Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 15, no. 8, August 2019, pp. 2217-2227
DOI: 10.23940/ijpe.19.08.p22.22172227

A Scalable Load Balancing Scheme for Software-Defined Datacenter Networks based on Fuzzy Logic

Guoyan Li[a], Xinqiang Wang[b,*], Zhigang Zhang[a], Yadong Chen[a], and Shudong Liu[a]

[a]*School of Computer and Information Engineering, Tianjin Chengjian University, Tianjin, 300384, China*
[b]*School of Software and Communication, Tianjin Sino-German University of Applied Sciences, Tianjin, 300350, China*

Abstract

In order to solve the problem that traditional load balancing technology lacks sufficient scalability and flexibility, we propose a dynamic and scalable load balancing scheme based on fuzzy logic for datacenter networks using the SDN architecture characteristics of control and forwarding separation, named LBSFL. The main parameters affecting the server load performance are analyzed and used as the input variables of fuzzy logic system to evaluate the server load, and then SDN controller schedules current network requests to achieve server load balancing. Additionally, we take into account the issue of server activation and deactivation under various traffic loads in this scheme. The experimental results show that the response time of the system can be increased by about 15% compared with LBBSRT, and it can also maintain the system load between the lower (30%) and normal (50%) boundaries. The scheme can greatly improve the load balance of servers under the premise of guaranteeing the network performance.

Keywords: software-defined networking; load balancing scheme; fuzzy logic

(Submitted on June 10, 2019; Revised on July 7, 2019; Accepted on August 11, 2019)

1. Introduction

In recent years, the Internet has been overloaded with the explosive growth of its scale and traffic, and a single server is unable to handle all network requests. At the same time, the rapid development of virtualization and cloud computing technology is also a huge challenge to the network [1]. Many Internet enterprises usually use load balancing technology to achieve a rational use of resources and provide users with reliable and high-quality services. The limitations of traditional network architecture and load balancing technology lead to poor network flexibility and low resource utilization, which ultimately lead to the decline of QoS in the network.

Software-defined networking (SDN) is a new network architecture that connects applications to network services and devices more closely, whether they are physical or virtualized [2-3]. As one of the core technologies of SDN, the OpenFlow protocol can realize the independence of control planes from traditional switching devices, thus achieving more flexible control of networks [4-5]. The OpenFlow protocol is an interface standard to describe the information used in the interaction between controllers and OpenFlow switches. In the OpenFlow network, the OpenFlow switch only has the function of the data plane, and the work of the control plane is completed by the SDN controller. The controller centralizes the control of switches in the network through the OpenFlow protocol and realizes data forwarding. Therefore, the load balancing algorithm can be deployed in the controller to achieve reasonable forwarding of traffic.

At present, some scholars have studied load balancing technology based on SDN. Reference [6] proposed a publish/subscribe system based on SDN. It can construct and store overlays and achieved a better trade-off between optimization objective and forwarding cost. Handigol et al. [7] proposed a comprehensive load balancing algorithm that can minimize system response time. A polling algorithm was used to achieve network load balancing in [8]. Reference [9] proposed a static and a dynamic load balancing algorithm for server clusters based on SDN. Reference [10] studied dynamic load balancing technology in the OpenFlow network, and the algorithm reduced the response time of the server. Reference [11]

* Corresponding author.
E-mail address: waffchiang@163.com

proposed a load balancing algorithm based on SDN. Bradai et al. [12] proposed a new mobility management called SDMM, which can reduce the handover latency and delivery delay. Reference [13] presented a mix integer programming model that was subjected to the minimization of the total supply chain service cost. The optimization solution was obtained using the improved heuristic algorithm. Reference [14] proposed a load balancing algorithm based on the response time of the server. However, it is unreliable to determine the server's load state only by response time. Reference [15] proposed a sequenced switch migration algorithm using sequenced manner. It considered delay, residual capacity, and failure probability for multi-objective optimization. However, current research is mainly reflected in the application of SDN architecture, and few improvements have been made to the load balancing algorithm. For the improved algorithm, the server load is evaluated according to a single performance parameter, which cannot effectively reduce the server response time and improve system resource utilization. Meanwhile, the existing load balancing scheme lacks scalability.

Fuzzy logic can solve many complex problems that cannot establish accurate mathematical models [16], such as security of cloud environments [17]. In this server cluster scheduling model, the load on each server is non-linear and unpredictable. Due to the restriction of information technology in nodes, it takes extra time to obtain and store information [18]. In this way, the information stored in the middleware can only represent the server's load status in the past, but not the current load status. The system experiences a certain delay. In order to collect the server load status more accurately, it is more suitable to use fuzzification language to evaluate the server load status.

According to this analysis, we propose a dynamic scheme based on fuzzy logic theory, load balancing theory, and virtual server scalability. In this scheme, we analyze the performance parameters that can reflect the load status of the virtual server and obtain the server load using the improved fuzzy logic load balancing algorithm. Then, the forwarding and control of network traffic requests can be determined through the average load of the entire server. When the load of the server is unbalanced, the lightest-loaded server will handle the current network traffic requests and, if necessary, activate or deactivate a virtual server. In order to validate the correctness and effectiveness of the LBSFL scheme, we conduct several simulation experiments. The experimental results show that the proposed scheme has high scalability and reliability. The system also shows better resource utilization and response time.

The rest of this paper is organized as follows. The architecture of the system is described in Section 2. Section 3 details the design of our proposed scheme. The performance evaluation of LBSFL scheme is presented in Section 4. The paper is summarized in the last section.

