Load Balancing in DCN Servers through SDN Machine Learning Algorithm

G. Sulthana Begam1 · M. Sangeetha2 · N. R. Shanker1

Received: 18 January 2021 / Accepted: 17 June 2021 / Published online: 10 July 2021 © King Fahd University of Petroleum & Minerals 2021

Abstract

Development in Internet technologies increases Internet users exponentially. Increase in users leads to more data center network (DCN) and heavy data traffic in servers. Data traffic in servers is managed through software-defined networking (SDN). SDN improves utilisation of large-scale network resource and performance of network applications. In SDN, load balancing technique optimises the data flow during transmission through server load deviation after evaluating the network status dynamically. However, load deviation in network needs optimum server selection and routing path with respect to less time and complexity. In this paper, we proposed a multiple regression-based searching (MRBS) algorithm for optimum server selection and routing path in DCN to improve performance even under heavy load conditions such as message spikes, different message frequencies, and unpredictable traffic patterns. MRBS selects the server based on regression analysis, which predicts types of traffic and response time based on the server data parameters such as load, response time, and bandwidth and server utilisation. MRBS combines heuristic algorithm and regression model for efficient server and path selection. The proposed algorithm reduces the delay and time more than 45% and shows better sever utilisation of 83% when compared with traditional algorithms due to stochastic gradient decent weights estimation.

Keywords Data center networks (DCN) · Software-defined networking (SDN) · Load balancing · Regression model · Server selection · Openflow

1 Introduction

The DCN[1] increases due to more usage from Internet applications such as streaming videos, e-commerce, social networking, and data storage from these applications expands up to 40 ZB every year. This growing demand for data storage and access needs efficient load balancing methods for reducing latency and response time. Any request forwarded to the server needs a load balancing mechanism to select the optimum server and path to achieve maximum throughput, as well as reduce resource utilisation for better user satisfaction. The load balancing mechanism is for the selection of the optimum server and also depends on the routing path from the request of a node to a server. The DCN applies SDN algorithms for load balancing mechanism.

SDN [2] has emerged as an efficient network technology, capable of supporting innovation in future network functions and applications, which can be implemented quickly and efficiently. Major benefits include low operating costs through simplified hardware and network management, effectively utilising all available server resource. SDN consists of data and control plane, simple forwarding devices represent a data plane, and control plane consists of a controller program to perform various network control functions involving monitor and control behavior of the underlying network. The SDN with Floodlight controller improves the load balancing in DCN server clusters, which forwards a request to the destination server based on the heuristic algorithm, designed to select the best path and server. Performance of open-source controllers in SDN such as POX, NOX, Trema, Ryu, OpendayLight, Floodlight, and ONOS

1 CSE, Aalim Muhammed Salegh College of Engineering, Chennai 600055, India
2 CSE/IT, Coimbatore Institute of Technology, Coimbatore 641014, India

* G. Sulthana Begam
  sulthanabshafi@aalimec.ac.in
M. Sangeetha
  msangeethsdr@gmail.com
N. R. Shanker
  nr_phd@yahoo.co.in
controllers is evaluated based on latency and throughput. However, the performance of any SDN controller is also influenced by various factors such as security, scalability, load balancing, processing capacity, controller placement, flow monitoring, flexibility, programmability, latency, and throughput.

In DCN, load balancing algorithm works during overload condition in the server based on threshold levels and through prediction of loads. The overload measurement consumes more time for decision-making and switching of the server becomes more difficult during dynamic load variations. Load prediction, in the server, is identified through stochastic work load characterisation. The stochastic method [3] estimates the load condition of the server during dynamic changes of the resource utilisation and provides a probabilistic-based approach in handling overloads. Parameters like bandwidth usages and migration path are not considered while decision-making and that leads to inefficient load balancing during demanding workload patterns. Workload scenarios like different message frequencies and unpredictable traffic patterns in the server such as web requests, video requests, social networking, and data mining requests demand for efficient algorithm for estimation and prediction. Proposed controller model uses stochastic gradient decent weights estimation of loads, which considers heterogeneous bandwidth of links in DCN. Server bandwidth in DCN is affected due to multiple paths routing among servers. Equal-cost multipath (ECMP) [4] load balancing algorithm used in commodity switches manages the traffic among the servers without considering the bandwidth for the dynamic traffic of DCN. The shortest path routing algorithms such as Dijkstra, Bellman Ford, and A* never consider parameters of topology in the network, complexity, and computing time. Multipath transmission control protocol (MPTCP) [5] is used for congestion control and used as software updates. MPTCP selects alternate multiple paths to forward subflows and improves throughput. CONGA (congestion-aware balancing) [6] minimises the complexity of centralised approach of MPTCP, which is a congestion-aware load balancing mechanism that needs changes in the hardware. LetFlow[7] considers load balancing flowlet, which causes load imbalance, when traffic burst occurs. Loads of DCN server are dynamic in nature because the flow length and flow size vary with time, which are never considered and addressed in load balancing techniques of [4–7].

Algorithm for load balancing in DCN should consider both server selection and routing but the conventional algorithms contemplate either the routing path or server selection. The routing path algorithm influences the appropriate server selection based on the network topology and computational time. Increase in computational time for selecting router influences the server selection to avoid overloading and for faster round trip time.

