Dynamic Load Balancing for Software-Defined Data Center Networks

Yun Chen, Weihong Chen, Yao Hu, Lianming Zhang(✉), and Yehua Wei

Key Laboratory of Internet of Things Technology and Application,
College of Physics and Information Science, Hunan Normal University, Changsha 410081, China
611cy@sina.com, 1018450155@qq.com, huyao9403@163.com,
zlm@hunnu.edu.cn, yehuahn@163.com

Abstract. In recent years, along with the increasing demand for cloud services, the network traffic is increased inside data center networks (DCNs). The inherent defects of TCP/IP network architecture have hindered the development of network technologies for a long time, and the software-defined network (SDN) has recently gained unprecedented attention from industry and research communities, and it has completely overturned the existing network architecture. In this paper, we have proposed a new network frame for software-defined data center network (SDDCN), and developed a dynamic schedule strategy of the network traffic by calculating available path coefficient of the SDDCN, then presented a schedule algorithm for the dynamic load balancing (SDLB) based on the path coefficient of network available bandwidth in real-time, and developed the components of the Ryu controller of the SDDCN, which is responsible for running the SDLB algorithm. Based on Mininet platform, the experimental results show that the SDLB algorithm has better performance of dynamic load balancing in the SDDCN, which has provided an effective solution for the load balancing of the existing DCNs.

Keywords: Software-defined data center network · Load balancing · Path coefficient

1 Introduction

With the rapid development of cloud computing, there are more and more data centers, which are pools of computational, storage and network resources using interconnection networks [1, 2], such as Internet. As the growing demand for the cloud services, the network traffic inside a data center network (DCN) has been increased, and it makes further demands on DCNs [3]. Today's data centers are constrained by the interconnection network [4]. For example, how to maximize the use of limited network resources has become an urgent problem needed to be solved in the field of the Internet, which uses TCP/IP architecture. One of the most commonly used methods is the load balancing technology. However, some inherent defects of the Internet architecture have hindered the development of network technology. At the same time, the problem of low bandwidth

S. Wang and A. Zhou (Eds.): CollaborateCom 2016, LNICST 201, pp. 286–301, 2017.
DOI: 10.1007/978-3-319-59288-6_26

utilization also exists for a long time, which hinders the development of load balancing technology in the existing networks [5].

Against this background, the three-layer architecture of DCNs based on traditional interconnection networks is being challenged. Flat, virtualized, programmable and definable networks become the new trend in the DCNs architecture. Software-defined network (SDN) is an architecture which decouples network control and forwarding functions, and enabling to be directly programmable, dynamic, manageable, cost-effective and adaptable, and seeking to be suitable for the high-bandwidth, dynamic nature of today's applications, and providing a solution to the above problems [6–8]. For example, applying the software-defined idea can overturn the existing network architecture into DCNs, and can be better to solve the technical problem of load balancing in the distributed network architecture [9]. Paper [10] presents a load balancer for OpenFlow based the DCNs, and implements a dynamic routing algorithm in the load balancer. This paper extends these existing works.

The technical aim of the paper is to achieve dynamic load balancing for DCNs using software-defined method. In comparison with the existing literature on similar work, this paper has the following major contributions:

- To present a new software-defined frame for DCNs with fat-tree structure.
- To present a schedule algorithm for dynamic load balancing (SDLB) of software-defined data center networks (SDDCNs) based on the path coefficient of network available bandwidth in real-time.
- To develop the components of the Ryu controller for the SDLB algorithm, and to evaluate the performance of the SDLB algorithm under the simulation environment based on the DCNs with the fat-tree topology.

In the following sections, we first discuss existing research works on the load balancing technology for the DCNs and SDN in Sect. 2. Section 3 presents a SDLB algorithm based on the path coefficient of network available bandwidth in real-time. Section 4 presents the method for developing the Ryu controller component to running the SDLB algorithm. The experimental results and performance analysis of the SDLB algorithm are presented in Sect. 4. The paper is concluded in Sect. 5.

2 Related Works

With the popularization of the concept of SDN and the rapid development of OpenFlow technology, the researchers have gradually deepened the network innovation of traffic engineering, network security and load balancing in OpenFlow networks. The top–ranking network equipment manufacturers have introduced a number of SDN devices with markedly different characteristics, and the global deployment of SDN has gradually expanded from a small range.

OpenFlow-based SDN technology applied to data center deployment has become a hot topic of discussion on the SDN. A number of vendors, service providers and companies got into this field. For example, VMware is actively developing the software-defined data center approach [11]. Avaya, H3C, Big Switch Networks, Cisco, Dell and IBM are

developing components and standards that enable the software-defined data center [12]. CloudGenix, VeloCloud and Viptela have already deployed a SDN in the data center and are now looking to expand those benefits to wide area networks [13], and so on.

