Load Balancing Researches in SDN: A Survey

Lin Li and Qiaozhi Xu†
College of Computer Science , Inner Mongolia Normal University
Hohhot , China
{1019925027,29776716}@qq.com

Abstract—**Compared with the traditional networks, the SDN networks have shown great advantages in many aspects, but also exist the problem of the load imbalance. If the load distribution is uneven in the SDN networks, it will greatly affect the performance of network. Many SDN-based load balancing strategies have been proposed to improve the performance of the SDN networks. Therefore, this paper summarizes and classifies the load balancing schemes in SDN networks, and analyzes the advantages and disadvantages of them. This paper will provide a guidance for other researchers in this field and promote the development of the SDN networks.**

Keywords-Load Balancing; SDN; Software Defined Network

I. INTRODUCTION

With the rapid development of the technologies of Internet, IoT and mobile network, more and more users share various resources through the network which bring a large number of information and data, thus, the storage and processing of these huge amounts of data pose challenges to the traditional hardware devices. Compared with the traditional technologies, virtualization and cloud computing technologies have better performance and advantages because they can reasonably allocate resources according to users demands and avoid the problem of insufficient space. However, the flexibility and dynamicity of virtualization technology also bring more pressures and challenges to the configuration of the network. Therefore, in the context of cloud computing and big data, how to schedule the limited network resources to meet the dynamic needs has become the foundation of the further development of cloud computing and big data.

The traditional network architectures use the static mode to work and exist some shortcomings, such as, the management and maintenance inconvenient, lack of flexibility etc. In 2008, Nick McKeown proposed the OpenFlow protocol and the concept of SDN to solve some issues in data centers and attract the attention of academia and industry [1].

If the load distribution is uneven,it will greatly affect the performance and efficiency of the whole network.Load balancing technology is one of the important ways to effectively allocate network resources and improve the network performance. Compared with the traditional network architectures, the SDN architectures have higher performance in many aspects, but still exit the problem of the load imbalance [2]. Therefore, it is very meaningful to study the issue of load balancing for promoting the development of the SDN networks. This paper makes a survey to the load balancing strategies in SDN, classifies and analyzes the advantages and disadvantages of these strategies, and it can provide a guidance for other researchers in this field.

978-1-5090-3025-5/17/$31.00 ©2017 IEEE

The remainder of this paper is structured as follows. Section 2 gives an overview of the related work. Section 3 classifies the load balancing strategies in SDN, and analyzes the problems of these strategies. Section 4 concludes the load balancing technology in SDN.

II. RELATED WORK

The core idea of SDN is that network control is decoupled from forwarding[3], and the standard interaction protocol (such as OpenFlow) defines southbound interfaces for the control layer to interact with the forwarding layer.

SDN has been applied to a variety of network scenarios to meet the requirements of centralized automation management, multi-path forwarding, green energy-saving and load balancing [4]. For example, by using OpenFlow to plan the path in the Google's B4, the utilization rate of multiple links were closed to 100% and the average link utilization rate was as high as 70%, which greatly reduced the cost and enhanced the network stability[5]. Jupiter used the SDN technology to make the network bandwidth up to Pbps level, and it can meet the requirements of the large data centers for bandwidth[6]. Microsoft Corp's SWAN system used the SDN architectures to achieve the efficient use among data centers and ensure the resource utilization rate more than 60% [7]. ElasticTree used the SDN technology to control the global information of the data centers and reduced the energy consumption to 50% [8]. SoftRAN used the global information of SDN to quickly and accurately coordinate the wireless equipment managed by the RAN, and reasonably allocated the spectrum resource and reduced the energy consumption [9].

Compared with the traditional network architectures, SDN technology can effectively improve the network performance and flexibility, but there are still some unresolved issues, for example, the standardization of the northbound interfaces, the scalability of the control plane and the load balancing among the multiple controllers etc. In order to enable researchers to quickly understand the research progress of SDN technology in a certain field, it is necessary to summarize and analyze the research progress in these field. C. K. Zhang et al. summarized the consistency, fault tolerance and other aspects in SDN[10]; T. Q. Zhou et al. analyzed the relative studies of traffic engineering based on SDN[11]. S. Scott-Hayward et al. summarized the security problems in SDN[12]. A. Blenkd et al. surveyed network virtual hypervisor based on SDN[13]. But so far, there is still no summary and analysis of the load balancing algorithms and strategies in SDN networks. This paper investigates, analyzes and summarizes the latest researches status and progresses of load balancing in SDN. Also, it can provide some help to the researchers in the related fields.

