INTERNATIONAL JOURNAL OF CIRCUITS, SYSTEMS AND SIGNAL PROCESSING
DOI: 10.46300/9106.2020.14.111 Volume 14, 2020

A Path-Server Traffic Scheduling Algorithm for Wireless Network Load based on SDN

Jie Zhou

Anhui Vocational College of Electronics and Information Technology, Bengbu, Anhui, China

Received: August 17, 2020. Revised: November 11, 2020. Accepted: November 20, 2020. Published: November 24, 2020.

Abstract—**At present, the research on flow scheduling optimization around SDN has become the focus of international attention. Based on the current traffic control algorithm, especially the shortcomings of traffic scheduling optimization, Path-Server Traffic Scheduling (PSTS) algorithm is proposed in this paper to solve problems. Based on the real-time monitoring of network on Ryu controller, network topology, network load and server load information were obtained to choose the optimal routing scheme, thus achieving the goal of improving the overall network performance. Through the interaction between the components, the work flow of the Ryu controller is designed, and the channel quality information among users is maintained by the link state management module. When the small base station receives the data packet sent by the SDN exchange price, it will temporarily store it in the data cache module. Then, according to the amount of cached data and the channel quality of each user, the optimal time slot of the wireless network resource allocation scheme is comprehensively determined, and the data packet is sent to the corresponding client. In the view of proposed design, the Mininet simulation platform is used to simulate the SDN network in this paper. Based on the simulation platform, the performance of the algorithm is analyzed and compared. Besides, bandwidth utilization, average transmission delay, system utility and terminal usage change, interrupt rate and terminal usage change of different algorithms in SDN wireless network are analyzed and compared through data. All experiments have proved that the research content proposed in this paper has an obvious effect on network load control, which shows network performance.**

Keywords—SDN, wireless network, load control, PSTS algorithm, Ryu controller

I. INTRODUCTION

WITH the rapid development of big data, cloud computing and social networks, the traffic within the data center has grown exponentially, and the bandwidth requirements for data center networks is increasing. In order to achieve higher bandwidth and better fault tolerance in data center networks, researchers have proposed a variety of new data center network architectures, such as Fat-Tree, BCube, DCell, etc., which all use multiple tree structures and provide multi-path transmission to obtain higher network bandwidth to ensure network reliability. Taking the miniaturized Internet of Things cloud computing environment as an example and using flow steering, the Internet of Things cloud service application covering the data path is dynamically adjusted based on SDN. SVirt is proposed to provide ideal virtual SDN services for tenants of public clouds in commercial 5G networks, actual deployment and use cases in the wireless sensor network, namely the Internet of Things. At present, all aspects of research around SDN are in full swing. In terms of network traffic control, SDN can better achieve centralized control of the global network logic since the control plane of SDN is separated from the forwarding plane, and it has the characteristics of an open programmable interface[1-2].

At present, the research on traffic management and control (flow scheduling optimization) around SDN has become the focus of international attention. Proposed traffic control (scheduling) schemes can be roughly divided into the following two categories.They are embodied in optimizing traffic scheduling and balancing server load,etc.. The idea of the first type is to design and build a new model or architecture to optimize traffic management and control. In literature [3-5], based on the optimal allocation of traffic, a new traffic distribution and scheduling model is proposed to achieve load balancing. The literature [6-7] takes the flow table in OpenFlow as the starting point and improves the existing route scheduling mechanism to achieve better load balancing effect. In literature [8], a real-time virtual machine management framework based on SDN is proposed. Moreover, the sequential network information is used as the core to drive the operation of the virtual machine, and the optimal scheduling of traffic in the whole network is realized. Experimental results show that the management structure can effectively reduce communication costs and improve network throughput. The idea of the second type is to study the corresponding algorithm to optimize the

flow management and control. Literature [9] proposed an effective resource management algorithm, which mainly optimizes the factors affecting the load balancing effect including traffic scheduling. What's more, based on the SDN framework, literature [10-11] analyzed the dynamics and staticity of the load balancing algorithm. Both algorithms can achieve load balancing, but the performance of dynamic algorithms is better than static algorithms. Compared with the first type of research scheme that needs to redesign and build the entire flow control model or architecture, the second type of research scheme only needs to design and improve the algorithm, which is simpler and more operable. However, the optimization of traffic control at the algorithm level only considers the link factors that affect data transmission, but does not take into account factors such as the server load that affects network data transmission. Most of the current traffic control algorithms consider influencing factors too singular, and only stay at the link level, which lack of the consideration at server level. The Dynamic Load Balance (DLB) algorithm determines the final forwarding path only by selecting the single link with the largest available bandwidth. Obviously, the influencing factors considered by the DLB algorithm are too singular.

Based on the current traffic control algorithm, especially the shortcomings of traffic scheduling optimization, Path-Server Traffic Scheduling (PSTS) algorithm is proposed in this paper to solve above problems. For most of the current researches on traffic scheduling optimization algorithms only consider the link level, the PSTS algorithm proposed in this paper adds additional consideration to the server level. Nearly comprehensive consideration of the influencing factors makes the algorithm achieve more reasonable traffic scheduling and higher network benefits.

