Using PageRank Algorithm to Predict and Prevent Congestion in Software Defined Networking

Jian-Bo Chen, Wei-Kang Cheng

Department of Information and Telecommunications Engineering, Ming Chuan University, Taiwan
jbchen@mail.mcu.edu.tw, cyes90111@gmail.com

Abstract

This research used the PageRank algorithm to predict and prevent congestion. Both the traffic volume and the importance of switches are considered. The most popular switch can to determined according to the ranking. After the most popular switch is determined, the weights can be adjusted to prevent congestion. To prevent congestion efficiently, the values of weight are also determined according to the value of ranking. Simulation results show that this method can evenly distribute loads and reduce response time.

Keywords: PageRank, Software defined Networking, Congestion

1 Introduction

In a traditional network, the routing protocol uses metrics to determine the best path. Metrics vary according to different routing protocols. For example, the metric for RIP (Routing Information Protocol) is hop count; the metric for OSPF (Open Shortest Path First) is the cost, which depends on bandwidth; the default metric for EIGRP (Enhanced Interior Gateway Routing Protocol) is bandwidth and delay, which also depends on bandwidth. Although these metrics are different, all metrics are static. Being static means that once the values of the metric are defined or calculated, these metrics become fixed values. When network traffic is generated by hosts, the loads on switches or links increases. In this situation, the metric should be adjusted to satisfy the current network environment needs. But these routing protocols do not capture the loading information, so cannot adjust the metrics immediately.

In SDN (Software Defined Networking), the Control Plane and Data Plane are separated from switches. Control Plane is a central controller which captures information from switches. Based on that information, the controller can determine the best path from source to destination. Once the best path is determined, the controller then adds flow entries to every switch along this path. Each switch has been added to an entry that instructs how traffic is to be processed or forwarded. In traditional networks, every router has its own routing table. The routing entries in the routing table are generated by the router itself according to specific routing protocols such as RIP, OSPF, or EIGRP. The traditional routing protocols use their own algorithms to find the best path, for example, RIP uses the Bellman-Ford algorithm; OSPF uses the Shortest Path First algorithm, and EIGRP uses the Diffusion Update Algorithm. In the SDN environment, the functions of a traditional router and switch are combined into a new device, called “SDN switch”. In the SDN switch, the table for instructing how traffic is to be processed or forwarded is called a “flow table”. Each entry in the flow table is called a “flow entry”. The flow entry content is added by the controller, not the switch itself. Hence, the main responsibility of the SDN switch is to forward the packet according to the flow table. The hardware design for the SDN switch focuses on fast switching only. All other software tasks are transferred to the controller, meaning that the switch is a Data Plane.

The controller can obtain real-time network loads so that it can re-calculate the metric to achieve a new best path instead of plying an old path. Then the new path is added to the switches by the controller. Therefore, we can dynamically adjust the metric according to current network loads. [1]

Once the controller collects the loading information, how to appropriately change to the best path is another challenge. On the topic of congestion prediction or congestion prevention, many researchers have used different methodologies to solve these problems. Some researchers have used statistical methods that can prevent congestion according to past or current traffic. [2-3] Some researchers have employed artificial intelligence methods which build a training model, then this training model predicts congestion. [4-5] But there are few researchers who have used some kinds of page ranking algorithms to solve these problems. Page ranking algorithm is used to rank the importance of a web page. The importance of a web page not only takes into consideration the number of ingress links but

*Corresponding Author: Jian-Bo Chen; E-mail: jbchen@mail.mcu.edu.tw

DOI: 10.53106/160792642021112206006

also considers the importance of the web page which creates this link. The main concept of this paper is to predict which switch is the most popular switch. The most popular switch is determined not only by considering the incoming traffic but also needs to consider the popularity of the switch which generates this traffic. This concept is rarely proposed in the literature. Most studies consider only the number of links, not the importance of nodes. When using the page ranking algorithm to predict and prevent congestion, both the traffic volume and the importance of switches are considered to achieve higher performance.