Paper [14] has firstly proposed the introduction of OpenFlow technology and the NOX controller is used for achieving the mechanism of effective addressing and routing in the DCNs, such as PortLand [15] and VL2. ElasticTree [16] is mainly based on energy saving and emission reduction, and achieves energy saving according to the rules of dynamic adjustment of network equipment. Paper [17] has designed a load balancing solution for DCNs. The controller obtains the real-time traffic information of switches, and calculates the costs using simulated annealing algorithm, and makes the load balancing decision, however, the solution is mainly for the specific DCN, such as PortLand [15].

The separation of forwarding element and control element based on OpenFlow has made the network innovation become a hot research direction, especially in traffic management, load balancing, and dynamic routing and so on. OpenTM gets the traffic information in the network node from the controller, and calculates the network load of each node, thus builds the traffic matrix for the whole network [18]. Paper [19] has applied the OpenFlow technology to the scalable streaming media application framework supporting the QoS, and the dynamic routing function of non-shortest path is used to schedule the video data stream. Paper [20] has proposed a server load balancing controller application to improve the QoS for video streaming over single operator OpenFlow network in case of server overload. Server load balance is achieved by continuously monitoring the load of each server, and dynamically redirecting activities or new service requests. In such a manner, the end user experiences the lowest delay and distortion when one or more servers are overloaded.

Paper [21] has described a dynamic load balancing method of server cluster based on OpenFlow to solve the problem of how to take load balancing into network virtualization data center. But there is no consideration of how the different services to make load balancing on the same path.

Paper [22] has presented a load balancing solution based on SDN for their campus network. Their aim is to cut costs and improve flexibility of the network. In this solution, their application shared elements with the distributed control plan of Kandoo [23], and there were a root controller and multiple local controllers. When an event comes, if it is not required to get overall information of the network, the local controller will handle it. Otherwise, it will be forwarded to the root controller, which keeps network-wide state and can process it best. However, the proposed scheme is mainly aimed at the load balancing in the campus network, and does not consider the scene of the data center.

Paper [10] has presented a dynamic load balancing algorithm based on the fat-tree topology for the DCN, and this algorithm calculates the shortest path for the network traffic by using of the network available bandwidth in real-time. Traffic from the source node to the top node is needed to visit in the fat-tree topology, and it determines the destination node of the downward transfer. For any traffic generated from the network, this algorithm firstly determines the source address, the destination address, and determines the highest layer node which the traffic needs to reach. It has realized dynamic path selection strategy based on the characteristic of the fat-tree topology, but only used

the greedy algorithm for the real-time available bandwidth of the single hop path, then got the traffic adjusting decisions. This cannot consider the status of other link on the path, and may lead to the over load of part link on that path, causing the network congestion.

In this paper, we have proposed a new frame for the software-defined data center network (SDDCN), and discussed the link bottleneck problems caused by disequilibrium distribution of the available bandwidth of all the links on the path, and presented a schedule algorithm for the dynamic load balancing based on the path coefficient of network available bandwidth in real-time, and developed the components of the Ryu controller of the SDDCN which is responsible for running the SDLB algorithm. The presented algorithm has solved the local congestion problem which the DLB algorithm can easily lead to, and also provided a more efficient scheduling scheme for the load balancing of the fat-tree network.

3 Dynamic Load Balancing for the SDDCN

In this section, a new frame for the SDDCN is proposed based on the fat-tree topology in the existing DCNs and using software-defined method, and has designed a SDLB algorithm based on the path coefficient of network available bandwidth in real-time.

3.1 A Network Frame for the SDDCN

In order to realize the flexible control of network traffic, and to maximize the use of limited bandwidth and other resources in the DCNs, this paper presents a new frame for SDDCNs using the software-defined method. In comparison with the existing DCNs, the SDDCN has the following features:

- All routers and switches are only responsible for forwarding traffic, while control functions are completed by a special SDN controller.
- The network currently built and put into use is a tree topology generally composed of two or three layers of routers and switches.

Figure 1 presents a network frame for the SDDCN, and it is different from the structure of the DCN based the fat-tree topology in [24]. In this SDDCN, all switches, including core layer switches (s_{17}, s_{18}, s_{19} and s_{20}), convergence layer switches (s_9, s_{10}, s_{11}, s_{12}, s_{13}, s_{14}, s_{15} and s_{16}), and access layer (edge) switches (s_1, s_2, s_3, s_4, s_5, s_6, s_7 and s_8), are connected with a special SDN controller such as the Ryu. These switches are only responsible for forwarding traffic and are controlled by the SDN controller. For three-layer fat-tree topology with k variables, which contains $(k/2)^2$ core switches and k arrays; and each array is composed of k switches with k ports, among which the number of the convergence layer switches and the number of edge layer switches are the same; each edge switch directly is connected to the $k/2$ hosts.