III. LOAD BALANCING IN THE SDN

Load balancing technology is one of the important ways to effectively allocate network resources and improve the network performance and quality of service. Compared with the traditional network architectures, the SDN architectures have higher efficiency and performance in many aspects, but also exit the problem of the load imbalance. This paper classifies the load balancing researches in SDN networks, as show in Fig. 1. The SDN architectures can be divided into centralized single controller architectures and distributed multiple controllers architectures according to the number and organization of the controllers in SDN networks. In the centralized architectures, load balancing researches are divided into the data plane and the control plane. The data plane mainly includes link load balancing and server load balancing. The distributed architectures are divided into the flat architecture and the hierarchical architecture. This paper introduces, analyzes and summarizes the above several types of load balancing researches, so that researchers can quickly understand the relevant knowledge in this field.

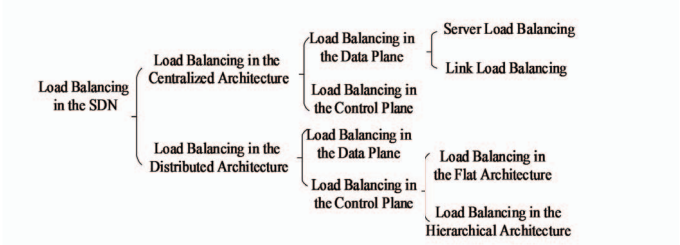

Figure 1. The Classification of Load Balancing in SDN

A. Load Balancing in the Centralized SDN Architecture

Generally, the centralized SDN networks have one controller, multiple SDN switches and servers. In this architecture, the problems of load balancing mainly focus on the servers and links. The controller controls the whole network and also manages and allocates all traffic in the network. In additional, the controller also can collect the load of the network in real time as well as dynamically adjust the load of servers and links according to the load balancing strategy, which can maximize the utilization of network resources and improve the network performance.

1) Load Balancing in the Data Plane

In the existing researches, load balancing of the data plane is mainly to solve the load imbalance of servers and links. In the following part, this paper introduces, summarizes and analyzes the server load balancing and link load balancing.

a) Server Load Balancing

When the traffic scheduling scheme in the network makes a node or link congestion, it may reduce the network performance and increase the transmission delay. By using the load balancing strategy to allocate traffic to different servers can avoid unnecessary network congestion.

For the load imbalance of servers, Stanford University proposed a web traffic load balancing scheme based on OpenFlow, called Plug-n-Server [14] (now known as Aster * x[15]). There were three components in the controller, and the host manager and network manager dynamically collected the CPU load of each server and network congestion status respectively. The traffic manager dynamically adjusted the load of the servers according to the collected information. The scheme reduced the network latency and improved the network performance. However, when the number of servers was increased, it would increase the load of controller and affect the user's experience and quality of service.

R. Wang et al. proposed a load balancing strategy which integrated matching forwarding rules of multiple IP prefixes into a forwarding rule with wildcard and used the wildcard rule to aggregate the service requests[16]. This strategy reduced the load of the controller, but some rules must be pre-installed in the switch which influenced the flexibility and scalability of the strategy.

The SBLA load balancing algorithm was proposed in [17] and it was suitable for server-cluster in virtual environment. Firstly, the controller used the SNMP protocol to collect the state information of the servers, then calculated the load of the servers according to the SBLA algorithm, and finally selected the lightest load server to respond to the users. The algorithm minimized the server's response time, but was unsuitable for unstructured networks and data centers.