In the cloud environment, improving the traffic in the server without any congestion and maximising throughput are challenging tasks, which drive the researchers to develop best routing algorithms for cloud server selection and routing. Server selection and routing are independent algorithms so far, but this should be analyzed alongside during the process of data communication in DCN. The correlation of routing and server selection parameters manages to avoid the congestion and complexity in server selection in the event of frequent messaging and unpredicted traffic patterns. To handle the server, operating frequent requests with unpredicted traffic patterns need to know the correlation between the routing environment and server which helps in understanding the behavior of traffic and the source of frequent messaging. The above problem can be solved only through hybridisation of algorithms that helps to measure the frequent messaging levels and predict the source of traffic patterns. Combined algorithm for routing and server selection paves the way toward improving the server and routing parameters such as the response time, throughput, server utilisation, and delay time.

1.1 Contributions

Algorithms for load balancing proposed in [8–12] have limitations with load calculations that include only response time thresholds and parameters thresholds. To identify imbalance in load and congestion the traditional methods sets threshold which leads to inefficient load balancing, computational complexity and processing delays. The performance of network measured in terms of throughput is greatly influenced by resilience to path failure, which is never considered during performance measurements. The proposed load balancing techniques in [14, 15] manage the traffic among the servers without considering the bandwidth of dynamic traffic of DCN. The load balancing mechanisms, which improve QoS, only weigh on routing and never consider optimum server selection for data flow. The proposed load balancing techniques in [16–20] use only multipath routing, and the load balancing algorithm in [21] harnesses the optimum server selection to maximise the efficient server utilisation. The proposed MRBS algorithm incorporates both the server and optimum path selection based on combination of heuristic and regression algorithms.

(i) To achieve congestion-free network demands, the heuristic estimator is used for selecting the best path to the server. In this paper, we propose the MRBS algorithm to improve the throughput through heuristic path search algorithm and uses heuristic estimation function.
f(n) = g(n) + h(n). MRBS algorithm efficiently searches best path with less overhead and quick turnaround. f(n) is the estimated lowest number of requests from clients to server. g(n) indicates the number of request in the path from the client to node n. h(n) represents the heuristic, exact number of requests processed along the path from node n to the server.

(ii) To improve more efficient usage of network resources, data for the servers related to load, response time, bandwidth must be collected over a period of time, and data are analyzed and correlated for quantifying the association between the load and bandwidth of the server with the response time. Servers with less response time should be selected to manage the optimal distribution of loads between servers.

(iii) In the proposed system, servers are divided into server clusters where each cluster can address different types of service. Periodic statistics of server cluster are collected and the number of servers in a cluster can be adjusted and servers can be migrated from one cluster to other based on the load of the cluster.

Section II discusses the related works. In Section III, we detail the design approach. Then, Section IV illustrates an implementation environment, and Section V discusses performance evaluation from four aspects. Finally, we conclude with our findings and future work.

2 Related Works

The load balancing in DCN is performed by various algorithms using the concepts of SDN. The SDN and load balancing hybrids to share the loads dynamically [8] based on network traffic and achieve higher throughput in cloud DCN using plug-n-server. Furthermore, utilisation based on queuing is achieved through RPL(QU-RPL) [9] algorithm queue utilisation. The RPL algorithm helps to achieve resource allocation and load balancing and improves packet delivery ratio during the queue environment for resource utilisation. The SDN improves the performance of fat-tree DCN through low-cost routing balance management (L2RM) framework [10], which offsets the load through adaptive route modification (ARM) and dynamic information polling (DIP) mechanism to reduce the cost and response time. Moreover, load balancing in DCN is attained through heuristic scheduling algorithm for OpenFlow network models. This model involves dynamic load balanced scheduling (DLBSS) algorithm [11] applying the concept of switching mitigation for maximising network throughput. Faster responses during load balancing would be achieved with SMCLBRT algorithm [12] that functions on real time and adaptive selection of response time threshold. For load management, the utilisation of resource in DCN needs to be awarded and analyzed.

Algorithms such as linear regression, support vector machines for regression, gradient boosting, and Gaussian process regression are used to analyze and estimate the resource utilisation of data center. An adaptive predictive threshold algorithm measures the resource utilisation under various work load condition [13]. To find the fastest path to the server, routing mechanism used in [14] sends TCP-SYN packet and optimal routing path is selected based on the quickest turnaround of receiving ACK. Load balancing and resource utilisation facilitate green computing in data centers through effective reduction in power obtained with sensible load balancing and reduce resource utilisation[15].

Moreover, resource allocation during multiple services and queuing in DCN is solved through explicit congestion notification (ENC) algorithm to improve throughput and reduce latency in DCN through isolation of traffic in every port during multiple queuing. This algorithm unravels the problem of load balancing, and similarly, balancing can be implemented through controllers.

Multiple controllers and single centralised controller ease the traffic in data centers through switch mitigations and select the shortest path. The controllers consist of a heuristic algorithm for implementing the load balancing system in DCN. Multipath selection in the data centers changes dynamically depending on the bottleneck and ideal throughput accomplished through MaxPass method [16]. NAMP [17] controller in DCN streamlines traffic flow through convex search algorithm and reduces the congestion. Minimising the flow group transmission time is done proficiently to address the limitations in the size of flow table, and it requires a series of feasible alternative courses of action for decision-making, determined by the resource constraints like flow table size.