II. PSTS Algorithm for Wireless Network Load Control

A. RYU Controller

YU is an open-source component-based software-defined networking framework project developed by Nippon Telegraph and Telephone (NTT). RYU is an SDN controller fully implemented in Python. RYU name comes from a Japanese word meaning "flow," which is a Japanese dragon, one of the water gods. It manages "flow" control to enable intelligent networking. The RYU Controller provides software components with well-defined applica-tion program interfaces (APIs) that make it easy for developers to create new network man-agement and control applications. RYU supports various protocols for managing network devices, such as OpenFlow, Netconf, OF-config, etc. About OpenFlow, RYU supports fully 1.0, 1.2, 1.3, 1.4, 1.5.

RYU SDN Controller has three layers. The top layer consists of business and network logic applications known as the application layer. The middle layer consists of network services known as the control layer or SDN framework, and the bottom layer consists of physical and virtual devices know as an infrastructure layer. The middle layer hosts northbound APIs and southbound APIs. The controller exposes open northbound APIs such as a Restful management API, REST, API for Quantum, User-defined API via REST or RPC, which are used by applications. RYU provides a group of components such as OpenStack Quantum, Firewall, OFREST, etc. useful for SDN applications. The objective of these applications is to gathered network intelligence by using a control-ler, performed analytics by running algorithms, and then orchestrated the new rules by using the controller.

The architecture of the RYU SDN Controller shown in Figure 1.

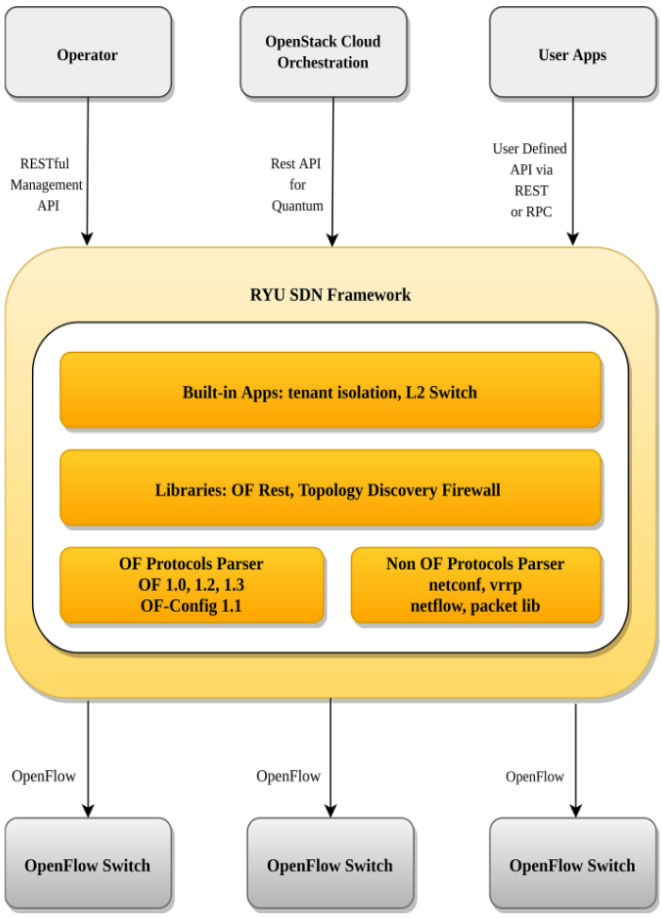

Fig.1 RYU SDN controller architecture

B. The SDN-based Network Architecture

Due to the Real-time monitoring of the network on Ryu controller, network topology information, network load information, and server load information are obtained to achieve an optimal routing scheme, which thereby realizes the goal of improving overall network performance. The SDN-based network architecture is shown in Figure 2.

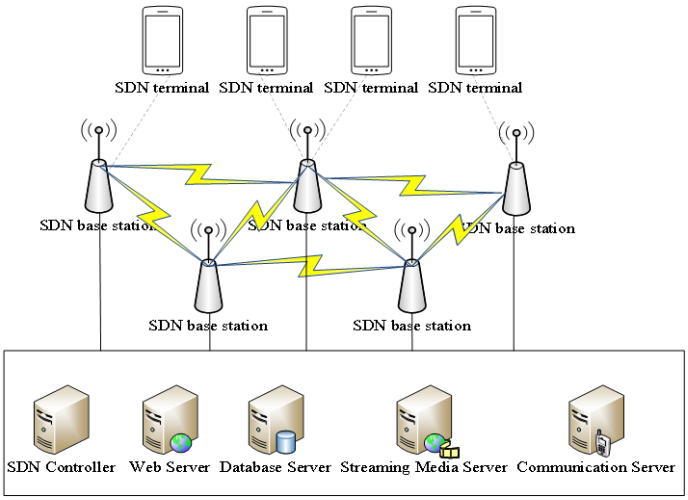

Fig.2 SDN-based network architecture

In this paper, PSTS algorithm is implemented in Ryu controller through custom module. The interaction among components can be used to design the workflow of the Ryu controller, which is the initialization phase, the state acquisition phase, the calculation and screening phase, and the policy execution phase. They are shown in Figure 3.

