SDN-BASED LOAD BALANCING STRATEGY FOR SERVER CLUSTER

Hailong Zhang[1], Xiao Guo[2]

[1]School of Information Engineering, Communication University of China, Beijing 100024, China
[2]Computer and Network Center, Communication University of China, Beijing 100024, China
zhlcuc@163.com, xguo@cuc.edu.cn

Abstract: Growing network traffic brings huge pressure to the server cluster. Using load balancing technology in server cluster becomes the choice of most enterprises. Because of many limitations, the development of the traditional load balancing technology has encountered bottlenecks. This has forced companies to find new load balancing method. Software Defined Network (SDN) provides a good method to solve the load balancing problem. In this paper, we implemented two load balancing algorithm that based on the latest SDN network architecture. The first one is a static scheduling algorithm and the second is a dynamic scheduling algorithm. Our experiments show that the performance of the dynamic algorithm is better than the static algorithm.

Keywords: Load balancing; SDN; OpenFlow; Server cluster

1 Introduction

The rapid development of Internet technology makes the server cluster of large Internet service providers facing more severe challenges. As the growth of users and network bandwidth, server need to handle a large number of access request in a short time. If the server cannot timely process user access request, extending the user's waiting time, it will affect the user experience, greatly reduce the Quality of Service. Questions like these make server become the new bottleneck in network and force researchers began to study how to improve server performance.

To improve server performance, enterprises have adopted a number of measures, such as improve CPU's processing speed, increase server's cache capacity, use high speed disk array, as well as build server cluster [2]. Simply upgrade the hardware system will not only cause the existing resources idle, but when the business continued to expand, companies will face the same difficult situation. By establishing the server cluster, companies can forward access request to multiple servers. This method improves the performance of the server in a certain extent. However, such solution also has a new problem that when the server cluster receives an access request, which server should response. If the control system cannot assign access request reasonably, the load imbalance situation between these servers will appear. Based on this, researchers have proposed the load balancing technology.

978-1-4799-4719-5 /14/$31.00 ©2014 IEEE

The common method of load balancing is to deploy one load balancer in front of the server cluster, thus it can reasonably allocate load to several server in order to make full use of the server resource. Load balancing technology can effectively solve the problem of load imbalance between servers, thereby reducing the response time of access request and improving system throughput and fault tolerance. The classical load balancing techniques have to use expensive hardware devices and cannot achieve precise control of the traffic load. These limitations make the traditional load balancing technology is not suitable for large-scale application. The emergence of Software Defined Network [4] (SDN) technology offers managers a traffic management technology that has advantage of low cost and flexible operation.

SDN is a new approach for computer networking that allows network administrators to manage network services through abstraction of lower level functionality. It separates the control plane and the forwarding plane of traditional network architecture. The controller running on the control plane can achieve the control objective for data forwarding through managing the switches' flow table. SDN requires southbound interface for the control plane to communicate with the data plane. OpenFlow protocol is the most suitable southbound interface until now. OpenFlow was first proposed in [1] as a way to enable researchers to conduct experiments in production networks. Because OpenFlow protocol can realize centralized control and efficient data transmission, it will get more extensive application.

In this paper, we implemented two classic load balancing algorithms. In the fourth section, we create a virtual network topology and compared the performance differences of two algorithms in response time.

The remainder of this paper is structured as follows. In Section II, we give an overview of the load balancing technology and OpenFlow technology; briefly summarizes the research status of OpenFlow-based load balancing. Section III introduced the implement of our two algorithms in more detail. The results about our experiments are described in Section IV. We summarize the main contribution of this paper in Section V.

2 Related work

2.1 Load balancing technology

Load balancing technology is mainly used in distributed systems to improve the overall performance of the cluster [5, 6]. It can optimize the allocation of resources, minimize response time and maximize the throughput of system.

Figure 1 is a model diagram of the server load balancing. Instead of being sent directly to the server cluster, the access request of clients is first sent to the load balancer through Internet. The load balancer selects an optimal server by using specified algorithm and forwards the request to the server.