In this study, the researchers want to predict which switch is the most likely to be congested, which is referred to as a "popular" switch. If the most popular switch can be identified in advance, the controller can determine the new best path, so that future traffic will avoid going through this switch. Consequently, network congestion can be prevented and traffic can be evenly distributed. In order to implement this kind of prediction, the PageRank algorithm is used as the prediction method in this paper. [6-7]

PageRank algorithm was proposed by Larry Page in 1998. [8] The original design for the PageRank algorithm was to analyze the importance of web pages. PageRank algorithm gives one PageRank value, denoted as PR value, for each web page. The web page with the highest PR value is considered the most popular web page. In the original version of the PageRank algorithm, there are no weights, that is, each hyperlink is counted as one. Some researchers proposed a weighted PageRank algorithm that can add weights to each hyperlink. [9]

Some researchers use the PageRank algorithm to solve problems other than those associated with web pages [10-11]. Other researchers have improved the efficiency of the PageRank algorithm [12-13]. In this paper, the weighted PageRank algorithm is used to identify the popular switch, which will likely be congested. When the network traffic is generated from source host to destination host, the traffic will go through a path from source to destination and pass from switch to switch. In the mechanism proposed in this paper, the switch acts as a web page; the network traffic acts as the hyperlink. After the controller captures the network traffic, the controller can use these switches and traffic to generate a directional graph. The weights of each link are the volume of network traffic requested. Based on this directional graph, a weighted PageRank algorithm can be adapted to calculate the PR value for each switch. The switch with the highest PR value is defined as the most popular switch, which is predicted as likely to be congested. The controller can re-calculate a new best path which avoids going through this switch if at all possible. Thus, the network traffic can be redistributed to other switches, and congestion will be prevented.

This paper is organized as follows. Section 2 describes some related background information. Section 3 presents the proposed mechanism. Section 4 shows the simulation results. Section 5 comprises the conclusions from this research.

2 Related Background Information

In this section, some background information is introduced, including shortest path first algorithms and web ranking algorithms. Then the PageRank algorithm and the weighted PageRank algorithm are introduced.

2.1 Shortest Path First Algorithms

In order to find the shortest path in a given graph, Edsger Dijkstra proposed the Dijkstra algorithm in 1959. This algorithm can find the shortest paths between nodes in a graph. The basic operation is described as: In a given source node, when finding the shortest path to destination nodes, the distance from this source node to all neighbor nodes is updated with the minimum distance. After some iterations, the shortest path from the source node to all other nodes in the graph can be found. [14]

Dijkstra algorithm can find the shortest path, but it is a kind of Breadth-First Search algorithm. Peter Hart, Nils Nilsson, and Bertram Raphael proposed the A* algorithm in 1968 which is a kind of Depth-First-Search algorithm. A* algorithm can achieve better performance by using a heuristic function to reduce the search time. [15-16] In the A* algorithm, if $g(n)$ represents the actual distances from the source node to any node n; $h(n)$ represents the estimated distances from any node n to the target node. Then the formula for the A* algorithm is defined in Eq. (1).

$$f(n) = g(n) + h(n) \tag{1}$$

This formula follows some performance characteristics. First, if $g(n)$ equal to zero, the algorithm only needs to calculate the heuristic function $h(n)$. In this case, this algorithm is a Best-First-Search algorithm consuming less execution time but it may not find the best solution. Second, if $h(n)$ is not larger than the actual distance from node n to the target node, the best solution can be found. In the case of being less than $h(n)$, the more nodes need to be calculated which will decrease performance. Third, if $h(n)$ is equal to zero, the algorithm only needs to calculate $g(n)$ without a heuristic function. In this case, the A* algorithm becomes a single source Dijkstra algorithm.

2.2 Web Ranking Algorithms

In this section, some web ranking algorithms are introduced. The first web ranking algorithm is the PageRank algorithm which was proposed by Larry Page in 1998. It is the kind of ranking algorithm used in the Google search engine. The concept of the

PageRank algorithm is to consider the quantity and quality of relationships between web pages. Hence, a higher ranking page means that it has a large number of hyperlinks with higher popularity directed to it.