In order to quickly respond to the users' requests, many data centers used the IMKVS (In-Memory Key-Value Storage) cache mechanism. In order to improve the performance of IMKVS, a two-phase load balancing method was proposed in [18]. In the first-phase, the load balancer dispersed the IMKVS traffic and assigned the load to the virtual machine with minimum load, so as to avoid the second-phase load balancer became the bottleneck of the network. In the second-phase, the load balancer balanced the load and alleviated the servers load by using several cache servers and replicating the popular targets. The scheme effectively reduced the load of the servers. However, it increased the command execution time when added the virtual machines. In addition, the scheme was only to handle the IMKVS requests and had some limitations.

b) Link Load Balancing

When multiple flows are mapped to the same link, some of data flows may suffer from some problems, such as the increased queuing time, long transmission delay and so on. By using a certain scheduling strategy to distribute the data flows to different paths to improve the utilization of the link.

Hedera was proposed in [19] and aimed at the phenomenon of link load imbalance caused by multiple flows being mapped to the same path. The scheme run the global first-fit algorithm and simulated annealing algorithm to make the load balancing decision and improve the network resource utilization. However, the scheme was only suitable for the data centers of Portland topology and lacked of the flexibility and scalability.

ECMP (Equal Cost Multipath Routing) [20]was one of the most common used schemes for link load balancing. But the hash algorithm in ECMP only considered the equivalence of the path and could not dynamically adjust according to the traffic on the path. The CONGA [21] proposed by Alizadeh et. al., which used the DRE (Discount Rate Estimator) algorithm

to estimate the load of per link, then the source node balanced the network load according to the congestion situation of each link. The CONGA could quickly respond to the burst traffic in the data centers, but it was poor scalability.

For the defects of CONGA, HULA architecture was proposed in [22]. By using the distribution of the network link utilization information and the periodic probe, HULA performed the distance vector algorithm on the switch, and the data forwarding decision was made by the next jump address, but not by the entire path. HULA had good scalability and could adapt to the dynamic change of the load, but it was quite difficult to implement.

Y. Li et.al. proposed a dynamic load balancing algorithm (DLB) in fat-tree topology [23]. According to the single hop greedy strategy, DLB transferred all traffic from the source node to the highest level, and then transferred it down to the destination node. Although this method implemented the dynamic routing strategy, it only used the single hop greedy strategy to select the path and did not consider the states of other links, which could lead to the overload of part of links and network congestion.

According to the distribution features of the long flows and the short flows in the data centers, F. Carpio et al. proposed a DiffFlow scheme[24]. In DiffFlow, the ECMP strategy was used for short flows to minimize the flow completion time and improve throughput, and the RPS(Random Packet Spraying) strategy was used for long flows to efficiently balance load of the entire network and avoid the congestion. However, when the switches in ToR layer detected long flows, the controller needed to inform all the switches in the Agg layer and the Core layer, it would increase the communication overhead.

R. Gandhi et al. proposed a DUTE scheme which combined the hardware with the software[25]. The scheme used the existing switches to build a hardware load balancer to effectively increase the capacity, reduce the cost and delay, but the flexibility was poor. Especially, when the switch failed, the disadvantage was especially obvious. The hardware load balancer dealt with a large number of traffic, while the software load balancer served as a backup to ensure high availability and flexibility, but it was difficult to implement.

J. Li et al. proposed a path load balancing decision scheme.The fuzzy synthetic evaluation mechanism (FSEM)[26] was divided into two phases: initial mode and periodic mode. In the initial mode, there was no traffic in the network, so the shortest path Floyd algorithm was used to compute k shortest paths between any two points. When there was traffic in the network, the scheme went into the periodic mode and periodically executed the fuzzy evaluation algorithm.The scheme was simple and easy to implement,however, it assumed that all nodes and links had the same capacity to handle the flows which was not consistent with the actual situation.

2) Load Balancing in the Control Plane

The centralized SDN architectures can effectively improve the network management efficiency and performance, but when the network's scale is large and the interaction between the controller and the switches increase, which would also greatly increase the load of the controller so that the controller cannot process the switches' requests in time. Therefore, some researchers are dedicated to solve the overload problem of single controller in the centralized architectures.