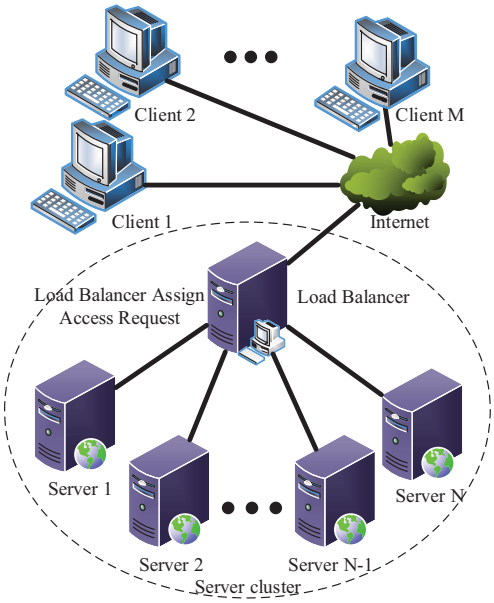

Figure 1 Model diagram of the server load balancing

Load balancing algorithms can usually be divided into static algorithms and dynamic algorithms [3, 7]. Static load balancing algorithm does not consider the load status of server, so it is suitable for the condition that load status can be predicted in advance. The static algorithms mainly include Round-Robin algorithm, Ratio algorithm and Priority algorithm.

- Round-Robin: Access request are sequentially allocated to each server according to the arrival sequence.

- Ratio: In this algorithm, according to the preset ratio, each server get the corresponding proportion of load.

- Priority: When the load is unbalanced, the backup server will be enabled.

Dynamic load balancing algorithm refers to that the load balancer can dynamically distribute the load to the optimal server according to the load status of the server. The typical algorithms mainly include Least-Connections algorithm, Predictive algorithm and Response-Time algorithm.

- Least-Connections: The load balancer periodically detect the number of connections that has established between the server and the user. When a new access request arrives, the load balancer forwards this request to the server that has the least connections.

- Predictive: Through the predictive model, controller can predict the load status of the server in the next period.

- Response-Time: Load balancer estimate the load status of each server by sending a probe instruction (such as Ping).

Because of the limitations of traditional networks, the development of load balancing technology has encountered bottlenecks gradually.

- Customized load balancer: Most load balancers are customized for a particular application or product. Their reusability of code and applicability of different architecture is poor.

- Complex hardware and software systems: Complex software or hardware system used for load balancing increases the operation cost of enterprise.

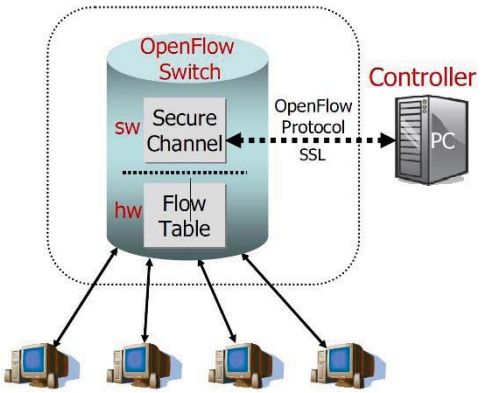

Figure 2 OpenFlow architecture diagram

2.2 Software defined network technology

As the improvement of functionality and performance, traditional network inevitably exposed many drawbacks. In order to achieve better service, more and more solutions are added to the architecture of router. These behaviors caused routing system large and fat and also resulted in the difficulties of improve the network performance. To address these limitations, Software Defined Networking (SDN) was proposed in 2003. SDN architecture is a new network architecture. It separates the control plane and the forwarding plane of traditional network architecture. The controller running on the control plane can achieve the control objective for data forwarding through managing the switches' flow table. OpenFlow protocol is the most suitable standard to implement SDN architecture until now. The main feature of OpenFlow is its programmability. By programming the controller, researchers can achieve their required network functionality. OpenFlow architecture diagram is shown in Figure 2.

OpenFlow network consists of three parts: OpenFlow switch, FlowVisor and Controller. OpenFlow switch takes forwarding function of data flow. FlowVisor realize the network virtualization. Controller can control the whole network.

The OpenFlow switch itself holds a flow table which stores flow entries. A flow entry consist of three fields: (1) A packet header that defines the flow, (2) The action, which defines how the packets should be processed, and (3) Statistics, which keep track of the number of packets and bytes for each flow, and the time since the last packet matched the flow (to help with the removal of inactive flows).

When a packet arrives at the OpenFlow switch, it will first be checked. If the packet matches a flow entry in a flow table, the corresponding action is executed. These actions include forward packet to port, forward packet to a controller or drop the packet. If a packet does not match a flow entry in a flow table, the default behavior is to send packets to a controller via a packet-in message, another options is to drop the packet. This behavior could be set by the controller. When the controller receives a packet-in message, its behavior depends on the program. E.g., if we want to forward this packet, the controller could sent a flow-mod message to switch to add a flow entry in switch's flow table. Researchers can send stats-request message to query statistics of packets or flow entries.

