https://doi.org/10.51153/kjcis.v6i2.181

SD-ALB: Software Defined Adaptive load balancing in Data Center Network

Rizwan Fazal[1], Syed Mushhad M. Gilani[2], Muhammad Junaid Khalid[3]

Abstract

Network monitoring has crucial importance in data center networks to analyze the behavior of the underlying network. This analysis is used for working on multiple network parameters and load balancing is one of them. This article proposes an adaptive load-balancing approach to balance the load between the servers while changing its behavior with a change in traffic. Software Defined Networking (SDN) provides the single point of network configuration called SDN controller. This approach is facilitating the easy implementation of adaptive load balancing in Data Center Networks. The proposed approach is an extension to the LBBCLT load balancing approach that uses a dynamic probe generator to probe the servers about the response time and link bandwidth. We incorporate a path selection module in it and the path is selected using the Ant Colony Optimization. The results show that the bandwidth consumption and throughput have been improved and servers are receiving the load according to their capacities.

Keywords: Adaptive load balancing; Software Defined Networking; Ant Colony Optimization; Data Center Networks; Path Selection; Response Time; Bandwidth .

1 Introduction

In the Data Center Networks (DCN), several servers are providing a variety of services over the Internet. The flow pattern of the network traffic inside the DCN is not the same it differs with the time because of the variety in the types of users such as businesses using servers for their purposes like cloud services, consumers using the servers for downloading, or fetching the data on the servers [1]. That is why it is not an easy task to analyze the network traffic of a DCN. The servers inside DCN also communicate with each other for different purposes for example a web server might also access the file server in user requests to download some file or to stream some video. In the DCN the number of users' interactions at a time is very huge the sending the receiving of the data has to be managed by some manager [2].

There are two types of traffic flows inside the DCN mice flow and elephant flow [3]. The mice flow comprises packets of small sizes for example text messages (e.g. WhatsApp), ICMP packets (REQUEST or REPLY), and TCP or UDP packets. The elephant flow is the flow of larger messages

[1]*UIIT, PMAS Arid Agriculture University, Rawalpindi (rizwanfazal101@gmail.com)*
[2]*Department of Computer Science, University of Agriculture Faisalabad: (mushhad@uaf.edu.pk) (Corresponding Author)*
[3]*UIIT, PMAS Arid Agriculture University, Rawalpindi (mjk22071998@gmail.com)*

like image files, audio files, video files, or some other files. The network traffic manager must differentiate between these two traffic flows while managing the traffic.

Due to the huge amount of network traffic in DCN, it faces different kinds of issues like low bandwidth, low throughput, and delay in packet transmission [4]. DCN supports multiple paths to the server and the optimal path to the server has to be chosen to minimize the issues being faced [5]. Usually, DCN uses a fat-tree topology because it increases the number of possible paths to a server. Figure 1 shows the servers in the fat tree topology [6].

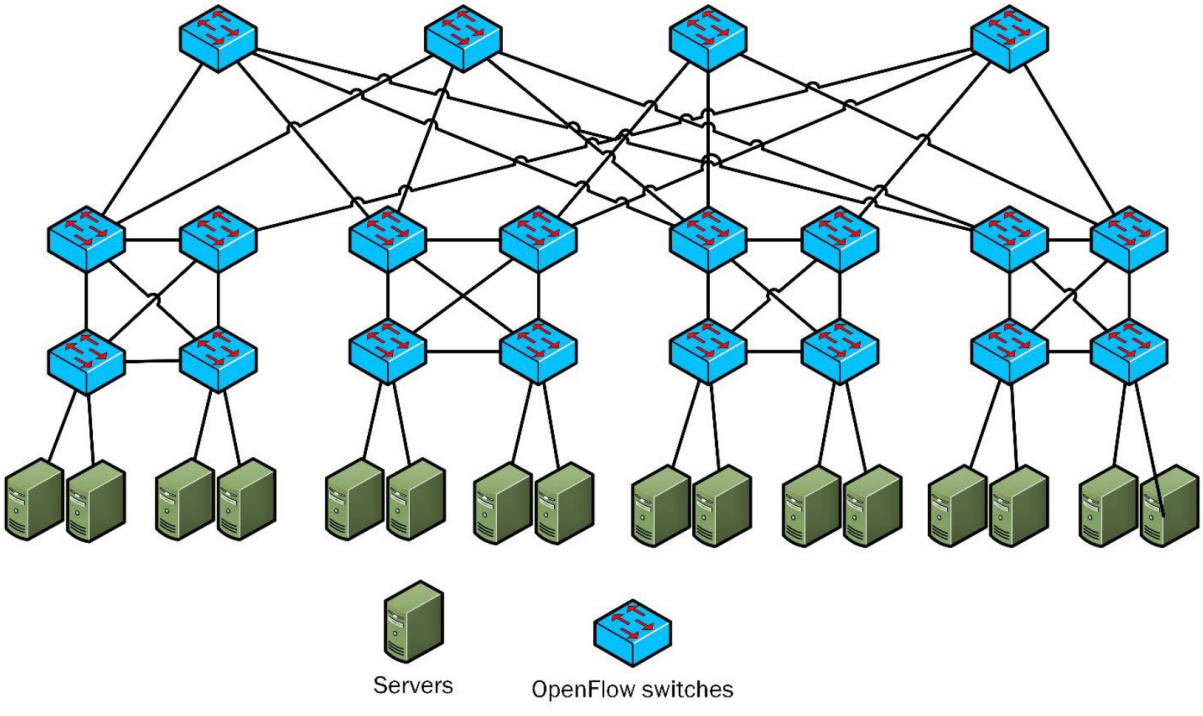

Figure 1: Data Center Fat Tree Topology

The number of requests handled by the server at a time is called load. Each server has its capacity for handling the load the load increases the server capacity the delay in reply to the requests and the data loss increases and it is also possible that the server gets down. This implies that the load needs to be balanced. Load balancing is a technique that divides the network traffic load into the resources of the network to avoid overloading conditions on any of the resources present in the system [7]. For Load balancing, the network manager requires knowledge of the network, links, server capacity, and the current load of the server in the DCN. The load balancing algorithms applied need to be adaptive, this means that the algorithm should be able to change its behavior with the change in network traffic. The major goal of load balancing is to manage the load in such a way that the load distributes proportional to server capacities and the bandwidth consumption

remains as minimum as possible [8].