Hyperlink-Induced Topic Search (HITS) algorithm is a kind of web link analysis algorithm that was proposed by Jon Kleinberg in 1999. [17] In this algorithm, two kinds of web pages are defined, including the Authority page and the Hub page. Authority page is a page with many incoming links from other web pages; Hub page is a page that has many links out to other important web pages. In this algorithm, both the Authority score and Hub score can be computed to rank the importance of Authority pages and Hub pages. Authority score is calculated by the summation of the total Hub scores that are pointed to this page; Hub score is calculated by the summation of the total Authority values that this page points to.

TF-IDF (Term Frequency - Inverse Document Frequency) is a kind of text-mining technology that can rank the keyword of any document. [18] The importance of a keyword will increase according to the number of times this keyword appears in a document, and it will decrease according to the number of times this keyword appears in the set of documents. Some variations of TF-IDF algorithms can rank the keywords in search engines.

Due to the increase of blogs, the EigenRumor algorithm was proposed by K. Fujimura, T. Inoue, and M. Sugisaki in 2005. [19] The focus of this algorithm is to find the most appropriate blogs in the search environment. EigenRumore algorithm scores each blog page according to the weights of hub and authority values based on eigenvector calculations. Then the rank scores of blogs can be obtained.

The comparison between these ranking algorithms is shown in Table 1. [20] From this table, it can be seen that only the PageRank algorithm deals with web structure without considering web contents. The aim of this research is to predict and prevent congestion based on traffic flow, which is only concerned with network topology (web structure), not requested content (web content). Therefore, PageRank is selected for the proposed method, and the operational details are described in later sections.

Table 1. Comparison of ranking algorithms

Algorithm	PageRank	HITS	TF-IDF	EigenRumor
Author	Larry Page	Jon Kleinberg	K. Spärck Jones	K. Fujimura, et.al
Year	1998	1998	1972	2005
Web Structure	Yes	Yes	No	No
Web Content	No	Yes	Yes	Yes

2.3 PageRank Algorithm

In 1998, Larry Page stated the main purpose of the PageRank algorithm is to analyze the relationship and importance of web pages. Each web page has its own PageRank value (PR value) which represents the importance of this web page. There are two key concepts for the PageRank algorithm. First, if a web page is linked by another important web page, then this web page is more important and its PR value should be increased. Second, if a web page with a high PR value links to another web page, then the linked web page should have increased PR value because it is linked by an important web page.

The PR value is calculated according to the quantity and quality of links. PageRank algorithm is based on a probability operation. The formula for PR value is shown in Eq. (2) and is described as follows.

$$PR(A)=\frac{1-d}{N}+d\left(\frac{PR(B)}{L(B)}+\frac{PR(C)}{L(C)}+\frac{PR(D)}{L(D)}+\cdots\right) \quad \textbf{(2)}$$

If there are three web pages (B, C, and D) linked to web page A, $PR(A)$ denotes the PR value of web page A. $L(B)$ denotes the sum of links which are egress links by web page B. $PR(A)$ is the sum of these three web pages' PR values divided by the sum of its egress links. N is the total number of web pages in this network topology and d is the damping factor which always is 0.85 in order to avoid the black hole page which has no egress link to other web pages. In the PageRank algorithm, the sum of all PR values of these web pages is 1, and the web page with highest the PR value is the most important web page in the network topology.

2.4 Weighted PageRank Algorithm

In the original design of the PageRank algorithm, the weight for each link is the same, that is, there is no concept of weights in the links. This means that the probability of each link to another web page is the same. Some researchers proposed an improvement of the PageRank algorithm with weights in each link. If the relationship between the two web pages is more significant, a higher weight is assigned to this link. Otherwise, the link is assigned with a lower weight. The formula for weighted PR value is shown in Eq. (3) and is described as follows

$$PR(A)=\frac{1-d}{N}+d\left(\frac{W_{BA}PR(B)}{WL(B)}+\frac{W_{CA}PR(C)}{WL(C)}+\frac{W_{DA}PR(C)}{WL(C)}\cdots\right) \quad \textbf{(3)}$$

In Eq. (3), W_{BA} is the weight of the link which is from page B to page A. $WL(B)$ is the total weights of all egress links from page *B*.