2. System Architecture

The architecture of the system, which is shown in Figure 1, consists of an OpenStack cloud platform, OpenFlow switch network, SDN controller, firewall and anti-virus module, server cluster, database, and so on. We adopt SDN network architecture, which supports the OpenFlow protocol. The controller can obtain the topology of the entire OpenFlow network and control the forwarding of OpenFlow flow tables in a unified way [19]. Based on the data forwarding module of the existing controller, a load balancing module is added to the system architecture. OpenFlow switches collect performance information of virtual servers regularly and feedback the information to the controller. Then, the load status of the virtual server is calculated according to the proposed fuzzy logic algorithm.

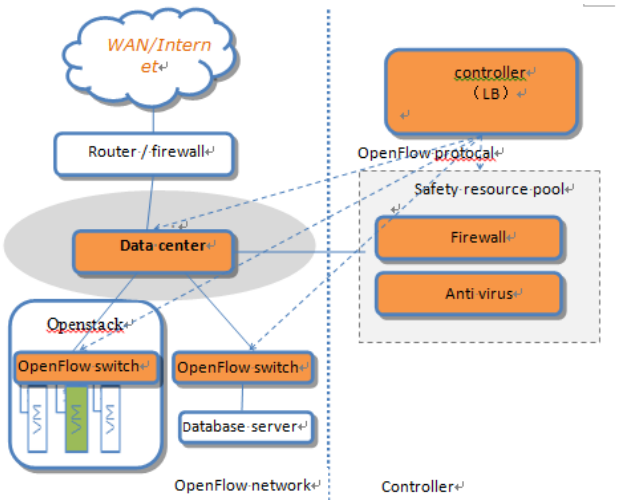


Figure 1. System architecture

3. Load Balancing Algorithm based on Fuzzy Logic

The server load can be obtained through some performance parameters of system, such as CPU utilization and memory utilization. However, the server load status expressed by these parameters is not a certain value. For example, we can consider that the server is in a high load state when the CPU utilization is 70% or 75%. This means that there are no precise mathematical models or control rules. Considering the advantages of fuzzy mathematics in describing emergent and uncertain problems, we introduce fuzzy logical theory into the load balancing scheme to evaluate the server's load. The fuzzy logic system consists of three-step: fuzziness, fuzzy reasoning, and defuzzification.

3.1. Fuzziness

Many performance parameters can be used to express the load status of virtual servers, such as the main frequency, CPU utilization, memory utilization, number of processes currently executing, and response time of each server. When traffic requests are scheduled, both call requests and return results are transmitted over the network. CPU and memory resources are occupied when tasks are executed. In this paper, CPU utilization, memory utilization, and I/O utilization are used as the performance parameters to evaluate server load. At the same time, these three parameters are used as input variables of the fuzzy logic system. Subsequently, the input variables are expressed as the fuzzy membership of related resources by the fuzzification function.

Fuzzy sets can assign a value to individuals in the universe through a mathematical way, which represents their membership in the fuzzy set. These degrees of membership are expressed by the numbers between "0" and "1". The process of fuzzification needs to determine the corresponding rank of fuzziness for each input variable, such as {low, middle, high}. Each variable has a membership degree corresponding to the rank sequence, so it is necessary to design the membership function to map the membership relationship between variables and sequences. Define the fuzzy membership function as follows:

(1) CPU utilization

Define the domain for CPU utilization of the server: Take three mappings from C to interval [0,1]: μ_l: $C\rightarrow[0,1]$, $C\rightarrow\mu_l(C)$; μ_m: $C\rightarrow[0,1]$, $C\rightarrow\mu_m(C)$; and μ_h: $C\rightarrow[0,1]$, $C\rightarrow\mu_h(C)$. The three fuzzy subsets of C are l, m, and h. μ_l, μ_m, and μ_h are three membership functions of C. Among them, the fuzzy subsets l, m, and h represent the load statuses light, medium, and high respectively in the load balancing algorithm. The fuzzy membership functions of CPU utilization are shown in Equations (1) to (3).

$$\mu_l(C)=\begin{cases}1, & C\le 25\%\\ 1.5\text{-}2\times C, & 25\%<C\le 75\%\\ 0, & C>75\%\end{cases} \tag{1}$$

$$\mu_m(C)=\begin{cases}0, & C\le 25\%\\ 4\times C\text{-}1, & 25\%<C\le 50\%\\ 3\text{-}4\times C, & 50\%<C\le 75\%\\ 0, & C>75\%\end{cases} \tag{2}$$

$$\mu_h(C)=\begin{cases}0, & C\le 25\%\\ 2\times C\text{-}0.5, & 25\%<C\le 75\%\\ 1, & C>75\%\end{cases} \tag{3}$$

(2) Memory utilization

Define the domain for memory utilization of the server: Take three mappings from M to interval [0,1]: μ_l: $M\rightarrow[0,1]$, $M\rightarrow\mu_l(M)$; μ_m: $M\rightarrow[0,1]$, $M\rightarrow\mu_m(M)$; and μ_h: $M\rightarrow[0,1]$, $M\rightarrow\mu_h(M)$. The three fuzzy subsets of M are l, m, and h. μ_l, μ_m, and μ_h are three membership functions of M. Among them, the fuzzy subsets l, m, and h represent the load statuses light, medium, and high respectively in the load balancing algorithm. The fuzzy membership functions of memory utilization are shown in Equations (4) to (6).