One of the major advantages of load balancing is that if a server is overloaded then the load balancer redirects network traffic from the overloaded server to another server. This signifies its importance in the DCN [9]. The algorithms are traditionally [10] on a load-balancing server or some other hardware load balancer. But in modern networking, there is a comparatively novel approach named Software Defined Networking (SDN) that allows configuring the network at a centralized point called an SDN controller [11].

SDN is a very cost-effective solution and has more benefits other than traditional network monitoring techniques. In traditional networking, all the network devices like routers and switches have two logical planes [12]. Data plane and Control plane. Control Plane is the intelligence of a device through this they learn the map or topology of the network. Data Plane is responsible to forward the traffic or action the traffic that is coming to the device. if the data plane does not know what to do with the packet it consults the control plane. The SDN has separated these two planes. It means the intelligence of every networking device has been taken away from them and is placed at a centralized configuration device called an SDN controller [13], only the data plane is left on the device. It means the device can only receive and forward the packet according to the flow rules deployed by the controller. If there is a packet that does not have any flow rule on it then it is forwarded to the controller, then the controller deploys the flow rule and the packet is forwarded to the destination. SDN uses an OpenFlow protocol to communicate with the switches. Open flow is the control plane protocol developed by the Open Networking Foundation (ONF) and this is the standard protocol that is available to all network devices today [10]. SDN provides a programming interface to the developer and provides the easiest management. Figure 2 depicts abstract level orchestration of SDN Architecture.

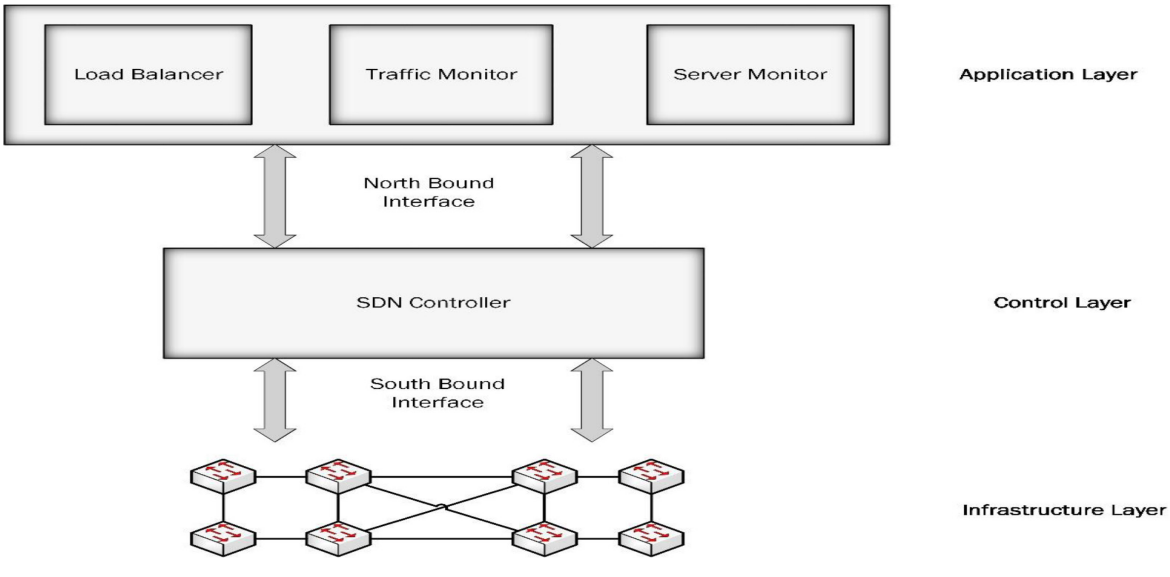

Figure 2: SDN Architecture

The centralized control of the control plane means that the network administrator has to configure the whole network only once and all the devices will follow those configurations. This is why this approach is mostly used in DCN [14].

The proposed approach is an extension to Load Balancing Based on the Closed Loop Control Theory (LBBCLT) [15] load balancing approach that uses a dynamic probe generator to probe the servers about the response time and link bandwidth. This article introduces a path selection module in it and the path is selected using the Ant Colony Optimization (ACO).

2 Literature Review

The authors proposed an algorithm that gets the server response time and then distributes the network traffic according to the server response time [15]. In this algorithm first, measure the flow of the traffic. If the traffic flow increases a threshold value, then the proposed algorithm works otherwise round robin work by default. In the proposed algorithm there are four modules. Server performance monitor that monitors the server response time and remaining capacity, switch port traffic accumulator module analyzes the traffic flow from each switch port to check the link utilization, throughput, and remaining bandwidth. The server Selector selects the optimal server for the incoming traffic. The probe interval generator generates the probe interval to get the response time from the server after analyzing the change in the network traffic. This algorithm is based on memory and CPU utilization and the response time of the servers. Their approach did not consider the bandwidth and throughput of the link. There has to be another module link selector as well.