Genetic programming-based load balancing (GPLB) [18] distributes workload across servers through real-time least-loaded and shortest path; also, throughput, link utilisation, and latency metrics are evaluated constantly. QoS-aware routing [19] in hybrid SDN/OSPF industrial Internet applies mechanism like single-path least cost forwarding and multipath forwarding based on K-path partition algorithm. DCMPTCP[20] fulfills load balancing of top-of-rack switches using multipath transport routes rack-local, inter-rack, and inter-rack many to one short flows to balance workloads across multiple paths thereby improving congestion control. GB-PANDAS[21]. Generalized Balanced Priority Algorithm for Near Data Scheduling, dispatches task to server with the minimum weighted workload, and this load balancing technique is recommended in the case where task arrival rates and the service rate matrix are unknown.
2.1 Inferences from Literature Survey

Load balancing and server path selection implemented in separate algorithms are so far based on the factors like traffic, resources, queuing, and congestion. However, the server path selection and load balancing need to be addressed simultaneously, to improve the efficiency and throughput. On the downside, traditional algorithms present lot of limitations and shortcomings such as controller mapping, which leads to inefficient load balancing in DCN and response time threshold increases the mitigation cost besides overloading the controller. Sophisticated algorithms are needed for the controller to handle congestion and to find the best path to reach the server.

3 Materials and Methods

The load balancing and server path selection are implemented through MRBS algorithm with the objective to improve the traffic during congestion, which efficiently changes the server path dynamically based on the resource utilisations and reduces the overload condition in DCN. The MRBS algorithm is implemented in Mininet simulator and supports the OpenFlow protocol. The simulation is performed in python language, and the topology of DCN is spine-leaf architecture, which facilitates reduced network failures and easier to add additional servers in DCN, when compared to other architectures such as fat-tree, Jellyfish, BCube, DCell, Xpander. The spine-leaf architecture consists of two switching layers such as leaf or access and spine or aggregation. The leaf layer consists of access switches, which aggregate the server traffic. Spine switches in Layer 2/3 (L2/3) or aggregation layer interconnect all of the leaf switches in a full mesh topology and act as a boundary between DCN servers. AS9716 Spine switches have 32 ports of 40G. AS5916 Leaf is attached to the spine with four ports of 40G. The uplink port count on the leaf switch determines the maximum number of spine switches and downstream port multiplying the leaf switch number determines the number of servers. In building this leaf spine topology with 40G interfaces, it has 1280 10G servers at 2.5:1 oversubscription.

Fig. 1 shows the entire block diagram of the proposed MRBS algorithm. Furthermore Floodlight OpenFlow controller is connected to the Mininet through port 6653.

4 Proposed MRBS Algorithm

The proposed algorithm functions as an SDN application and collects dynamic information of the topology, and devices also gather flow statistics using the REST API (Representable State Transfer API) as the northbound API. The servers are divided into clusters, where each server cluster manages different types of traffic or services. The request for the service is forwarded to a particular server in server cluster based on regression analysis performed based on the variable such as load L, response time RT, bandwidth B, and server utilisation. The request is forwarded to the selected server through the best path with lower traffic flow selected

Fig. 1  Block diagram of MRBS algorithm implemented in spine-leaf architecture of DCN
by heuristic path searching algorithm and regression analysis based on historic values of routing data. The algorithm operates based on server and path selection for any request that arises from the user end or client. The below explains the MRBS algorithm into two modules.

**MRBS algorithm**

**Input:** Network Traffic statistics T (Traffic Matrix), DCN Topology, Load LS/Tn, Response time RTS/Tn, Bandwidth BS/Tn of Server Si at Time Tn  
**Output:** Select Destination Server Si from SC, Best Path from Client C to Si  
**Assign**  
//Server Selection  
set SC (SERVERCLUSTER) = { Si, S2, S3, ..., Sn }  
set START = [C] END {}  
g(C) = 0 \ f(C) = h(C)  
If controller receive a Packet in messages at Time Tn then  
Parse messages;  
For Each Si in the SC do  
Fetch LS/Tn, BS/Tn  
Calculate RTS/Tn = R1 * LS/Tn + R2 * BS/Tn = R3  
End For  
Select Si with \ min( RTS/Tn, RTS/Tn, ..., RTS/Tn) and set this as Goal Node  
//Routing  
If Request r from Client Node C at Time Tn do  
While Goal is reached Then Return PATH  
Select a node N with \ min f(N) from START[] and save it in END[]  
Select Successor M not in loop with \ min h(M) of N  
if M not in START or END  
move M to START  
set g(M) = g(N) + No of request (C, M)  
set f(M) = g(M) + h(M)  
move M to START at Time Tn  
Update the Flow Switch Table  
End While

| Table 1 Average bandwidth utilisation [22] |
|-------------------------------------------|
| Reason for sudden message spikes | Bandwidth in Gbps |
| Scheduled backups | > 150 |
| Updating software | > 200 |
| Peak load due to sudden events | > 300 |
| Hacking attempts | > 500 |

| Table 2 Summary of notations |
|--------------------------------|
| Parameters | Description |
| Si | Server |
| SCi | Server cluster |
| Bi | Bandwidth of server Si |
| B | Average bandwidth of server cluster |
| BSi/Tn | Bandwidth of the server Si at time Tn |
| RTS/Tn | Response time of server Si at time Tn |
| LS/Tn | Load of server Si at time Tn |
| LSCi/Tn | Load of the server cluster SCi at time Tn |

- Sever selection  
- Path selection
4.1 Message Spikes and Different Message Frequencies–MRBS Algorithm

The sudden message spikes and different message frequencies because of running scheduled backups, updating software, mail server problems, and during hacking attempts can trigger unusual bandwidth usage and overloading of server. The network will be a bottleneck during a traffic spike. Table 1 shows the average bandwidth utilisation during various message spikes and conditions which rises above the normal.

