International Journal of Grid and Distributed Computing
Vol. 9, No. 1 (2016), pp.25-36
http://dx.doi.org/10.14257/ijgdc.2016.9.1.03

Research on Load Balance Method in SDN

Cui Chen-xiao[1] and Xu Ya-bin[1, 2]

[1] *School of Computer, Beijing Information Science and Technology University, Beijing 100101, China*
[2] *Beijing Key Laboratory of Internet Culture and Digital Dissemination Research, Beijing 100101, China*
cuichenxiao990@126.com, xyb@bistu.edu.cn

Abstract

In order to improve the performance of data transmission in SDN, this paper proposes a load balance solution scheme by taking advantage of the global network view of SDN. We collect 4 load features from each transmission path. These features are bandwidth utilization ratio, packet loss rate, transmission latency and transmission hops. By using this 4 load features, BP Artificial Neural Network model is trained to predict the integrated load for different path and to choose one with least load as the data-flow transmission path. The contrast experiment results show that load balancing strategy proposed in this paper can select more rational transmission path for data-flow and achieve 19.3% network latency decrease at most.

Keywords: *Software Defined Network; Load Balance; BP Artificial Neural Network*

1. Introduction

With the trend of amplifying in network scale and application, traditional network protocols and functions are becoming increasingly large and complex. In the meanwhile, the strong coupling of traditional network devices between logic control and data transmission triggers much complexity in device management and control. This factor makes it hard to maintain traditional network operation and management. In order to improve the flexibility and effectiveness to manage network, researchers at Stanford University proposed OpenFlow**Error! Reference source not found.** technology in 2008 and opulated it as the concept of Software Defined Network or SDN for short.

The core idea of SDN is to decouple network control from data transmission. OpenFlow switches implement data transmission function, so as to simplify the design of switches, and control functions are provided by controllers. As shown in Figure 1, SDN architecture is composed of three layers [1], which referred as data layer, control layer and application layer.

As the basic network device in data layer, OpenFlow switches implement data transmission function according to flow-tables allocated from controller. Being as the "brain" of SDN, controller acquires application information from upper layer through the unified northbound interface. Flow-tables are generated in controller and allocated to OpenFlow switches through OpenFlow protocol. By acquiring network topology information, SDN controller provides the global network view for OpenFlow switches and implements the flexible network configuration and network management. As with traditional network, key link redundancy technology in SDN can effectively solve the problems of network congestion and provide the robustness and stability for network. By evenly distributing traffic among multiple paths, load balance can be achieved in SDN. But how to achieve this goal has become an urgent problem for network researchers and network managers. Hence, load balance research in SDN will make lot of sense.

ISSN: 2005-4262 IJGDC

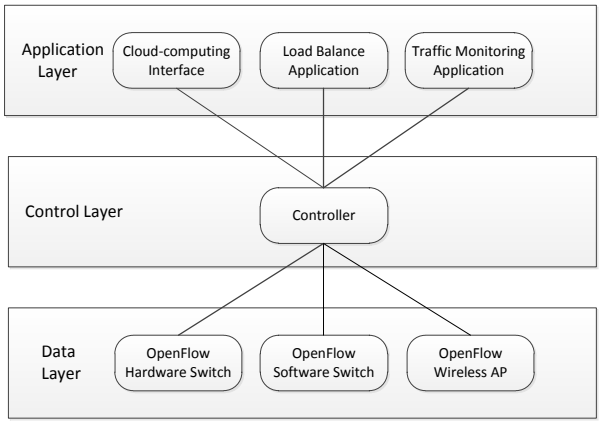

Figure 1. SDN Architecture

In this paper, we propose a strategy to keep network load balanced in SDN. The strategy is adaptive to network traffic, and schedules flows based on the integrated path load which computed based on BP artificial neural network. The proposed load balance strategy aims at optimizing path load and selects a least loaded path for newcomer data-flow.

The rest of this paper is organized as follows. Section II presents related works and Section III shows a framework of the proposed system. Section IV describes the method to acquire four load features and the major mechanism of BP Artificial Neural Network to achieve integrated path load. Section V presents the experimental results. Finally, Section VI concludes this paper.

2. Related Works

Many researches have been proposed on load balance in traditional multipath network. Two load balance strategies have been widely used in multipath network at present: Equal-Cost MultiPath (ECMP)[3] and Valiant Load Balance (VLB)[4]. The core idea of ECMP is to evenly distribute data-flow to next-hop switches, and VLB distributes traffic among all available paths and randomly picks the next-hop switch. These two strategies both use fixed methods and cannot pick transmission path adaptively to the path load condition.