3 Proposed Method

In this section, the proposed method is introduced, including how to find the popular switch, and how to adjust the weights.

3.1 Find the Popular Switch

An example network topology is used to describe the proposed method, as shown in Figure 1. In this topology, the topology graph $G = (V, E)$ contains six switches, and the bandwidth for each link is assumed the same so that the initial weight can be defined as 1. This topology is described below.

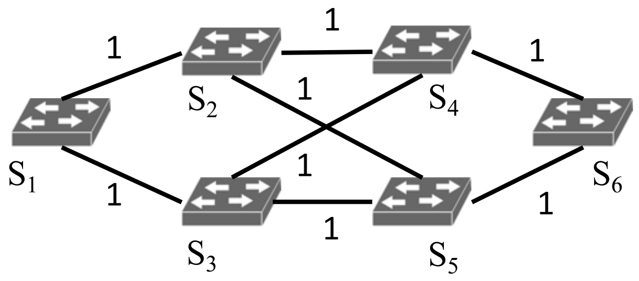


Figure 1. Original graph of network topology

$G = (V, E)$
$V = \{S_1, S_2, S_3, S_4, S_5, S_6\}$
$E = \{(S_1, S_2), (S_1, S_3), (S_2, S_4),(S_2, S_5), (S_3, S_4), (S_3, S_5), (S_4, S_6), (S_5, S_6)\}$
$W[i][j]=1$ for $(S_i, S_j) \in E$

where *G* is the topology; *V* is the set of switches; *E* is the set of links; *W* is the weight of links.

When the traffic is generated, the controller captures the traffic for each link. In an SDN environment, the controller can use *PortStats* and *FlowStats* events to obtain the amount of traffic easily. After a specific time period, the controller uses this traffic volume to generate a directional graph, $G'(V', E')$. The generation process of G' is described below.

Firstly, there is no traffic, so that $V' = \{\}$ and $E' = \{\}$. When the first traffic was generated, it is assumed that the traffic is sourced from S_1 to S_6, based on the shortest path first algorithm, the traffic will go through S_1->S_2->S_4->S_6, so that $V' = \{S_1, S_2, S_4, S_6\}$ and $E' = \{<S_1, S_2>, <S_2, S_4>, <S_4, S_6>\}$. The weights $w[1][2] = w[2][4] = w[4][6] = 1$. When the second traffic was generated, it is assumed that the second traffic is sourced from S_1 to S_5, based on the shortest path first algorithm, the traffic will go through S_1->S_2->S_5, so at this time, $V' = \{S_1, S_2, S_4, S_6, S_5\}$ and $E' = \{<S_1, S_2>, <S_2, S_4>, <S_4, S_6>, <S_2, S_5>\}$. The weights $w[1][2] = 2$, $w[2][4] = w[4][6] = w[2][5] = 1$. After a period of time, the final directional graph will be generated.

In this directional graph, the PageRank algorithm can be adopted to find the switch with the highest PR value. A switch with the highest PR value means that some important switches have a higher amount of traffic going to this one switch, and this switch is the most popular switch. This popular switch is predicted to have more and more traffic and soon be overwhelmed. Hence, the original weights are adjusted, the SPF algorithm is executed again, and new flow entries are added to switches. Then the congestion of this popular switch will be prevented.

An example can illustrate how to generate the directional graph and how to use the weighted PageRank algorithm. Assuming that after a period of time, the controller captures traffic, a directional graph is generated, as shown in Figure 2. In this graph, for example, assume that S_1 sends 5 requests to S_3 and S_3 sends 8 requests to S_1. The weights in the directional graph between S_1 and S_3 are 5 and 8.