$$\mu_l(M)=\begin{cases}1, & M\le 25\%\\ 1.5\text{-}2\times M, & 25\%<M\le 75\%\\ 0, & M>75\%\end{cases} \tag{4}$$

$$\mu_m(M) = \begin{cases} 0, & M \le 25\% \\ 4\times M\text{-}1, & 25\% < M \le 50\% \\ 3\text{-}4\times M, & 50\% < M \le 75\% \\ 0, & M > 75\% \end{cases} \tag{5}$$

$$\mu_h(M) = \begin{cases} 0, & M \le 25\% \\ 2\times M\text{-}0.5, & 25\% < M \le 75\% \\ 1, & M > 75\% \end{cases} \tag{6}$$

(3) I/O utilization

Define the domain for I/O utilization of the server: Take three mappings from *IO* to interval [0,1]: μ_l: $IO \to [0,1]$, $IO \to \mu_l(IO)$; μ_m: $IO \to [0,1]$, $IO \to \mu_m(IO)$; and μ_h: $IO \to [0,1]$, $IO \to \mu_h(IO)$. The three fuzzy subsets of *IO* are *l*, *m*, and *h*. μ_l, μ_m, and μ_h are three membership functions of *IO*. Among them, the fuzzy subsets *l*, *m*, and *h* represent the load statuses light, medium, and high respectively in the load balancing algorithm. The fuzzy membership functions of I/O utilization are shown in Equations (7) to (9).

$$\mu_l(IO) = \begin{cases} 1, & IO \le 30\% \\ 1.5\text{-}2\times IO, & 30\% < IO \le 70\% \\ 0, & IO > 70\% \end{cases} \tag{7}$$

$$\mu_m(IO) = \begin{cases} 0, & IO \le 25\% \\ 5\times IO\text{-}1.5, & 30\% < IO \le 50\% \\ 3.5\text{-}5\times IO, & 50\% < IO \le 70\% \\ 0, & IO > 70\% \end{cases} \tag{8}$$

$$\mu_h(IO) = \begin{cases} 0, & IO \le 30\% \\ 2\times IO\text{-}0.75, & 30\% < IO \le 70\% \\ 1, & IO > 70\% \end{cases} \tag{9}$$

Through the processing of the membership function, we can make a comprehensive evaluation for the virtual server's load. The result of the fuzzy logic algorithm is the possibility that the virtual server will deal with the current traffic request, expressed as *R*. In the fuzzy comprehensive evaluation of server load, the factor set is defined as $E = \{C, M, IO\}$ and the comment set is defined as {high, medium, low}.

3.2. Fuzzy Reasoning

Fuzzy reasoning is a key step in the fuzzy logic algorithm. The reasoning process establishes the reasoning rules for each evaluation index based on the relations between input variables and output variables. The most important part of fuzzy reasoning is the rule database of fuzzy control. The classical fuzzy rules are formed by numerical or linguistic variables. In this paper, we build a fuzzy matrix to realize the evaluation of factor set to comment set. Comprehensive fuzzy evaluation is based on the following steps:

(1) We evaluate each variable in μ and then construct fuzzy matrices representing fuzzy relations between *E* and *F*, as shown in Equations (10) to (12).

$$R1 = [\mu_h(C), \mu_m(C), \mu_l(C)] \tag{10}$$

$$R2 = [\mu_h(M), \mu_m(M), \mu_l(M)] \tag{11}$$

$$R3 = [\mu_h(IO), \mu_m(IO), \mu_l(IO)] \tag{12}$$

Three vectors constitute the fuzzy matrix from *E* to *F*, $R = [R1, R2, R3]$.

(2) The weight vector is defined as $P = [p1, p2, p3]$, and $p1$, $p2$, and $p3$ represent the importance of CPU utilization, memory utilization, and I/O utilization in μ, respectively. $p1 + p2 + p3 = 1$.

(3) Make the fuzzy transformation operation, $Q = P.R$. The fuzzy vector Q is the result of each virtual server's evaluation on the set of comments, and the three components indicate the degree of possibility that the virtual server is the candidate server.

3.3. Defuzzification

Since the output obtained by fuzzy reasoning is still a fuzzy vector F, $F = (L, M, H)$, it is necessary to obtain the exact value by defuzzification. The classical area center method is used for defuzzification in this paper, and the centroid of the membership function corresponding to each fuzzy grade is taken as the exact output of the fuzzy grade. The corresponding centroids of fuzzy grade L, M, and H are 0.15, 0.5, and 0.85, respectively. Then, we use Equation (13) to obtain the output of the fuzzy logic algorithm.

$$Fuzzy_{\_out} = \frac{\sum_i w_i M_i}{\sum_i \mathrm{M_i}} \tag{13}$$

Where M_i indicates the centroid value of each fuzzy grade and w_i indicates the weight of the corresponding fuzzy grade for M_i. The higher the result, the heavier the virtual server load. When the average load of the servers exceeds the threshold, the fuzzy logic algorithm is implemented to achieve server load balancing. A scalable load balancing scheme for software-defined datacenter networks will be introduced in the next section.

3.4. Load Balancing Scheme based on SDN

Figure 2 shows the setting of threshold of the scheme proposed in this paper. Through the improved load balancing algorithm based on fuzzy logic, we can obtain the load on each of the servers and then obtain the average load of all servers. This load can also be expressed as the resource utilization of the system.

