Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 14, no. 4, April 2018, pp. 709-716
DOI: 10.23940/ijpe.18.04.p14.709716

SDN Load Balancing Method based on K-Dijkstra

Xiaohui Yang* and Lei Wang

School of Cyberspace Security and Computer, Hebei University, Baodin, 071002, China

Abstract

In order to solve the problem that control and forwarding are closely coupled in traditional network, the network lack of innovation and programmability, and the network management and maintenance difficulty, an SDN load balancing method based on K-Dijkstra is proposed. By using SDN technology to achieve the separation of control and forwarding, the controller is responsible for the global scheduling, making the network flow and task scheduling more flexible than the traditional network. Through the integration of flow management, traffic monitoring, dynamic load balancing and load calculation in the SDN control layer, the K-Dijkstra algorithm and the HRRF algorithm are combined in the load balancing module to solve the problem of path selection. The traffic environment also has a better load balancing effect, optimizing the control layer structure, improving network management efficiency and achieving dynamic load balancing of network traffic. The simulation results on Mininet show that the method can significantly improve network delay, packet loss and throughput compared with traditional networks.

Keywords: software definition network; dynamic load balancing; flow management

(Submitted on January 5, 2018; Revised on February 23, 2018; Accepted on March 27, 2018)

1. Introduction

With the development of the network showing a trend of fast, diversified services and wide branches, the traditional network structure and network equipment have gradually restricted the birth of new network technologies. The emergence of the new network architecture Software Defined Network (SDN) has provided a new research direction for the new Internet architecture [15]. Since the introduction of SDN [1], many research institutes and scholars have done a lot of meaningful research [5]. The OpenFlow network has been deployed on a large scale in many countries and educational institutions [4,8]. In recent years, through the work of scholars, the SDN security model has gradually matured and started to be deployed on a large scale in the field of security [2]. The working principle of SDN is to divide the network structure into the application layer, control layer and data layer. Causing the control function to be separated from the traditional network, switches and routers in the data layer achieve the forwarding function. The application layer provides a programmable interface, thereby reducing the workload of the switch and the router and increasing the network flexibility [16].

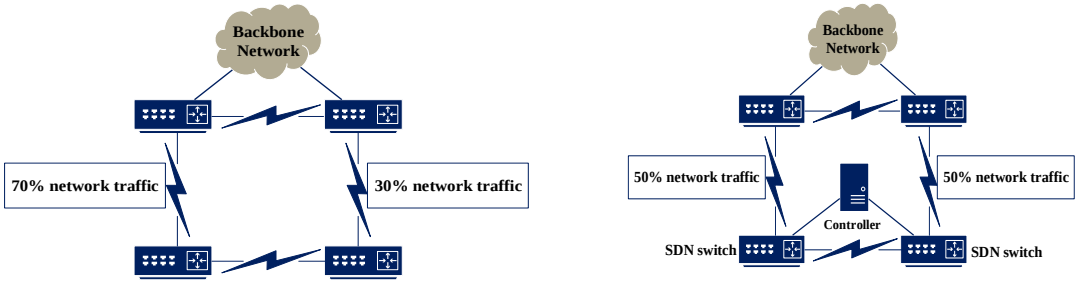
With the rapid growth of network users, more and more network application traffic into the network, data generated by various data servers makes serious server resource consumption, and the processing speed of server and the speed of memory access is gradually unable to satisfy the user's requirements. Therefore, the disadvantages of the traditional networks structure gradually appear [13]. Methods of traditional networks achieve load balancing include: random algorithm, polling algorithm, weighted polling algorithm and the minimum connection algorithm. Traditional networks use load balancer to achieve load balancing. Load balancer is expensive and complex. In addition, because all requests are delivered through a single hardware load balancer, the single point nature of the load balancer results in load balancing, and any failure on the device will cause the entire site to crash. In order to improve efficiency, traditional networks need to increase network bandwidth, upgrade network devices and re-purchase new infrastructure. This method is not desirable when funds are limited, and the network service quality will not greatly improve. In addition, the data forwarding and control in the traditional network architecture are closely coupled, so there are deficiencies in network innovation and programming, and the time for device development, testing and application is relatively long. New applications in the