Currently, there are several OpenFlow controllers available, e.g., NOX [11], Beacon, Floodlight [18], Pox, and OpenDaylight [13] etc. Some are Open-source and others are commercial and proprietary. In general it is possible to program an own controller using the OpenFlow specification [14]. The Floodlight Open SDN Controller is an enterprise-class, Apache-licensed, Java-based OpenFlow Controller. It is Java-based and intended to run with standard jdk tools and can optionally be run in Eclipse. The Floodlight core architecture is modular, with components including topology management, device management (MAC and IP tracking), path computation, infrastructure for web access (management), counter store (OpenFlow counters), and a generalized storage abstraction for state storage. The Floodlight controller realizes a set of common functionalities to control and inquire an OpenFlow network, while applications on top of it realize different features to solve different user needs over the network. Floodlight can be configured to "load" different modules to accommodate different applications. Currently, applications have been built to work with Floodlight through three main APIs, REST applications, Module applications, OpenStack [17] applications. There are already more mature module can be used. This facilitates developers to develop applications.

2.3 Research status

Due to the high cost of traditional network load balancing technology, it is difficult to get widely used. The emergence of OpenFlow technology offers managers a traffic management technology which has advantages of low cost and flexible operation. There are already some studies focus on load balancing problem in OpenFlow network, such as Plug-n-Serve [9] and SBLB load balancing strategy [12]. Plug-n-Serve proposed an OpenFlow based server load balancing system that effectively reduce response time of web services in unstructured networks built with cheap commodity hardware. Their paper did not introduce specific algorithm for load balancing. SBLB load balancing strategy collects the server load to determine which server the packets will be forwarded. Such implements have a higher complexity. Some studies proposed a load balancing solution to handle the load of multiple services by using multiple OpenFlow controllers [10]. It implemented a round robin load-balance algorithm as a plugin for the OpenFlow controller NOX. Other studies proposed a "partitioning" algorithm [11]. Their load-balancing architecture proactively maps blocks of source IP addresses to servers so client requests are directly forwarded through the load balancer with minimal intervention by the controller. Those static methods cannot achieve real load balancing.

Until now, there is not enough research work in OpenFlow-based load balancing technology. In this paper, we investigated how to load balance through OpenFlow technology and implemented two load balancing algorithm.

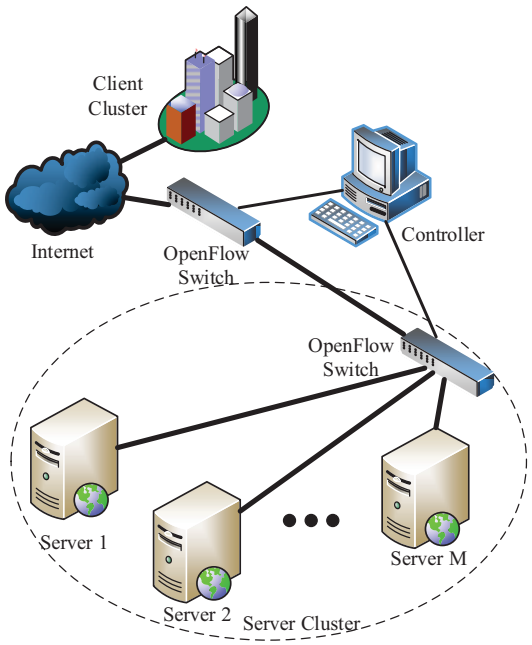

Figure 3 Structure diagram of OpenFlow load balancing

3 OpenFlow load balancing strategy

The key idea of OpenFlow load balancing concept is the replacement of the expensive and statically defined hardware components in clusters through OpenFlow controller using the local network infrastructure for implementation of the load balancing strategy. Figure 3

is the structure diagram of OpenFlow load balancing. The controller can control data transmission by operating flow table of OpenFlow switches.

We choose Floodlight as the controller. Floodlight provides developers with a number of modules. In the scheme, we have developed a load balancing module on the basis of existing modules. This module implements the Round-Robin algorithm and Least-Connections algorithm of traditional load balancing algorithms.

In our module, server cluster is assigned a virtual IP (VIP, 10.0.0.211 in our scheme) address. Servers also has its own real IP (RIP) address. All clients access server cluster through this virtual IP address. When a client's access request packets first arrives at the OpenFlow switch, it will be forwarded to the controller. Then it's handling procedure shown in Figure 4.