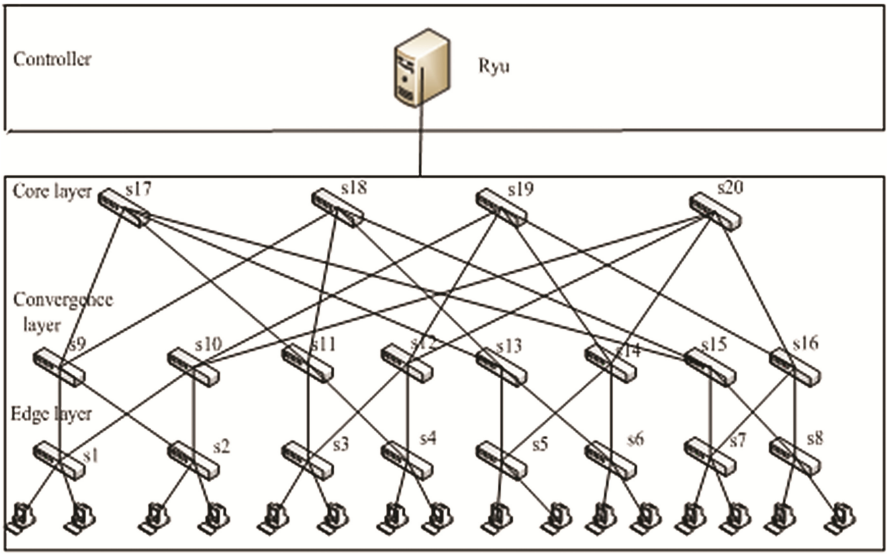

Fig. 1. A frame for the SDDCN

In general, we set the edge switches connecting to the host as the starting points of the paths. One of the important features of the fat-tree topology is that the traffic between all the hosts and the same edge switch in the same array only needs to visit the edge switches as the highest; the traffic between all the hosts and different edge switches within the same array only needs to visit the convergence layer switch as the highest; for the traffic between different arrays starting from the edge switches only needs to upward transfer to the highest core switches, and the downward path of the traffic is determined thereof. In this paper, the SDLB algorithm is designed based on the fat-tree topology of the SDDCN.

3.2 Detail Method

The SDLB algorithm designed in this paper needs to create and maintain three path tables: upward path table (UPT), downward path table (DPT), link load table (LLT) inside the SDN controller, thus to save all the up and down paths and the load information of each link, then the SDN controller accordingly calculates out the best path for the traffic transfer.

The structure of UPT is given in Table 1: Edge represents the start edge switch of the traffic, and each of the edge switches (s_1, s_2, …, s_8) can be uploaded to one of the core layer switches through several paths; Core and Aggregation represent the high level switches that the data flow can reach, and A_1, A_2 respectively represents the first or second set of the convergence layer switches of each array; s_{17}, s_{18}, …, s_{20} respectively represents the core layer switches; each cell represents the path of the traffic to the top switch. Through the load balancing decision of the UPT, the path information is

provided, and the current minimum available bandwidth in each path for upward and downward is calculated.

Table 1. UPT

Top \ Edge / Path	Aggregation		Core			
	A_1	A_2	s_{17}	s_{18}	s_{19}	s_{20}
s_1	{1,9}	{1,10}	{1,9,17}	{1,9,18}	{1,10,19}	{1,10,20}
s_2	{2,9}	{2,10}	{2,9,17}	{2,9,18}	{2,10,19}	{2,10,20}
s_3	{3,11}	{3,12}	{3,11,17}	{3,11,18}	{3,12,19}	{3,12,20}
s_4	{4,11}	{4,12}	{4,11,17}	{4,11,18}	{4,12,19}	{4,12,20}
s_5	{5,13}	{5,14}	{5,13,17}	{5,13,18}	{5,14,19}	{5,14,20}
s_6	{6,13}	{6,14}	{6,13,17}	{6,13,18}	{6,14,19}	{6,14,20}
s_7	{7,15}	{7,16}	{7,15,17}	{7,15,18}	{7,16,19}	{7,16,20}
s_8	{8,15}	{8,16}	{8,15,17}	{8,15,18}	{8,16,19}	{8,16,20}

The structure of DPT is shown in Table 2: contrast to UPT, in DPT, the Aggregation and Core represent the starting point of the downward traffic flow, which is the highest level node for upward flow to reach, and the node can be obtained by UPT; Edge represents the path needed to reach the destination; once the highest level nodes are determined, the downward flow transferring path is determined, and the load balancing algorithm determines the downward paths to reach the destination nodes based on the highest level nodes.

Table 2. DPT

Top \ Edge / Path	**Aggregation**		**Core**			
	A_1	A_2	s_{17}	s_{18}	s_{19}	s_{20}
s_1	{9,1}	{10,1}	{17,9,1}	{18,9,1}	{19,10,1}	{20,10,1}
s_2	{9,2}	{10,2}	{17,9,2}	{18,9,2}	{19,10,2}	{20,10,2}
s_3	{11,3}	{12,3}	{17,11,3}	{18,11,3}	{19,12,3}	{20,12,3}
s_4	{11,4}	{12,4}	{17,11,4}	{18,11,4}	{19,12,4}	{20,12,4}
s_5	{13,5}	{14,5}	{17,13,5}	{18,13,5}	{19,14,5}	{20,14,5}
s_6	{13,6}	{14,6}	{17,13,6}	{18,13,6}	{19,14,6}	{20,14,6}
s_7	{15,7}	{16,7}	{17,15,7}	{18,15,7}	{19,16,7}	{20,16,7}
s_8	{15,8}	{16,8}	{17,15,8}	{18,15,8}	{19,16,8}	{20,16,8}