* Corresponding author.
E-mail address: yxh@hbu.edu.cn

network need to be re-established in protocols and standards, and network management and maintenance are difficult. The emergence of SDN is expected to solve the above problems. In order to solve the problem of disorder of receiving end stream due to different delay of transmission path, Yang Yang [6] and others proposed a kind of dynamic routing algorithm F-TAM with fuse mechanism. By contrasting the proportions of control channels and data channels in a single domain, Xiaomao Wang [10] and others adopted a dynamic mix of source routing and direct delivery, so it will reduce the controller pressure and dynamically adjust the control plane. It also reduces the probability of inconsistent control logic when the flow table is issued. Ying Wang [11] and others proposed a load balancing mechanism consisting of four components: load measurement, load announcement, balance decision and switch migration. This mechanism realizes the load balancing by using multi-controller management and utilizing the capability of load announcement when the traffic is overloaded. Each controller can make balanced decision as soon as possible without relying on the load information provided by other controllers. Meanwhile, in order to reduce the communication load and processing load, which is caused by load announcement, a suppression algorithm is proposed to reduce the load announcement frequency. Although all literature takes advantage of the flexibility of SDN to solve the problem of out-of-order in the flow table issuance to varying degrees, the management load brought by the centralized management is under the multi-controller. In the aspect of path decision, the above literature fails to solve the problem of path selection when there are too many network nodes. In order to solve this problem, this paper combines the K-Dijkstra algorithm with the HRRF algorithm to solve the problem of path selection when the network nodes are overloaded and the traffic is overloaded.

2. SDN Technology Architecture Work

For the first time in 2007, a team led by Martin Casado, a Stanford student, implemented SDN. They try to add a centralized controller to the traditional network so that network administrators can directly monitor, forward the network traffic through the controller and apply it to the network devices; therefore, the entire network is safe control. Architecture is shown in Figure 1.



(a) Traditional network link load is not balanced (b) SDN network link traffic balance forwarding

Figure 1. Traditional network and SDN network link load comparison

Comparing with the traditional network, the SDN technology architecture is a new type of network structure, not a specific protocol. The traditional network needs to pass the router and the switchboard, and follow the TCP / IP agreement to carry on the data transmission. The agreement rule is written in the router and the switchboard, but writing the content outward is not transparent. This shows that in order to modify functions or add services to existing networks, the entire network protocol needs to be modified. Due to the huge workload, the traditional networks are difficult to move under the current rapid network development. The SDN separates control and forwarding, and provides an API interface that can be programmed by the user to set requirements on the controller. The data layer only needs to forward the data according to the instructions issued by the controller. This architecture enables the network to be fully transparent to users, enables users to plan the network environment and achieves the target functions according to their own needs[14]. SDN architecture is shown in Figure 2.

Data layer: Consists of a number of switches and routers that support the OpenFlow protocol. All the forwarding entries and packets of user data are processed and forwarded there. Southbound interface: A interface between the control layer and the data layer. The control layer delivers the forwarding policy to the data layer through the currently mainstream OpenFlow protocol. Control layer: The controller in the control layer which has the functions of coordinating the underlying network equipment, provides the whole network view, formulates the flow table forwarding strategy, manages the network topology structure, implementing updating and maintaining the equipment state information. Northbound interface: There is no standardization for this interface between the control layer and the application layer. However, the northbound interface provided by the application layer of the mainstream controller uses the REST API. Application layer: Contains a number of application software that enable third-party users to design software with different functions as required to manage the network, including load balancing, security, network operation monitoring, topology discovery and other services.