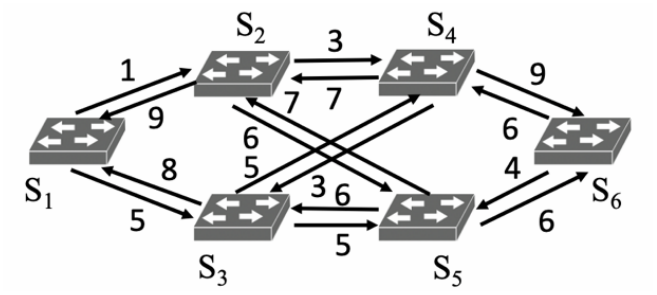


Figure 2. Directional graph generated by traffic

In this example, the ingress traffic can easily be counted to find out which one is the highest traffic switch. The ingress traffic count is shown in Table 2, and S_1 is the highest volume traffic switch.

Table 2. Ingress traffic for each switch

	Traffic Count
S_1	17
S_2	15
S_3	14
S_4	14
S_5	15
S_6	15

In this proposed method, the highest traffic switch may not be the popular switch. In the weighted directional graph, the weighted PageRank algorithm is used to obtain the PR values for each switch. In this example, there are six switches so that the initial PR values for each switch is 1/6, that is, $PR(S_1) = PR(S_2) = PR(S_3) = PR(S_4) = PR(S_5) = PR(S_6)=1/6$. After several iterations of the weighted PageRank algorithm, the final converged PR values are shown in Table 3.

Table 3. Converged PR values

	PR value
$PR(S_1)$	0.1697
$PR(S_2)$	0.1525
$PR(S_3)$	0.2116
$PR(S_4)$	0.1659
$PR(S_5)$	0.1644
$PR(S_6)$	0.1359

In Table 3, one can see that the switch with the highest PR value is S_3. But based on Table 2, the highest traffic switch is S_1, showing that the highest traffic switch is not the most popular switch. In the example, switch S_3 is the most popular switch, which will be overwhelmed. In order to prevent congestion in S_3, the initial weights are adjusted for some links. After the SPF algorithm is re-executed, the traffic will be prevented going through S_3 as much as possible, and the traffic can be evenly distributed to all other switches.

3.2 Adjust the Weights

Using the PageRank algorithm with the amount of traffic, one can predict which switch will be congested. The goal is to prevent this switch from being congested. In the proposed mechanism, SPF algorithm is used to find the best path, so one only needs to adjust the initial weights and execute the SPF algorithm again to obtain a new path, and then add these flow entries to the switches. Two methods are proposed to adjust the weights. The first is to add a fixed value to the weights; the second is to add a value that is dependent on the PR value.

In the first method, based on the previous section, the switch with the highest PR value is S_3, so the weights for all links connected to S_3 must be adjusted. In this example, 1 is added to each link so that the weights for S_1 to S_3, S_4 to S_3, and S_5 to S_3 are changed to 2, as shown in Figure 3.

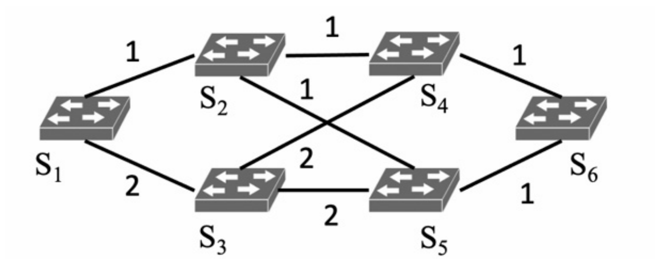


Figure 3. Adjust weights with a fixed value

In the second method, a specific value is added to the weight of each link. The value added is the maximum PR values for its adjacent switches, that is, $max(PR(i), PR(j))$, where i, j are the two adjacent switches connected by this link. When the weight of each link is adjusted, a new weighted graph is generated as shown in Figure 4. For example, the weight is 1.1659 for the link between S_2 and S_4, because $PR(S_2)$ is 0.1525 and $PR(S_4)$ is 0.1659, $max(PR(S_2), PR(S_4))$ is 0.1659, so the weight for S_2 and S_4 is adjusted to 1.1659.