In SDN architecture, limited researches have been proposed on network load balance. A number of load balance system have been proposed in [5]-[7]. In these systems, controller is used to analyze replying information from OpenFlow switches and modify the flow-tables by specific load balance strategy, so as to efficiently plan data transmission path and achieve load balance in SDN. But these strategies belong to static load balance method which cannot make dynamic routing plan according to real-time network load condition. Besides, these several methods take little advantages of SDN to make a better load balance desig. A dynamic load balance algorithm, known as DLB, has been proposed in [8]. The DLB algorithm simply applies greedy selection strategy to pick next-hop link which transmits least data load. Although these algorithm implements load balance on multipath SDN, this routing strategy is only decided by link load of every next-hop without combining the superiority of global network view in SDN. Hence, this routing strategy may not find the best transmission path in global view so that may not achieve the best load balance effect.

The load balance strategy proposed in **Error! Reference source not found.** collects hops f each path, transmission packet number, key switch number and transmission rate in every switch port, and combine these four features to calculate the load condition of each path through fuzzy comprehensive evaluation. Finally, choose the transmission path based on the load condition of each path. As for fuzzy comprehensive evaluation, artificial

subjective mark process is needed in this strategy. This load balance algorithm cannot objectively show the accurate real-time load condition of each path as well. [10] implements the heuristic method of load balance by classifies the best path and best servers combining with global network view of SDN. This heuristic method applies Ant Colony Optimization to decide transmission path and servers .This method collects the information to calculate the link usage and monitors the delay in links. But this strategy applies a simple method to collect network path information with a single evaluation criterion.

In summary, load balance research in current SDN architecture has been just carried out. Most existing strategies employ simple methods to decide best global transmission path with poor performance. The path information collection shows a lack of comprehensive factors and cannot achieve ideal effects. Faced with these problems, this paper takes advantage of the global view of SDN architecture to collect 4 factors, which referred as bandwidth utilization ratio, packet loss rate, transmission latency and transmission hop, of each path and employs Artificial Neural Network to calculate the integrated load condition of each path.

3. Load Balance System Design

Although traditional routers store routing tables, routing tables only contain destination network information and routers can only calculate the next-hop network information without global network routing view. However, SDN controller obtains the ability to show the global view of network at the beginning of network construction. By updating topology information of global network, SDN controller can discover all paths between each source node to each destination node. By utilizing this superiority of global network view in SDN, the load condition of every global path can be evaluated.

Network architecture for proposed SDN load balance system is shown in Figure 2.

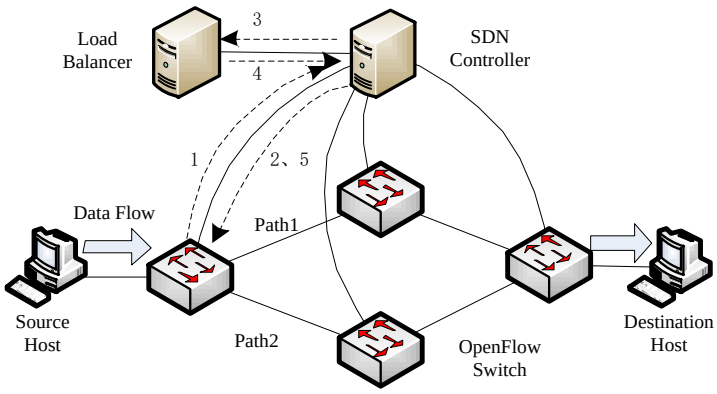

Figure 2. Network Architecture for Proposed System

In order to lighten the load of SDN controller, the proposed architecture employs a dedicated load balancer server to balance the load in each path. For the purpose of choosing real-time least loaded path, load balancer immediately calculate the integrated load condition of multiple path when receiving the path information transmitted from SDN controller. Hence, in the proposed design, controller periodically transmits the load information of each path to load balancer. And when the SDN controller need to process the load balance function, the load balancer return a least loaded path back to controller according to the calculated load condition of each path. After SDN controller receives the chosen path for transmission, it will allocate flow-tables for OpenFlow switches to achieve the plan of data-flow transmission.

The procedure of control data in proposed system is shown in Figure 2:

1. When a new data-flow transmitted into SDN domain, OpenFlow switches process the matching between packet head information and flow-tables. If the flow-table matches the packet head information, this data-flow will be transmitted by the Action field in flow-table. And if there is no flow-table to match this packet, OpenFlow switches will transmit this packet's head information to SDN controller to decide the transmission path.
2. When finding only one path for data transmission, the SDN controller will create new flow-tables and allocate them to OpenFlow switches to active data transmission
3. When finding multiple paths for data transmission, the SDN controller will transmit multiple path load information to the load balancer.
4. The load balancer calculates the integrated load for every path and chooses one least loaded path as the result to return back to SDN controller.
5. SDN controller receives the chosen path from load balancer and creates flow-tables to allocate to OpenFlow switches.