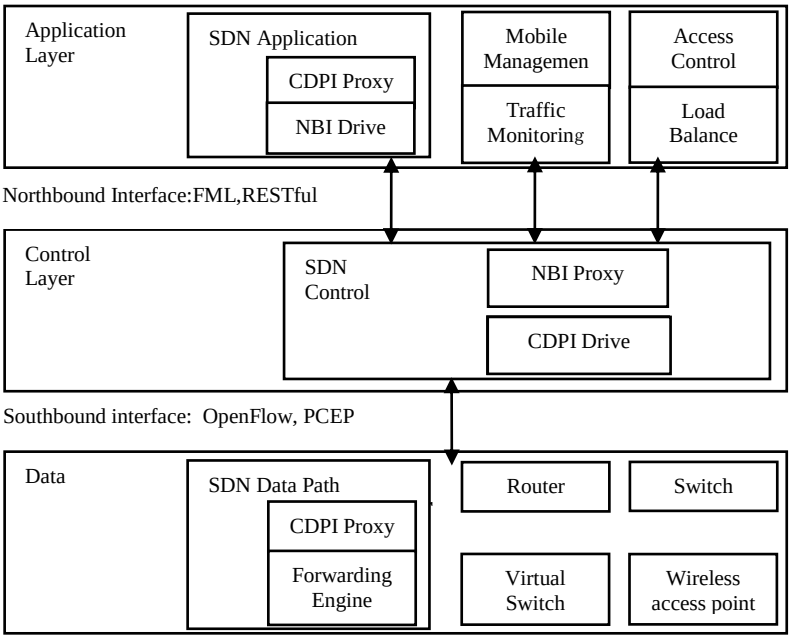


Figure 2. SDN architecture diagram

3. Load Management Design in SDN Architecture

This paper draws on the above system architecture, designs a new SDN flow control structure, and mainly consists of three parts: a number of servers and clients consisting of the underlying network, a number of OpenFlow protocol switches formed OpenFlow switching network, and SDN controller composed of streaming data forwarding decision-making center. This is shown in Figure 3.

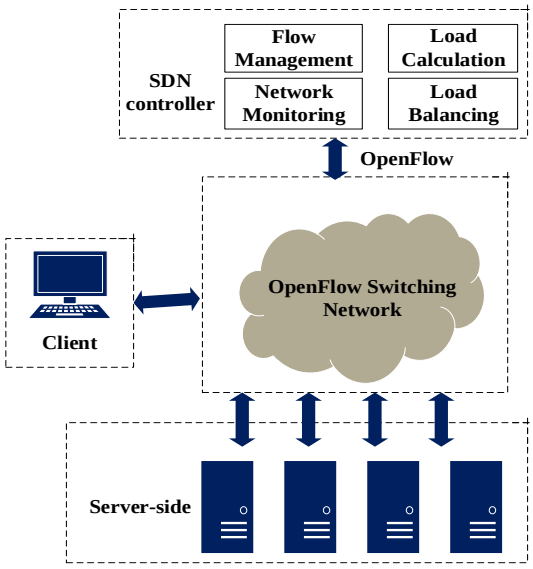


Figure 3. SDN architecture under a new load management structure

3.1. Flow Management Module

This module deploys and executes the path required by the flow during load balancing forwarding. When the controller delivers the load balancing forwarding policy, the flow management module is responsible for sending the forwarding policy in the form of a flow table to the switch in the data layer that supports the OpenFlow protocol. When data flows enter a designated switch in a flow table, the switch forwards the packet only according to the rules in the flow table.

3.2. Flow Monitoring Module

In the traditional network architecture, in order to achieve the statistical detection of a link traffic, all traffic on this link can only be assigned by port mirroring. This method makes the traditional network in the traffic detection detect the flow of great. In the SDN, the controller is responsible for controlling the entire network, so that real-time access to traffic information within the data layer switches. This centralized control makes SDN accurate, fast, convenient and low overhead in dynamic traffic monitoring [9].

3.3. Load Calculation Module

The significance of designing this module is that SDN takes full use of the advantages of statistical convenience over traffic monitoring in traditional network architectures. It can directly extract and analyze the traffic load collected by the traffic monitoring module of each link, calculate whether the current time needs to load balance the network. This module also can base on part of the link service needs. Current traffic conditions are used to make global judgments and reasonable analysis, in order to balance the distribution of network resources.

In order to measure the link load in the network at a certain moment, the concept of Coefficient of Variation (C.V) is introduced. Although the variance can represent the imbalance of network load, each calculation needs to obtain the flow information of each link. So, the calculation work is relatively complicated. The coefficient of variation only needs to obtain partial data randomly to measure the degree of data dispersion.