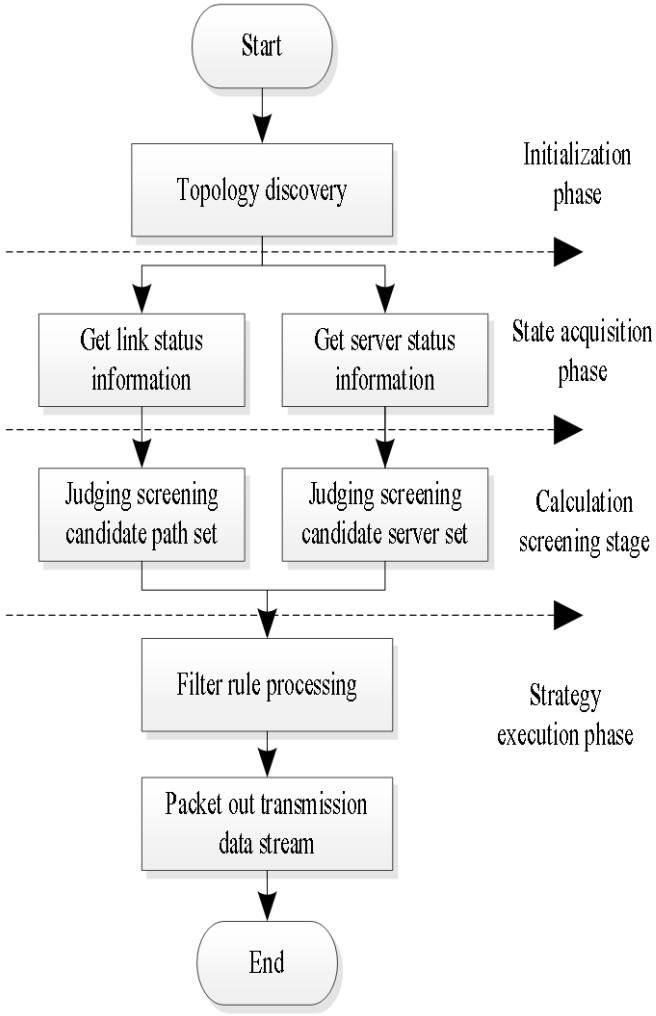

Fig. 3 Ryu controller workflow

C. SDN Controller

SDN switch sends the video stream to the corresponding small base station according to the flow table sent by the SDN controller, and SDN controller mainly includes a network state management and transmission scheduling module. When SDN network receives the data sent by the video server, the video stream is scheduled according to the number of video layers to be sent in combination with the channel quality of the wireless network so that DSCN can fully utilize the wireless network resources. In DSCN, each small base station mainly includes link state management, network resource management, and data cache module where the link state management module maintains channel quality information among users. When small base station receives the data packet sent by the SDN exchange price, it will temporarily store it in the data cache module. Then, according to the amount of buffered data and the channel quality of each user, the optimal wireless network resource allocation scheme of the current time slot is comprehensively determined, and the data packet will be sent to corresponding user[12-14].

SDN network topology is represented by undirected graph $G=(V, E)$ where V and E respectively represent the set of nodes and the set of links in the network. *G* contains *M* controllers $C=\{c_1, c_2, \cdots, c_M\}$ and *N* switches $S=\{s_1, s_2, \cdots, s_N\}$ where $|V|=M+N$. It is assumed that the controller can be optimally deployed in the network topology, and one controller manages multiple switches to form a SDN sub-domain. Controller C_m has a processing capacity of $\eta_m, U_{cm} \in S$ is defined as a switch set managed by the controller C_m, and d is the number of hops among network devices. Let connection relationship between the switch S_i and the controller C_m is X_{im}, which is as shown in the formula (1).

$$X_{im}=\begin{cases}1, S_i \quad is\ connected \quad to\ C_m \\ 0, \quad others\end{cases} \tag{1}$$

In SDN network, the controller load mainly includes data interaction, routing rules and state synchronization.

Definition 1. Data interaction cost.

To implement centralized control, the sub-domain controller needs to exchange periodic information with the switch to collect hop count and traffic information of each switch. Let the data interaction cost of the controller C_m be $\ell_{data}^{C_m}$, which is as shown in equation (2).

$$\ell_{data}^{C_m}=\sum_{i=u_{c_m}} d_{im} \cdot v_{SW} \cdot x_{im} \tag{2}$$

where V_{SW} is the average rate of a switch, which depends on the number of links connected to each switch.

Definition 2. Routing rule cost.

When the switch receives new traffic, it sends a PACKET-IN message to the controller, and asks the controller to calculate the traffic path and deliver the routing entry so that the routing rule cost $\ell_{route}^{C_m}$ is generated in the controller, which is as shown in Equation (3).

$$\ell_{route}^{C_m} = f_{PACKET} \cdot \sum_{S_k \in S} \sum_{C_m \in C} \varphi_k \cdot d_{km} \cdot X_{km} \quad (3)$$

where f_{PACKET} is the average size of the PACKET-IN packet sent by the switch, and φ_k is the stream request rate of the switch S_k.

Definition 3. State synchronization cost.