As shown in Table 3, each row of LLT represents a link between two adjacent nodes in the topology, which is represented by the starting and end points of the link, i.e., <*src*, *dst*>; each column represents an adjacent time interval. For definiteness and without loss of generality, we take the current available bandwidth of each link in a fixed cycle statistical topology, that is, B_i represents the current available bandwidth of link *i*. In the initialization phase, we set the available bandwidth as the maximum bandwidth of the link, and then calculate and update the LLT based on the port flow statistical information obtained at each cycle.

Table 3. LLT

Time / B_i / Link	$\mathbf{T_0}$	$\mathbf{T_-}$	$\mathbf{T_+}$
$<s_1,s_9>$	1000	520	220
$<s_1,s_{10}>$	1000	395	395
$<s_3,s_{11}>$	1000	355	55
$<s_3,s_{12}>$	1000	545	545
$<s_9,s_{17}>$	1000	415	115
$<s_9,s_{18}>$	1000	600	600
$<s_{10},s_{19}>$	1000	300	300
$<s_{10},s_{20}>$	1000	350	350
$<s_{11},s_{17}>$	1000	450	150
$<s_{11},s_{18}>$	1000	175	175
$<s_{12},s_{19}>$	1000	485	485
$<s_{12},s_{20}>$	1000	220	220
...	...	...	...

3.3 Problem Descriptions

Definition 1: Available bandwidth of the path. For all the possible paths that the traffic flow transfers to the highest layer or from the highest layer to the downward destination switches, the SDN controller will make a statistic on the available bandwidth of all the links inside each path, and through the principles of maximum and minimum to calculate the available bandwidth of that path [25]. We assume the available bandwidth of the k^{th} upward and downward path are respectively Bu_k and Bd_k, and then we have

$$Bu_{\mathrm{k}} = min\{b_0, b_1, \cdots, b_i\} \tag{1}$$

$$Bd_k = min\{B_0, B_1, \cdots, B_i\} \tag{2}$$

where B_i is the current available bandwidth for the i^{th} link in the path, and $min\{\}$ is the minimum value in the sequence.

The size of the available bandwidth of the path can be used as the load state of the local network, which can be used as the basis of the load balancing algorithm. The greater the available bandwidth of the path is, the smaller the load on the path, the more difficult to cause local network congestion for choosing that path to transmit data packets.

Definition 2: Path variation coefficient. Based on the uploaded paths calculated from the source switch and the downward paths calculated according to the destination switch, the complete path is formed. But how to choose the best path for the traffic flow needs to calculate out the path variation coefficients (CV) respectively, that is

$$CV = \frac{\sigma_{AB_k}}{\overline{AB_k}} \times 100\% \tag{3}$$

where AB_k is the available bandwidth of the k^{th} path, and $\overline{AB_k}$ is the average value of the available bandwidth of the upward path and the downward path in the k^{th} path, the calculation formula is as follows:

$$\overline{AB_k} = \frac{Bu_k + Bd_k}{2} \tag{4}$$

Obviously, the smaller the $\overline{AB_k}$ value is, the greater the path variation coefficient is. The greater the available bandwidth variation of the path is, the worse the path transmission performance is, and the more prone to cause network congestion.

Among which, σ_{AB_k} is the standard deviation between the available bandwidths of upward path and the downward path inside the k^{th} path, the calculation formula is as follows:

$$\sigma_{AB_k} = \sqrt{\frac{\left(Bd_k - \overline{AB_k}\right)^2 + \left(Bu_k - \overline{AB_k}\right)^2}{2}} \tag{5}$$

The smaller the σ_{AB_k} value is, the smaller the path variation coefficient is. The smaller the available bandwidth variation of the path is, the better the path transmission performance is, and the less prone to cause network congestion.

3.4 SDLB Algorithm

In order to dynamically select the best path and avoid local congestion in the network, the SDLB algorithm is designed, which is based on the real-time bandwidth of the link, and the pseudo code of the SDLB algorithm is as follows.