Coefficient of Variation formula:

C.V = (SD ÷ MN) × 100%

Among them, SD represents the standard deviation and MN represents the average. The standard for measuring load balancing as shown in Equation (1) and Equation (2):

$$\tau(t) = \sqrt{\frac{\sum_{i=1}^{n}[F_a(t) - F_i(t)]^2}{n}} \Big/ F_a(t) \tag{1}$$

$$F_a(t) = \frac{\sum_{i=1}^{n} F_i(t)}{n} \tag{2}$$

Where n is a randomly generated number used to obtain the load of n links, n can be taken multiple times. I is the service node number, $F_a(t)$ is the average load of n links. $F_i(t)$ is the load at moment i. The larger value of $\tau(t)$, the more uneven the load of each node, the worse the load balance. The smaller value of $\tau(t)$, the more uniform the load of each node, the better the load balance. When the value of $\tau(t)$ is equal to 0, the link is in perfect balance and the load condition is completely ideal. Therefore, the value of $\tau(t)$ is used to directly reflect the load on the network link as one of the parameters that triggers link load balancing.

3.4. Load Balancing Module

This module is the core module in the SDN controller. When the link load in the network is unbalanced ($\tau(t)$>threshold), this module is called to perform link load balancing. Traditional routing algorithm for load balancing mostly adopts Floyd or Dijkstra algorithm to obtain a shortest path, so that overloaded traffic in the network is diverted to other links to reach the link road load balancing purposes [7]. In addition to ensuring the optimal performance of a single link, whether the link is beneficial to the load balancing of the global link should also be taken into consideration. In this module using the K-Dijkstra algorithm for load scheduling, combined with High Response Ratio First(HRRF) algorithm to achieve the desired load balancing effect in the case of heavy traffic, such as metropolitan area network and wide area network.

4. SDN Load Balancing Method Based on K-Dijkstra Algorithm

4.1. The Basic Idea of Algorithm

This article refers to Martins Deletion Algorithm[12], and makes some improvements on the Dijkstra's shortest path algorithm. The core idea of K-Dijkstra algorithm is to delete an edge on the shortest path in the digraph. The weighted value w of the edge in the digraph is composed of the ternary function group $w=\alpha x+\beta y+\gamma z$ ($\alpha+\beta+\gamma=1$), where x represents the

average delay, *y* represents the average packet loss rate, and *z* represents the number of hop counts. The specific values are provided by the traffic monitoring module at the control layer. Through the application layer software, users or administrators can set the value of three parameters to achieve the current network delay, packet loss rate and the proportion of the number of hop counts. It shows that the smaller the *w*, the better the current load of the path, and the higher the priority. Then, the shortest path that can replace the deleted edge is found by calculation, and achieve the purpose of finding a backup link for load balancing.

If the current traffic is too large and has too many network nodes when several spare links are calculated by K-Dijkstra's algorithm, excessive backup links may appear in the calculation process through the K-Dijkstra algorithm. These excess backup links occur because the K-Dijkstra algorithm realizes the selection of a new path by deleting some nodes. As the number of deleted nodes increases, the node that the flow passes through will also increase. This will cause the network delay and the packet loss rate to increase. In order to solve this problem, it draws on the High-Response-Ratio First (HRRF) algorithm commonly used in job scheduling by CPU[3]. The controller detects the corresponding time of each link by sending a stream to all backup paths at regular intervals *t*. When the link receives the probe stream message, the link will encapsulate the stream into a Packet_In message and feed it back to the controller. The controller is determined by collecting the time t_{back} from sending the probe stream to receiving feedback messages from each alternate link. The smallest time interval is regarded as the link, which is most favorable to the current load condition among all alternative paths. At the same time, it no longer accepts the time t_{back} of the feedback message, and suspends the sending of the stream message to the congestion link instead of sending the message to the optimal path, to achieve the purpose of load balancing.

