v17i1.10129.
- [6] N. Zhang, "Software-Defined Networking Enabled Wireless Network Virtualization: Challenges and Solutions," *IEEE Network*, vol. 31, no. 5, pp. 42-49, 2017, doi: 10.1109/MNET.2017.1600248.
- [7] Mujiono S., "Load balancing clustering on moodle LMS to overcome performance issue of e-learning system," *TELKOMNIKA Telecommunication Computing Electronics and Control*, vol. 17, no. 1, pp. 131-138, Feb. 2019, doi: 10.12928/TELKOMNIKA.v17i1.10284.
- [8] M. Shafiee and J. Ghaderi, "A Simple Congestion-Aware Algorithm for Load Balancing in Datacenter Networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3670-3682, 2017, doi: 10.1109/TNET.2017.2751251.
- [9] Z. Benlalia, "Comparing load balancing algorithms for web application in cloud environment," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 17, no. 2, pp. 1104-1108, 2020, doi: 10.11591/ijeecs.v17.i2.pp1104-1108.
- [10] D.K.N. Venkatramana, S.B. Srikantaiah, J. Moodabidri, "SCGRP: SDN-enabled connectivity-aware geographical routing protocol of VANETs for the urban environment," *IET Journal*, vol. 6, no. 5, pp. 102-111, 2017, doi: <https://doi.org/10.1049/iet-net.2016.0117>.
- [11] C.C. Chuang Ya-Ju Yu, and Ai-Chun Pang, "Flow-Aware Routing and Forwarding for SDN Scalability in Wireless Data Centers," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1676-1691, 2018, doi: 10.1109/TNSM.2018.2865166.
- [12] J. Pang Gaochao Xu, and Xiaodong Fu, "SDN-Based Data Center Networking with Collaboration of Multipath TCP and Segment Routing," *IEEE Access*, vol. 5, pp. 9764-9773, 2017, doi: 10.1109/ACCESS.2017.2700867.
- [13] Q. Qin, et al., "SDN Controller Placement With Delay-Overhead Balancing in Wireless Edge Networks," *IEEE Transactions On Network And Service Management*, vol. 15, no. 4, pp. 1446-1459, 2018, doi: 10.1109/TNSM.2018.2876064.
- [14] S. Zhang, et al., "Online Load Balancing for Distributed Control Plane in Software-Defined Data Center Network," *IEEE Access*, vol. 6, pp. 18184-18191, 2018, doi: 10.1109/ACCESS.2018.2820148.
- [15] M. Amiri, et al., "SDN-Enabled Game-Aware Routing for Cloud Gaming Datacenter Network," *IEEE Access*, vol. 5, pp. 18633-18645, 2017, doi: 10.1109/ACCESS.2017.2752643.
- [16] J. Zhang, et al., "Load Balancing in Data Center Networks: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2324-2352, 2018, doi: 10.1109/COMST.2018.2816042.

- [17] You-Chiun Wang, and Siang-Yu You, "An Efficient Route Management Framework for Load Balance and Overhead Reduction in SDN-Based Data Center Networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1422–1434, 2018, doi: 10.1109/TNSM.2018.2872054.
- [18] K. Alhazmi Abdallah Shami, and Ahmed Refaey. "Optimized Provisioning of SDN-enabled Virtual Networks in Geo-distributed Cloud Computing Datacenters," *Journal of Communications and Networks*, vol. 19, no. 4, pp. 402–415, 2017, doi: 10.1109/JCN.2017.000064.
- [19] P.R. Iyer, et al., "Adaptive real time traffic prediction using deep neural networks," *International Journal of Artificial Intelligence (IJ-AI)*, vol. 8, no. 2, pp. 107-119, 2019, doi: 10.11591/ijai.v8.i2.pp107-119.
- [20] J. Zhang et al., "Load Balancing in Data Center Networks: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2324–2352, 2018, doi: 10.1109/COMST.2018.2816042.
- [21] C. Jittawiriyankoon, "Evaluation of load balancing approaches for Erlang concurrent application in cloud systems," *TELKOMNIKA Telecommunication, Computing, Electronics and Control*, vol. 18, no. 4, pp. 1795-1801, 2020, doi: 10.12928/TELKOMNIKA.v18i4.13150.
- [22] H. Tussyadiah, Ridha M.N., and Danu D.S., "Distributed gateway-based load balancing in software defined network," *TELKOMNIKA Telecommunication, Computing, Electronics and Control*, vol. 18, no. 5, pp. 2352-2361, Oct. 2020, doi:10.12928/TELKOMNIKA.v18i5.14851.
- [23] X. Yu Hong Xu, and Huaxi Gu, "Thor: A Server-level Hybrid Switching Data Center Network with Heterogeneous Topologies," *ACM TUR-C '17*, pp. 1-10, 2017, doi: 10.1145/3063955.3063992.
- [24] D. Sun, et al., "Dynamic Traffic Scheduling and Congestion Control across Data Centers Based on SDN," *Future Internet*, vol. 10, no. 7, 2018, doi: 10.3390/fi10070064.
- [25] Xi Huang, et al., "Dynamic switch-controller association and control devolution for SDN systems," *IEEE International Conference on Communications (ICC)*, 2017, doi: 10.1109/ICC.2017.7997427.
- [26] W. Prakash and Deepalakshmi P., "DServ-LB: Dynamic server load balancing algorithm," *International Journal of Communication Systems*, 2018, doi: 10.1002/dac.3840.
- [27] M. Alla'Anzy and M. Othman, "Load balancing and server consolidation in cloud computing environments: A meta-study," *IEEE Access, Digital Object Identifier*, 2019, doi:10.1109/ACCESS.2019.2944420.
- [28] V. Priya, et al., "Resource scheduling algorithm with load balancing for cloud service provisioning," *Applied soft computing, Elsevier*, vol 76, pp. 416-424, 2019.

BIOGRAPHIES OF AUTHORS



C. Fancy is currently pursuing Doctoral research in the Computer Science Engineering department, SRM Institute of Science and Technology, Kattankulathur. She is also a Teaching faculty in the same university. Her research field is Software Defined Networking. She had accomplished UG and PG courses in the years 2009 and 2011 respectively. She is currently working with other research aspects such as Cryptography, TCP/IP services, and the Internet of Things.



M. Pushpalatha is working in SRM Institute of Science and Technology, Kattankulathur. She holds the Professor designation in Computer Science and Engineering department. She had completed UG and PG courses in the years 1994 and 2001 respectively. She completed Doctoral research in the Computer Science Engineering domain, in the year 2014. Her area of interests include Software Defined Data Center Networks, Middleware services, and Wireless Networks, Internet of Things, 5G and future networks. She has several publications in reputed journals. She had presented papers in various national and international conferences. She is currently guiding research scholars in various domains. She is a professional member of various reputed standard bodies such as ACM etc. She has guided several UG and PG level student projects.