4. Path Features Extraction

The ultimate purpose of load balance strategy is to choose one best load condition path for data transmission. In order to acquire the load condition of each path, OpenFlow protocol should be used to obtain the link load condition between two switches. In the proposed system, whenever a new data-flow transmitted into the SDN domain, SDN controller will inquire the flow-table information and collect each port statistics from OpenFlow switches along the decided transmission path. After extracting the load feature information of each link, Artificial Neural Network should be used to calculate the load condition of this path. At last, the path with least load condition will be chosen as the final transmission path.

In OpenFlow protocol, the SDN controller can collect the port statistics from every OpenFlow switch via OFPT_STATS_REQUESTS message. This information is kept in flow-tables that defined by the OpenFlow protocol.

Acting as the major foundation of data transmission in OpenFlow switches, one or more flow-tables are maintained in every OpenFlow switch. A flow-table is composed of several flow-table entries which are regarded as the basic match objects. Specified in OpenFlow v1.4 [36][11], the flow-table entry contains 6 components, shown in Figure 3.

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie

Figure 3. Flow-table entry instruction in OpenFlow v.14

These components are: Match Fields, Priority, Counters, Instructions, Timeouts and Cookie. Match Fields match against packet head information, including ingress port, packet headers and optionally metadata. When transmitted into OpenFlow switch, packets will be matched against the Match Field in flow-tables. If some flow-table entry matches this packet, the Counters will update the statistics about this data-flow, including the number of matched packet, the transmitted bytes number, duration time, *etc*., and the packet will execute the actions that defined in Instructions. Timeouts defines the maximum amount of time or idle time about this flow-table entry. For example, there is a flow-table in one OpenFlow switch, shown in Table 1. When some data-flow transmitted into OpenFlow switch contains the header information as: ip_src=10.0.0.1, ip_dst=10.0.0.2, the flow-table entries will be matched against this header information successively. Firstly, flow-table entry with Priority=1 will be matched. Because the match fields in this flow-table entry cannot match with data-flow header information, this data-flow header information will be matched against the next flow-table entry. Finally, it will be matched with the flow-table entry with NO=3, and then the OpenFlow switch will

execute the actions defined by the Instruction field that the packets will be transmitted via the switch port1, and update counters.

Table 1. Flow-Table Example in OpenFlow Switch

NO.	Match Fields	Priority	Instructions	Timeouts	Counters
1	ip_src=192.168.1.*	1	Drop	60s	n_packets=30 n_bytes=2893
2	mac_dst=AA:BB:CC:DD:EE:FF ip_dst=192.168.3.101	2	Output: port2	60s	n_packets=60 n_bytes=3730
3	ip_src=10.0.0.1 ip_dst=10.0.0.2	2	Output: port1	60s	n_packets=10 n_bytes=169
…	…	…	…	…	…
20	mac_dst=AA:BB:CC:DD:EE:FF	10	Output: port3	60s	n_packets=25 n_bytes=2450

After periodically collecting the path information via OpenFlow protocol, the SDN controller will immediately transmit this information to load balancer server for further process. In this way, the load of SDN controller can be lightened and the system becomes conformed to the standard SDN architecture.

In order to adequately express the load condition of each path, this paper utilize bandwidth utilization ratio (*BWRatio*), packet lose rate (*PLoss*), transmission latency (*Latency*) and transmission hop (*Hop*) as the evaluation criteria for path load condition.

1) Bandwidth Utilization Ratio (*BWRatio*): Bandwidth utilization ratio is the significant evaluation criterion in network transmission and it can reflect the load condition of one link in a sense. When BWRatio→1, this link may be in a heavy load condition and it is with high possibility to occur link congestion. In order to obtain the link bandwidth utilization ratio, the SDN controller needs to collect cumulative transmitted bytes B_T at corresponding OpenFlow switches port. By calculating subtraction between B_T and the last period cumulative transmitted bytes B_{T-1}, we can get the transmitted data bytes in this period, meaning the utilized bandwidth of the link in this period. At last, we use the maximum bandwidth $BWRatio_{MAX}$ to divide the utilized bandwidth of the link in this period. Thus, the bandwidth utilization ratio can be calculated by Equation (1):