4.2. The Basic Description of The Algorithm

K-Dijkstra algorithm is described as follows:

- Step 1. The Dijkstra algorithm is used to calculate the shortest path with *m* as the root node and *s* as the end point in the directed graph G(N,A), and this path is denoted as S_n, $n=1$
- Step 2. If there is a candidate path that exists in this path at this time, and the value of *n* less than the maximum number of required shortest paths *N*, then let the current path $S=S_n$, go to Step 3; otherwise, the procedure ends
- Step 3. Starting from the first node, traversing all the nodes in the graph to find the node whose first degree of entry is greater than 1 is denoted as k_a , and if there is no extended node k_a' of k_a in all nodes, go to Step (4); otherwise find all the nodes following k_a , whose corresponding extended nodes are not in the first node of point set *P*, denoted as k_i , then go to Step (5)
- Step 4. Generate the expansion node k_a of k_a' and add it to the set of points *P*, at the same time, connect all the precursor nodes of k_a other than the previous node k_{a-1} in the path *S* to generate one edge to k_a' , and save the arc the same weight, add these arcs to arc set *A*. Calculate the shortest path from the initial node m to the extended node k_a' and that is denoted as $k_i=k_{a+1}$
- Step 5. Mark kc for all nodes traversed from k_a after path *S*, and perform the following operations:
  (1) Add an expansion node k_c' of k_c to the node set *P*
  (2) Connect the precursor node of all k_c in path *S* to an arc of its extended node k_c (except for the previous node k_{c-1} of k_c), and add these arcs to arc set *A* with the weights held constant
  (3) Calculate the shortest path from the start node s to k_c'. If an extended node k_{c-1}' exists in the previous node k_{c-1} of k_c in the path *S*, an arc connecting k_{c-1}' to k_c' is generated, and the weight is equal to the arcs (k_{c-1}, k_c)
- Step 6. Find the shortest path between the start node m and the current expansion node $t(n)'$ of the end node as the section *n* shortest path, and let $n=n+1$, and perform (2) and continue.

Expansion node: The last node on the basis of the collection to add the corresponding new node.
Precursor node: The previous node of a node in the shortest path.

The HRRF algorithm is described as follows:

- Step 1. Set the time interval Δt
- Step 2. Each interval Δt , send the probe stream *F* to the candidate paths *S1*, *S2*, ... , *Sn*.
- Step 3. Start the timer after sending probe stream *F*
- Step 4. After receiving the probe stream F, the alternative paths *S1*, *S2* , ... , *Sn* encapsulate the stream into a Packet_In message and feed it back to the controller

- Step 5. After receiving the first Packet_In message fed back by the alternative path S_f, the controller stops receiving other feedback messages, and after modifying the flow table information, forwards the stream to be forwarded through the path S_f. If the controller receives multiple Packet_In messages at the same time, the controller forwards the stream to be forwarded randomly
- Step 6. Repeat Step 2-Step 5 after the time interval.

4.3. Algorithm Feasibility Analysis

Combined with the above two algorithms, K-Dijkstra algorithm searches globally to find the link most conducive to the current load situation. When the current traffic is too large, it is optimized by HRRF algorithm. Then, it can also have a better load balancing effect in a larger traffic environment. When $\tau(t)$ is not equal to 0, the load is unbalanced. The flow scheduling policy in the load balancing module is triggered to dynamically adjust the overall situation. Instead of forwarding the flow to the overload link, the flow is scheduled to the chain that is most conducive to the current load, which can achieve the purpose of balancing the load of the link.

5. Simulation Test and Experimental Result

Experimental environment is: Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz 4.00GB of Memory. The operating system is Ubuntu14.02. The controller adopts OpenDayLight controller to form a controller cluster, which is used to form the control plane. The bottom data layer uses Mininet to deploy the network environment to generate a simulated network topology map. The host IP address of the running controller and the switch are 10.188.14.139. In the same network environment, the traditional network the polling algorithm, the randomized algorithm and the SDN load balancing method based on K-Dijkstra proposed in this paper are respectively executed 100 times under different loadings. The contents of the testing are the network delay, the network packet loss rate, the network throughput and the execution time. Take the average value, and compare the results obtained by taking different values of the three parameters α, β, γ in the weighted value w of the directed graph respectively.