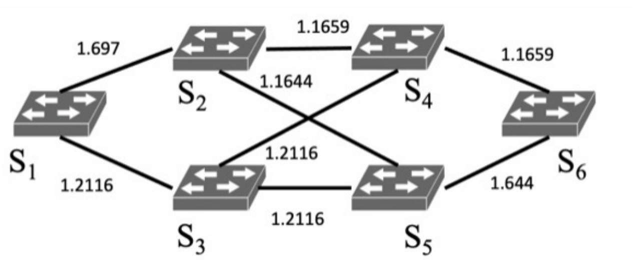


Figure 4. Adjust weight with PR values

For adoption of both method 1 and method 2, the SPF algorithm was executed in this new weighted graph. The new path was added to these switches by the SDN controller. The pseudo-code for the proposed method is shown below.

```
G is the network topology with node V, edge E, and weighted matrix W
G' is the directional graph with node V', edge E', and weighted matrix W'

initialize the network topology G(V, E) with weighted matrix W
while loop
    find the shortest path for each node in G
    add flow entries to each node
    for every generated traffic
        find the shortest path from source to destination
        for every node in this path
            if node is not in V'
                add this node to V'
        for every link in this path
            if link is not in E'
                add this link to E' and update W'
            else
                update W'
    execute the weighted PageRank algorithm with G' and W'
    update W for all links connected to the node with maximum PageRank
```

4 Simulation Results

In the experimental simulation, a scale-free topology was used. A scale-free topology is one type of complex topology with some specific features. The most important feature is that most of the nodes connect to little nodes, and little nodes connect to most of the nodes. These key nodes are called "hubs", and the hub nodes can support the failure of network nodes. Internet topology is a kind of scale-free topology so scale-free topology was adopted for simulation topology in this study. In this simulation, a scale-free topology with 100 nodes was used.

The purpose of the simulation is to predict which switch will be overwhelmed and to prevent congestion. If the loads can be evenly distributed to all switches, then congestion will not occur. Therefore, the percentage of loads and standard deviation for each link will be calculated. If the standard deviation is low, it means that the loads are distributed evenly to all switches.

Some methods are compared with the proposed model. The first just uses the SPF algorithm with original weights, denoted as "Dijkstra/A*". In this method, in spite of the loads added in this topology, the initial weights still remain the same; that is, this method does not consider current loads. The second method is choosing the largest amount of traffic as the popular switch; that is, this method counts all the traffic and finds the switch with the highest traffic, denoted as "Highest Traffic". This method only considers the amount of traffic, rather than the relationship of traffic loads. The third method is using the proposed PageRank algorithm to find the most popular switch, denoted as "PageRank_fix". When the most popular switch is identified, the weights have a fixed value added for all links connected to this switch. The fourth method is also the PageRank algorithm, denoted as "PageRank_PR". But, different from the third method, this method adds the PR values to each link to adjust the weights.

When the controller gathers the topology, it uses the SPF algorithm to find the shortest path and add flow entries to switches. Then traffic is randomly generated in this topology. To get the loads and to calculate the percentage for each link, multimedia requests are added to generate streaming contents in the topology. The number of requests was increased from 50 to 500; and this traffic was used to find the largest volume of traffic and PR values. Then weights were adjusted and the standard deviation calculated, as shown in Figure 5. It can be seen that "PageRank_fix" and "PageRank_PR" can achieve lower standard deviations than "Dijkstra/A*" and "Highest Traffic". Moreover, "PageRank_PR" has a lower standard deviation than "PageRank_fix" because the weights for all links were adjusted to improve the distribution of loads.

Although standard deviation can prove that the loads are evenly distributed, the aim is to know whether the response time will be decreased when the proposed methods are adopted. Using the same experimental environment, the number of requests was increased from 50 to 500. When requests increase, the response time increases. In Figure 6, it can be seen that "PageRank_fix" and "PageRank_PR" can achieve better response times than the other two methods.