The DIFANE[27] combined the active and passive installation of flow tables, and the flows were maintained in the data plane as far as possible. Only when the network changed, the rules in DIFANE would be modified, so that avoided the frequent communication between the controller and the switches and reduced the load of the controller. But the structure of the switch was too complex to be suitable for the large-scale networks. A. R. Curtis proposed DevoFlow on the basis of DIFANE [28]. DevoFlow used rule replication and local operation to reduce the information exchange between switches and controller, but it was not realistic because the flow table structure and the hardware structure of the switch needed to be modified in order to implement the function.

Pre-installed rules in the flow table of the switch,which may reduce the interaction between the controller and the switches, but it would consume a lot of resources of the TCAM table, and need to modify the internal structure of the switch, so it was difficult to realize. In addition, if the controller was attacked or failed, the network would be out of control due to the simple processing of the underlying devices. Therefore, some researchers proposed the distributed SDN architectures, such as Onix [29] and Hyperflow [30]. The distributed SDN architectures can effectively solve the problems of single controller failure, overload and scalability in the centralized architectures. However, when the load imbalance occurred among the multiple controllers, it would also reduce network performance and increase the delay. In this following, this paper summarizes, induces and analyzes the load balancing schemes in the distributed multiple controllers architectures.

B. Load Balancing in the Distributed SD*N Architecture*

The distributed SDN architectures are divided to organize the controllers: flat and hierarchical[31]. In the flat, all the controllers are on the same layer. In the hierarchical, the controllers are located at different layers.

1) Load Balancing in the Flat Architecture

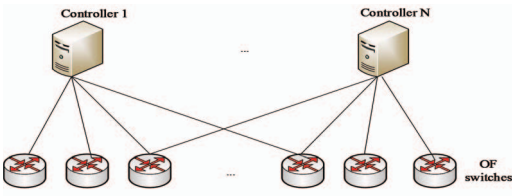

Figure 2. The Flat Architecture

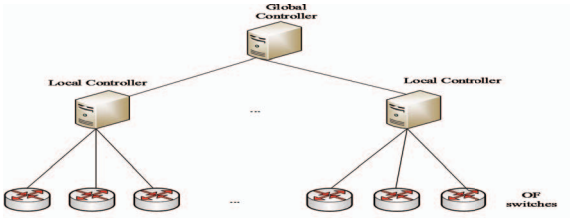

Figure 3. The Hierarchical Architecture

a) Load Balancing based on Switch Migration

Although the OpenFlow protocol does not explicitly specify the switch migration protocol, the OpenFlow1.3 protocol clearly states that the controller has three roles: master, slave and equal. The master/equal controller has full control to the switch, and the slave controller has only read permission to the switch[32]. When the switch connect to multiple controllers, the switch has only one master controller and multiple slave controllers. When the switch needs to be migrated, the switch needs to choose a slave controller as its new master controller and the old master controller changes to the slave mode.

The ElastiCon[33] scheme migrated some of switches connected to the overload controllers to the lighter load controller according to the switch migration protocol, and realized the dynamic load balancing by continually optimizing the load of the switches to the controllers. When the load exceeded the capacity of all the controllers, ElastiCon added new controller and triggered the switch migration protocol; Similarly, when the load reduced to a certain level, it shut down some controllers to reduce the cost. This scheme could balance the load of controllers through migrating the switch dynamically, but when the load was unstable, it would increase the cost and latency to repeat calculation and migration.

The Pratyaastha[34] scheme divided the data flows into different regions according to the applications, and the different regions was separately corresponded to different controllers. Each controller had different flow processing function and reported flow arrival rate of the switches and the state of storage partition in real-time. When the load was imbalance, it used the switch migration protocol in the[33]. It was convenient for the controller to process the requests according to the types of data flows, and reduced the time of flow created. However, the migrations would be more complex and increase the cost and latency when multi-controller were load imbalance.

A load balancing scheme based on role was proposed in [35]. By counting the flow number of requesting to be processed, the scheme migrated the switch generating most of the flow requests over a period of time to the idle controller. However, the paper did not explain how to deal with the problem of all controllers overload or light load, and it would migrate switch frequently when the load was unstable.