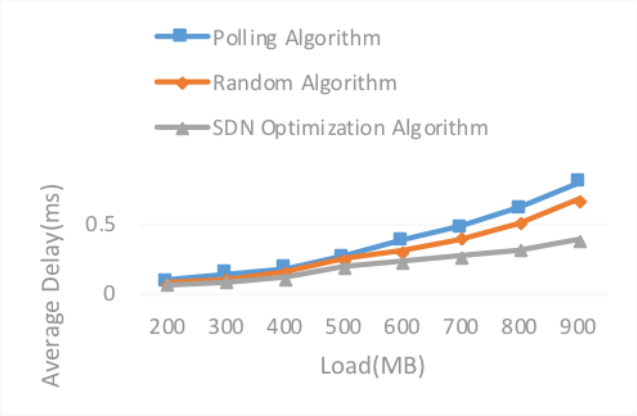


Figure 4. Average delay

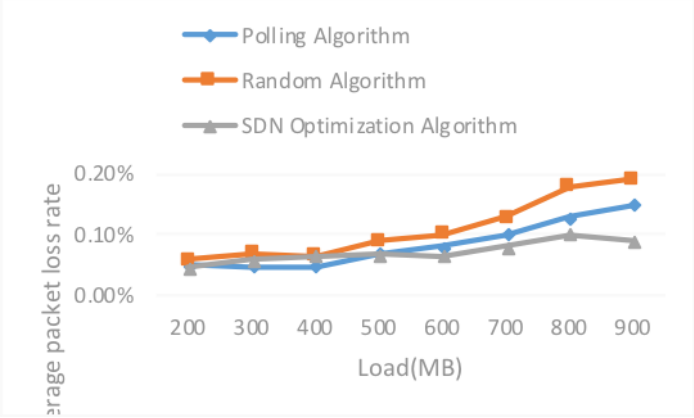


Figure 5. Average packet loss rate

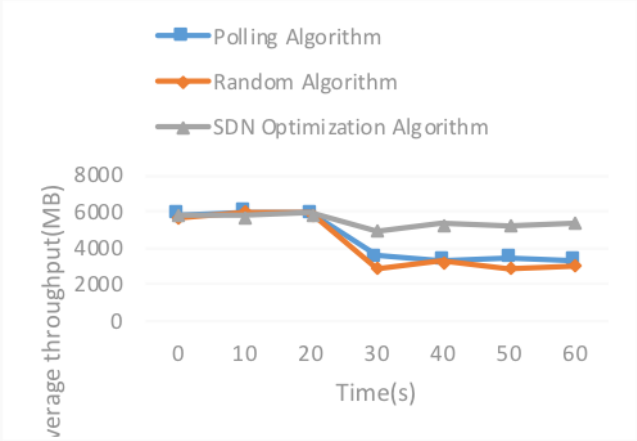


Figure 6. Average throughput

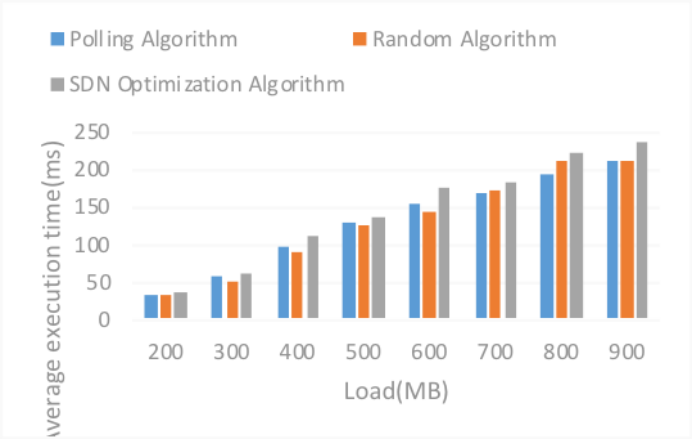


Figure 7. Average execution time

Figure 4, Figure 5 and Figure 6 show that when the load is in the range of 200 ~ 500MB and the time is within 0 ~ 20s, the packet or stream processing of the server is less difficult. Therefore, the average delay, the average packet loss rate and the average throughput of the three load balancing algorithms are not significantly different. As the load and the time increase, the server needs to handle more and more packets or streams. In this case of the high load, the traditional network using the polling algorithm and the randomized algorithm to deal with the network packet increases the difficulty. The link load unbalanced leads to the overload of some links, so the growth rate of the average delay and the average packet is more obvious. The proposed SDN algorithm uses K-Dijkstra algorithm to find some alternative paths, then uses HRRF algorithm to select the optimal path in the part of the alternative path. Inform the controller through the feedback message, and achieve the load balancing. Even if the network load is too large or after a certain period of time, there is an evident advantage in the average delay and the average packet loss rate, which making the network can provide better service for users.