There is another approach where authors proposed an algorithm that loads balance HTTP traffic that is generated from the client to the data center over the internet [16]. In the proposed algorithm there are three modules. The first module is the server response time collector, which gets the server status and response time of the server. The second module gets the flow of traffic that is received by the OpenFlow switch. This module checks the traffic flow and if this is a new flow forward the network traffic to the controller where the load balancing application works and checks the flow and then told the switch path of flow to the server according to the flow size. The last component is the server selector component which selects the best server in the data center servers that handles the user request according to the flow size. The average response time improves through this algorithm and also improves the throughput compared to other algorithms, round robin, and random selection. Also, get the server CPU utilization and Memory usage to improve the load balancing effect. This approach has a similar problem that there is no path selection algorithm there exists a high possibility that the path with low throughput will be selected.

A traffic-aware load balancing algorithm was proposed by the authors named TA-ASLB [17]. Through this algorithm select a server whose computational capacity is high in the server farm.

In the proposed methodology there are three-phase. The first controller monitors the entire topology of the data center. The proposed method takes three details, Network traffic flow size, the residual bandwidth of the link, and the capacity of all the servers in the data center. In the second phase, an adaptive decision phase where the algorithm checks if the traffic flow is increased by a threshold value then the proposed algorithm is used to balance the traffic otherwise default round-robin method is used to balance the network traffic to their server. The third phase is the server selection phase where the most appropriate server is selected and user queries are sent. Through this algorithm, throughput is improved and compared with another algorithm Round Robin(RR), Weighted-RR method, and the statistics method. Latency is also improved through this algorithm and also latency is compared to Round Robin and Weighted-RR methods. This methodology will decrease the throughput with the increase in the traffic size in a linear fashion that is very rapid and is very unsuitable for data center networks.

The authors proposed an algorithm that used the shortest path first. In this algorithm first, find out the total connected host [18]. Then using the Shortest Path First find the route information. After that find the total link cost for all the routes. After that get the transmission rate and then forward the traffic to the switch to the best path. Through this algorithm, the network is improved and the packet transmission rate is improved successfully. But in this research author has not worked on the throughput of the network traffic which is a very important aspect of load balancing. So when both throughput and response are better we achieve better load balancing to the server in the data center network. Also used different algorithms to improve the load balancing. They claimed to use the Dijkstra algorithm for path selection which is very similar to the shortest path selection algorithm this algorithm will lose performance with an increase in the number of hosts that used only 4 hosts whereas in DCN there are far more hosts available.

The authors proposed a system model that gets the server response from the server pools [19]. Traditionally the response time was taken using the ping facility but the response time from the ping facility is not accurate which is why this model is proposed. The system has three important parts. The first one is terminal equipment which is clients and servers. The second part is the network services that include OpenFlow switches. The third part is the decision phase where control finds the best servers whose response time is better and sends all the traffic to those servers. Through this approach get a better response time from the server but they did not include the throughput of the network. In load balancing, there are two most important aspects the throughput of the servers and the response time of the servers, their approach has a very fluctuating throughput.

The author proposed a framework that is based on software-defined networking [14]. The framework consists of two parts: infrastructure and control. and this approach is applied to well-known data center topologies. Also used a Proportional-Integral-Derivative (PID) controller for load balancing between the servers in the data center. To get server response time this research uses the Network Statistics module to determine the server that has the lower response. The

Flow Manager module adjusts the flow and sends the flow to the appropriate path. Through this approach, throughput is increased and response time is better than the old traditional load balancing technique and this approach is cost-effective. Their approach has very high fluctuation in response time and throughput which is a very bad sign because the higher the fluctuation the lower the performance because which will cause connection instability.

Authors proposed a Deep reinforcement learning approach for load balancing in SDN-based data center networks [20]. Their approach proposed an artificial plane above the control plane of SDN architecture. The AI plane is collecting network statistics continuously and is using Graph Convolutional Networks (GCN) and Deep Q Networks (DQN) combined for load learning from the environment and performing the load balancing. This load-balancing approach is very optimal. The throughput observed is good and improved. Thus, the bandwidth consumption is optimal here. The use of deep reinforcement learning makes this approach adaptive to the change in network parameters and it behaves accordingly. The problem found in this research is, they only did path load balancing and did not consider server capacity.

The authors proposed a simple path selection algorithm using SDN to perform the load balancing for data center networks [21]. Their approach distributes the data to every possible path to make sure that maximum throughput is available for use. Nevertheless, the data receiving time has been increased and is fluctuating because the data received is not in through uniform paths, which creates a fluctuation in time, taken. This approach is very impractical for time-sensitive applications. For example, when the web server is being load balanced. Because the web server load balancing directly affects the user experience.

3 Proposed Methodology

The proposed approach is focused on balancing the load among the servers by selecting an optimal server and selecting the better path toward the selected server. To perform this, the proposed approach is divided into five modules. Probe interval generator, this module will select the time interval after which the servers will be probed for their performance. Server performance monitor, this module will monitor the server response times and saves them in the local database. Server selector, this module will read the local database for the server response times and will select the server with the least response time as the optimal server, Switch port accumulator, this module will probe the switches for the traffic size passing through their ports and will store the data in the local database. In addition, the Path selector, in this module will take the selected server and traffic data from the database and will execute the ant colony algorithm on the network to find a better path from the sender to the receiver. These modules are shown in Figure 3.