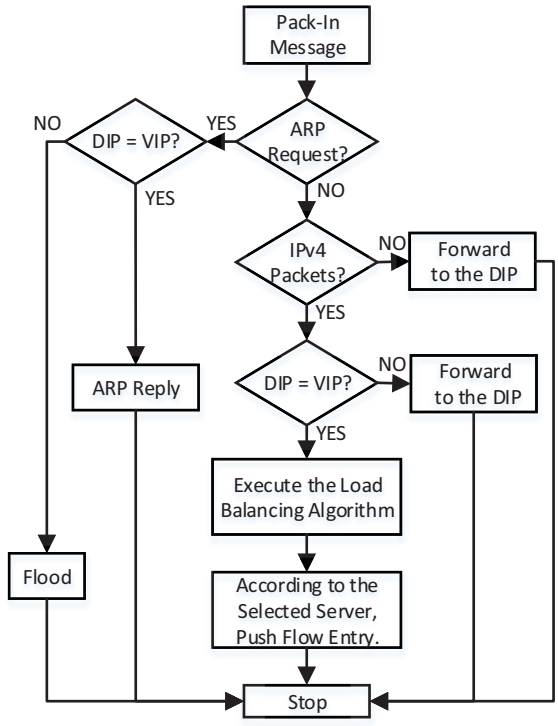

Figure 4 Handling procedure of access request packets

In Figure 4, we define DIP as the destination IP address of request packets. When the destination address of the packet is not equal to VIP, the packet will be treated as ordinary data packet for transmission; but if two of the address are equal, the packet will be forwarded to the server that computed by the load balancing algorithm. The specific forwarding process is divided into three steps:

1) Retrieve route: This will retrieve both in-bound and out-bound routes of packets.

2) Set flow-mod message: For the in-bound route, the action of flow entry is set to rewrite DIP of packet from VIP to server's real IP address; the match fields is set to match source IP address, in port and Ethernet type of the packet. For the out-bound route, the action of flow entry is set to rewrite source IP address of packet from server's real IP address to VIP; the match fields is set to match destion IP address, in port and Ethernet type of the packet.

3) Push flow entry: Use static flow entry pusher to push flow entry into the OpenFlow switches.

In this paper, we implemented two OpenFlow load balancing algorithms. There are OpenFlow Round-Robin algorithm and OpenFlow Least-Connections algorithm. The first algorithm is a static load balancing algorithm and the second algorithm is a dynamic load balancing algorithm. In section IV, we will prove that the performance of the dynamic algorithm is better than the performance of the static algorithm.

3.1 OpenFlow Round-Robin algorithm

When the controller is initialized, we first statistics all servers' information (IP address, MAC address, ID, port) in the cluster and stored it in a HashMap. When the module is calling OpenFlow Round-Robin algorithm, this algorithm will determine which server to use according to the last selected server's ID. This method ensures that all servers will be visited in a loop. The drawback of this algorithm is that it does not consider the actual load status of each server, and therefore cannot make full use of server resources; it cannot achieve real load balancing effect.

3.2 OpenFlow Least-Connections algorithm

Different scenarios have different interpretations of connections, researchers generally considered that a 5 tuple information group from interactive hosts determines a connection. The 5 tuple information group includes source address, source port, destination address, destination port and protocol type. In the scheme, we count the connections of each server by using the flow entries information in the switch. When forwarding the flow packets, depending on the setting of the match fields information (source IP address, in port, Ethernet type), access request from different clients will be pushed different flow entries. The module periodically query the contents of flow entries by sending a stats-request message to the switch that connected to the server cluster directly. Then our module statistics the connections of each server by matching the Ethernet type of match field and output port of action set in the flow entries.

The specific steps are as follows. When the module is initialized, it will start a thread runs every 5 seconds. The thread will send a stats-request message to query the flow table contents. When the controller receives a stats-reply message, it will calculate all of the connections number. Finally, the controller selects the server with the smallest connections and forwards the data packets.

4 Experimental evaluation

4.1 Experimental environment

We conduct experiments with Mininet [15] on Linux host by using the Floodlight controller. Mininet is a network simulator that can conveniently create a network topology which composed of many virtual OpenFlow switch. We use Mininet to model the experimental network topology, as shown in Figure 5. This network topology has 2 switches, 3 servers, 121 clients. VIP is set to 10.0.0.211. Performance of three servers is the same.