Figure 7 shows that as the load increases, the average execution time of the three algorithms is different. As the load increases, the average execution time used by the randomized algorithm is the least, and the SDN optimization algorithm proposed in this paper is slightly larger. The reason is that the randomized algorithm and the polling algorithm are simple, and the execution time is short. The SDN optimization algorithm proposed in this paper is implemented on the basis of SDN architecture. The controller plays a decisive role and has a large workload. Therefore, the execution time is slightly larger than the traditional network the polling algorithm and the randomized algorithm.

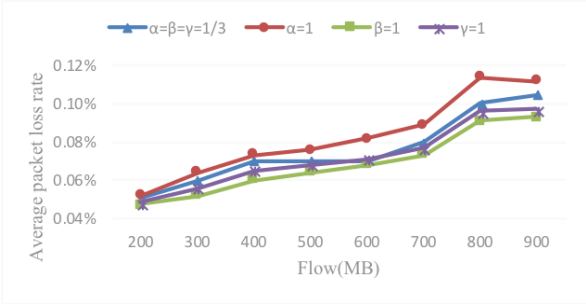


Figure 8. Average delay under different parameters

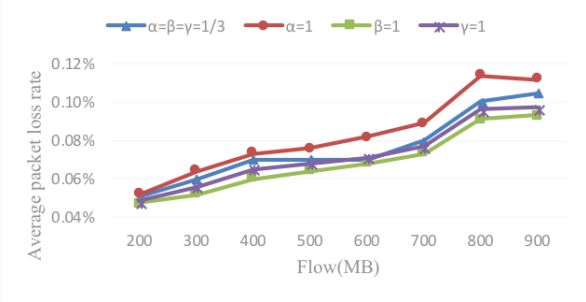


Figure 9. Average packet loss rate under different parameters

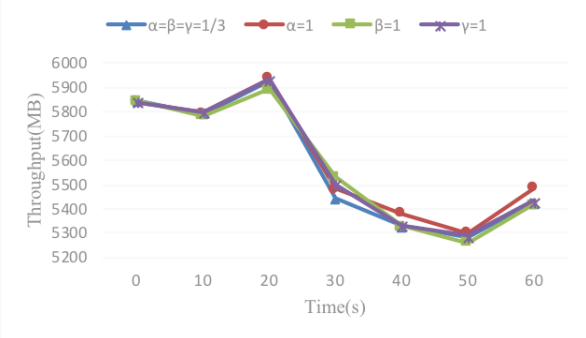


Figure 10. Average throughput under different parameters

Figures 8, 9 and 10 show that when the three parameters α, β, γ in the edge weight w of the directed graph take different values respectively, the results are different. In case of the average delay, if the users or administrators have the highest expectation of the low delay in the current situation, the value of α is equal to 1, the value of β and γ is equal to 0, and the

delay is the lowest. In the case of the packet loss rate, if the users or administrators have the highest expectation of the low packet loss rate, the value of β is equal to 1, the value of α and γ is equal to 0, and the rate of the packet loss is the lowest. There is no significant difference in throughput. The experimental results show that the weight w of the directed graph is determined by the ternary function, and the dynamic values of the three parameters α, β, γ are obtained through the application layer, which can meet the different needs of different users or administrators in the network delay and the packet loss rate.