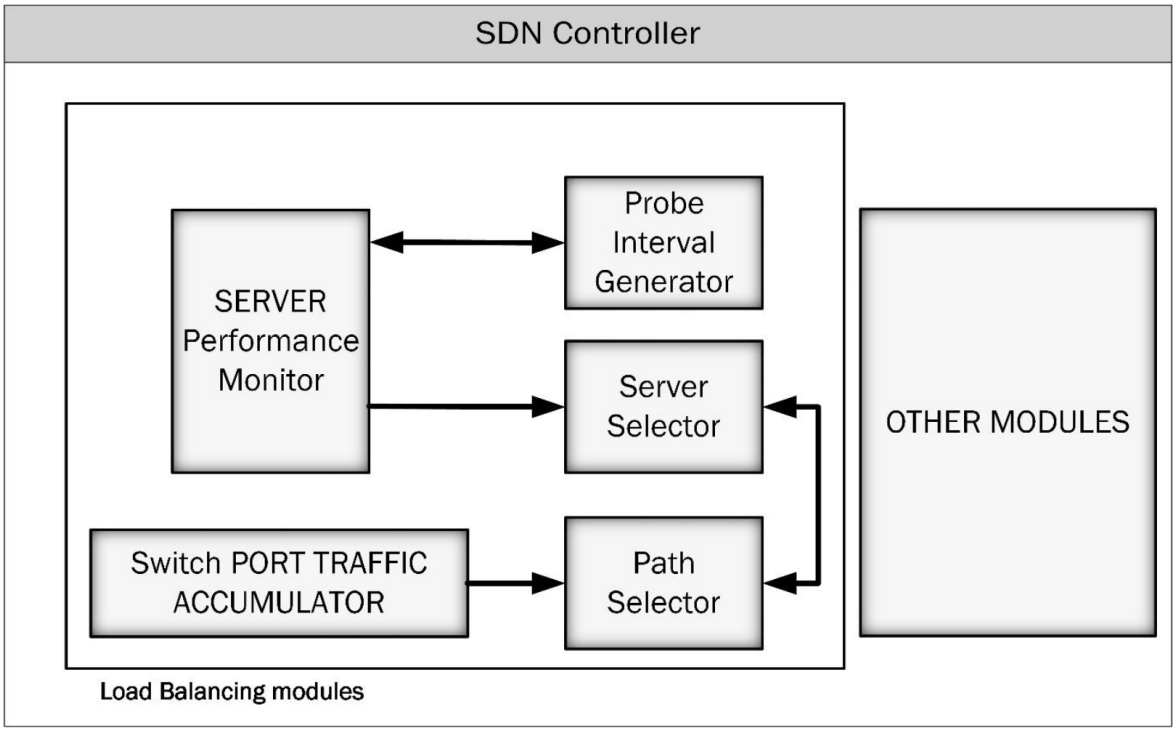

Figure 3: Proposed Methodology

3.1 Probe Interval Generator

This module has the responsibility to make the proposed approach adaptive. Initially, a base value of probe interval (I) is set by the network administrator. This module is taking the server load and probe interval (I) as input and stores, the server load for one iteration and takes the new server load (⟦Load⟧_new) on every iteration, and saves the previous load in the old server load variable (⟦Load⟧_old). This will tell whether the load is increased from the previous iteration or decreased. If the load is increased, then the probe interval (I) will be decreased by 25% and if the load is decreased then the probe interval (I) will be increased by 25% and if it is the same the same probe interval (I) will be used. This is given by equations 1, equation 2, and equation 3.

$$I=I \times 0.75 \text{ if } Load_{\_new} > Load_{old} \quad (1)$$

$$I=I \times 1.25 \text{ if } Load_{new} < Load_{old} \quad (2)$$

$$I=I \text{ if } Load_{new} = Load_{old} \quad (3)$$

The interval generated by this module will trigger all other modules in the proposed approach the path and server performance monitor and traffic accumulator will be immediately triggered, the next interval will be found and after server performance monitoring, the server selector will select the server and the traffic will be redirected using the selected path towards the selected server. This module is inspired by the LBBCLCT [15] approach.

3.2 Traffic Accumulator

This module will be triggered after every time interval set by the probe interval generator. The major task of this module is to gather the stats of the switch port traffic. The traffic data will be stored in a local database. In this research, SQLite is used as a local database. To retrieve the switch port statistical data in this module, send an OpenFlow Statistics Message (OFPPORTSTATREQUEST) to the OpenFlow switches in the data plane of the network. Then the switches send a reply message called OFPPORTSTATREPLY comprising the size of traffic passing through each port of the switch and the time taken to retrieve the data from the switch. This data is then forwarded to the path selector module. This module has been used previously and was originally inspired by the SD-WLB approach [16].

3.3 Server Monitoring

For server performance monitoring the server response time is captured because the load on the server directly affects the response time, if the response time is quick, then the load is low and if the response time is high then the load is high on the server. To find the response time instead of sending an ICMP ECHO message ARP REQUEST message is sent. This is done because the ICMP packet is first to be dropped if the server is highly loaded but this is not the case with ARP packets. For this reason, the ARP REQUEST packet is sent to every server in the data center. After the packet is sent, a tuple with the server's IP and the timeinmillis stamp is stored at the controller. As soon as the ARP reply from the corresponding IP address is received. A new timeinmillis stamp will be taken and the old stamp will be subtracted from the new one this is also demonstrated in equation 4

$$Response\ time = timeinmillis_{new}\ timeinmillis_{old} \quad (4)$$

Then the average response times of every server are found and sent as $Load_{new}$ to the probe interval generator. The idea of sending the ARP packet instead of sending the ICMP packet was inspired by LBBSRT [19] approach. After the experiments, it has been observed that is it relatively a better approach than sending the ICMP packets.