In a distributed SDN network, synchronization messages are sent among controllers to maintain a global network view, and result in a state synchronization cost $\ell_{state}^{C_m}$, which is as shown in equation (4)

$$\ell_{state}^{C_m} = \lambda_{syn} \cdot \sum_{C_m, C_n \in C} X_{mn} \cdot d_{mn} \quad (4)$$

where λ_{syn} related to f_{PACKET} is the network data shared among controllers, and $\lambda_{syn} < f_{PACKET}$ since the controller does not share all the information of the sub-domain.

Therefore, the controller load in the network is a linear aggregation of the three types of costs mentioned above, and the load ℓ_{c_m} of the controller C_m is calculated as shown in equation (5)

$$\ell_{c_m} = \ell_{data}^{C_m} + \ell_{route}^{C_m} + \ell_{state}^{C_m} \quad (5)$$

The load difference between controller C_m and controller C_n is

$$Q_{c_m c_n} = \frac{\ell_{c_m}}{\ell_{c_n}} \quad (6)$$

Based on the above settings, the load difference matrix of the controller can be obtained[15].

$$Q_{m \times m} = \begin{cases} 1 & Q_{c_1 c_2} & Q_{c_1 c_3} & \cdots & Q_{c_1 c_m} \\ Q_{c_2 c_1} & 1 & Q_{c_2 c_3} & \cdots & Q_{c_2 c_m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ Q_{c_m c_1} & Q_{c_m c_2} & Q_{c_m c_3} & \cdots & 1 \end{cases} \quad (7)$$

For a given load difference threshold u, the switch migration trigger condition is as shown in equation (8).

$$\int C_m, C_n \in C, \varepsilon_{mn} = \left(Q_{c_m c_n} - Q_{c_n c_m}\right) > u \quad (8)$$

where ε_{mn} is trigger factor.

If the trigger factor ε_{mn} exceeds the preset threshold u, it will be considered that the controller load imbalance occurs in the network and switch migration will be triggered. Therefore, the load difference threshold is important to adjust the controller load distribution in the network, and the value of the load difference threshold u is related to both the maximum load and the minimum load of the controller. The load difference threshold is calculated as shown in equation (9).

$$u = \frac{\max Q_{M \times M} - \min Q_{M \times M}}{\max Q_{M \times M}} \quad (9)$$

where $maxQ_{M \times M}$ is the maximum value of the element in the matrix $Q_{M \times M}$, and $minQ_{M \times M}$ is the minimum value of the element in the matrix $Q_{M \times M}$. In order to effectively improve the migration efficiency and reduce the migration complexity, the migration object is determined based on the load difference matrix $Q_{M \times M}$. The controller that migrates the switch out is defined as the migration-out controller, and the controller that receives the switch is defined as the migration-in controller. As a result, by detecting $Q_{M \times M}$, the migration-out controller set C_{OM} and the migration-in controller set C_{IM} are obtained. If $\lambda_{mn} > u$, the controller c_m and c_n will be respectively added to the set C_{OM} and C_{IM}.

The connection between switches constitutes an undirected graph. The composition of network equipment in data center can be abstracted as connection graph $G(S, L)$, as shown in Figure 3, $S=\{S_1, S_2, S_3,...,S_n\}$ represents a collection of network devices such as an openflow switch, $L=\{L_{12},...L_{ij}\}$ denotes the set of links. L_{ij} is the link between node i and node j, where $i, j \in S$. R_{ij} represents the path from node i to node j. Each link L_{ij} corresponds to a ω_{ij}, which represents the weight value of the link between node i and node j. ω_{ij} can be the link utilization or residual bandwidth, etc. The adjacency matrix of undirected simple graph can be used to represent the whole network topology and the corresponding relationship between the links between switches and the link weight value ω_{ij}. Basic mathematical model of routing module is shown in Figure 4.

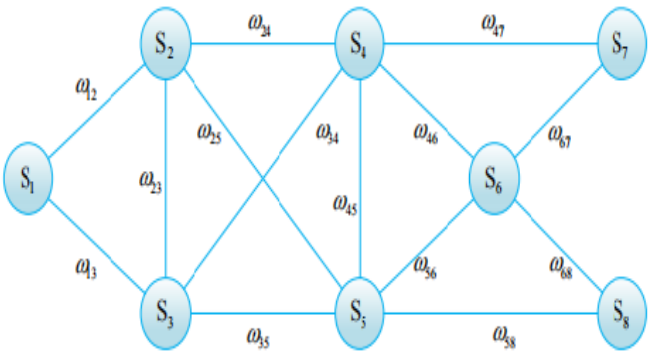

Fig.4 Basic mathematical model of routing module

D. Acquisition and Calculation of Link Delay

The NetworkMonitor part is responsible for monitoring the available bandwidth and delay of the network link, and storing the acquired data in the *networkx* graph structure. The NetworkHandler component reads the relevant parameter values in the *networkx* graph in real time, and filters out the qualified path set according to the PSTS algorithm [16-17].

Step1.The controller sends a *packet_out* message to the switch A. The data segment of the packet carries the LLDP protocol and is timestamped with the packet.

Step2.The action field of the *packet_out* message indicates that Switch A floods or forwards the packet to another port. In actual operation, the packet is forwarded to Switch B by Switch A.

Step3.After receiving the packet from Switch A, Switch B cannot match the corresponding flow entry. Therefore, Switch B sends *packet_in* message to the controller.