Algorithm 1. SDLB Algorithm

```
Input: Flow(srcHost, dstHost),LLT,UPT,DPT
Output: bestPath
1: SelectBestPath(){
2:    srcEdgeSw=getEdgeSwitch(srcHost);
3:    dstEdgeSw=getEdgeSwitch(dstHost);
4:    topLayer=getTopLayer(srcEdge, dstEdge);
5:    if srcEdgeSw==dstEdgeSw then
6:       return srcEdgeSw
7:    end if
8:    upwardPaths=getUpwardPaths(srcEdgeSw, topLayer)
9:    downwardPaths=getDownwardPath(topLayer, dstEdgeSw)
10:   paths=getPaths(upwardPaths, downwardPaths)
11:   bestPath=getBestPath(paths)
12:   return bestPath
13: }
```

According to the structural characteristics of the fat-tree network, the traffic flow only needs to determine the upward transmission path from the source node to the top layer node which is needed to reach, and then the downward transmission path is also determined. On the basis of the source address and destination address, the SDLB algorithm firstly calculates out the highest layer of switches that the traffic flow needs to reach in the shortest path, and provides basis for the calculation of the upward path. Once the highest layer needed to be visited by the traffic flow is determined, the SDLB algorithm can select out all the possible upward paths in the paths based on the source switch and the highest layer needed to reach of the traffic flow, then select the corresponding downward path according to the end of each upward path, and each upward path has only one downward path according to the characteristics of the fat-tree topology. Once all the possible upward path and downward path of the traffic flow has been determined, and formed a complete path, the SDLB algorithm will read the current available bandwidth B_i of all the links inside the link load table for each of the upward paths and its corresponding downward path, and calculate out the available bandwidth of that path based on the maximum and minimum principle, then calculate out the path coefficients of all the complete path through the calculation formula (3), and select the best transfer path for traffic flow accordingly. In the forwarding decision of traffic flow, the SDN controller can monitor the network state in real time and make statistics for the network information, then select the best path according to the SDLB algorithm, and update that path to all the switches of that path to realize scheduling of the traffic flow and achieve the load balancing effects.

For any n node of the fat-tree network, the time complexity of the SDLB algorithm is $O(n^3)$. Different from the case of single hop greedy of the DLB algorithm, the SDLB algorithm needs to maintain the link state information of the network to calculate the path coefficient, which leads to the computational overhead of the SDLB algorithm increasing larger continuously with the expansion of the network. However, the monitoring module of the SDLB algorithm in the initialization phase will complete all topology storage and update, so that the SDN controller can reach a balance between SDLB algorithm performance and the computational overhead.

4 Performance Evaluation and Discussion

In order to analyze the performance of the SDLB algorithm, we have established the experiment system. The SDN controller is to use Ryu, the switches are to use OpenFlow which will be integrated into the Ryu controller [26] to realize three relevant components, including *load_balance*, *load_sbalance* and *simple_monitor*. The component of the *load_balance* will realize the SDLB algorithm, and the component of the *load_sbalance* will realize the DLB algorithm, and the component of the *simple_monitor* will realize the all the basic statistical information needed for the load balancing algorithm. The three components exist under the Ryu controller's application files can communicate by using of the Ryu controller's API and other components in order to obtain the topology information and the load state of all the switches of the current network in time; then the component of the *load_balance* uses the component of the *simple_monitor* to obtain the data update link load table, upward path table and the downward table, and calculate the value of path variation coefficient *CV* to make path decision for the traffic flow and seal that path into *FlowMod* message and updated onto the flow tables of all the switches. Through the development of components, the SDLB algorithm can be realized in the Ryu controller which can be successfully started. The SDN controller can be connected to the network, and the traffic flow in the network can be dynamically scheduled. Then the performance of the SDLB algorithm is checked by comparing with the existing DLB algorithm.

4.1 Testing Environment

In this paper, we use the *Mininet* simulation platform [27] to test the performance of the SDLB algorithm. We generated a fat-tree network with $k = 8$ and connected to the Ryu controller. Two kinds of UDP traffic mode [28, 29] are set up for the fat-tree network to restore the traffic characteristics of the DCN.

(1) Random mode: the data flow randomly starts from one host, and is transferred to another host randomly.
(2) Centralized mode (i, j, k) (P_t, P_a, P_c): the host m sends data flow to the hosts: $m + i$, $m + j$ and $m + k$ at probabilities of P_t, P_a and P_c, and $P_t + P_a + P_c = 1$. Without loss of generality, the setting values are (1, 4, 64) (50%, 30%, 20%).

A host computer running Ryu 3.6 controller, and the other host running the fat-tree simulation environment generated by Mininet 2.1.0, and reintegrate the OVS into the virtualized OpenFlow switch of the OpenvSwitch 2.3.0 and connected to the Ryu controller. In order to evaluate the performance, we assume the maximum bandwidth of the host to the edge layer switch is 0.05 Mbps, and the data stream of different traffic loads is initiated in two modes, and each test time is 50 s.

4.2 Evaluation Parameters

In order to evaluate the performance of the SDLB algorithm, the following performance evaluation parameters are set up.

(1) Average bandwidth utilization ratio ρ. It is referring to in the average value of the ratio between the actual bandwidth and the specified bandwidth obtained by all data flows. The real bandwidth value that each flow can reach is often below the specified bandwidth when transferred, because the network status and performance is not ideal. By comparing the changes of these two, it can carry on the evaluation of the current network status and performance. The calculation formula of the average bandwidth utilization rate ρ is as follows:

$$\rho = \frac{\sum_{i=1}^{n} \frac{B_i}{B_0}}{n} \tag{6}$$

where B_i is the actual bandwidth of the i^{th} flow, B_0 is the specified bandwidth of that flow, and n is the total amount of the flows.