3.4 Server Selection

This module selects the server with the least response time as the best-performing server and redirects the traffic toward this server. In this module when a new flow is coming to the controller. The OpenVswitch checks the recent flow of traffic and the recent response of all the servers. If the flow is match any existing flow, then forward the user traffic to the server whose response time is less throughput is high. If the flow is not matched, then switch forward the traffic flow to the controller. Then the controller installs the flow rules to redirect this new traffic toward the best server selected by this module via the best path chosen by the path selector module.

3.5 Path Selector

This module gets triggered parallel to the server performance monitor by the probe interval generator. This module uses the Ant colony optimization (ACO) algorithm for optimal path selection toward every server in the network topology. The Sending node of the network is considered to be the ant colony and the selected optimal server is the food source. The switches and their connections are the edges and vertices. The ACO will send the ants to every possible path in search of food. One ant traversing one edge will increase its weightage which will evaporate as well after the fixed time interval. Then the quickest and coming back to the ant colony will leave more weightage on the edges. This will make other ants go over the more weightage edge than the less weightage edge. This is how when all the ants will use one path that will give the optimal path. This research has optimized the ant colony algorithm for load balancing. After the complete execution, this algorithm will give the best path toward every server from the SDN controller.

4 Simulation and Results

For simulation purposes, the FloodLight SDN controller is used. The controller is then integrated with the mininet SDN simulation tool. The mininet is not being used as a virtual machine, instead, it is installed on the physical machine. The machine has a core i5-10210 processor with 8GBs of RAM with Ubuntu 22.04 installed as the primary operating system. The fat tree topology of mininet with a depth of 3 and fanout of 5 is used for simulation because a fat tree is the most common topology of the data center networks [22] The depth of 3 and fanout of 5 will result in 125 hosts, 12 of them are configured as SMB file server. All the other nodes i.e. 113 hosts are accessing the servers. The higher amount of load is intentionally diverted toward one server to test the performance of the proposed approach. Figure 4 shows the test topology.

The Load balancer package of the controller is modified to integrate the proposed approach with the SDN controller. The link discovery module is used by the path selector to get aware of the network topology and construct the graph of the network topology. The traffic from the port traffic accumulator is used to assign the costs on the edges. Then the ACO is operated on this graph. Similarly, the traffic port accumulator sends the OFPORTSTATREQUEST to the OpenFlow switches, then switches replies with OFPORTSTATREPLY to the SDN controller, Then the SDN controller uses the statistics module to store the data in SQLite local database.

When a packet arrives for a request from a new user, the IP address will be unknown to the flow rules installed on OpenFlow switches. In this case, the PacketIn message is generated and it is forwarded to the SDN controller. Now SDN controller installs the flow rule towards the selected optimal server using the selected optimal path. After the flow rule has been installed, the SDN controller generates the PacketOut message.

The load balancing module is working continuously and does not wait for the PacketIn event to get triggered because in that case the time consumed to handle the user request will get increased

and there might get a chance that the user request might get timed out. This affects the user experience of the services. The load balancing module is working in the background the optimal server is always known to the SDN controller. and the optimal path from the sending node of the network to the optimal server is also computed at the same time when the optimal server is chosen.

This means in the case of the PacketIn event both the path and the server are already known to the SDN controller and the SDN controller just installs new flow rules in the OpenFlow switches according to the path and destination server.