Step4.After receiving the *packet_in* message, the controller parses the data packet, extracts the timestamp of the data field, and subtracts the current time from the timestamp to obtain T_1.

Step5.Similarly, the total time from the controller to switch B, then from switch B to switch A and finally back to the controller can be obtained. It is recorded as T_2. Moreover, round-trip time (RTT) between the controller and switch A plus the RTT between switch A and switch B plus the RTT between controller and switch B equals to (T_1+T_2) .

Step6.Whereafter, controller sends a *echo_request* message with timestamp to switch A and switch B. After receiving the message, the switch immediately replies to *echo_reply* message with the *echo_request* timestamp. Let t_a and t_b be the RTT between switch A, switch B and the controller. Finally, the forward and backward average delay of the link between switch A and switch B can be calculated. The calculation is.

The location of the quantum particle is represented with the quantum bit and is defined as

$$T_d = \frac{(t_1 + t_2) - (t_a + t_b)}{2} \tag{10}$$

The class that implements the probe network delay function in the above work is the NetworkDe-layDetector class. The main functions in the class are that the *get_total_delay* function is responsible for superimposing the delay of each link to obtain the delay of each path, and saving the result in a two-dimensional array of paths. The *show_delay_statis* function is used to display the link delay information, including the source address, destination address, and delay.

E. Acquisition and Calculation of Link Bandwidth

In this paper, the OpenFlow protocol is used to obtain information in this paper such as switch ports and flow tables through packets, which thereby obtains the available bandwidth of the link in the SDN network.

Step1. Calculating the data transmission rate of the port.

When the controller queries the port data flow information, the type field of the *ofp_stats_request* is set to *OFPST_FLOW* and sent to the switch. Moreover, the switch responds by uploading the *ofp_stats_reply* message. The controller matches the *dura-tion_sec, duration_nsec, byte_count* values of the body field in *ofp_stats_reply* message, where *duration_sec, du-ration_nsec* represents the duration of the data stream, and *byte_count* represents the bit size of the data stream. Thus the actual transmission rate of the data stream can be calculated.

$$\text{speed} = \frac{\text{byte\_count}}{\text{durationsec} + \text{duration}_{\text{nsec}}/10^9} \times 8 \tag{11}$$

Step2. Calculating Link Throughput.

Calculating link throughput is similar to obtaining the actual transmission rate of the data stream. The controller sends an *ofp_stats_request* message whose type field is *OFPST_PORT* to the switch, and the switch answers the message. In order to calculate the link throughput, two moments t_1, t_2 are defined, and the link throughput can be derived from

$$\text{capacity} = \frac{[(\text{tx\_bytes}_{t_2} + \text{rx\_bytes}_{t_2}) - (\text{tx\_bytes}_{t_1} + \text{rx\_bytes}_{t_1})]}{(t_2 - t_1) \times 8} \tag{12}$$

where *tx_bytes* represents the number of bytes sent and *rx_bytes* is the number of bytes received.

Step3. Calculating the Available Bandwidth of the Link.

After obtaining the current occupied bandwidth *speed* and the link throughput *capacity*, available bandwidth of the current link can be calculated.

$$free\_bandwidth = capacity - speed \tag{13}$$

After obtaining the available bandwidth of the current link, the above operations can be performed cyclically, and the available bandwidth of all links in the entire network will be gained[18].

F. Acquisition and Tradeoff of Server Status

In ServerInfo part, command component called, each corresponding command is executed on each server in *ac-cess_port* to respectively obtain the memory usage *mem* and the CPU load *cpu*. *free* command is used to get the memory usage. When calculating the memory usage, a rough formula will be adopted. The calculation is as follows.

$$p_{\text{usedmem}} = \frac{\text{used}}{\text{total}} \tag{14}$$

uptime command is used to obtain CPU load information, and the three numbers after *loadaverage* parameter respectively represent the average system load within 1 min, 5 min, and 15 min. According to the average system load, development trend of the system load can be judged. What's more, to ensure CPU performance, CPU load threshold $Th_{i,cpu}$ can be defined as follows.

$$Th_{i,cpu} = cpu_s \times cores \times 0.8 \tag{15}$$

III. Simulation and Analysis

The Mininet simulation platform is used to simulate the SDN network and PSTS algorithm. In addition, the performance of the algorithm is analyzed and compared based on the simulation. Due to experimental conditions and the fact that the Mininet simulation platform does not have a virtualized server device, only link module of the PSTS algorithm is tested in this experiment. To evaluate the performance of the PSTS algorithm and the DLB algorithm, following two indicators are defined [19-20].

The average bandwidth utilization η refers to the average ratio of the actual bandwidth *real_bwi* obtained by each data stream *i* to the specified bandwidth *assigned_bwi o*f the data stream. The calculation formula is as follows.

$$\eta = \frac{\sum_{i=1}^{n} \frac{real\_bwi}{assigned\_bwi}}{n} \tag{16}$$

The average transmission delay τ refers to the average time when each data stream *i* arrives at the server minus the time when data stream is sent from the client. The calculation formula is as follows.

$$\tau = \frac{\sum_{i=1}^{n} (server\_t_i - client\_t_i)}{n} \tag{17}$$

In order to verify the PSTS controller algorithm proposed in this paper, traffic load is first used as an independent variable in the simulation. Besides it, the evaluation of the algorithm is embodied to analyze changes in network performance under different traffic loads. What's more, set h_0 as the Iperf client, h_a~h_z as the Iperf server, test 200 sets of data, and the experimental results will be gotten.