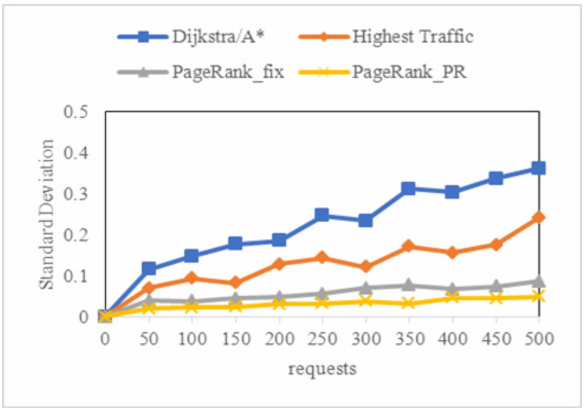


Figure 5. Comparison of standard deviations

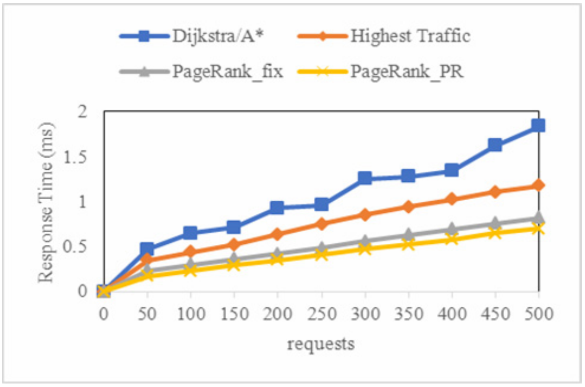


Figure 6. Comparison of response time

5 Conclusion

This paper proposed a congestion prediction and prevention method using the PageRank algorithm. PageRank algorithm can consider both the traffic volume and the importance of switches. The simulation results illustrate that this proposed method can evenly distribute the loads to every switch and reduce response time better than other methods. However, in order to use the PageRank algorithm, the controller must collect all the traffic in the topology, that is, there must be a high volume of Packet-in / Packet-out traffic from switches to controller. In a heavy loading environment, the channels between the controller and switches will be the bottleneck. In addition, the computing resources of the controller must also be considered. In future studies, the researchers will use out-of-band channels and multi-controllers to solve these issues.

References

[1] E. Akin, T. Korkmaz, Comparison of Routing Algorithms with Static and Dynamic Link Cost in SDN, *16th IEEE Annual Consumer Communications & Networking Conference*, Las Vegas, NV, USA, 2019, pp. 1-8.

[2] G. Saldamli, H. Mishra, N. Ravi, R. R. Kodati, S. A. Kuntamukkala, L. Tawalbeh, Improving Link Failure Recovery and Congestion Control in SDNs, *10th International Conference on Information and Communication Systems*, Irbid, Jordan, 2019, pp. 30-35.

[3] B. Zhu, J. Zhao, D.-A. Li, R. Bai, Application of Congestion Avoidance Mechanism in Multimedia Transmission over Mesh Networks, *International Journal of Ad Hoc and Ubiquitous Computing*, Vol. 31, No. 4, pp. 266-276, 2019.

[4] J. Wu, Y. Peng, M. Song, M. Cui, L. Zhang, Link Congestion Prediction using Machine Learning for Software-Defined-Network Data Plane, *International Conference on Computer, Information and Telecommunication Systems*, Beijing, China, 2019, pp. 1-5.

[5] Z. Yuan, P. Zhou, S. Wang, X. Zhang, Research on Routing Optimization of SDN Network Using Reinforcement Learning Method, *2nd International Conference on Safety Produce Informatization*, Chongqing, China, 2019, pp. 442-445.

[6] L. Davoli, L. Veltri, P. L. Ventre, G. Siracusano, S. Salsano, Traffic Engineering with Segment Routing: SDN-Based Architectural Design and Open Source Implementation, *Fourth European Workshop on Software Defined Networks*, Bilbao, Spain, 2015, pp. 111-112.