(2) Average transmission delay τ. It is referring to the average value of time spend for all the data flow from the sending end to the receiving end, the flow transmission time is affected by the influence of network status and performance, through comparing the transmission delay of the flow can give effectively judgment on the network real-time performance status. The average transmission time delay calculation formula is as follows:

$$\tau = \frac{\sum_{i=1}^{n} t_{i-receive} - t_{i-send}}{n} \tag{7}$$

where $t_{i-receive}$ is the time of receiving the data flow at the receiving end, and t_{i-send} is the time of sending the data flow at the sending end.

4.3 Results Analysis

In order to verify the performance of the SDLB algorithm, we use the DLB algorithm to carry out the comparison tests, and performed several test on the two algorithms respectively in the centralized flow mode and the random flow mode. Figure 2 shows the relationship between the average bandwidth utilization and the traffic load. Under

different traffic loads in random mode, the average bandwidth utilization of the SDLB algorithm is better than the DLB algorithm. When the traffic load is low, the bandwidth utilization rates of the DLB algorithm and the SDLB algorithm are all close to 1, and network performance is good. With the increase of traffic load, the performance of the two algorithms are all decreased, especially the traffic load is more than 0.5, a downward trend of the two algorithms is more obvious; but the utilization rate of the average bandwidth under the SDLB algorithm scheduling decreased slightly and gently, and was higher than that of the DLB algorithm. This is because the SDLB algorithm fully based on the available bandwidth of each link in the network to calculate the path coefficient of each complete path, and then select the best path for the flow scheduling, which is more comprehensive than the DLB algorithm, and can be better in avoiding the local network congestion and improving the network bandwidth utilization rate.

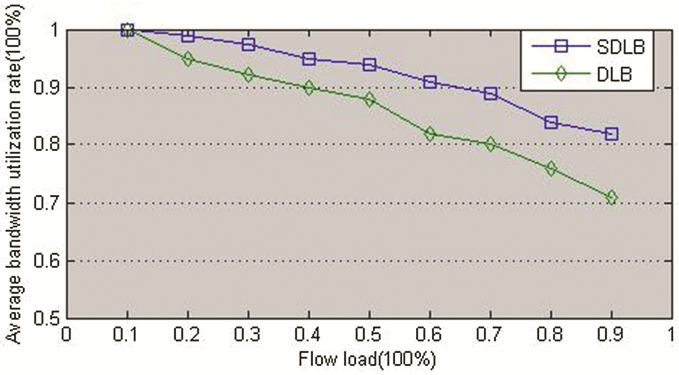

Fig. 2. Average bandwidth utilization in random mode

Figure 3 shows the relationship between the average transmission delay and the traffic load. Under different traffic loads in random mode, the average transmission delay of the SDLB algorithm is better than the DLB algorithm. When the traffic load is low, the available bandwidth of the network is enough, and the average transmission delay of the two algorithms is lower, and with the increase of the traffic load, the average transmission delay of the two algorithms is increasing, especially in the traffic load of more than 0.6, the average transmission delay is obviously high; when the network is close to full load, the network congestion is serious with higher average transmission delay. However, the SDLB algorithm can calculate the path coefficient to select path for the flow according to the actual available bandwidth, and can make a better load balancing decision than the DLB algorithm under the same flow load, and reduce the possibility of congestion in the network, and effectively decrease the average transmission delay in the network.

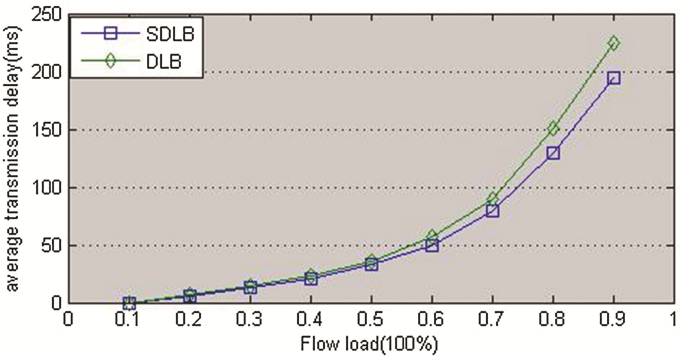

Fig. 3. Average transmission delay in random mode

Figure 4 shows the relationship between the average bandwidth utilization and traffic load under the centralized mode. The equilibrium effect of the SDLB algorithm is close in the random mode and the centralized mode, but the performances of the SDLB algorithm are better than the DLB algorithm in the two modes. In the centralized mode, the average bandwidth utilization rate of the two algorithms decreases with the increase of the traffic load, but the equalization performance of the DLB algorithm decreases more quickly, especially in the traffic load of over 0.3, the decrease of average bandwidth utilization rate of the DLB algorithm is more obvious, this is because the bandwidth of all the link paths are efficient during the low flow load, and the congestion rate is low. Based on simple greedy selection path, the DLB algorithm has increased the possibility of the local link congestion, which also confirms the effectiveness of the SDLB algorithm for solving this problem.