In Figure 5, as the traffic load increases, the average bandwidth resource utilization of the DLB algorithm and the PSTS algorithm proposed in this paper are increasing, but average bandwidth utilization increases at a slower rate in PSTS algorithm. When the traffic load factor reaches 3, the utilization of PSTS algorithm is about 46% lower than that of DLB algorithm, and then DLB algorithm increases at a faster rate.

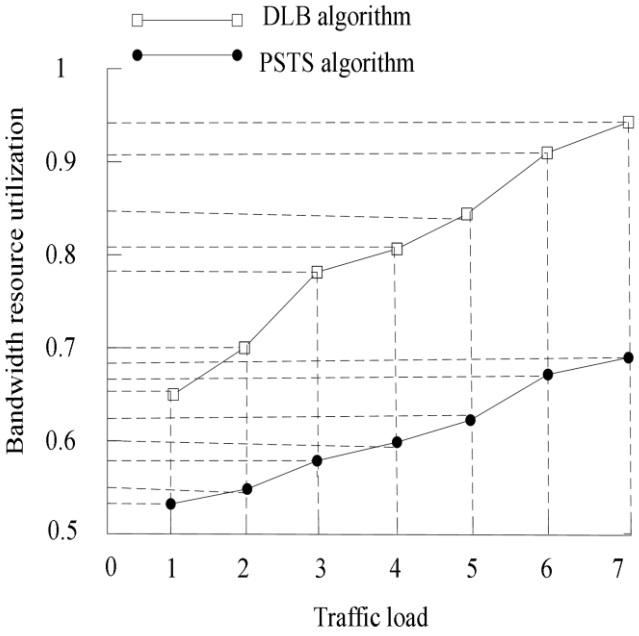

Fig. 5 Comparison and analysis of average bandwidth utilization data

The average transmission delay of DLB algorithm and PSTS algorithm under different traffic loads is shown in Figure 6. It can be seen that the performance of the PSTS algorithm is significantly better than the DLB algorithm.

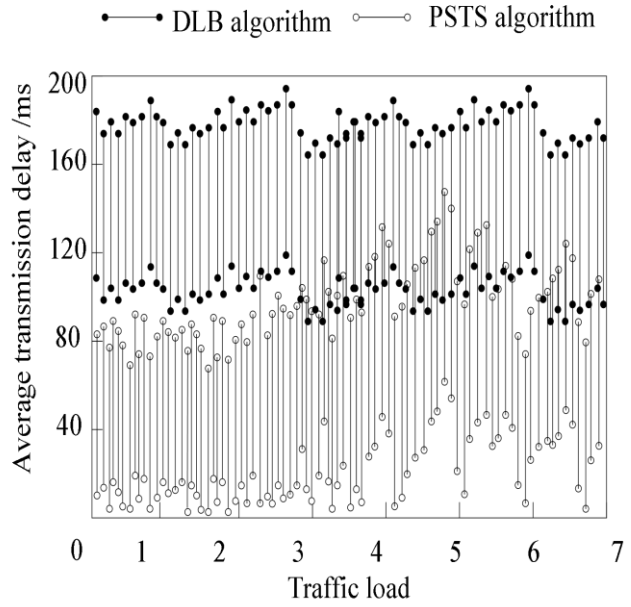

Fig. 6 Comparison of average transmission delays

According to the above result, the traffic load is set to 0.5. Moreover, terminal transmits a data stream with a bandwidth of 100M for 20 times, and the data stream transmission path and the destination host are determined by the controller. The trend

of system utility changes as the amount of terminal usage increases. Therefore, the change trend of system utility will decrease as the amount of terminal usage increases. The trends in wireless network system utility and terminal usage changes are shown in Figure 7.