During this period of overloaded traffic, the network throughput exceeds the bandwidth of the network and the performance of the server reduces in terms of response time. The load balancing technique identifies the issues using historical data collected using monitoring tools. Table 2 describes the parameters used in the formation of MRBS algorithm.

Therefore, the load of the server cluster $S_i$ denoted as $LSC_i$ can be calculated by the sum of all load of the server in the cluster the total number of received request messages $F_{Stn}$ at time $T_n$ is for measurement of the sudden spikes. As shown in Eq. (1).

$$LSC_{i}T_{n} = \sum_{i=1}^{n} F_{Stn}$$  

(1)

In the proposed system, the load of the server $S_i$ denoted as $LS_i T_n$ at Time $T_n$ is represented as the number of request messages forwarded to that server from edge switches. The module periodically calculates the mean of the load of all server clusters and compares with the set threshold. The threshold level determines the discrepancy in the server, and corrective actions like reshuffling the number of servers in the server cluster or shutting the servers from the cluster can be applied.

Furthermore, the network performance is measured by its standard deviation of load of server cluster and calculated using Eq. (2).

$$\sigma = \frac{1}{N} \sqrt{\sum_{i=1}^{N} (B_i - B)^2}$$  

(2)

where $N$ is the total number of servers in a cluster, $B_i$ is the bandwidth of the server $S_i$, and $B$ represents the average bandwidth of the cluster.

A load balancing algorithm is in the router to address the overloading of the server by changing the traffic through selecting different server and path. The proposed MRBS algorithm in the router performs efficiently in server selection and provides more efficient usage of the network and resources. The proposed MRBS algorithm is designed with high flexibility in usage of server and addresses different types of servers.

The server selection module in MRBS algorithm analyzes various parameters such as load, response time, bandwidth, and server utilisation of server in the form of data sets. From the above data sets, the regression analysis is used for load balancing and shown in Eq. (3) as follows:

$$RTS_{iTn} = R_1 \ast LS_i T_n + R_2 \ast BS_i T_n + R_3$$  

(3)

$RTS_{iTn}$ is the predicted or expected value of the response time of server $S_i$ at time $T_n$, $LS_i T_n$ is load, and $BS_i T_n$ is the bandwidth of the server $S_i$ at time $T_n$ that are two distinct independent or predictor variables, $R_1, R_2$ and $R_3$ are the estimated regression coefficient. The regression model allows the analysis of hypothesis using simple correlation of dependent variables that quantifies the association between the load and bandwidth of the server with the response time.

Detecting load imbalance of server in the traditional network is based on workload, and threshold values are calculated based on the various parameters like load, response time, and bandwidth. The response time is one of the factors for analyzing the QoS of the network, which surges with increase in the load of the servers that also influences the bandwidth utilisation and latency of the network. To achieve quick turnaround, it is appropriate to select the server with low estimated response time $RTS_{iTn}$ at time $T_n$ that influences successful load balancing on servers. The selected server is the optimum server to which the request is forwarded. Multiple regression analysis predicts the response time based on the load to the server and bandwidth of the network, which helps to forecast the congestion in the network. The forecast is based on the multiple regression equation outputs for given input parameters such as load and bandwidth of current values obtained from the network. The prediction in the router at time $T_n$ is for measurement of the sudden spikes. Such sudden spikes lead to the traffic overload and extend to heavy bandwidth consumption of the network. The heavy bandwidth leads to overload in the server.

4.2 Traffic and Path Selection after MRBS Prediction

The traffic in DCN is measured in terms of bits, packets, and flows. Table 3 shows the characteristics of the traffic flow such as flow size and flow duration. The number of active flows relies on the application, which creates the traffic.

| Traffic data set | <50% | <80% | <100% |
|------------------|------|------|-------|
| Flow duration    | 100 ms | 10 s | 100 s |
| Flow size        | 1 KB  | 10 KB| 10 MB |
The traffic generating applications are Internet, cloud, custom applications, and the duration of mice traffic flows is more than 50% with less than 100 ms, and the elephant flows are between 1 and 100 s. The flow size mostly depends on the tenants hosted in DCN. Nearly half of the traffic flow is less than 1 KB size, most of the flows are less than 10 KB in size, and all the flows are less than 10 MB in size. The traffic flow in the routes plays a vital role in the selection of a server in the DCN. Moreover, the path selections of server depend on minimum link cost, lower traffic flow. In this paper, heuristic path search technique is proposed for the server route selection based on the memory constraints and achieves optimality and completeness based on the individual node’s information. The heuristic function is defined as follows.

\[ f(n) = g(n) + h(n) \]  (4)

where \( f(n) \) denotes total request processed by the server, \( g(n) \) is the total request processed while reaching intermediate node \( n \), and \( h(n) \) is the exact number of pending request from node \( n \) to selected server. The heuristic algorithm identifies the best path based on the forwarded request based on the theorem.