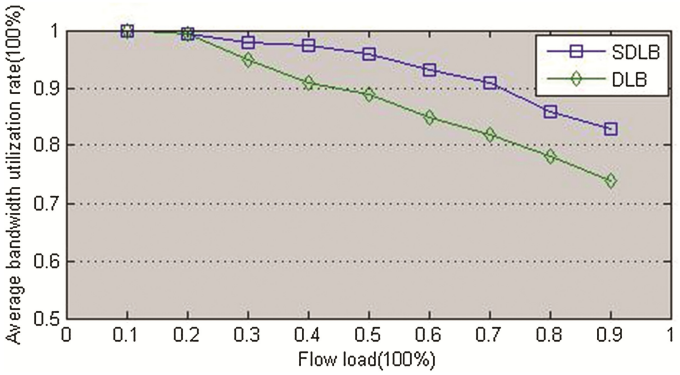

Fig. 4. Average bandwidth utilization in centralized mode

Figure 5 shows the relationship between the average transmission delay and the traffic load under the centralized mode. Quite similar to the situations in the random mode, the average transmission delay of the two algorithms increases with the increase

of the traffic load, and the SDLB algorithm has a better performance than the DLB algorithm.

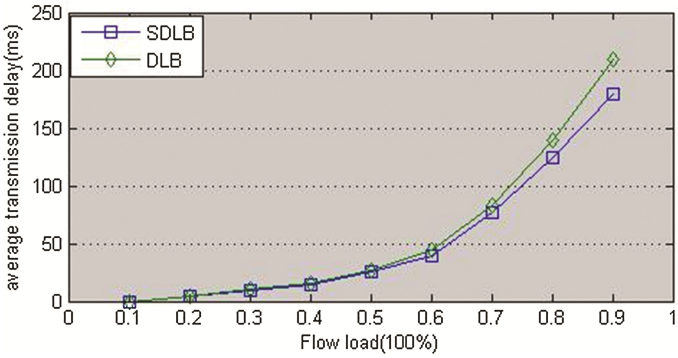

Fig. 5. Average transmission delay in centralized mode

In the meantime, we find that the tendency of the average bandwidth utilization along with flow load appears to be very similar though few difference (see Figs. 2 and 4), and so does the tendency of the average transmission delay along with flow load (see Figs. 3 and 5). This means that the performance of the SDLB algorithm is not affected by traffic mode, and this will show that the algorithm has well stability and applicability.

In summary, the SDLB algorithm has better average bandwidth utilization and average transmission delay in the above two test modes, which reflects the better performance of the network under that algorithm, and verifies the effectiveness of the proposed SDLB algorithm.

5 Conclusion and Future Work

The transfer-control-separation ideology of SDN architecture is one of the most promising technologies in the field of computer networks. In this paper, a new SDDCN architecture is proposed, based on this, a dynamic scheduling load balancing (DSLB) algorithm for the SDDCN based on fat-tree topology is designed. Based on the all possible available bandwidth of the upward paths and the downward paths, the SDLB algorithm has calculated the path coefficient of each complete path, and selected the best path for the traffic flow. The SDLB algorithm can make full use of the network bandwidth, ease the local congestion in the DCN, and provide a better solution for the traffic load balancing in the DCN.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment. This research is supported in part by the grant from the National Natural Science Foundation of China (61572191 and 61402170), the Hunan Provincial Science and Technology Program Project of China (No. 2013FJ4051), and the Hunan Provincial Education Department Scientific Research Fund of China (No. 13B065).