$$BWRatio_{Link} = \frac{B_T - B_{T-1}}{BWRatio_{MAX}} \quad (1)$$

If one path contains several links, L_1, L_2, L_3 … L_n, with the corresponding bandwidth utilization ratio as $BWRatio_1$, $BWRatio_2$, $BWRatio_3$ … $BWRatio_n$, the bandwidth utilization ratio of this path can be calculated by Equation (2):

$$BWRatio = \frac{1}{n}\sum_{i=1}^{n} BWRatio_i \quad (2)$$

2) Packet Loss Rate (*PLoss*): Packet loss always occurs during the packet processing period in data transmission device. If switches in the network is too busy to process the incoming packets, the packets may be dropped by switches. The packet loss rate indicates the busy condition of the switch and the load condition of the path. The SDN controller can collect cumulative number of transmitted packet $Packet_T$ and cumulative number of received packet $Packet_R$ at corresponding OpenFlow switches ports. Thus, the packet loss rate of one path can be calculated by Equation (3):

$$PLoss = \frac{Packet_T - Packet_R}{Packet_T} \quad (3)$$

3) Transmission Latency (*Latency*): Transmission latency is the time spent by host switch on data transmission. It is related to the performance of switches, the size of data packets and the congestion condition of the transmission queue. The transmission latency can indicates the congestion condition of a link and load condition of the switch in some way. The SDN controller can collect the transmitted bytes *Byte* in this period and the transmission rate *tr_Rate* at corresponding OpenFlow switches port. Then, the transmission latency can be calculated by Equation(4):

$$Latency_i = \frac{Byte}{tr_{Rate}} \quad (4)$$

If one path consists of several links L_1, L_2, L_3 … L_n with the corresponding transmission latency $Latency_1$, $Latency_2$, $Latency_3$ …$Latency_n$, then the total latency of this path can be presented as

$$Latency = \sum_{i=1}^{n} Latency_i \quad (5)$$

4) Transmission Hop (*Hop*): Transmission hop is the important factor considered by routing strategy. Large number of transmission hop will increase the congestion probability. In the same network condition, transmitting packet along path with fewer hops will decrease packet loss probability and reduce transmission delay. Because the global network topology is maintained in the database of the SDN controller, we can get the number of hops between two switches using source switch and destination switch as the query factor to search the database.

5. Load Balance Approach Based on Artificial Neural Network

In order to pick the least loaded path, we need to confirm the path load condition by analyzing the integrated path information. In the network, traffic evaluation and estimation play a significant role in improving network performing and quality of service. In this paper, machine learning method is employed to integrate this 4 path information features to evaluate path load condition. Because using different methods or different features may cause different accuracy and rationality for path load condition, it is very important to choose an accurate and rational machine learning method.

Artificial Neural Network computing model is a nonlinear and self-adapted information process system composed of multiple interconnected process units. It has nonlinearity and self-learning characteristics. Compared with Logistic Regression and other probability statistics machine learning, Artificial Neural Network has no limit on input vectors. The input vectors need no limit on particular probability distribution or independence among variables. Due to the uncertainty of the traffic in network, a large amount of data can be collected to form a dataset. This dataset is complex and may follow no particular probability distribution. Therefore, Artificial Neural Network is applicable for process the uncertain network traffic data.

Artificial Neural Network is composed of multilayer interconnected neurons. One Artificial Neuron can be depicted by Equation (6) and (7):

$$u_k = \sum_{i=1}^{n} w_{ik} x_i \quad (6)$$

$$y_k = f(u_k + b_k) \quad (7)$$

Where x_i $(i = 1, \ldots, n)$ presents the input vectors. w_{ik} $(i = 1, \ldots, n)$ presents the weight of neuron *k*. And if w_{ik}>0, it indicates excited state, and if $w_{ik} < 0$, it indicate the restraint state. m denotes the number of input vector. u_k is defined as the linear combination of input vector. b_k is denoted as the threshold value of a neuron. Depending on the positive or negative value of b_k, the input of activation function will be increased or decreased. b_k is used feedback parameter to adjust the neural network. *f(x)* is defined as the activation function. y_k presents the output of neuron.

The 3-layer Back Propagation Artificial Neural Network (BPANN) is employed as the process model to integrate path information. The process procedure of BPANN contains two phases. The first one is forward propagation learning phase. During this phase, the input neurons process the input vectors with preset weights through the action function. Then, the output from hidden layer will be conducted to the output layer, and finally output layer output the ultimate results. This learning process is forward propagated, and the condition of every node will affect the output of the next layer nodes. After the learning phase, BPANN will be in the weight adjustment phase. In this phase, the error between the ultimate results and the expected output will be utilized to modify the weights. If the results of the output layer presents too much large, the error will be make back propagation to modify neuron weights in every neural layer. As a consequence, the back propagation modification will keep adjusting the neural network until the error meets in the satisfying interval. By utilizing a large number of data to keep learning and adjusting, the BPANN will eventually form a fittest neural network for the application scenario.