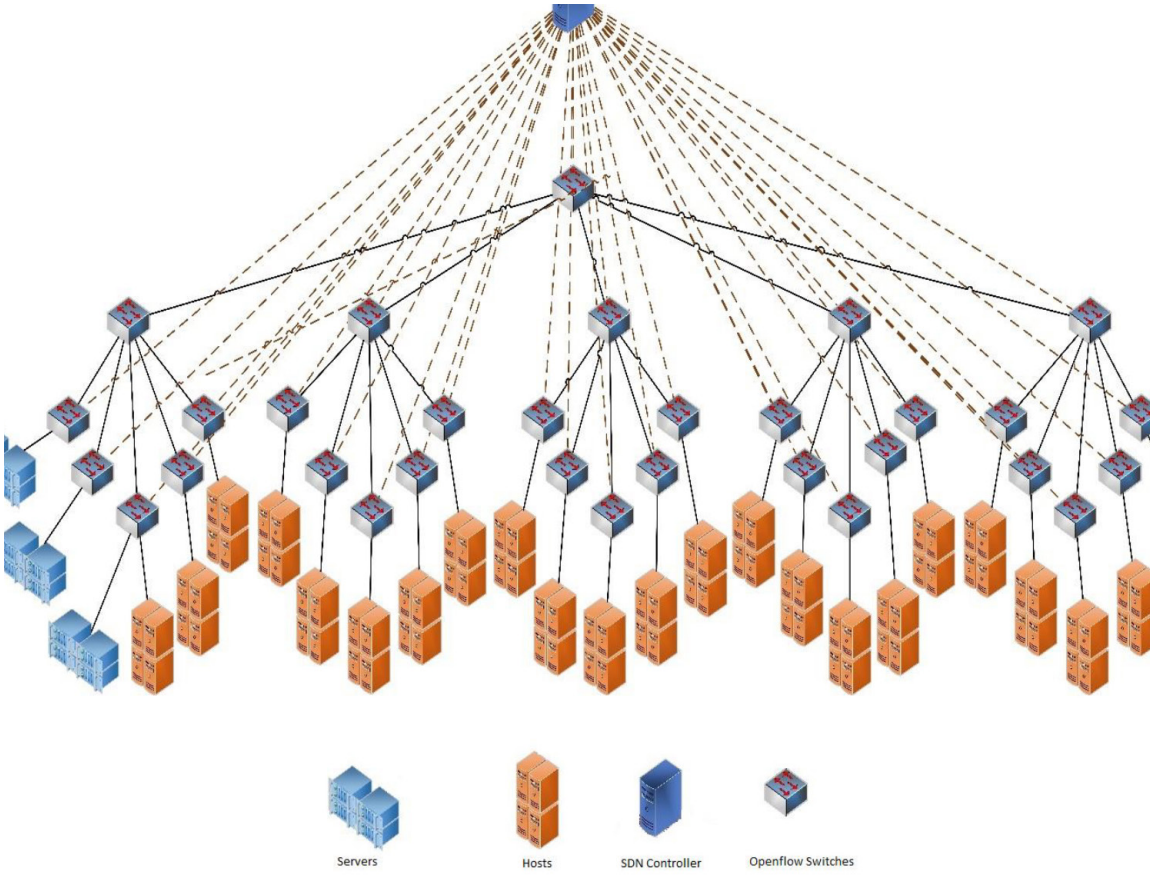

Figure 4: Test Simulation Topology

The throughput of the network is monitored by the iperf Linux command on the controller which gives an overview of how much a server is being loaded. The approach worked very well, the throughput of the network has comparatively improved the results, and its comparison with other approaches is given in Figure 5. Table 1 shows the response time of servers.

The load balancing modules start working as soon as the controller is executed the network

administrator gives the initial probe interval. While simulating the approach when the topology is initiated, there is no load on the servers. Then the nodes start requesting the files and started to upload the files as well. When the first client will request, the server selector will choose the server with the least response time. Similarly, at the second request if the probe interval was reached the server selection module will choose then the next server with the least response time.

This is indicating that initially, the load at the data center will increase which will result in a decrease in probe interval. But after some time the load will get stabilized, and the probe interval will also get stabilized. Then the behavior of decreasing the probe interval with the increase in network traffic size and increasing the probe interval with the decrease in network traffic will be observed. The throughput is monitored by monitoring the network performance of the servers continuously.

This research also monitored the server response time of the approaches as well. The proposed approach has outperformed the previous approaches in terms of response time as well. Because the path obtained by the ant colony optimization is a better path than the path provided by Dijkstra. The comparison is given in Figure 6. Table 2 shows the throughput achieved by different approaches as discussed in literature The results show that the proposed approach has the least response time even in an increasing number of requests. This is because the probe interval generator is generating the optimal time for probing. The load gets balanced before even getting increased,

Table 1: Average response time (seconds) of servers under different approaches

No. of Requests	Round Robin	SD-WLB	LBBSRT	LBCLCT	Proposed Approach
10	0.0451	0.0455	0.0451	0.0451	0.0441
20	0.0445	0.0454	0.0453	0.0425	0.0410
30	0.0431	0.0456	0.0455	0.0420	0.0405
40	0.0432	0.0455	0.0495	0.0455	0.0415
50	0.0431	0.0457	0.0445	0.0425	0.0385
60	0.0562	0.0445	0.0412	0.0415	0.0380
70	0.0761	0.0455	0.091	0.0405	0.0375
80	0.059	0.0576	0.0576	0.0395	0.0370
90	0.093	0.0658	0.0586	0.0385	0.0371
100	0.0721	0.0468	0.0751	0.0395	0.0368
110	0.0730	0.0460	0.0760	0.0393	0.0364

Table 2: Average Throughput (Mbps) available on servers

No. of Requests	Round Robin	SD-WLB	LBBSRT	LBCLCT	Proposed Approach
10	180	180	180	180	180
20	175	178	180	180	180
30	163	173	175	180	180
40	158	169	171	175	180
50	147	164	168	173	178
60	139	157	164	168	175
70	124	150	159	164	172
80	120	141	154	160	169
90	113	131	150	155	165
100	103	125	145	150	162
110	100	124	144	150	163

The graphical representation of the data represented in the tables above is given in Figure 5 and Figure 6. The graphs visualize that how much proposed approach is better than the previous approaches.