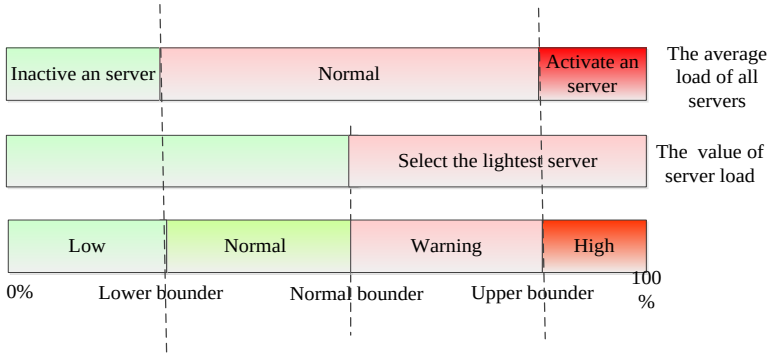


Figure 2. Expression of threshold

If the average load of all servers is between 0 and the low boundary, it is regarded as "low". In this case, the lowest load server will be deactivated to increase the utilization of the system. If the average load of all servers is between the lower and normal boundaries, the system is in a "normal" state. If the average load of all servers is between the normal and warning boundaries, the load between servers is unbalanced, and the current traffic request will be handled by the lightest load server. If the average load of all servers exceeds the upper boundary, the status of the system is regarded as "high". In this case, an inactive virtual server will be activated to reduce the system utilization. We can set the lower, normal, and upper boundaries as required.

Figure 3 is the flow chart of the LBSFL scheme in this paper. The execution process of LBSFL is shown below.

Step 1 The load balancing module in the SDN controller collects and analyzes the load information of each virtual server at intervals. The information includes CPU utilization, memory utilization, and I/O utilization.

Step 2 If the average load of servers is in the "low" state, the lowest load server will be deactivated to increase the utilization of system resources.

Step 3 If the average load of servers is in the "warning" state, the current network traffic request will be forwarded to the lightest load server, and the lightest server is acquired using the improved fuzzy logic algorithm in the paper.

Step 4 If the average load of servers is in the "high" state, an inactive virtual server will be activated to reduce the utilization of system resources.

Step 5 If the SDN controller continues to monitor the status of all servers, return to Step 1; otherwise, end.

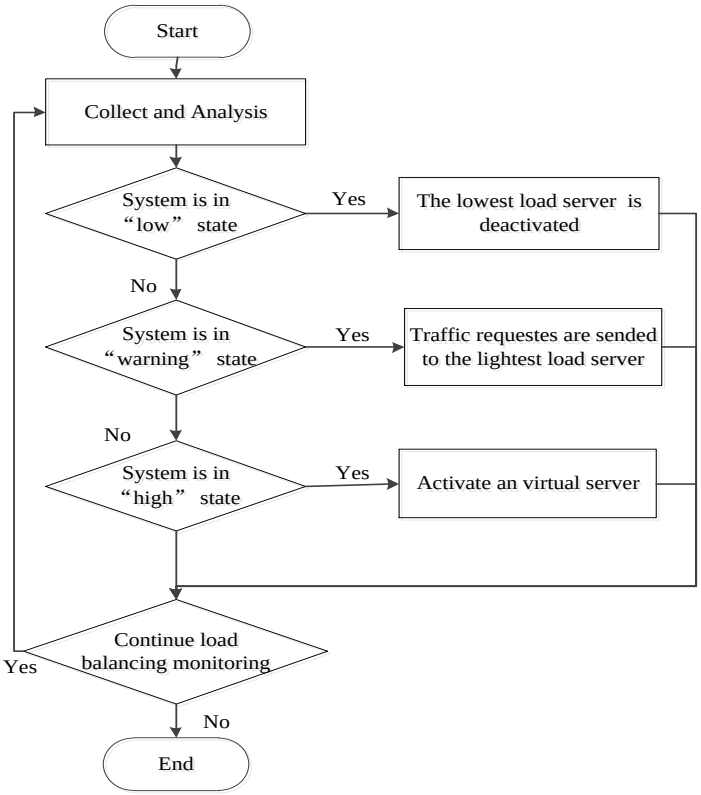


Figure 3. Flow chart of the LBSFL scheme

4. Experiment Result and Performance Analysis

To prove the advantages of the proposed LBSFL scheme for datacenter networks, this section introduces the experimental prototype setup and the experimental scenarios in real life. Then, the simulation results are analysed.

4.1. Experimental Prototype Setup

Figure 4 shows the topology of the datacenter. Four virtual servers with the same configuration are set up in the experiment. Considering that frequently deactivating or activating the server will affect the performance of the system, three servers are initialized to provide web services. The configuration of the three virtual servers used in the simulation environment is exactly the same.

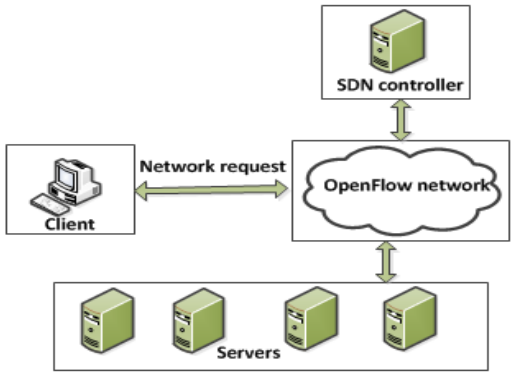


Figure 4. Topology of datacentre

OpenDayLight is chosen as the SDN controller in our experiments. It is a model-driven controller based on java programming language [20]. It provides a northward interface to the application layer, and the communication between the controller and the application can be achieved through the Java interface or REST API. The basic configuration for the experiment is Intel Core 3.4 GHZ CPU 8 GB RAM. iPerf tools are used to generate network traffic in mininet [21-22].