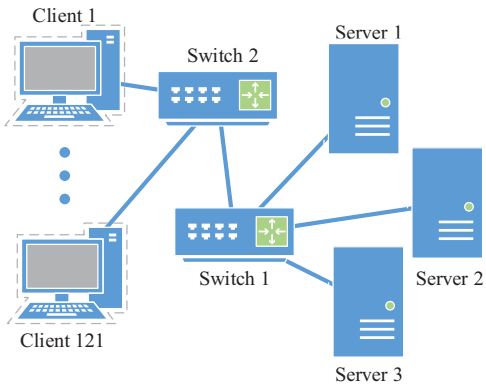

Figure 5 Experimental network topology

4.2 Measurement

Three servers provide a simple web service. The client uses httping [16] to simulate real web access and measure the response time. To improve accuracy, the client access the web server for 100 times and calculate the average response time.

4.3 Experimental program

In order to compare the performance of OpenFlow Round-Robin algorithm and OpenFlow Least-Connections algorithm, we designed three experiments. In different experiments, servers in server cluster have different load status. We measured the average response time of the three load status to prove that OpenFlow Round-Robin algorithm has better performance than OpenFlow Least-Connections algorithm. Load status of server cluster in three experiments are shown in Table I.

1) First experiment: In the first experiment, the load status of each server are the same. We set each server has 30 clinets to access.

2) Second experiment: In this experiment, different servers have different load status. Server 1 has 90 clients to access; server 2 has 30 clients to access; server 3 has no client to access.

3) Third experiment: Similar to the second experiment, the load status of each server are different. Server 1 has 120 clients to access while server 2 and server 3 has no host to access.

Table I Load status of server cluster

Experiment	Load status(clients)		
	Server 1	Server 2	Server 3
First experiment	30	30	30
Second experiment	90	30	0
Third experiment	120	0	0

In the experiment, we use an idle client to access the server cluster's web service and measure the average response time for access; We compare the performance of two different load balancing algorithm through this average response time.

4.4 Results

1) First experiment

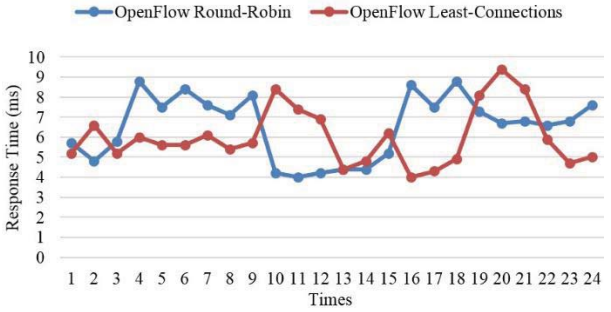

Figure 6 Average response time of first experiment

Figure 6 shows the average response time when we run two load balancing algorithms separately.

In Figure 6, we found that when run two algorithms separately, the response time of access request is almost the same. This is because the load status of all servers approximately equal. So no matter which server the access request is forwarded to, the response time is very similar.

2) Second experiment

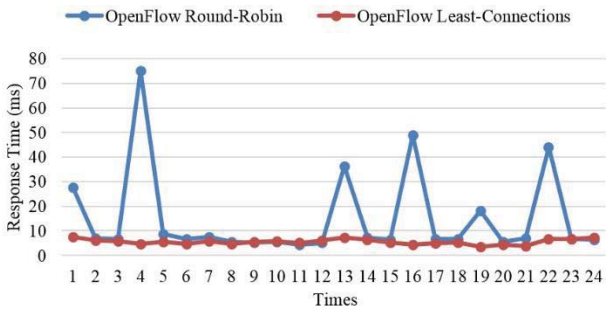

Figure 7 Average response time of the second experiment

Figure 7 shows the average response time when we run two load balancing algorithms separately.

We can clearly see that when the controller run the OpenFlow Round-Robin algorithm, the response time of access request is very unstable. Most of the time, response time are higher than the OpneFlow Least-Connections algorithm. This is because the load of

server 1 is relatively heavier while server 3 is lighter. When run Round-Robin algorithm, if the access request is forwarded to server 1, it is possible to cause congestion, thus resulting in increased response time. But when run Least-Connections algorithm, every access request will be forwarded to server 3. Because the load of server 3 is very light, the response time of the access request will be very small.