In this paper, 4 path information features are selected as the input vector, so the neural network contains 4 input neuron nodes correspondingly. And we set one neuron node to indicate the ultimate output. As for hidden layer, the number of neurons in hidden layer has great effect on the output error of neural network. At present, there is no unified approach to determine the number of neurons in hidden layer. In this paper, we employ Equation (8) from [14] to set the number of neurons in hidden layer:

$$N = \sqrt{m+n} + a \tag{8}$$

Where N denotes as the number of neurons in hidden layer, m denotes as the number of neurons in input layer, n denotes as the number of neurons in output layer and a is the constant between 1 and 10.

In experiment design, this paper set the value of 1000 as the number of training times and the training target is set as 0.001. As for the number of neurons in hidden layer, we respectively choose $a=1$, $a=3$, $a=5$, $a=7$ and $a=9$. And correspondingly, we will get the number of 3, 5, 7, 9 and 11 as the number of neurons in hidden layer. And finally, we select the best performance neural network parameters to create the BP Artificial Neural Network. Consequently, the structure of the BP Artificial Neural Network can be depicted in Figure 4:

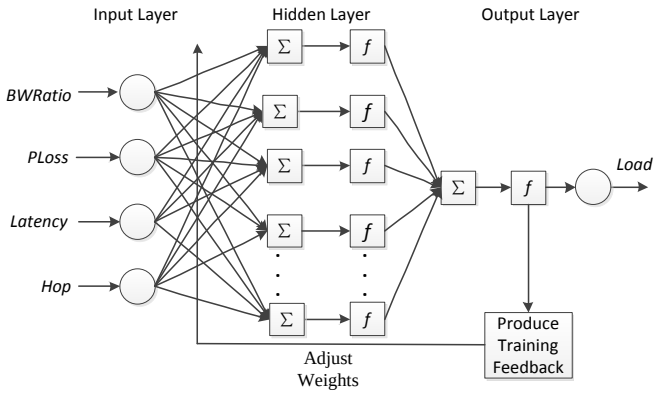

Figure 4. Path Load Balance Neural Network Structure

6. Experimental Results and Evaluation

6.1 Experimental Environment

In this paper, we utilize mininet[16] network simulation tool and Floodlight[17] SDN Controller to simulate multipath SDN environment to evaluate the performance of proposed load balance strategy. The simulated SDN topology is shown in Figure 5:

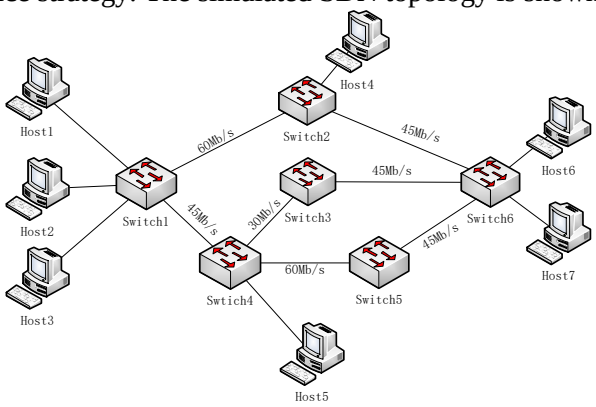

Figure 5. Experimental SDN Topology

Figure 5 shows a multiple path network topology. This topology contains three paths from Host1 to Host7:

1）Switch1->Switch2->Switch6；

2）Switch1-> Switch4-> Switch3-> Switch6；

3）Switch1-> Switch4-> Switch5->Switch6。

Link maximum bandwidth is shown in Figure 5.

In the reality network environment, network traffic presents strong randomness and uncertainty. So, in this paper, the experiment is designed to simulate the network environment in reality as possible. Traffic will be transmitted randomly among Host2, Host3, Host4, Host5, Host6 and Host7. During the traffic transmission, the load conditions of three paths from Host1 to Host7 are quite different. Then, we start to control Host1 to send traffic to Host7, which is regarded as the new incoming data-flow in SDN domain. The ultimate purpose for the experiment is to select one least loaded path from Host1 to Host7 as the new incoming transmission path, and to achieve load balance in network.