**Theorem 1.**

Given an admissible and exact heuristic without constant relative error, then heuristic best path will be guaranteed and finds the best path optimal in terms of solution cost, time, and space over the class of admissible best-first searches on a tree as in Eq. (5) as follows:

\[ f(s) = \sum_{n=1}^{m} \min(f(n)) + h(n) \text{ path from } c \text{ to } s \]  (5)

\( f(s) \) denotes total request processed by the server in the best path from client \( c \) to the server \( s \), which is the sum of all the minimum request processed by the intermediate nodes comparing with adjacent in the path. Heuristic function \( h(n) \) uses the exact number of the request processed along the path from node ‘\( n \)’ to the server. The heuristic \( h(n) \) is admissible if for every node, \( n \), in the path from the client to server satisfies the condition defined as follow.

\[ h(n) \leq h^*(n) \]  (6)

where \( h^*(n) \) is the maximum number of allowed request to be processed by server without congestion and always finds a best path. \( h^*(n) \) is calculated as number of request processed by server when the standard deviation of the server cluster (Eq. 2) to which the server belongs reaches a threshold value of (0–5)%.

While calculating \( f(s) \), which represents total request processed in the best path, the heuristic function \( h(n) \) of all the adjacent router is considered. \( B \) is the factor that defines the number of adjacent routers \( r \) processed calculating minimum of \( f(n) \). The depth \( m \) is the number of routers passed in the best path reaching from client to server in DCN, while processing the request. Total path generated while finding the best path from client to server is calculated by using Eq. 7 [23] given as follows:

\[ B^m + 2B^{m-1} + 3B^{m-2} + \ldots + mB \]  (7)

Factoring out \( B^m \) from Eq. 7, it becomes

\[ B^m(1B^0 + 2B^{-1} + 3B^{-2} + \ldots + mB^{1-m}) \]  (8)

A series is the sum of the terms of an infinite sequence of numbers that is convergent if the sequence of its partial sums tends to a limit; hence, Eq. 8 becomes 9.

\[ B^m(\sum_{i=1}^{m} iB^{1-i}) = B^m(\sum_{i=1}^{m} i(B^{-1})^{i-1}) \leq B^m(\sum_{i=0}^{\infty} (iB^{-1})^{i-1}) \]  (9)

Geometric progression for \(|X|<1\) is defined as follows.

\[ \sum_{i=0}^{\infty} iX^i = \frac{1}{(1-X)^2} \]  (10)

Applying Eq. 10 in the convergent series 9, it becomes Eq. 11.

\[ B^m \frac{1}{(1-B^{-1})^2} = B^m \frac{1^2}{(1-B^{-1})^2} = B^m \left( \frac{1}{(1-B^{-1})} \right)^2 \in O(B^m) \]  (12)

Time complexity of the heuristic best path finding algorithm is of order \( O(B^m) \).

The above Theorem1 follows the best path in a short time based on a minimum value of \( f(n) \), which includes the heuristic \( h(n) \). The combination of iteration and heuristic search calculates the shortest path to reach the server, and the algorithm maintains two lists such as Start and End. The Start list contains list of nodes not processed completely and arranges based on f value and End list. The list maintains the node list according to the expansion not in the optimal path. For example, consider a node from Start list with min ‘f’ value, in case of tie, get the node with less ‘g’ value as the current node, and expansion nodes are called as children. If the child node is not in Start/End list, then calculate the number of request in the path reaching child node through current node, i.e., new ‘g’ value of the child node. In case, if the child node present in either of the list, finds the minimum number of request in the path reaching child node.
Moreover, calculate the ‘f’ value based on the lowest number of requests from child node to server. If ‘f’ value is less than the child node, move to Start list for further processing. MRBS algorithm efficiently searches the best path until the Goal Node is reached, which is optimum server selected in the server selection module.

5 Simulation Environment

System Configuration: MRBS algorithm is implemented in virtual environment setup of the physical machine with i5 Intel processor and 16 GB RAM. A virtual machine is created using oracle virtual box 6.0 with 4 GB RAM and 10 GB hard disk. In VM ubuntu14.04, operating system is installed, and DevStack, a free and open-source cloud platform which enables the system administrator to monitor network resources, is installed.

Algorithm Implementation:
In the simulation environment for implementing the proposed MRBS algorithm, mininet4.0 is used to create a virtual network of DCN topology using python3.7 script. The DCN topology contains multiple virtual client nodes connected to the virtual switch, which are executed by Open vSwitch and web server to process the HTTP requests. SDN controller Floodlight which is open source is used and works with the principles of OpenFlow protocol mainly to coordinate the traffic flows in an SDN environment. The MRBS algorithm is configured in the controller, which consists of two major components: best server selection and best path selection. The requests are forwarded along the best path with minimum transmission cost and the route which already processed lesser number of requests.

The effect of sudden message spikes leads to the traffic overload and bandwidth consumption in the network that can be solved through MRBS algorithm based on prediction of sudden spikes and bandwidth utilisation. The performance evaluation of the proposed MRBS algorithm under different message spikes is covered in the next section.
The traffic generation and message frequency for load balancing are tested in two scenarios through MRBS algorithm. In the first scenario, the clients send HTTP request resources to HTTP servers. iPerf is an open-source tool that allows generating TCP, and UDP traffic/load between two hosts is also used to measure the maximum network bandwidth (throughput) between a server and a client. While sending TCP packets, bandwidth and packet count can be set. The rate of request varied from 5 to 30 Mbps. In the second scenario, the clients send UDP requests to receive files of different sizes and types. iPerf can be used to measure jitter between a server and a client. The static flows are then pushed into each switch of the selected best path. Information such as In-Port, Out-Port, Source IP, Destination IP, Source MAC, Destination MAC is fed to the flows. Load balancing becomes dynamic as we keep the algorithm running for continuous update of the information.