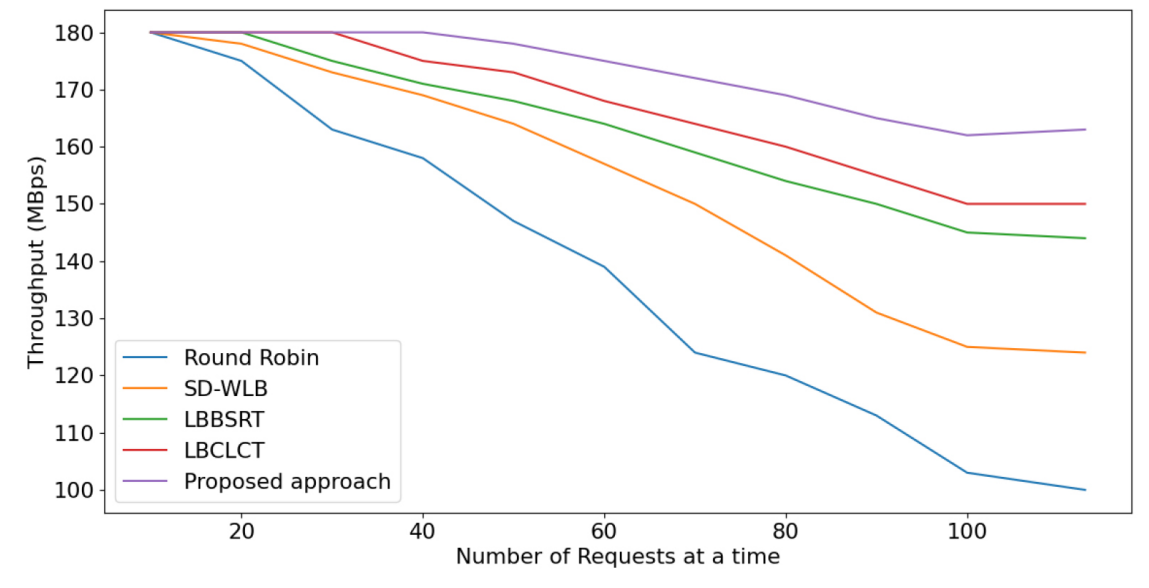

Figure 5: Average Throughput Comparison

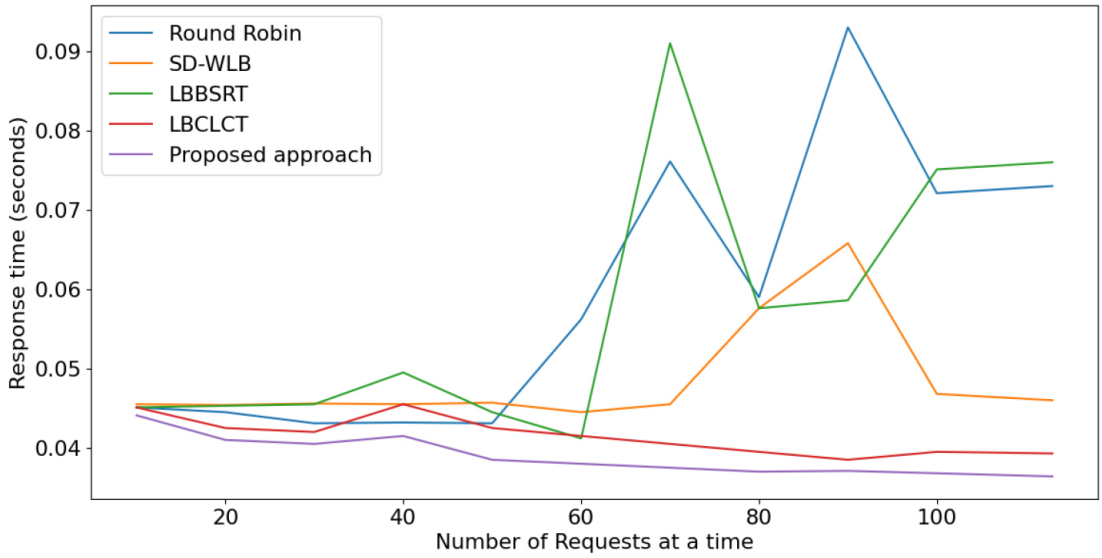

Figure 6: Response Time Comparison

5 Conclusion and Future Work

This article proposes a load-balancing approach for SDN-based data center networks. This approach has five different modules, Server selector, Probe interval generator, Path selector, Server performance monitor, and port traffic accumulator. The server selector selects the optimal server, the probe interval generator generates the efficient interval to probe the servers about the performance and find out the paths from worst performing to best-performing server. The path selector finds out the optimal path using the ACO algorithm. The server performance monitor is monitoring the server's performance and response time. The traffic port accumulator is giving the details about the traffic running through different ports of the switch. These modules are inspired by the literature review and the best of them are integrated to make a better approach for server load balancing in data center networks.

The innovation of this approach is the use of the Ant Colony algorithm. The approaches discussed in the literature review use the Dijkstra algorithm for path selection. Dijkstra algorithm is a greedy algorithm. It is not necessarily the path chosen by it is always optimal. However, Ant colony optimization ensures the selection of an optimal path.

The proposed approach has outperformed many previous approaches in terms of throughput and server response time but requires lots of computational resources. In the future, an optimal approach is required that will optimize the computational resources of the load-balancing approach. For optimization of this approach, deep reinforcement learning can be employed.

Reference

[1] Guo, Pengfei, et al. "A Congestion Control Algorithm Based on Deep Reinforcement Learning in SDN Data Center Networks." *International Conference on Neural Networks, Information, and Communication Engineering (NNICE 2022),* vol. 12258, SPIE, 2022, pp. 211–16. www.spiedigitallibrary.org, https://doi.org/10.1117/12.2639258.

[2] Pang, Shuanglong, et al. "Research on SDN-Based Data Center Network Traffic Management and Optimization." *2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA),* 2022, pp. 600–04. IEEE Xplore, https://doi.org/10.1109/ICPECA53709.2022.9718973.

[3] Tang, Qian, et al. "Elephant Flow Detection Mechanism in SDN-Based Data Center Networks." *Scientific Programming,* vol. 2020, Sept. 2020, pp. 1–8. DOI.org (Crossref), https://doi.org/10.1155/2020/8888375.

[4] Wang, You-Chiun, and Ting-Jui Hsiao. "URBM: User-Rank-Based Management of Flows in Data Center Networks through SDN." *2022 4th International Conference on Computer Communication and the Internet (ICCCI), 2022,* pp. 142–49. IEEE Xplore, https://doi.org/10.1109/ICCCI55554.2022.9850240.

[5] Vani, K. A., and K. N. RamaMohanBabu. "An Intelligent Server Load Balancing Based on Multi-Criteria Decision-Making in SDN." *International Journal of Electrical and Computer Engineering Systems,* vol. 14, no. 4, Apr. 2023, pp. 433–42. hrcak.srce.hr, https://doi.org/10.32985/ijeces.14.4.7.