3) Third experiment

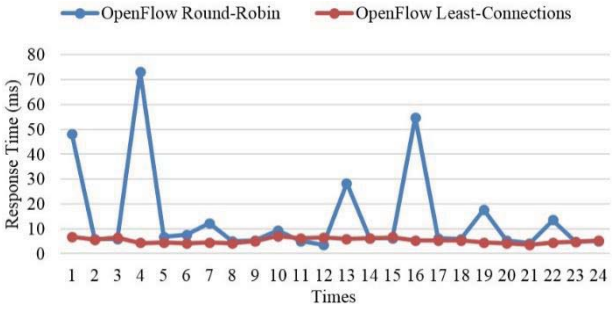

Figure 8 Average response time of the third experiment

Figure 8 shows the average response time when we run two load balancing algorithms separately.

The results in Figure 7 and Figure 8 are very similar. This is because in both experiments, the load status of the server cluster is substantially similar. However, because the load of server 1 is much heavier, when run the Round-Robin algorithm, the response time of access request may be longer.

It can be seen from the above results that for OpenFlow Round-Robin algorithm, when the load status of the server cluster varies greatly, the response time of access request is very unstable. This will seriously reduce the effect of load balancing. Compared to the Round-Robin algorithm, the response time of OpenFlow Least-Connections algorithm is shorter and the effect of load balancing is better. This is benefit from OpenFlow Least-Connections algorithm that it always able to forward load to server which has the least connections.

5 Conclusions

The traditional load balancing technology is expensive and inefficient. In order to replace the expensive load balancing devices in the network, our work proposed two OpenFlow load balancing algorithm that based on the latest SDN network architecture. The OpenFlow Round-Robin algorithm is the implementation of traditional Round-Robin algorithm. The OpenFlow Least-Connections algorithm is able to calculate the connections number of current server through querying the flow entries information of switch. According to the connections number, this algorithm can forward access request dynamically. Our experiments show that the performance of the OpenFlow Least-Connections algorithm is better than OpenFlow Round-Robin algorithm.

References

[1] Nick McKeown, Tom Anderson, Hari Balakrishnan, et al. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-4.

[2] Trevor Schroeder, Steve Goddard, Byrav Ramamurthy. Scalable web server clustering technologies. IEEE Network Magazine, 2000, 14(3): 38-45.

[3] Pushpendra Kumar Chandra, Bibhudatta Sahoo. Performance analysis of load balancing algorithms for cluster of video on demand servers. IEEE International Advance Conputing Conference, 2009: 408-412.

[4] Thomas D. Nadeau, Ken Gray. SDN: Software Defined Networks, O'Reilly Media, Inc. 2013.

[5] Wenyu Zhou, Shoubao Yang, Jun Fang, et al. Vmctune: A load balancing scheme for virtual machine cluster using dynamic resource allocation. International Conference on Grid and Cloud Computing, 2010: 81-86.

[6] Mei Lu Chin, Chong Eng Tan, Mohd Imran Bandan. Efficient load balancing for bursty demand in web based application services via domain name services. Information and Telecommunication Technologies, Asia-Pacific Symposium on, 2010: 1-4.

[7] Y. Zhang, H. Kameda, S.L. Hung. Comparison of dynamic and static load-balancing strategies in heterogeneous distributed systems. Computers and Digital Techniques, 1997. Proceedings. 1997,144(2): 100-106.

[8] Natasha Gude, Teemu Koponen, Justin Pettit, et al. NOX: towards an operating system for networks. ACM SIGCOMM Computer Communication Review, 2008, 38(3): 105-110.

[9] Nikhil Handigol, Srinivasan Seetharaman, Mario Flajslik, et al. Plug-n-Serve: Load-balancing web traffic using OpenFlow. ACM SIGCOMM Demo, 2009.

[10] Marc Koerner, Odej Kao. Multiple service load-balancing with OpenFlow. International Conference on High Performance Switching and Routing, 2012: 210-214.

[11] Richard Wang, Dana Butnariu, Jennifer Rexford. OpenFlow-based server load balancing gone wild. Hot-ICE, 2011. Proceedings. 2011.

[12] Zhihao Shang, Zhihao Shang, Qiang Ma, et al. Design and implementation of server cluster dynamic load balancing based on OpenFlow. International Joint Conference on Awareness Science and Technology and Ubi-Media Computing, 2013: 691-697.

[13] Open daylight. http://www.opendaylight.org.

[14] OpenFlow Switch Specification, Version 1.1.0. http://www.openflow.org/wp/documents.

[15] Mininet. http://mininet.org.

[16] httping. http://www.vanheusden.com/httping.

[17] Open stack. http://www.openstack.org.

[18] Project Floodlight. http://www.projectfloodlight.org/floodlitght.