6.2 Experimental Results Analyze

6.2.1 Training Results of BPANN: Before utilizing the Artificial Neural Network to indicate the integrated path load, it is indispensible to train the neural network with a large number of dataset to achieve the least error Artificial Neural Network. In this paper, we utilize mininet network simulation tool to simulate the SDN topology as shown in Figure 5 . Hosts in the topology will transmit traffic randomly to other hosts to simulate the network traffic, while the SDN controller utilizes OpenFlow protocol to record the path load features. We collect path load features in 180s to generate the datasets to train the Artificial Neural Network and select a part of the dataset as the testset. As depicted in Section **Error! Reference source not found.**, the number of neurons in hidden layer annot be easily determined. In the procedure of training, we train the BP Artificial Neural Network with the value of 3, 5, 7, 9 and 11 as the number of neurons in hidden layer. And we can get the results of these different neural networks to evaluate the performance of these 5 neural networks. The experimental results are shown in Figure 6.

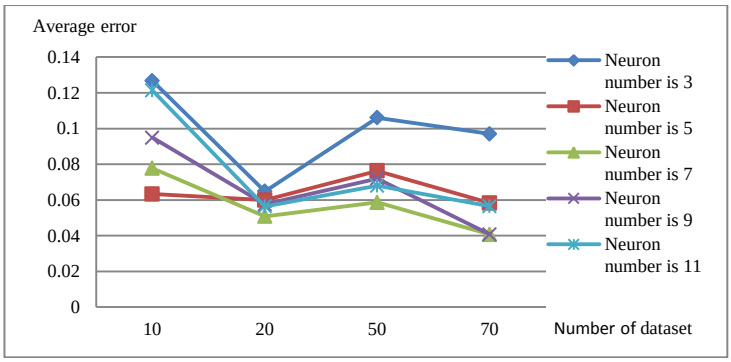

Figure 6. Average Error for Different BPANN

Figure 6 shows the average error in the five BPANNs with different neuron numbers in hidden layer. As shown in Figure 6 when the number of the neurons in hidden layer is too small (namely 3), the error of the trained BPANN is relatively the largest. Along with the increase of the size of datasets, all the error results of BPANN are decreasing. In general, the error results show big differences in BPANN which contains least neurons (namely 3) and most neurons (namely 11). When the number of neurons comes into 7, the average error results are relatively least and keep steady. Take all this into consideration, BPANN with 7 neurons in hidden layer is selected, meaning this type of BPANN is the fittest one for SDN network case.

6.2.2 Experimental Results of Load Balance based on BPANN: In order to evaluate the load balance strategy based on BPANN (ANNLB), we have used two other load balance strategy as comparisons. They are the DLB load balance method proposed in [8] and static Round Robin load balance strategy.

In the early 90s of the experiment, Host2, Host3, Host4, Host5, Host6 and Host7 keep transmitting data randomly to each other, so that to make some traffic in multiple paths in the network. Then, Host1 transmits traffic to Host7 to simulate the new incoming data-flow in SDN domain. During the experiment, the average bandwidth utilization ratio and the transmission latency will be recorded. After the comparisons with DLB proposed in [8] and static load balance Round Robin, the experimental results show in Figure 7, Figure 8 and Figure 9:

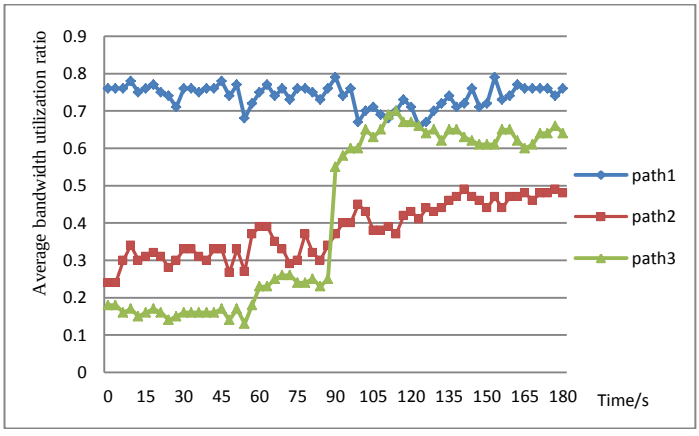

Figure 7. Average Bandwidth Utilization Ratio in ANNLB

As depicted in Figure 7, when a new incoming data-flow needs to be transmitted, load balance strategy proposed in this paper select path3, which in a good load condition, as

the transmission path. And the entire alternative paths are maintained in a good bandwidth utilization ratio condition.