[6] Sridevi, K., and Md Abdul Saifulla. "LBABC: Distributed Controller Load Balancing Using Artificial Bee Colony Optimization in an SDN." *Peer-to-Peer Networking and Applications,* vol. 16, no. 2, Mar. 2023, pp. 947–57. Springer Link, https://doi.org/10.1007/s12083-023-01448-2.

[7] Ramasubbareddy, Somula, and R. Sasikala. "RTTSMCE: A Response Time Aware Task Scheduling in Multi-Cloudlet Environment." *International Journal of Computers and Applications,* vol. 43, no. 7, Aug. 2021, pp. 691–96. DOI.org (Crossref), https://doi.org/10.1080/1206212X.2019.1629098.

[8] Xu, Chen, et al. "A Survey of SDN Traffic Management Research." *2022 11th International Conference on Communications, Circuits and Systems (ICCCAS),* 2022, pp. 231–36. IEEE Xplore, https://doi.org/10.1109/ICCCAS55266.2022.9824926.

[9] Zaher, Maiass, et al. "Sieve: A Flow Scheduling Framework in SDN Based Data Center Networks." *Computer Communications, v*ol. 171, Apr. 2021, pp. 99–111. *ScienceDirect,* https://doi.org/10.1016/j.comcom.2021.02.013.

[10] Schaller, Sibylle, and Dave Hood. "Software Defined Networking Architecture Standardization." *Computer Standards & Interfaces,* vol. 54, Nov. 2017, pp. 197–202. DOI.org (Crossref), https://doi.org/10.1016/j.csi.2017.01.005.

[11] Hamdan, Mosab, et al. "A Comprehensive Survey of Load Balancing Techniques in Software-Defined Network." *Journal of Network and Computer Applications,* vol. 174, Jan. 2021, p. 102856. ScienceDirect, https://doi.org/10.1016/j.jnca.2020.102856.

[12] Semong, Thabo, et al. "Intelligent Load Balancing Techniques in Software Defined Networks: A Survey." *Electronics,* vol. 9, no. 7, July 2020, p. 1091. DOI.org (Crossref), https://doi.org/10.3390/electronics9071091.

[13] Abdelrahman, Abdallah Mustafa, et al. "Software-Defined Networking Security for Private Data Center Networks and Clouds: Vulnerabilities, Attacks, Countermeasures, and Solutions." *International Journal of Communication Systems,* vol. 34, no. 4, 2021, p. e4706. Wiley Online Library, https://doi.org/10.1002/dac.4706.

[14] Montazerolghaem, Ahmadreza. "Software-Defined Load-Balanced Data Center: Design, Implementation and Performance Analysis." *Cluster Computing,* vol. 24, no. 2, June 2021, pp. 591–610. *DOI.org (Crossref),* https://doi.org/10.1007/s10586-020-03134-x.

[15] Malavika, Rajadurai, and Muniappan Lakshapalam Valarmathi. "Load Balancing Based on Closed Loop Control Theory (LBBCLCT): A Software Defined Networking (SDN) Powered Server Load Balancing System Based on Closed Loop Control Theory." *Concurrency and Computation: Practice and Experience,* Feb. 2022. DOI.org (Crossref), https://doi.org/10.1002/cpe.6854.

[16] Soleimanzadeh, Kiarash, et al. "SD-WLB: An SDN-Aided Mechanism for Web Load Balancing Based on Server Statistics." *ETRI Journal,* vol. 41, no. 2, 2019, pp. 197–206. *Wiley Online Library,* https://doi.org/10.4218/etrij.2018-0188.

[17] Fancy, C., and M. Pushpalatha. "Traffic-Aware Adaptive Server Load Balancing for Software Defined Networks." *International Journal of Electrical and Computer Engineering (IJECE),* vol. 11, no. 3, June 2021, p. 2211. *DOI.org (Crossref),* https://doi.org/10.11591/ijece.v11i3.pp2211-2218.

[18] Emad Ali, Tariq, et al. "Load Balance in Data Center SDN Networks." *International Journal of Electrical and Computer Engineering (IJECE),* vol. 8, no. 5, Oct. 2018, p. 3084. DOI.org (Crossref), https://doi.org/10.11591/ijece.v8i5.pp3084-3091.

[19] H. Zhong, Y. Fang, and J. Cui, "Reprint of 'LBBSRT: An efficient SDN load balancing scheme based on server response time,'*Future Gener. Comput. Syst.,* vol. 80, pp. 409–416, Mar. 2018, doi: 10.1016/j.future.2017.11.012.

[20] Xiangyun, Zeng, et al. "Deep Reinforcement Learning with Graph Convolutional Networks for Load Balancing in SDN-Based Data Center Networks." *2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP),* IEEE, 2021, pp. 344–52. *DOI.org (Crossref),* https://doi.org/10.1109/ICCWAMTIP53232.2021.9674074.

[21] Chakravarthy, V. Deeban, and Balakrishnan Amutha. "A Novel Software-Defined Networking Approach for Load Balancing in Data Center Networks." *International Journal of Communication Systems,* vol. 35, no. 2, 2022, p. e4213. *Wiley Online Library,* https://doi.org/10.1002/dac.4213.

[22] Zhang, Jiao, et al. "Load Balancing in Data Center Networks: A Survey." *IEEE Communications Surveys & Tutorials,* vol. 20, no. 3, 2018, pp. 2324–52. *IEEE Xplore,* https://doi.org/10.1109/COMST.2018.2816042.