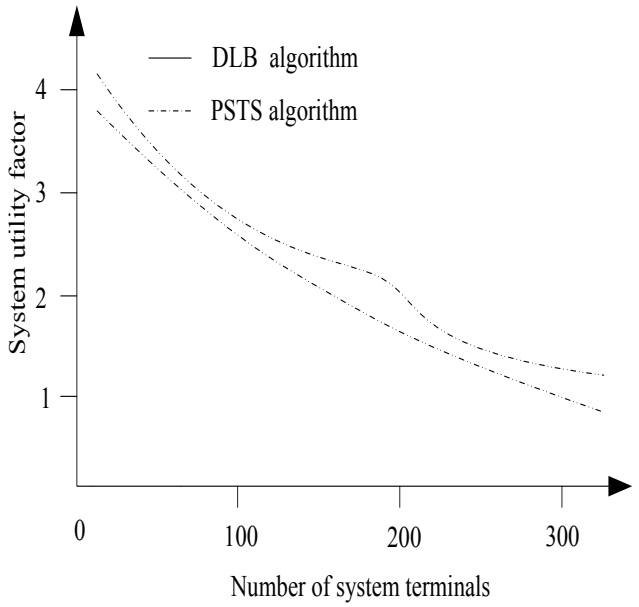

Fig.7 Trends in wireless network system utility and terminal usage changes

This is because wireless resources are limited, and the increase in terminal usage will reduce the average resource rate, which thereby reduces the wireless transmission rate, and fails to provide sufficient bandwidth to transmit high-capacity data. Therefore, it is shown that the algorithm adjusts the layer numbers of data to be sent and received according to changes in system status. In addition, according to Figure 5, it can be judged that the system utility of PSTS algorithm is higher than DLB algorithm since the classic DLB control algorithm is suitable for SDN network environment in lower density, and the goal of the DLB algorithm is to stabilize the buffer length of the user-based short data instead of maximizing the system utility. On the contrary, when building mathematical models, PSTS algorithm clearly aims to maximize system utility.

Streaming video interruptions in wireless networks can greatly affect the experience of network playback. Then define the video playback interruption rate which is the ratio of video interruption and recovery due to network delay during the total playback time to evaluate the video playback quality. In the two algorithms, the relationship between the video playback interruption rate and the number of used terminals is as shown in Figure 8.

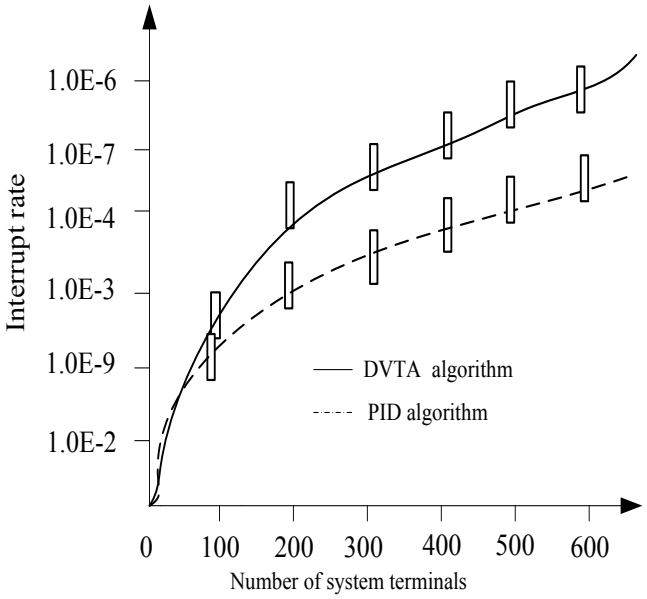

Fig. 8 Trends in wireless network outage rate and terminal usage change

Compared with DLB algorithm, PSTS algorithm has a lower video playback interruption rate since when deriving the PSTS algorithm, rate constraint is considered in this paper to reduce the video playback interruption rate by ensuring that the long-term average video transmission rate is greater than the long-term average video playback bit rate.

In DLB algorithm, controllers almost select the same data transmission path so that the load of h_1 increases sharply, and the waiting queue of data becomes longer, which affects the performance of h_1, reduces the data processing speed, and even causes packet loss due to local congestion. However, In PSTS algorithm, each time data streams arrive on different hosts as when the load of each host is the same, polling will be used by the PSTS algorithm to select the destination host to relieve the pressure among hosts. Therefore it can be seen from the above four statistical comparison figures that the performance of PSTS algorithm proposed in this paper is significantly better than DLB algorithm.

Based on the modular function implementation of the Ryu controller, PSTS proposed in this paper can effectively obtain the influencing factors at the link level and server level. During implementation process, the sorting and filtering of each link and each server is implemented to support the ultimate optimal traffic scheduling solution through measuring the impact factors (performance indicators) at the link level and server level and introducing the impact factors that have been obtained before to calculate the weight. The simulation results show that under the same traffic load PSTS algorithm proposed in this paper can achieve higher average bandwidth utilization and lower average transmission delay than DLB algorithm which is widely accepted now. In terms of load distribution, when there

are a large number of parallel data streams in the network, PSTS algorithm can distribute the data streams to each server more efficiently, which greatly avoids the local congestion in the network, improves the processing speed of the data stream and thereby improves the overall performance of the network.

IV. Conclusion

For the widely accepted DLB algorithm which only takes the link with the largest bandwidth in each hop as the final selected path, it is believed in this paper that traffic scheduling should consider the global network conditions. On one hand, that the bandwidth of a certain link in one hop is the largest cannot represent other conditions in this link are better than the remaining links in the hop. On the other hand, factors affecting traffic scheduling include server load conditions in addition to network link conditions. Based on the above considerations, an SDN-based load balancing network controller algorithm is proposed in this paper, which takes into account the network link status and server load status. Furthermore, the simulation results show that the PSTS algorithm designed in this paper can obtain larger average bandwidth utilization, smaller average transmission delay, and more balanced network load, which effectively avoids the local congestion of the network and shows good network performance.

With the increasing expansion of network scale, a robust and efficient dynamic routing system design is very important. In the future, the whole system will be further improved to achieve independent module function, reduce coupling, and develop efficient interfaces to meet more business requirements. In addition, a single controller can not deal with the problem of snd network across multiple regions. It needs a distributed cluster composed of multiple SDN controllers to avoid the reliability, scalability and performance problems of a single controller node. Therefore, it is necessary to improve the whole routing and traffic scheduling algorithm so that it can be successfully transferred to the SDN architecture of the controller cluster.