In order to make the experiments more representative, we evaluate the efficiency of LBSFL against the LBBSRT scheme. LBBSRT is a dynamic balancing scheme comparing with the static load balancing scheme and achieves better load balancing performance compared with other schemes [14]. The input variables of fuzzy logic system include CPU utilization, memory utilization, and I/O utilization. It is important to note here that we set the weight factor of three parameters as 0.4, 0.3, and 0.3. CPU utilization is used to determine the current load of the server compared with memory utilization and IO utilization in our daily lives. The weight value of CPU utilization is also higher than that of memory utilization and IO utilization.

4.2. Experimental Scenarios

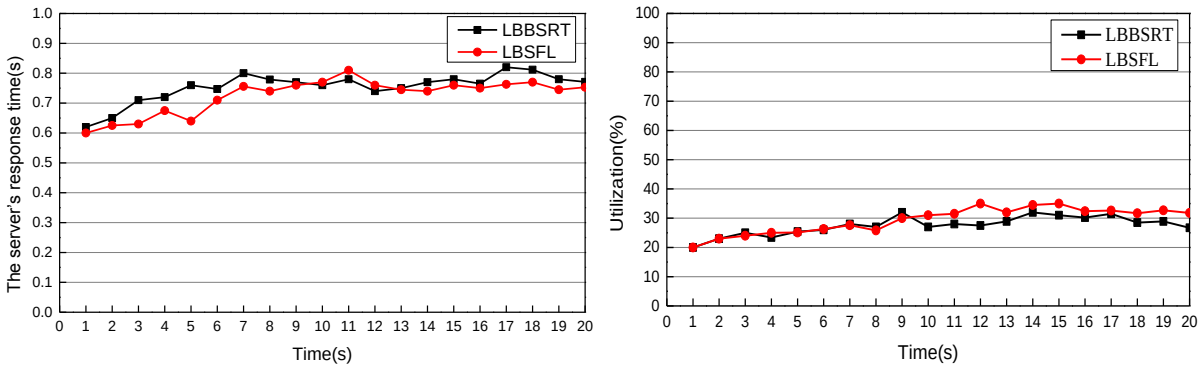
In order to analyze the performance of the LBSFL scheme for datacenter networks, we have designed different scenarios under various traffic loads based on the experimental prototype. According to the measurement of [23], the request rate of the switch in the datacenter is up to 5000K/s in the peak period. In our experiment, three scenarios are considered, as listed in Table 1. The goal of the load balancing scheme is to maintain the average load of servers between the lower boundary (30%) and the normal boundary (50%). The average response time for each server is taken as the response time of the system. The utilization of resources and response time of the system are used as two parameters to compare with LBBSRT, and the performance of each server in the system is analysed.

Table 1. Experimental scenarios

Scenario	Time (*s*)	The request of network traffic (*/s*)
1	20	1000
2	20	2000
3	20	5000

Scenario 1 The implementation results are shown below. Figure 5 illustrates the average response time of the system and the utilization of system resources in the first scenario.

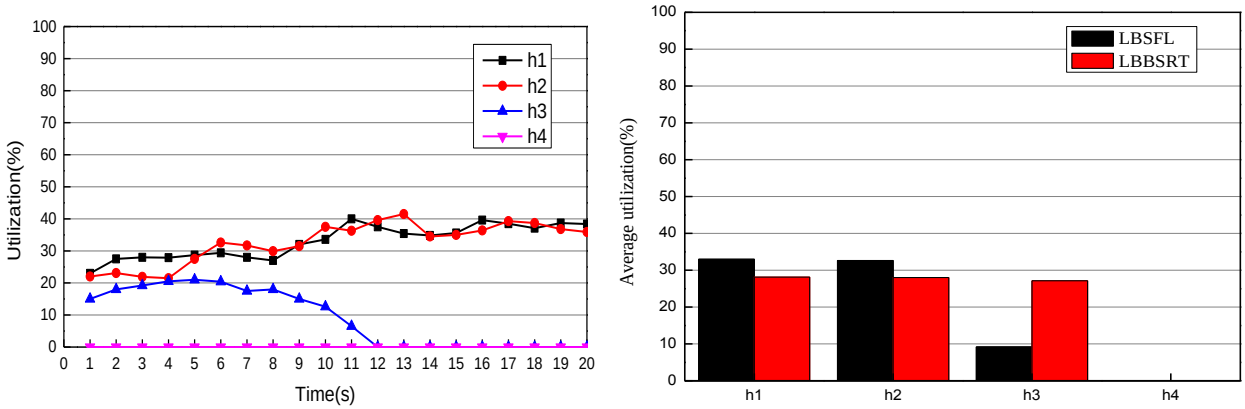
Figure 5(a) shows that the average response time of the system for LBBSRT and LBSFL was 0.754s and 0.725s, respectively. Compared with LBBSRT, LBSFL has a slight advantage in system response time. Figure 5(b) shows the analysis of system resource utilization, and the results show that the system resource utilization of LBSFL is higher than that of LBBSRT after 9 seconds. This is because the resource utilization of the system remains below the low boundary. In order to keep the average load of the system between the lower and normal boundaries, the lightest server must be inactivated.