Figure 2 depicts throughput measures for the data, reaches the destination server under different rate of request and period of time. The proposed MRBS algorithm manages the data through routing in the best path based on response time (ms) in Eq. 2 as discussed in Section IV. During the web traffic data flow, slight changes in the throughput are seen in the proposed algorithm due to variation in server configuration. The proposed algorithms manage the response time for different configurations of server. The average throughput of server cluster is 90% in the proposed MRBS algorithm comparing with other algorithms such as ECMP and MPTCP, the proposed algorithm increases by 8–17%.

Figure 3 shows the average bandwidth utilisation, which is the ratio between the actual bandwidth utilised from the total bandwidth under a different rate of request which specifies number of flows. As the traffic rate increases, bandwidth utilisation also creeps up. The average bandwidth utilisation is 63% in the proposed MRBS algorithm comparing with other algorithms, the proposed algorithm increases by 14–20%. The sudden message spikes and different message frequencies create unusual bandwidth usage in the level ranging from 150–400Gbps and overloading of server. Furthermore, MRBS predicts the response time (ms) on the above scenario based on the load and bandwidth using correlation analysis. When a sudden hike in bandwidth occurs due

Table 5  Avg. Response Time of Controller (ms)

| Algorithm         | DCN architectures |
|--------------------|------------------|
|                    | Fat-tree | Jellyfish | Leaf-spine |
| OPTIMAL [25]       | 0.27     | 0.25     | 0.23       |
| (No specific flow size of loads) |          |          |            |
| GREEDY [25]        | 0.22     | 0.26     | 0.24       |
| (No specific flow size of loads) |          |          |            |
| MRBS (Proposed)    | 0.21     | 0.22     | 0.18       |
| (with sudden message spikes) |          |          |            |

Bold values taken for comparison with proposed algorithm

Fig. 5  Average delay of controller during web traffic

Fig. 6  Standard deviation of loads of 10 g servers due to data mining traffic

Table 6  Standard deviation of load of servers

| Algorithm         | DCN architectures |
|--------------------|------------------|
|                    | Fat-tree | Jellyfish | Leaf-spine |
| ECMP [24]          | 24       | 19       | 20         |
| (No specific flow size of loads) |          |          |            |
| MTSS [24]          | 23       | 17       | 19         |
| (No specific flow size of loads) |          |          |            |
| MRBS (Proposed)    | 13       | 10       | 11         |
| (with specific flow size of 1 kb, 10 kb, and 10 mb of loads) |          |          |            |

Bold values taken for comparison with proposed algorithm

5.1 MRBS Algorithm Performance during Message Frequency

The traffic generation and message frequency for load balancing are tested in two scenarios through MRBS algorithm. In the first scenario, the clients send HTTP request resources to HTTP servers. iPerf is an open-source tool that allows generating TCP, and UDP traffic/load between two hosts is also used to measure the maximum network bandwidth (throughput) between a server and a client. While sending TCP packets, bandwidth and packet count can be set. The rate of request varied from 5 to 30 Mbps. In the second scenario, the clients send UDP requests to receive files of different sizes and types. iPerf can be used to measure jitter between a server and a client. The static flows are then pushed into each switch of the selected best path. Information such as In-Port, Out-Port, Source IP, Destination IP, Source MAC, Destination MAC is fed to the flows. Load balancing becomes dynamic as we keep the algorithm running for continuous update of the information.

Figure 2 depicts throughput measures for the data, reaches the destination server under different rate of request and period of time. The proposed MRBS algorithm manages the
to message spikes, the proposed MRBS algorithm is able to stabilise the bandwidth that improves the throughput, based on alternative routing traffic with the minimal response time.

Table 4 shows the average bandwidth utilisation of servers for different implementations of DCN architectures with ECMP, MTSS, and proposed MRBS algorithm. In ECMP and MTSS, no specific sudden spikes are addressed and it takes time to streamline bandwidth usage in case of a sudden rise in utilisation due to message spikes. MRBS effectively addresses the bandwidth management by altering traffic based on response time.

Response time of servers includes the time taken by request to reach the server and the time taken by the server to process the request. Even during worst case scenarios that consumes more bandwidth as the load is balanced between the servers by MRBS algorithm, there is slight variation in the response time and it is stabilised. Figure 4 shows the comparison of average response time of servers in the sequence of time interval, and the average response time of the 10 g servers using MRBS algorithm is 0.183 ms.

Table 5 shows the average response time of servers for different implementations of DCN architectures with ECMP, MTSS, and proposed MRBS algorithm. In ECMP and MTSS, no specific sudden spikes are addressed; MRBS stabilises response time by re-routing the traffic once it finds sudden hike in bandwidth. ECMP and MTSS take time to bring down the response time in case of sudden rise in bandwidth due to message spikes.