References

1. Assunção, M.D., Calheiros, R.N., Bianchi, S., Netto, M.A.S., Buyya, R.: Big data computing and clouds: trends and future directions. J. Parallel Distrib. Comput. **79–80**, 3–15 (2015)
2. Bilal, K., Khan, S.U., Zhang, L., Li, H., Hayat, K., Madani, S.A., Min-Allah, N., Wang, L., Chen, D., Iqbal, M., Xu, C.Z., Zomaya, A.Y.: Quantitative comparisons of the state-of-the-art data center architectures. Concurrency Comput. Pract. Experience **25**(12), 1771–1783 (2013)
3. Gang, D., Gong, Z., Hong, W.: Characteristics research on modern data center network. J. Comput. Res. Dev. **51**(2), 395–407 (2014)
4. Bilal, K., Khan, S.U., Zomaya, A.Y.: Green data center networks: challenges and opportunities. In: 11th International Conference on Frontiers of Information Technology (FIT 2013), Islamabad, Pakistan, pp. 229–234. IEEE (2013)
5. Pan, J., Paul, S., Jain, R.: A survey of the research on future internet architectures. IEEE Commun. Mag. **49**(7), 26–36 (2011)
6. Software-defined networking definition. https://www.opennetworking.org/sdn-resources/sdn-definition
7. Kim, H., Feamster, N.: Improving network management with software defined networking. IEEE Commun. Mag. **51**(2), 114–119 (2013)
8. Zuo, Q.Y., Chen, M., Zhao, G.S., Xing, C.Y., Zhang, G.M., Jiang, P.C.: Research on OpenFlow-based SDN technologies. J. Softw. **24**(5), 1078–1097 (2013)
9. Riforgiate, S., Sydney, A.: The evaluation of software defined networking for communication and control of cyber physical systems. Dissertations & Theses – Gradworks (2013)
10. Yu, L., Pan, D.: OpenFlow based load balancing for fat-tree networks with multipath support. In: 12th IEEE International Conference on Communications (ICC 2013), Budapest, Hungary (2013)
11. Davidson, E.A.: The software-defined-data-center: concept or reality? http://tinyurl.com/omhmbfv
12. Knorr, E.: OpenDaylight: a big step toward the software-defined data center. http://www.infoworld.com/article/2614152
13. Burt, J.: Startup CloudGenix aims to bring SDN to WAN. http://www.eweek.com/networking/startup-cloudgenix-aims-to-bring-sdn-to-wan.html
14. Tavakoli, A., Casado, M., Koponen, T., Shenker, S.: Applying nox to the datacenter. In: The Eighth ACM Workshop on Hot Topics in Networks (HotNets-VIII), New York City, NY, October 2009
15. Niranjan Mysore, R., Pamboris, A., Farrington, N., Huang, N., Miri, P., Radhakrishnan, S., Subramanya, V., Vahdat, A.: Portland: a scalable fault-tolerant layer 2 data center network fabric. ACM Sigcomm Comput. Commun. Rev. **39**(4), 39–50 (2009)
16. Heller, B., Seetharaman, S., Mahadevan, P., Yiakoumis, Y., Sharma, P., Banerjee, S., McKeown, N.: Elastictree: saving energy in data center networks. In: 7th USENIX Conference on Networked Systems Design and Implementation (NSDI 2010), San Jose, CA, USA, pp. 249–264. USENIX Association Berkeley (2010)

17. Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N., Vahdat, A.: Hedera: dynamic flow scheduling for data center networks. In: 7th USENIX Conference on Networked Systems Design and Implementation (NSDI 2010), San Jose, CA, USA, p. 19. USENIX Association Berkeley (2010)
18. Tootoonchian, A., Ghobadi, M., Ganjali, Y.: OpenTM: traffic matrix estimator for OpenFlow networks. In: Passive and Active Measurement, International Conference (PAM 2010), Zurich, Switzerland, pp. 201–210, 7–9 April 2010
19. Egilmez, H.E., Gorkemli, B., Tekalp, A.M., Civanlar, S.: Scalable video streaming over OpenFlow networks: an optimization framework for QoS routing. In: 18th IEEE International Conference on Image Processing (ICIP 2011), Brussels, Belgium, pp. 2241–2244. IEEE, September 2011
20. Yilmaz, S., Tekalp, A.M., Unluturk, B.D.: Video streaming over software defined networks with server load balancing. In: International Conference on Computing, Networking and Communications (ICNC 2015), Garden Grove, CA, USA, pp. 722–726. IEEE, February 2015
21. Chen, W., Shang, Z., Tian, X., Li, H.: Dynamic server cluster load balancing in virtualization environment with openflow. Int. J. Distrib. Sens. Netw., Article ID 531538 (2015)
22. Ghaffarinejad, A., Syrotiuk, V.R.: Load balancing in a campus network using software defined networking. In: Third GENI Research and Educational Experiment Workshop (GREE 2014), Atlanta, GA, pp. 75–76. IEEE, March 2014
23. Yeganeh, S.H., Ganjali, Y.: Kandoo: a framework for efficient and scalable offloading of control applications. In: First Workshop on Hot Topics in Software Defined Networks (HotSDN 2012), Helsinki, Finland, pp. 19–24, ACM, August 2012
24. Li, D., Chen, G., Ren, F., Jiang, C., Xu, M.: Data center network research progress and trends. J. Comput. **25**(7), 87–89 (2014)
25. Amis, A.D., Prakash, R., Vuong, T.H.P., Huynh, D.T.: Max-min d-cluster formation in wireless ad hoc networks. In: IEEE International Conference on Computer Communications (INFOCOM 2000), Tel Aviv, pp. 32–41. IEEE, March 2000
26. Ryu. http://www.Ryu.org/
27. Mininet. http://mininet.org/
28. Benson, T., Anand, A., Akella, A., Zhang, M.: Understanding data center traffic characteristics. ACM SIGCOMM Comput. Commun. Rev. **40**(1), 65–72 (2009)
29. Kandula, S., Sengupta, S., Greenberg, A., Patel, P., Chaiken, R.: The nature of data center traffic: measurements and analysis. In: 9th ACM SIGCOMM Conference on Internet Measurement Conference (IMC 2009), Chicago, IL, USA, pp. 202–208. ACM, November 2009