6. Conclusions

The traditional network structure cannot meet the increasing network traffic. As to the problem of load balancing in network, this paper proposes a load balancing method based on K-Dijkstra algorithm and HRRF algorithm, which uses SDN as a network architecture. This method solves the problem of unbalanced load caused by excessive network traffic. This paper designs four modules: flow management, traffic monitoring, dynamic load balancing and load calculation in the controller of SDN architecture. The K-Dijkstra algorithm and HRRF algorithm are introduced into the dynamic load balancing module. Through K-Dijkstra algorithm, a number of alternative paths are calculated, and the optimal path in the alternative path is selected by HRRF algorithm to achieve load balancing, which effectively reduces the network delay and packet loss rate and improves the network throughput.

Acknowledgements

This work is supported by the National Key R&D Program of China under Grant (No. 2017YFB0802300).

References

1. J. A. Azevedo, J. J. E. R. S. Madeira, E. Q. V. Martins and F. M. A. Pires, “A Shortest Paths Ranking Algorithm.” Proceedings of the Annual Conference AIRO'90, Models and Methods for Decision Support, Operational Research Society of Italy. pp. 1001-1011
2. Feldmann “A. Internet Clean-slate Design: What and Why” *Acm Sigcomm Computer Communication Review*, vol. 37, no. 03, pp. 59-64, 2007
3. A. Gavras, A. Karila, S. Fdida, et al. “Future Internet Research and Experimentation: the FIRE Initiative.” *Acm Sigcomm Computer Communication Review*, vol. 37, no. 38, pp. 89-92, 2007
4. E. C. Geni, “Opening Up New Classes of Experiments in Global Networking.” *IEEE Internet Computing*, vol. 14, no. 01, pp. 39-42, 2010
5. R. Jain. “Internet 3.0: Ten Problems with Current Internet Architecture and Solutions for the next Generation.” *IEEE Conference on Military Communications*, IEEE Press, pp. 153-161, IEEE, 2006
6. N. Mckeown, T. Anderson, H. Balakrishnan, et al. “OpenFlow:Enabling Innovation in Campus Networks,” *Acm Sigcomm Computer Communication Review*, vol. 38, no. 02, pp. 69-74, 2008
7. H. C. Li. “HRRF Scheduling Strategy based on TinyOS.” *Computer Science*, vol. 37, no. 04, pp. 80-81, 2010
8. H. F. Li, C. Dong, Zheng, X. H. Zheng, et al. “Research and Implementation of Traffic Management Applications based on Software Defined Networks and Implementation of.” *Computer Applications and Software*, vol. 32, no. 05, pp. 17-19, 2015
9. T. Wang, H. C. Chen, G. S. Cheng. “Study on Software-Defined Network and Security Defense Technology.” *Journal on Communications*, vol. 38, no. 11, pp. 133-160, 2017
10. X. M. Wang, C. H. Huang, Q. Y. Fan, K. He. “Measurement Method based on Load Balancing in Streaming Networks.” *Huazhong University of Science and Technology Journal*, 2016,44 (11): 75-81.vol. 44, no. 11, pp. 75-81, 2016
11. Y. Wang, J. K. Yu, K. K. Pei, X. S. Qiu. “A Load Balancing Scheme for SDN Multi-Controller based on Load Bulletin.” *Journal of Electronics and Information Technology*, vol. 39, no. 11, pp. 2733-2740, 2017
12. S. Wu. “The Design of a Load Balancing Scheme based on SDN Network and Implementation.” *Fudan University*, 2014
13. Y. Yang, J. H. Yang, H. S. Wen, H. Wang. “Multi-path Transmission in Data Center based on SDN Traffic Measurement.” *Huazhong University of Science and Technology Journal*, vol. 44, no. 11, pp. 53-58, 2016
14. S. Zeng, G. Chen, F. Z. Qi. “Software Defined Network Performance.” *Computer Science*, vol. 42, no. s1, 2015
15. C. K. Zhang, Y. Cui, Y. Y. Tang, et al. “Research Progress on Software-defined Networks (SDNs).” *Journal of Software*, vol. 26, no. 01, pp. 62-81, 2015
16. Q. Y. Zuo, M. Chen, G. S. Zhao, et al. “Research on SDN based on OpenFlow.” *Journal of Software*, no. 05, pp. 1078-1097, 2013