(a) The server's response time in the first scenario (b) System utilization analysis in the first scenario

Figure 5. Analysis in the first scenario

Figure 6 shows that at almost 9s, the lightest server h3 was not allowed to process traffic requests. There is a small increase in the utilization of the system, and the average load of the system is between the basic and normal boundaries. Figure 6(a) shows the resource utilization of each server in the LBSFL scheme. Figure 6(b) presents the average utilization of four servers named h1, h2, h3, and h4 under the LBSFL and LBBSRT schemes, and h4 was dormant. According to the curve in Figure 6, h3 was inactivated at 12s. In comparison with the LBBSRT scheme, our scheme saves resources and achieves better load balancing.

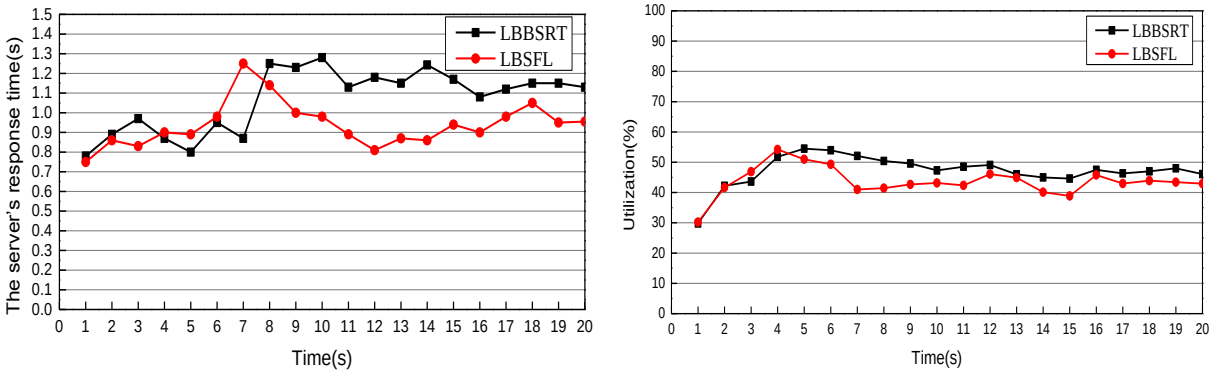


(a) LBSFL's performance analysis in the first scenario (b) Performance analysis of servers in the first scenario

Figure 6. Performance analysis in the first scenario

Scenario 2 The implementation results are shown below. Figure 7 illustrates the server's response time and system utilization in the second scenario, respectively.

In this scenario, the average response time of the system for LBBSRT and LBSFL was 1.07s and 0.939s, respectively. We can see from Figure 7(a) that compared to the LBBSRT scheme, LBSFL has significant advantages in terms of overall response times. Figure 7(b) shows the average resource utilization of the system for LBBSRT and LBSFL. The curve of LBSFL reveals that the average load of the system was between the lower and normal boundaries under the implementation of the fuzzy logic load balancing algorithm.



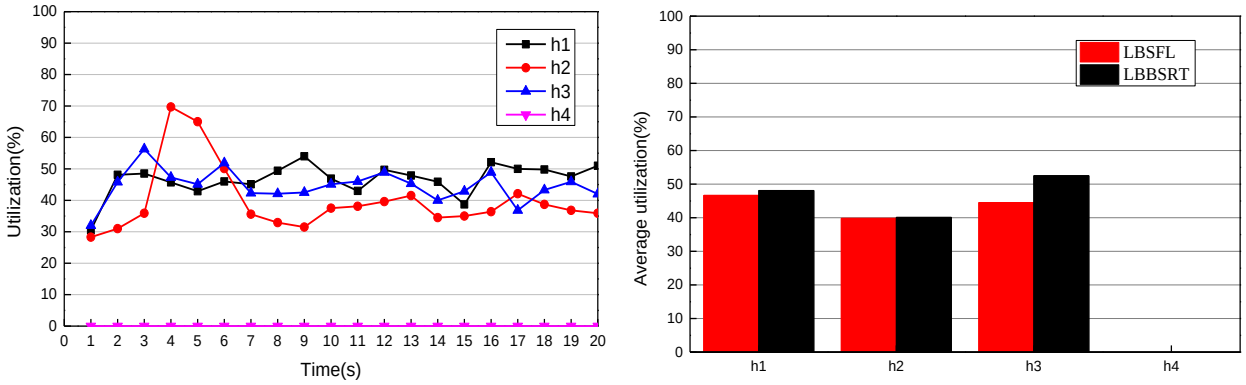
(a) The server's response time in the second scenario (b) System loading analysis in the second scenario

Figure 7. Analysis in the second scenario

Figure 8(a) shows the utilization of the four servers named h1, h2, h3, and h4 under the LBSFL scheme. Figure 8(a) shows that at almost 4s, h2 was overloaded and the scheme was triggered to determine which server should process the incoming traffic request. According to Figure 8(b), the resource utilization of LBSFL was better than that of LBBSRT, and the three servers achieved load balancing. The reason is that in our scheme, the algorithm takes CPU utilization, memory utilization, and IO utilization as input variables of the fuzzy logic algorithm to evaluate the server load more accurately, and then the SDN controller chooses the lightest load server to process current network traffic. It is more difficult to achieve in the traditional scheme. In comparison with the LBBSRT scheme, our scheme obtains a much better load balancing effect and reduces the response time of the system.