In order to solve the problems of low efficiency and high cost caused by the switch migration, PASMM [36] optimized the switch migration problem to the auction problem of the remaining resources of the controller. In this mechanism, the light load controller acted as an auctioneer to auction its remaining resources, and the migrated switch acted as the bidder. By improving the trading price of the over-demanded controller resources, the PASMM algorithm completed the auction process. Compared with the traditional switch migration strategy, PASSM had a better load balancing effect. But in the large scale network, when multiple controllers were overloaded at the same time, PASMM auctioned for many times and increased the migration time and migration cost of the switch.

b) Load Balancing based on Flow Redirection

The Balanceflow [37] would forward flows to X-controller by adding a flow-table item with controller-X action to the flow table of the switches, thus reducing the load of the original controller. Balanceflow could flexibly handle the flow requests, but the super controller may become the bottleneck of the entire network when it frequently handled the requests from the ordinary controllers.

P. P. Lin et al. proposed a hierarchical structure of multi-controller cooperation, named MSDN [38]. In the MSDN, the load balancer used the different strategies to split a large number of initialization flows and sent them to different controllers, and reduced the load of the controllers. The scheme was mainly aimed at the heavy load on the controllers caused by a lot of accesses of the switches to the controllers when the network began to run.

The ASIC system was proposed in [39],which consisted of three parts: load balancer, controller cluster and a distributed data sharing. The packets arrived at the load balancer firstly, and the load balancer executed the relevant algorithm to select a suitable controller to process the packets. Then, the controller updated the flow table according to the distributed global network view and completed the requests process of the packets. This scheme mainly solved the controller overloaded caused by a large number of requests in the initial state. In addition, the load balancer limited the network performance.

F. Cimorelli et al. proposed a distributed load balancing algorithm for the control traffic based on the game theory,which allowed the controllers to converge to a specific equilibrium, known as Wardrop equilibrium [40]. In the equal mode, all the controllers received and responded to messages of the switches, so the switches could judge the controllers' state according to the delay function and select an available controller to make the data flows to the Wardrop equilibrium, then achieved the rational use of resources, cost minimization, delay minimization.

c) Other Load Balancing

V. Yazici et al. proposed an algorithm to add or remove the controllers dynamically, which could flexibly manage the controller pools[41]. This scheme divided the whole network into two IP networks: controller-controller network and controller-switch network. Switches and controllers were mapped by IP aliases, so that a switch could be controlled only by a controller. When the controller failed, the switch migration may make the entire network more complex and difficult to manage.

H. Y. Fu et al. proposed a multi-controller model with dormancy mechanism [42]. When the load in the control plane was lighter, some idle controllers were allowed to enter into the dormant state for energy saving. But it only responded to the controllers at the light load, and did not handle the state while the controllers were overloaded.

M. F. Bari et al. realized the redeployment of the controller and the switch by dynamically changing the number and position of the controller in the different network conditions[43]. But this method was not suitable for the existing network topology, and did not explain the factors should be considered as migration .

2) Load Balancing in the Hierarchical Architecture

There are two forms in the hierarchical architecture: (1) In the triangle architecture, the controllers are divided into some local controllers and a global controller,as shown in Fig. 3; (2) In the inverted triangle architecture, there exists an intermediate layer between the controllers and the switches, as shown in Fig. 4 .

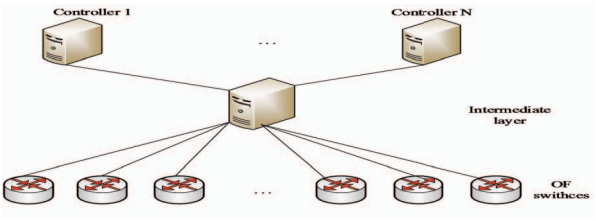

Figure 4. Inverted Triangle Architecture

a) Load Balancing in the Triangle Architecture

S. Hassas et al. proposed a two layer controller architecture Kandoo, the upper layer was a root controller, and the lower layer were local controllers [44]. The requests were processed by the local controller firstly, if the local controller could not handle, it was sent to the root controller which made the decision and sent back to the switch. The local controller would directly processed the request without accessing to the root controller when there was a similar request. The local controller could reduce the processing burden on the root controller, but the network performance would reduce when the root controller was overload or failure.