[7] M. Beshley, M. Seliuchenko, O. Panchenko, A. Polishuk, Adaptive Flow Routing Model in SDN, *14th International Conference on the Experience of Designing and Application of CAD Systems in Microelectronics*, Lviv, Ukraine, 2017, pp. 298-302.

[8] S. Brin, L. Page, The Anatomy of a Large-scale Hypertextual Web Search Engine, *Computer Networks and ISDN Systems*, Vol. 30, No. 1-7, pp. 107-117, April, 1998.

[9] S.-Y. Lee, Y.-G. Kim, S.-J. Lee, K. M. Lee, An Improvement of Weighted PageRank to Handle the Zero Link Similarity, *Joint 7th International Conference on Soft Computing and Intelligent Systems and 15th International Symposium on Advanced Intelligent Systems*, Kitakyushu, Japan, 2014, pp. 610-613.

[10] M. Tepper, G. Sapiro, A Short-Graph Fourier Transform Via Personalized PageRank Vectors, *IEEE International Conference on Acoustics, Speech and Signal Processing*, Shanghai, China, 2016, pp. 4806-4810.

[11] C.-C. Kuo, C.-L. Hou, C.-S. Yang, The Study of a Risk Assessment System based on PageRank, *Journal of Internet Technology*, Vol. 20, No. 7, pp. 2255-2264, December, 2019.

[12] H. Choi, J. Um, H. Yoon, M. Lee, Y. Choi, W. Lee, S. Song, H. Jung, A Partitioning Technique for Improving the Performance of PageRank on Hadoop, *2012 7th International Conference on Computing and Convergence Technology*, Seoul, Korea, 2012, pp. 458-461.

[13] Z. Zhu, Q. Peng, Z. Li, X. Guan, O. Muhammad, Fast PageRank Computation Based on Network Decomposition and DAG Structure, *IEEE Access*, Vol. 6, pp. 41760-41770, June, 2018.

[14] P. L. Frana, T. J. Misa, An Interview with Edsger W. Dijkstra, *Communications of the ACM*, Vol. 53, No. 8, pp. 41-47, August, 2010.

[15] A. Candra, M. A. Budiman, K. Hartanto, Dijkstra's and A-Star in Finding the Shortest Path: a Tutorial, *International Conference on Data Science, Artificial Intelligence, and Business Analytics*, Medan, Indonesia, 2020, pp. 28-32.

[16] J.-R. Jiang, H.-W. Huang, J.-H. Liao, S.-Y. Chen, Extending Dijkstra's Shortest Path Algorithm for Software Defined Networking, *16th Asia-Pacific Network Operations and Management Symposium*, Hsinchu, Taiwan, 2014, pp. 1-4.

[17] J. M. Kleinberg, Authoritative Sources in a Hyperlinked Environment, *Journal of the ACM*, Vol. 46, No. 5, pp. 604-632, September, 1999.

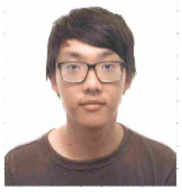
[18] G. Salton, C. Buckley, Term-weighting Approaches in Automatic Text Retrieval, *Information Processing and Management*, Vol. 24, No. 5, pp. 513-523, 1988.

[19] K. Fujimura, T. Inoue, M. Sugisaki, The EigenRumor Algorithm for Ranking blogs, *2nd Annual Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics on World Wide Web*, Chiba, Japan, 2005, pp. 1-6.

[20] D. K. Sharma, A. K. Sharma, A Comparative Analysis of Web Page Ranking Algorithms, *International Journal on Computer Science and Engineering*, Vol. 2, No. 8, pp. 2670-2676, 2010.

Biographies

Jian-Bo Chen received the Ph.D. degree in the department of computer science and engineering in Tatung University, Taipei, Taiwan in 2008. He is currently an assistant professor in the department of information and telecommunications engineering in Ming Chuan University, Taoyuan, Taiwan. His research interests include software-defined networking and load balance.



Wei-Kang Cheng received the B.S. and M.S. degrees in Information and Telecommunications Engineering from Ming Chuan University, Taiwan, R.O.C., in 2018 and 2019, respectively. His research interests include computer networking.