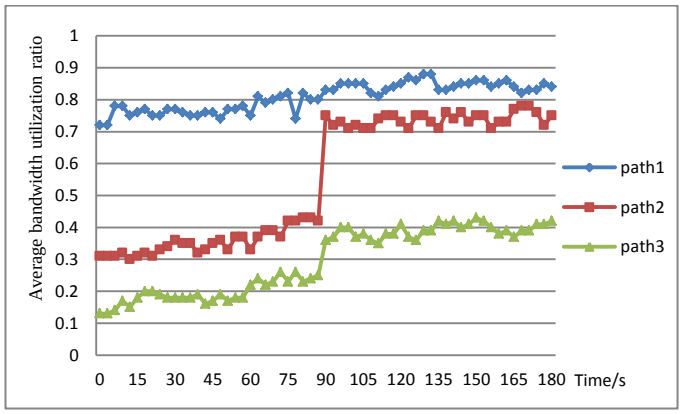

Figure 8. Average Bandwidth Utilization Ratio in DLB

We can see from Figure 8 that when a new incoming data-flow needs to be transmitted, DLB proposed in [8] can only consider the link condition of next-hop. So, this method selects path2 as the transmission path by mistake. Because of the poor maximum bandwidth of path2, the bandwidth utilization ratio increases rapidly. In the meanwhile, because there is a common link in path2 and path3, this load balance strategy also influences path3 and that increased the probability of congestion.

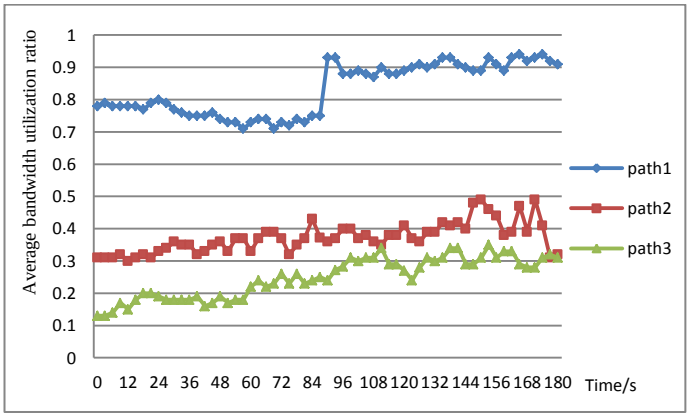

Figure 9. Average Bandwidth Utilization Ratio in Round Robin

Figure 9. shows the experimental results of Round Robin strategy. Due to the static strategy, Round Robin load balance cannot take the real-time link state into consideration and path1, with a dangerous bandwidth utilization ratio, is selected as the transmission path. This strategy leads to a worse load condition in path1 and increase the probability of congestion. Table 2 shows the average bandwidth utilization ratio and the bandwidth utilization mean-square deviation.

Table 2. Average Bandwidth Utilization Ratio and Bandwidth Utilization Ratio Mean-Square Deviation

	ANNLB	DLB	RR
Path1	0.728	0.845	0.907
Path2	0.440	0.734	0.398
Path3	0.634	0.385	0.301
Mean-	0.147	0.239	0.325

Square Deviation	

As depicted in Table 2, the Mean-Square Deviation of ANNLB is the least one among all others, which means bandwidth utilization ratio of all paths are close with each other and a favorable result is achieved by utilizing ANNLB. Because the DLB strategy ignores the global path load condition, the best condition path is not selected and other alternative paths suffer bad impact. But this load balance strategy still works to some extent. Round Robin belongs to the static load balance strategy, so it cannot take the real-time path load condition into consideration. And the result of this strategy shows to be the worst.

Figure 10 shows the latency of new incoming data-flow under different load balance strategy.

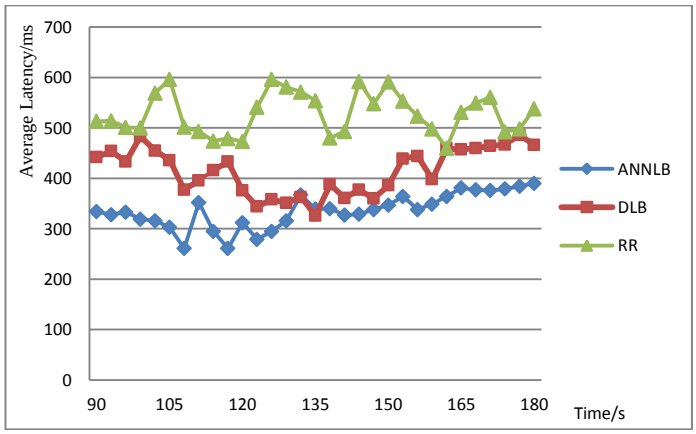

Figure 10. Transmission Latency from Host1 to Host7

As depicted in Figure 10, the ANNLB strategy and DLB strategy show better network performance than static load balance strategy. Because ANNLB strategy focus on global path condition and DLB only consider the next-hop link condition to provide local optimum transmission path, the ANNLB strategy provides better network performance than DLB does.