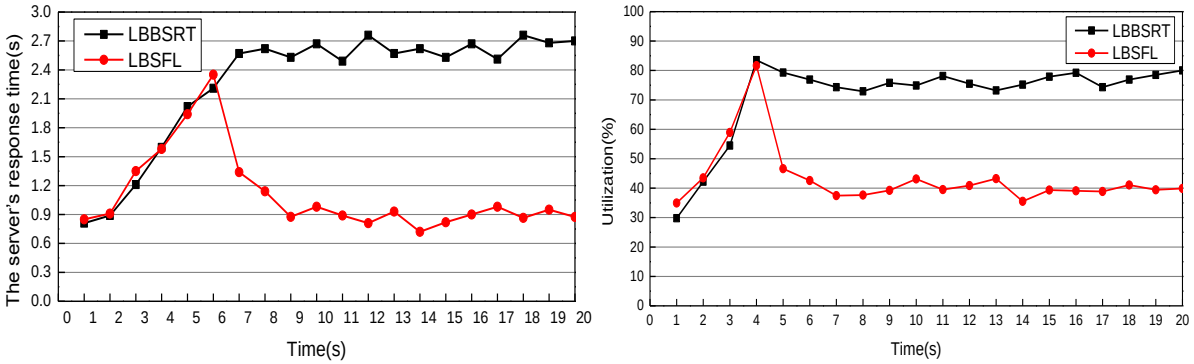
Scenario 3 The implementation results are shown below. Figure 9 illustrates the server's response time and system utilization in the third scenario.

Figure 9(a) shows that the average response time of servers for LBBSRT and LBSFL were 2.27s and 1.102s, respectively. Figure 9(b) shows that at almost 4s, the average of the servers was above the upper boundary. To reach system dynamic load balancing, the controller triggered the LBSFL scheme to activate a virtual server in the LBSFL scheme, and then the response time of the system declined and tended to be stable. At approximately 7s, the system utilization decreased from 81 % to 37% in the LBSFL scheme. The system utilization cannot achieve a good effect of load balancing under the LBBSRT scheme.



(a) Performance analysis of servers in the second scenario (b) Performance analysis of servers in the second scenario

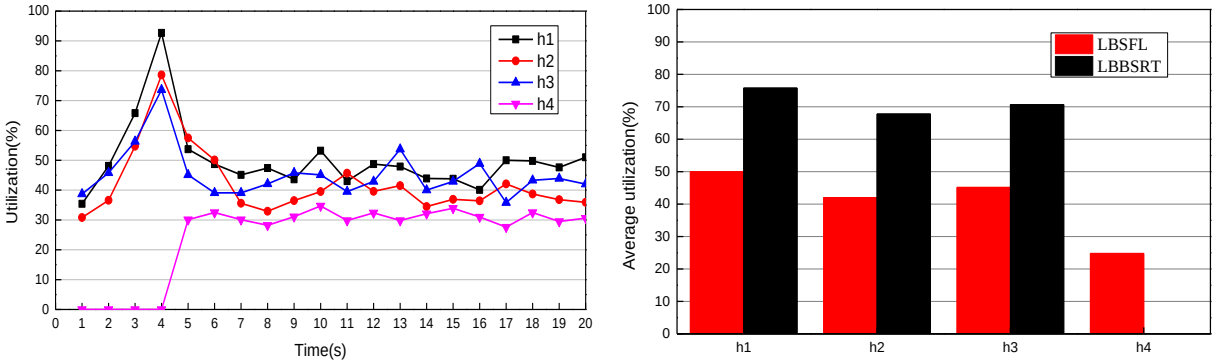
Figure 8. Performance analysis in the second scenario



(a) The server's response time in the third scenario (b) System loading analysis in the third scenario

Figure 9. Analysis in the third scenario

Figure 10(a) shows the resource utilization of each server in LBSFL. The h4 server was activated at 4s. Figure 10(b) shows the average utilization of system for LBSFL and LBBSRT. According to Figure 10(b), LBSFL has lower system resource utilization than LBBSRT, and the servers achieve load balancing. LBSFL has significant advantages compared to the LBBSRT scheme, especially in the case of heavy loads.



(a) Performance analysis of servers in the third scenario (b) Performance analysis of servers in the third scenario

Figure 10. Performance analysis in the third scenario

5. Conclusions

In this paper, we propose a scalable and dynamic load balancing scheme based on SDN for datacenter networks. The OpenFlow switches collect the status information of the server in real time and report it to the SDN controller. Three performance parameters are used as the input variables of the fuzzy logic system to evaluate the server load. We propose the definition of lower, normal, and upper thresholds of system utilization, and the scheme can activate or inactivate the server according to the current load of the system. The LBSFL scheme achieves high reliability and scalability compared with the dynamic LBBSRT scheme and demonstrates better load balancing performance compared with other schemes under various

traffic loads. The scheme not only solves the load balancing problem efficiently, but also takes into account the issue of server activation and deactivation to improve the system performance.

Acknowledgements

This work was partially supported by the Project of Natural Science Foundation of Tianjin (No. 17JCQNJC00500), the Tianjin Educational Science Project Planning for 13th Five-year (No. HE3045), the Fund for the Development of Science and Technology of Tianjin Education Committee (No. 2018KJ174), and the Project of Tianjin Science and Technology (No. 18JCTPJC61800).