H. Yao et al. proposed a hybrid load balancing method, called Hybridflow [45].In the Hybridflow, the super controller managed multiple controller-cluster, and each cluster consisted of a plurality of controllers and switches. A double threshold method was proposed in this paper to determine whether the load imbalance occurred in the external or the internal of the controller-cluster. The scheme could effectively solve the phenomenon of load imbalance in the cluster, but did not consider how to resolve the overload of the super controller.

b) Load Balancing in the Inverted Triangle Architecture

A. Basta et al. proposed a hypervisor model which added a hypervisor layer between the controllers and the switches [46]. The hypervisor layer consisted of three parts: hypervisor instance, hypervisor management and hypervisor proxy. When the controller-cluster was overload, the hypervisor instance of the hypervisor layer would be migrated according to the control path migration protocol proposed in the paper. The controllers were not aware of the migration of the hypervisor instance because the hypervisor proxy shielded the migration. This scheme could effectively hide the differences of the underlying switches, and the controller needed not to know the change of the underlying. However, the paper did not mention of how to handle a hypervisor instance when it was overload.

R. Sherwood et al. proposed the FlowVisor which sliced the traffic among multi-controller, and each slice could have different forwarding strategies and respectively correspond to different controllers[47]. M. Koerner and O. Kao proposed a multi-service load balancing scheme by using the FlowVisor in which the different controllers controlled different servers and selected different load balancing methods[48]. This scheme could make different processing for different services, but could not solve the problem of the FlowVisor was overload.

In the centralized SDN,load balancing studies focused on the data plane which could effectively improve the network performance and reduce congestion. In the distributed SDN ,the researches of load balancing mainly focused on the control plane in which included switch migration strategy and flow redirection strategy. These strategies could solve the load imbalance among multiple controller to improve the quality of service and reduce the response delay, but these strategies still exist some problems. It is important for the development of SDN to further study and improve these schemes.

IV. CONCLUSION

Compared with the traditional network architectures, the SDN networks have shown great advantages and become a research hotspot in recent years, but load imbalance is one of the unresolved problems. Based on the investigations and researches to the current load balancing strategies in SDN, this paper classifies load balancing schemes, analyzes the advantages and disadvantages of them, and it can provide guidance for other researchers in this field. Future we will continually study the load balancing schemes in the distributed SDN architectures and look for a better load balancing strategy to better reflection the advantage of the SDN architecture and promote the further development of the SDN.

ACKNOWLEDGMENT

This work was supported by the Inner Mongolia Autonomous Region Natural Science Foundation (Grant No. 2012MS0930), and the Scientific Research Project of the Inner Mongolia Autonomous Region Education Department (NJZY12032).

REFERENCES

[1] N. McKeown, “Software-Defined networking”, INFOCOM keynote talk, 2009, vol. 17, pp. 30-32.

[2] D. Kreutz,F. M. V. Ramos,P. Verissimo, “Towards secure and dependable software-defined networks”, Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking .Hong Kong, China, ACM, 2013, pp. 55-60.

[3] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, “Software-Defined Networking: A Comprehensive Survey,” Proceedings of the IEEE, 2014, vol. 103, pp. 10-13.

[4] Q. Y. Zuo, M. Chen, G. S. Zhao, C. Y. Xing , G. M. Zhang , P. C. Jiang, “Research on OpenFlow-based SDN technologies,”Journal of Software, 2013 , vol. 24, pp. 1078－1097 [Journal of Software China, p.1078-1097,2013].

[5] S. Jain , A. Kumar, et al, “B4: experience with a globally-deployed software defined wan,” ACM SIGCOMM 2013 Conference on SIGCOMM. ACM, 2013, pp. 3-14.

[6] A. Singh, J. Ong, A. Agarwal , et al, “Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network,” ACM Conference on Special Interest Group on Data Communication. ACM, 2015, pp. 183-197.

[7] C. Y. Hong ,S. Kandula, R. Mahajan, et al., “Achieving high utilization with software-driven WAN,” ACM SIGCOMM 2013 Conference on SIGCOMM.ACM, vol.43, pp.15-26.

[8] B. Heller, S. Seetharaman, P. Mahadevan, et al., "ElasticTree: saving energy in data center networks," Usenix Conference on Networked Systems Design and Implementation. USENIX Association, 2010, pp.17-17.