7. Conclusion

As the same with traditional network, path redundancy technology in SDN effectively provides the robustness and stability for network. But how to evenly distribute traffic among multiple paths has become an urgent problem for SDN researchers. Taking advantage of the global network view of SDN, this paper proposes a load balance solution scheme based on real-time path load condition. This strategy collects bandwidth utilization ratio, packet loss rate, transmission latency and transmission hops, and integrate them by Artificial Neural Network to indicate the path load condition. By utilizing this strategy, the transmission path can be selected for new incoming data-flow and the load balance can be achieved. We utilize mininet and Floodlight controller to simulate SDN network. The load balance strategy proposed in this paper is applied in the simulation experiment. Compared with strategy proposed in [8] and static Round Robin strategy, the experimental results show that the strategy proposes in this paper provides a better network performance and achieve 19.3% network latency decrease at most.

Acknowledgements

This work was supported by the National Natural Science Foundation of China(Grant No.61370139), Beijing Key Laboratory of Internet Culture and Digital Dissemination

Research (ICDD201309) and Beijing collages innovation team building and teacher occupation development program(IDHT20130519)

References

[1] N. Mckeown, S. Shenker, T. Anderson, L. Peterson, J. Turner, H. Balakrishnan, J. Re. "Openflow: Enabling Innovation In Campus Networks.", Acm Sigcomm Computer Communication Review38.2**, (2008)**, pp. 69-74.
[2] Open Networking Foundation. Software-Defined Networking: The new Norm for Networks. Apr. 13, **(2012)**.
[3] T. Wing Chim, and Y. K.L. "Traffic distribution over equal-cost-multi-paths.", IEEE International Conference on Communications IEEE, **(2004)**, pp. 465–475.
[4] D. William J., and B. Towles, "Principles and practices of interconnection networks", Principles and practices of interconnection networks. Elsevier, Morgan Kaufmann Publishers, **(2004).**
[5] H. Nikhil, *et al*. "Plug-n-Serve: Load-Balancing Web Traffic using OpenFlow." Acm Sigcomm Demo **(2009).**
[6] R. Wang, D. Butnariu, and J. Rexford. "OpenFlow-based server load balancing gone wild", Proceedings of the 11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services USENIX Association, **(2011)**, pp. 12-12.
[7] Y. Hu, W. Wang, X. Gong, X. Que, & S. Cheng, "BalanceFlow: Controller load balancing for OpenFlow networks.", Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on IEEE, **(2012)**, pp. 780-785.
[8] Y. Li, D. Pan. "OpenFlow based load balancing for Fat-Tree networks with multipath support", Proc. 12th IEEE International Conference on Communications (ICC'13), Budapest, Hungary. **(2013)**, pp. 1-5.
[9] J. Li, X. Chang, Y. Ren, Z. Zhang, & G. Wang, "An Effective Path Load Balancing Mechanism Based on SDN." Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on IEEE, **(2014)**, pp. 527-533.
[10] A M Koushika, S T Selvi. "Load balancing Using Software Defined Networking in cloud environment" Recent Trends in Information Technology (ICRTIT), 2014 International Conference on. IEEE, **(2014)**, pp. 1-8.
[11] OpenFlow v1.4 http://archive.openflow.org/
[12] S-y Yang, J-c Wang, and H-y Piao. "Optimization of Network Multipath Traffic Based on DEMT Algorithm." Computer Engineering 35**,(2009)**, pp. 99-100.
[13] H. Long, Y. Shen, M. Guo, & F. Tang. "LABERIO: Dynamic load-balanced Routing in OpenFlow-enabled Networks." 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA) IEEE Computer Society, **(2013)**, pp. 290-297.
[14] J. Gao, "Artificial neural network principle and simulation examples, Beijing: Mechanical Industry Press", **(2007).**
[15] L. Zhang, "Artificial neural network model and its application", Shanghai: Fudan University press, **(1993)**, pp. 55-70.
[16] Mininet http://mininet.org/
[17] Floodlight http://www.projectfloodlight.org/floodlight/

Authors

Cui Chenxiao. Cui Chenxiao was born in 1990. He is a master's candidate at School of computer from Beijing Information of Science and Technology University. His main research interests include cloud computing and future network.

Xu Yabin. Xu Yabin was born in 1962, master degree. He is a professor and a master supervisor in Beijing Information of Science and Technology University. His research interests include cloud computing & big data, social networking services and future network