References

1. Y. Chen, X. Gong, W. Wang, and Z. Que, "VNMC for Network Virtualization in OpenFlow Network," in *Proceedings of IEEE International Conference on Cloud Computing and Intelligent Systems*, pp. 797-801, 2013
2. E. Salvadori, R. D. Corin, A. Broglio, and M. Gerola, "Generalizing Virtual Network Topologies in OpenFlow-based Networks," in *Proceedings of Global Telecommunications Conference*, pp. 1-6, IEEE, 2011
3. R. D. Corin, M. Gerola, R. Riggio, F. D. Pellegrini, and E. Salvadori, "Vertigo: Network Virtualization and Beyond," *European Workshop on Software Defined NETWORKING*, pp. 24-29, IEEE, 2012
4. N. McKeown, "Software-Defined Networking," *INFOCOM Keynote Talk*, Vol. 17, No. 2, pp. 30-32, 2009
5. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. L. Peterson, J. Rexford, et al., "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Review*, Vol. 38, No. 2, pp. 69-74, 2008
6. Y. Wang, Y. Zhang, and J. Chen, "SDNPS: A Load-Balanced Topic-based Publish/Subscribe System in Software-Defined Networking," *Applied Sciences*, Vol. 6, No. 4, pp. 91, 2016
7. N. Handigol, S. Seetharaman, M. Flajslik, N. McKeown, and R. Johari, "Plug-N-Serve: Load-Balancing Web Traffic using OpenFlow," *ACM Sigcomm Demo*, Vol. 4, No. 5, pp. 6, 2009
8. S. Kaur, K. K. Saluja, J. Singh, and N. S. Ghumman, "Round-Robin based Load Balancing in Software Defined Networking," in *Proceedings of International Conference on Computing for Sustainable Global Development*, IEEE, 2015
9. H. Zhang and X. Guo, "SDN-based Load Balancing Strategy for Server Cluster," in *Proceedings of IEEE International Conference on Cloud Computing and Intelligence Systems*, pp. 662-667, 2014
10. Z. Shang, W. Chen, Q. Ma, and B. Wu, "Design and Implementation of Server Cluster Dynamic Load Balancing based on OpenFlow," in *Proceedings of International Joint Conference on Awareness Science and Technology and Ubi-Media Computing*, pp. 691-697, IEEE, 2014
11. Y. Zhou, L. Ruan, L. Xiao, and R. Liu, "A Method for Load Balancing based on Software-Defined Network," *Advanced Science and Technology Letters*, Vol. 45, pp. 43-48, 2014
12. A. Bradai, A. Benslimane, and K. D. Singh, "Dynamic Anchor Points Selection for Mobility Management in Software Defined Networks," *Journal of Network & Computer Applications*, Vol. 57, No. C, pp. 1-11, 2015
13. J. He, Y. Huang, and D. Chang, "Simulation-based Heuristic Method for Container Supply Chain Network Optimization," Elsevier Science Publishers B. V., 2015
14. H. Zhong, Y. Fang, and J. Cui, "LBBSRT: An Efficient SDN Load Balancing Scheme based on Server Response Time," *Future Generation Computer Systems*, Vol. 68, pp. 183-190, 2017
15. L. Yao, T. Hu, and J. Lan, "Sequenced Switch Migration for Balancing Controller Loads in Large-Scale Software-Defined Networking," *International Journal of Performability Engineering*, Vol. 14, No. 8, pp. 1848, 2018
16. L. A. Zadeh, "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 1, pp. 28-44, 1973
17. Z. Yang and J. Luo, "A Behavior Trust Model based on Fuzzy Logic in Cloud Environment," *International Journal of Performability Engineering*, Vol. 14, No. 4, 2018
18. K. J. Zhang, Q. L. Han, and Z. P. Cai, "RiPPAS: A Ring-based Privacy-Preserving Aggregation Scheme in Wireless Sensor Networks," *SENSORS*, Vol. 17, No. 2, 2017
19. F. Pakzad, M. Portmann, W. L. Tan, and J. Indulska, "Efficient Topology Discovery in OpenFlow-based Software Defined Networks," *Computer Communications*, Vol. 77, No. C, pp. 52-61, 2016
20. D. B. Hoang and M. Pham, "On Software-Defined Networking and the Design of SDN Controllers," *Network of the Future*, pp. 1-3, IEEE, 2015
21. R. Trestian, K. Katrinis, and G. M. Muntean, "OFLoad: An OpenFlow-based Dynamic Load Balancing Strategy for Datacenter Networks," *IEEE Transactions on Network and Service Management*, 2017
22. G. Sun, T. Chen, Y. Su, and C. Li, "Internet Traffic Classification based on Incremental Support Vector Machines," Mobile Networks and Applications, pp. 1-8, 2018
23. S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The Nature of Data Center Traffic: Measurements & Analysis," in *Proceedings of ACM SIGCOMM Conference on Internet Measurement Conference*, pp. 202-208, ACM, 2009

Guoyan Li received her M.S degree and Ph.D. in computer applications from Hebei University of Technology in 2009 and 2013, respectively. Her research interests include future internet architecture, network function virtualization, and software-

defined networking. Email: ligy@tcu.edu.cn

Xinqiang Wang received his M.S. degree from Nankai University in 2009. He is currently pursuing a Ph.D. in the School of Software at Tianjin University. His research interests include network optimization and routing in highly dynamic networks. Email: waffchiang@163.com

Zhigang Zhang received his M.S. degree in software engineering from Nankai University in 2005 and his Ph.D. in computer applications from Tianjin University of Technology in 2017. He is currently an associate professor at Tianjin Chengjian University. His research interests include internet architecture, network function virtualization, and network optimization. Email: zzg@tcu.edu.cn

Yadong Chen received her M.S. degree in computer applications from Chang'an University in 2003. She is currently an associate professor at Tianjin Chengjian University. Her research interests include network function virtualization and network optimization. Email: cyd@tcu.edu.cn

Shudong Liu received his Ph.D. degree from Harbin Institute of Technology in 2001. He is currently a professor at Tianjin Chengjian University. His research interests include communication technology and Internet of things technology. Email: lsd@tcu.edu.cn