When different load balancing algorithms are executed, the number of requests transmitted and bandwidth utilisation are not the same in a link. Delay time of the controller is the time taken by controller to find the path and forward the request to the server. As the rate of transmission increases, the delay time of ECMP algorithm increases steadily, MPTCP algorithm has a sharp rise in the middle and manages to stabilise delay time as it finds optimal path. Figure 5 depicts that the average delay time of controller in MRBS algorithm is 23.5 µs which is 19—45% less compared with the other two algorithms.

5.2 MRBS Performance for Different Traffic Patterns

Figure 6 shows the standard deviation of loads in 10 g server, and the proposed algorithm MRBS performance is evaluated. The proposed MRBS load balancing algorithm performs load balancing dynamically according to the traffic network and avoids overloading of individual transmission link and improves network performance. Average standard deviation of proposed MRBS algorithm is 11.6 comparing the other two algorithms such as ECMP, MPTCP which is 20.2 and 19.2, respectively. The specific load such as web traffic and data mining load is more balanced in all the servers in the cluster so the average standard deviation is minimum in the proposed MRBS algorithm.

Table 6 average server utilisation in different types of DCN architectures DCN architectures. JellyFish performs better than fat-tree and leaf-spine architectures as its average throughput is higher than fat-tree.

Figure 7 shows the bandwidth utilisation of the 10 g servers for existing load balancing algorithms such as Round Robin(RR) and Weighted Round Robin(WRR) and proposed MRBS algorithm by varying the different types of traffic such as web and data mining traffic. During the utilisation of higher bandwidth by the 10 g servers, the proposed algorithm selects the alternate server based on the response time and avoids congestion during data mining and web traffic.

However, the proposed algorithm MRBS shows better sever utilisation of 83% when compared to the traditional algorithms Round Robin(RR) and Weighted Round Robin(WRR) which are 58 and 75%, respectively.

Table 7 shows the average server utilisation in different types of DCN architectures. MRBS algorithm outperforms than the existing load balancing algorithms due to congestion avoidance and stabilises the bandwidth to improve the throughput based on alternative routing traffic with minimal response time.

In JellyFish architecture, existing load balancing algorithms work better as it supports more servers than fat-tree architecture and the servers are 2% greater than leaf-spine architecture.
6 Conclusion

Load balancing algorithm controls and coordinates the distribution of workloads across the server in datacenters for efficient use of network resources and focuses the problem of traffic scheduling and congestion control. The routing and server selection is performed through correlation of parameters such as response time, load, and bandwidth during message spikes and unpredicted traffic pattern conditions in DCN. In this paper, we proposed an MRBS algorithm that detects the various load conditions including web and data mining traffic and senses traffic patterns such as message spikes and different message frequencies through bandwidth monitoring. After the detection of traffic and load, the proposed MRBS algorithm plays a vital role in forwarding the request to the servers having least number of connections and through the least cost path based on minimum link cost and less traffic flow. The evaluation results of the MRBS algorithm using mininet and Floodlight controller prove that it effectively improves bandwidth utilisation, reduces delay, and balances the loads of the servers in server cluster. In future, the load balancing and traffic-based decisions through MRBS algorithm can be evaluated for green cloud computing.

Acknowledgements We would like to thank Mr. Sarfarz N Gudumian, Senior Performance Architect, Mphasis Corp, Chicago, USA for his valuable comments and suggestions that helped us significantly improve the quality of this paper.