[9] A. Gudipati, D. Perry, EL. Li, S. Katti, et al., "SoftRAN:software defined radio access network," ACM SIGCOMM Workshop on Hot Topics in Software Defined NETWORKING. ACM, 2013, pp.25-30.

[10] C. K. Zhang, Y. Cui ,H. Y. Tang ,J. P. Wu . "State-of-the-Art survey on software-defined networking (SDN)," Journal of Software, 2015,vol. 26, pp. 62−81[Journal of Software China, p. 62-81, 2015].

[11] T. Q. Zhou, Z. P. Cai, J. Xia, M. Xu, "Traffic engineering for software defined networks," Journal of Software, 2016, vol. 27, pp. 394−417 [Journal of Software China, p. 394-417,2016].

[12] S. Scott-Hayward, S. Natarajan, S. Sezer, "A Survey of Security in Software Defined Networks," IEEE Communications Surveys & Tutorials, 2016, vol. 18, pp. 623-654.

[13] A. Blen, A. Basta, M. Reisslein, "Survey on Network Virtualization Hypervisors for Software Defined Networking," IEEE Communications Surveys & Tutorials, 2016, vol. 18, pp. 655-685.

[14] N. Handigol, S. Seetharaman, M. Flajslik, N. Mckeown, R. Johari, "Plug-n-Server:load-balacing Web traffic using OpenFlow," In ACM Sigcomm Demo,2009, pp. 268- 270.

[15] N. Handigol, S. Seetharaman, M. Flajslik, N. Mckeown, R. Johari, "Aster* x: Load-Balancing Web Traffic over Wide-Area Networks," Geni Engineering Conference. 2010.

[16] R. Wang, D. Butnariu, J. Rexford, "OpenFlow-based server load balancing gone wild," Usenix Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services. USENIX Association, 2011, pp. 12-12.

[17] W. Chen, Z. Shang, X. Tian, H. Li, "Dynamic server cluster load balancing in virtualization environment with openflow," International Journal of Distributed Sensor Networks, 2015.

[18] A. F. R. Trajano, M. P. Fernandez, "Two-phase load balancing of In-Memory Key-Value Storages through NFV and SDN," IEEE, 2015, pp. 409-414.

[19] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," NSDI. vol. 10, pp.19, 2010.

[20] C. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm," RFC Editor, 2000.

[21] M. Alizadeh, T. Edsall, S. Dharmapurikar, et al., "CONGA: distributed congestion-aware load balancing for datacenters," ACM Conference on SIGCOMM. ACM, 2014, pp. 503-514.

[22] N. Katta, M. Hira , C. Kim , et al., "HULA: Scalable Load Balancing Using Programmable Data Planes," the Symposium. 2016, pp. 1-12.

[23] Y. Li, D. Pan, "OpenFlow based Load Balancing for Fat-Tree Networks with Multipath Support," IEEE International Conference on Communication(ICC), 2013

[24] F. Carpio, A. Engelmann, A. Jukan, "DiffFlow: Differentiating Short and Long Flows for Load Balancing in Data Center Networks," 2016.

[25] R. Gandhi, H. H. Liu, Y. C. Hu, et al., "Duet: cloud scale load balancing with hardware and software," ACM, 2014.

[26] J. Li, X.. Chang, Y. Ren, Z. Zhang, G. Wang, "An Effective Path Load Balancing Mechanism Based on SDN," IEEE, International Conference on Trust, Security and Privacy in Computing and Communications. IEEE, 2014, pp. 527-533.

[27] M. Yu, J. Rexford, M. J.Freedman, J. Wang, "Scalable Flow-based Networking with DIFANE," ACM SIGCOMM Computer Communication Review,2010, vol. 41, pp. 351-362.

[28] A. R. Curtis, J. C. Mogu, J. Tourrilhes, et al. "DevoFlow:scaling flow management for high-performance networks," ACM SIGCOMM 2011 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Toronto, On, Canada, August. DBLP, 2011, pp. 254-265.

[29] T. Koponen, M. Casado, N. Gude, et al, " Onix : a distributed control platform for large-scal production networks," In OSDI , 2010, pp. 1-6.