References

[1] P.P. Devi, B. Jaison. "Protection on Wireless Sensor Network from Clone Attack using the SDN-Enabled Hybrid Clone Node Detection Mechanisms", *Computer Communications*, 2020, vol.152, pp. 316-322.

[2] Kim D, Kim J, Wang G, et al. "K-GENI testbed deployment and federated meta operations experiment over GENI and KREONET". *Computer Networks*, 2014, 61(14), pp. 39-50.

[3] Ronald P. Bianchini Jr, John Paul Shen. "Interprocessor Traffic Scheduling Algorithm for Multiple-Processor Networks". *IEEE Transactions on Computers*, 1987, 36(4), pp. 396-409.

[4] Zhang Y J. "A Multi-server Scheduling Framework for Resource Allocation in Wireless Multi-carrier Networks". *Wireless Communications IEEE Transactions on*, 2007, 6(11), pp. 3884-3891.

[5] Kavitha T, Rajamani V. "A Modified Efficient Traffic Scheduling Algorithm for Routing in Optical WDM Mesh Networks". *Journal of Optical Communications*, 2013, 34(3), pp. 193-200.

[6] Chiasson J, Tang Z, Ghanem J, et al. "The effect of time delays on the stability of load balancing algorithms for parallel computations", *IEEE Transactions on Control Systems Technology*, 2005, 13(6), pp. 932-942.

[7] Georgi Enchev, Nikolay Djagarov, Zhivko Grozdev, Selecting Type of Communication for Wireless Sensor Network on Board of a Vessel, *WSEAS Transactions on Systems and Control*, pp. 384-389, Volume 13, 2018

[8] Ho, Chen L , Jason R , et al. "The effect of local scheduling in load balancing designs". *Acm Sigmetrics Performance Evaluation Review*, 2008, 36(2), pp. 110-112.

[9] Seo J, Kim K H. "A Prototype of Online Dynamic Scaling Scheduler for Real-Time Task based on Virtual Machine". *International Journal of Electrical & Computer Engineering*, 2016, 6(1), pp. 205-211.

[10] Lu J B, Hu W D, Yu W X. "Resource Management Algorithm Based on Covariance Control for Phased Array Radars". *Acta Electronica Sinica*, 2007, 35(3), pp. 402-408.

[11] Harsha M B, Sarojadevi H, "A Comparative Study of Different Static and Dynamic Load Balancing Algorithm in Cloud Computing with Special Emphasis on Time Factor". *Journal of Engineering Research and Applications*, 6(2), pp. 61-65, 2016.

[12] Heirich, A. "A Scalable Diffusion Algorithm For Dynamic Mapping And Load Balancing On Networks Of Arbitrary Topology". *International Journal of Foundations of Computer Science*, 1997, 8(3), pp. 329-346.

[13] Ramanathan P , Sivalingam K M , Agrawal P , et al. "Dynamic resource allocation schemes during handoff for mobile multimedia wireless networks". *IEEE Journal on Selected Areas in Communications*, 2006, 17(7), pp. 1270-1283.

[14] Pahalawatta P V , Katsaggelos A K . "Review of content-aware resource allocation schemes for video streaming over wireless networks". *Wireless Communications & Mobile Computing*, 2010, 7(2), pp. 131-142.

[15] Sheu J P, Kao C C, Yang S R, et al. "A Resource Allocation Scheme for Scalable Video Multicast in WiMAX Relay Networks". *IEEE Transactions on Mobile Computing*, 2013, 12(1), pp. 90-104.

[16] Jun-Hong Z, Zhen-Shui L I, Meng-Quan Z, et al. "LQG/LTR Multivariable Controller Design for a Gust Load Alleviation System". *Flight Dynamics*, 2007, 25(4), pp. 33-36.

[17] Radzi N A M , Suhaimy N , Ahmad W S H M W , et al. "Context Aware Traffic Scheduling Algorithm for Power Distribution Smart Grid Network". IEEE Access, pp. 99:1-1, 2019.

[18] Younes, Maram, Bani. "An efficient dynamic traffic light scheduling algorithm considering emergency vehicles for intelligent transportation systems", Wireless Networks,vol. 45, no. 2, pp. 79-88, 2018.

[19] Sargento S, Valadas R. "Capacity and Cross-Traffic Estimation of All Links in a Path Using ICMP Timestamps". *Telecommunication Systems*, 2006, 33(1-3), pp. 89-115.

[20] Kapoor R, Zanella A, Gerla M. "A Fair and Traffic Dependent Scheduling Algorithm for Bluetooth Scatternets". *Mobile Networks & Applications*, 2004, 9(1), pp. 9-20.

[21] Xue D, Qin Y, Siew C K. "Performance analysis of a novel traffic scheduling algorithm in slotted optical networks". *Computer Communications*, 2007, 30(18), pp. 3559-3571.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)

This article is published under the terms of the Creative Commons Attribution License 4.0
https://creativecommons.org/licenses/by/4.0/deed.en_US