Funding The authors received no specific funding for this study.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Greenberg, A.; Hamilton, J.; Maltz, D.A.; Patel, P.: The cost of a cloud: Research problems in data center networks. ACM SIGCOMM Comput. Commun. 39, 68–73 (2008). https://doi.org/10.1145/1496091.1496103
2. Kreutz, D.; Ramos, F.M.V.; Verissimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S.: Software-defined networking: a comprehensive survey. Proc. IEEE. 103(1), 14–76 (2015). https://doi.org/10.1109/JPROC.2014.2371999
3. Yu, L.; Chen, L.; Cai, Z.; Shen, H.; Liang, Y.; Pan, Y.; Stochastic load balancing for virtual resource management in datacenters. IEEE Trans. Cloud Comput. 8(2), 459–472 (2016). https://doi.org/10.1109/TCC.2016.2525984
4. Greenberg, A.; Hamilton, J. R.; Jain, N.; Kandula, S.; Kim, C.; Lahiri, P.; Maltz, D. A.; Patel, P.; Sengupta, S.: VI2: a scalable and flexible data center network. In: Proceedings of the ACM SIGCOMM Conference on Data Communication, pp. 51–62 (2009). https://doi.org/10.1145/1592568.1592576
5. Raiciu, C.; Barre, S.; Pluntke, C.; Greenhalgh, A.; Handley, M.: Improving datacenter performance and robustness with multipath TCP. In: Proceedings of the ACM Conference on SIGCOMM, pp. 266–277 (2011). https://doi.org/10.1145/208436.208467
6. Alizadeh, M.; Edsall, T.; Dharmapurikar, S.; Vaidyanathan, R.; Chu, K.; Fingerhut, A.; Matus, F.; Pan, R.; Yadav, N.; Varghese, N. G.: CONGA: Distributed congestion-aware load balancing for datacenters. Proceedings of the ACM Conference on SIGCOMM, vol. 44, no. 4, pp. 266–277 (2014). https://doi.org/10.1145/2740070.2626316
7. Vanini, E.; Pan, R.; Alizadeh, M.; Taheri, P.; Edsall, T.: Let it flow resilient asymmetric load balancing with flowlet switching. In: Proceedings of the NSDI, pp. 407–420 (2017)
8. Yong, W.; Xiaoling, T.; Qian, H.; Yuwen, K.: A dynamic load balancing method of cloud-center based on SDN. China Communication. 13(2), 130–137 (2016). https://doi.org/10.1109/CC.2016.7405731
9. Kim, H.-S.; Kim, H.; Paek, J.: S Bahk Load balancing under heavy traffic in RPL routing protocol for low power and lossy networks. IEEE Trans. Mob. Comput. 16(4), 964–979 (2016). https://doi.org/10.1109/TMC.2016.25851
10. Wang, Y.-C.; You, S.-Yu.: Load balance and overhead reduction in sdn-based data center networks. IEEE Trans. Netw. Serv. Manage. 15(4), 1422–1434 (2018). https://doi.org/10.1109/TNSM.2018.2872054
11. Tang, F.; Yang, L.T.; Tang, C.; Li, J.; Guo, M.: A dynamical and load-balanced flow scheduling approach for big data centers in clouds. IEEE Trans. Cloud Comput. 6(4), 915–928 (2018). https://doi.org/10.1109/TCC.2016.2543722
12. Cui, J.; Lu, Q.; Zhong, H.; Tian, M.; Liu, L.: A load-balancing mechanism for distributed sdn control plane using response time. IEEE Trans. Netw. Serv. Manage. 15(4), 1197–1206 (2018). https://doi.org/10.1109/TNSM.2018.2876369
13. Baig, S.-u-R.; Iqbal, W.; Berral, J.L.; Erradi, A.; Carrera, D.: Adaptive prediction models for data center resources utilisation estimation. IEEE Trans. Netw. Serv. Manage. 16(4), 1681–1693 (2019). https://doi.org/10.1109/TNSM.2019.2932340
14. Alvarez-Horcajo, J.; Lopez-Pajares, D.; Martinez-Yelmo, I.; Carral, J.A.; Arco, J.M.: Improving multipath routing of TCP flows by network exploration. IEEE Access. 7, 13608–13621 (2019). https://doi.org/10.1109/ACCESS.2019.2893412
15. Khalil, M.I.K.; Ahmad, I.; Almazrooi, A.A.: Energy efficient indistinguishable workload distribution in geographically distributed data centers. IEEE Access, Special Section Mobile Edge Compute Mobile Cloud Compute Addressing Heterogeneity Energy Issues Comput and Netw. 7, 82672–82680 (2019)
16. Park, M.; Sohn, S.; Kwon, K.; Kwon, T.T.: MaxPass: credit-based multipath transmission for load balancing in data centers. J. Commun. Networks 21(6), 558–568 (2019)
17. Cheng, Y.; Jia, N.: NAMP: Network-aware multipathing in software-defined data center networks. IEEE/ACM Trans. Netw. 28(2), 846–859 (2020). https://doi.org/10.1109/TNET.2020.2971587
18. Jamali, S.; Badirzadeh, A.; Siapoush, M.S.: On the use of the genetic programming for balanced load distribution in software-defined networks. Digital Commun. Netw. 5(4), 288–296 (2019). https://doi.org/10.1016/j.dcan.2019.10.002
19. Bi, Y.; Han, G.; Lin, C.; Peng, Y.; Pu, H.; Jia, Y.: Intelligent quality of service aware traffic forwarding for software-defined networking/open shortest path first hybrid industrial internet. IEEE Trans. Industr. Inf. 16(2), 1395–1405 (2020). https://doi.org/10.1109/TII.2019.2946045
20. Dong, E.; Fu, X.; Xu, M.: Low-cost datacenter load balancing with multipath transport and top-of-rack switches. IEEE Transac. Parallel Distrib Syst. 31(10), 2232–2247 (2020). https://doi.org/10.1109/TPDS.2020.2989441
21. Yekkehkhany, A.; Nagi, R.: Blind GB-PANDAS: a blind throughput-optimal load balancing algorithm for affinity scheduling. IEEE/ACM Trans. Netw. 28(3), 1199–1212 (2020). https://doi.org/10.1109/TNET.2020.2978195

22. Noormohammadpour, M.; Raghavendra, C.: Datacenter traffic control: understanding techniques and trade-offs. IEEE Commun. Surv. Tutorials. 20(2), 1492–1525 (2018). https://doi.org/10.1109/COMST.2017.2782753

23. Korf, R: Analyzing the performance of pattern database heuristics. In: Proceedings of the National Conference on Artificial Intelligence, pp. 1164–1170 (2007)

24. Xiaolong, X.; Yun, C.; Liuyun, H.; Anup, K.: MTSS: multi-path traffic scheduling mechanism based on SDN. J. Syst. Eng. Electron. 30(5), 974–984 (2019). https://doi.org/10.21629/JSEE.2019.05.14

25. Zhang, S.; Lan, J.; Sun, P.; Jiang, Y.: Online load balancing for distributed control plane in software-defined data center network. IEEE access. 6, 18184–18191 (2018). https://doi.org/10.1109/ACCESS.2018.2820148

26. Subramanian, R.; Manoranjitham, T.: Dynamic scheduling for traffic management and load balancing using sdn. Int. J. Cont. Theory Appl. 9(2), 919–925 (2016)