[30] A. Tootoonchian, Y. Ganjali, "HyperFlow: a distributed control plane for OpenFlow," Internet Network Management Conference on Research on Enterprise NETWORKING. USENIX Association, 2010.

[31] S. Schmid, J. Suomela, "Exploiting locality in distributed SDN control," ACM SIGCOMM Workshop on Hot Topics in Software Defined NETWORKING. ACM, 2013, pp. 121-126.

[32] Consortium O F. OpenFlow switch specification[J]. 2015.

[33] A. Dixit, H. Fang, S. Mukherjee, T.V. Lakshman, R.R. Kompella, "ElastiCon: an elastic distributed sdn controller," Proceedings of the tenth ACM/IEEE symposium on Architectures for networking and communications systems.ACM, 2014, pp. 17-28.

[34] A. Krishnamurthy, S. P. Chandrabose, A. Gember-Jacobson, "Pratyaastha: an efficient elastic distributed SDN control plane," The Workshop on Hot Topics in Software Defined NETWORKING. ACM, 2014, pp. 133-138.

[35] Y. P. Yu , H. Qin, "Research on load balancing strategy of controller in OpenFlow network," Network Security Technology and Application, 2015, pp. 6-7.[Network Security Technology and Application China, p. 6-7, 2015].

[36] F.Y. Chen, B.Q. Wang, B.W. Wang, Z. M. Wang, "Progressive auction based switch migration mechanism in software define network," Journal of Computer Applications, 2015, vol. 35, pp. 2118-2123.[Journal of Computer Applications, 2015].

[37] Y. Hu, W. Wang, X. Gong, X. Que, S. Cheng, "BalanceFlow: Controller load balancing for OpenFlow networks"[C]// Cloud Computing and Intelligent Systems. IEEE, 2012, vol. 2, pp. 780-785.

[38] P.P. Lin, J. Bi , H.Y. Hu, X. K. Jiang, "MSDN : a Mechanism for Scalable Intra-domain Control Plane in SDN," Journal of Chinese Computer Systems, 2013, vol. 34, pp. 17-20.

[39] P.P. Lin, J. Bi, H. Hu. "ASIC:an architecture for scalable intra-domain control in OpenFlow," International Conference on Future Internet Technologies. 2012, pp. 21-26. [International Conference on Future Internet Technologies China, p. 21-26, 2012].

[40] F. Cimorelli, F. D. Priscoli, A. Pietrabissa, et al., "A distributed load balancing algorithm for the control plane in software defined networking," Mediterranean Conference on Control and Automation. 2016, pp. 1033-1040.

[41] V. Yazici, M. O. Sunay, A. O. Ercan, " Controlling a Software-Defined Network via Distributed Controllers," Eprint Arxiv, 2014.

[42] H.Y. Fu, J. Bi, J.P. Wu, Z. Chen, Wang K, Luo M. "A Dormant Multi-Controller Model for Software Defined Networking," China Communications, 2014. [China Communication,2014].

[43] M. F. Bari, A. R. Roy, S. R. Chowdhury,Q. Zhang, "Dynamic Controller Provisioning in Software Defined Networks," International Conference on Network and Service Management. IEEE, 2013, pp.18-25.

[44] S. Hassas Yeganeh, Y. Ganjali, " Kandoo: a framework for efficient and scalable offloading of control applications," The Workshop on Hot Topics in Software Defined Networks. ACM, 2012, pp. 19-24.

[45] H. Yao, C. Qiu, C. Zhao, L. Shi, "A multicontroller load balancing approach in software-defined wireless networks," International Journal of Distributed Sensor Networks, 2015, pp. 1-8.

[46] A. Basta, A. Blenk, H. B. Hassine, "Towards a dynamic SDN virtualization layer: Control path migration protocol," International Conference on Network and Service Management. 2015, pp. 354-359.

[47] R. Sherwood, G. Gibb, K. K. Yap, et al. "Can the production network be the testbed?," OSDI 2010, vol. 10, pp. 1-6.

[48] M. Koerner, O. Kao, "Multiple service load-balancing with OpenFlow," IEEE, International Conference on High PERFORMANCE Switching and Routing. IEEE, 2012, pp. 210-214