Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control
Journal homepage: http://kinetik.umm.ac.id
ISSN: 2503-2267
Vol. 6, No. 2, May 2021, Pp. 141-148

Comparison analysis of multipath routing implementation in software defined network

Syaifuddin*[1], Muhamad Fathul Azis[2], Fauzi Dwi Setiawan Sumadi[3]
Universitas Muhammadiyah Malang, Indonesia[1,2,3]

Article Info
Keywords:
Multipath Routing, Software Defined Network, QoS, Djikstra, Port statistic

Article history:
Received: March 05, 2021
Accepted: April 07, 2021
Published: May 31, 2021

Cite:
Syaifuddin, S., Azis , M. F. ., & Sumadi, F. D. S. (2021). Comparison Analysis of Multipath Routing Implementation in Software Defined Network. Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, 6(2). https://doi.org/10.22219/kinetik.v6i2.1228

*Corresponding author.
Syaifuddin
E-mail address:
saifuddin@umm.ac.id

Abstract
Multipath routing is a path search method used as a data transmission process utilizing more than one available path on a network. The multipath routing concept is directed to substitute the single path routing concept for reducing network traffic congestion by distributing data transmission through several available paths. It can be implemented in the Software Defined Network (SDN) environment that separates the control plane and the data plane, which provides flexibility by deploying application-based solutions to resolve the problem. This paper is directed to implement the modified Deep First Search (DFS) multipath routing algorithm and compare the proposed method with Dijkstra and normal DFS multipath algorithm. The contribution was designed by combining the node, edge, path, and bucket weight using port statistics available in OpenFlow standard and manual calculation. The results of the system's emulation showed that the overall algorithm could determine more than one path for the data transmission process. The average execution time on all algorithms produced 0.0903 ms for the modified DFS multipath algorithm, 0.0858 ms for DFS multipath algorithm, and 0.901 ms Dijkstra multipath algorithm, respectively. The QoS parameter testing results illustrated that the proposes method was better than another multipath routing algorithm in terms of throughput and jitter. However, based on packet loss percentage, the modified method was placed after normal DFS but still generated better results than Dijkstra. Overall, the implement multipath routing concept in SDN with all algorithms could be deployed to provide more than one data transmission path.

1. Introduction

Along with the rapid development of network topology [1], it will also create an inevitable impact on the infrastructure and complexity of network management. The network management based on the programable concept is an effort to replace the traditional network architecture model in order to be dynamic and adaptive [2]. SDN is a solution to actualize the mentioned notion with a new paradigm that separates the control plane as a controller and the data plane as forwarding devices [3]. In SDN architecture, the communication between the control plane and data plane is regulated by Southband interface protocol, e.g., OpenFlow [4]. OpenFlow is a sufficiently popular protocol and a standard protocol in the SDN network used to communicate securely. The separation between the control plane and data plane makes SDN architecture easy to be managed and developed because it only needs to modify the application on the controller, e.g., Floodlight [5], Onos [6], OpenDayLight [7], POX [8], and Ryu [9].

Routing becomes one of the important parts in network management focused on providing the path for data transmission. Generally, routing is correlated directly with path searching algorithms such as Dijkstra [10] and DFS [11]. Dijkstra's algorithm implements a greedy strategy that choose the path with the lowest cost, while the DFS performs the depth expansion on the left node then conducts backtracking when destination node has not been found. The goal for both algorithms is directed to discover only one path used for data transfer and it is the best path within the end-to-end topology prominently known by the term single path routing. Commonly, the single path routing can produce a network congestion because it only utilizes single path to forward data. Therefore, multipath routing may become a solution for that problem [12]. In several research, multipath routing implementation has been used on networking topology which supports multiple paths such as Fat-Tree [13], [14]. In papers [15] [16], showed the benefits of multipath routing such as load balancing to data transmission that was better than single path routing.

Several studies have been conducted before about implementation concept of multipath routing on SDN. In [17], implemented Equal Cost Multipath (ECMP) routing scheme with modified Dijkstra algorithm to find all of available paths from Fat-Tree network topology. In [18], the researcher implemented modified DFS to adapt multipath routing concept using path calculation like Open Shortest Path First (OSPF). Other research, [19] simulated data transmission using TCP and UDP transmission protocol to test the network performance and compare which was better using Iperf tools and Wireshark as network traffic monitoring application and analysis. In [20], the experiment of testing for calculating

the quality of network services is performed by measuring certain performance metrics such as throughput, jitter, and packet loss.

Based on previous research, this paper proposed the modification of DFS algorithm to adapt multipath routing and using Dijkstra algorithm as comparison. The contribution pointed from the research is explained as follow:

1. The DFS algorithm will add port statistics parameter as routing metric on path calculation to define the shortest path.
2. The variable that will be included are the average number of packets being received and transferred on the defined links.
3. The total routing metrics is the addition of the node, edge, path, and bucket weight.
4. The proposed method will be comparatively analyzed with the regular DFS and Dijkstra algorithms using QoS variable (throughput, jitter, and packet loss).

The remaining section of the paper will show the detail of the methodology, the emulation results and analysis, and the conclusion of the research.

2. Research Method

Our research aims to provide an effective traffic management solution during data transfer. Therefore, the DFS algorithm was modified to be able find all available paths to adapt concept of multipath routing [17] [18]. Multipath routing over SDN can provide more flexible routing method, which conserves the network throughput effectively [15]. Compared with the single path routing method, multipath routing was better in terms of load balancing on network traffic [16], because it used more than one path for data transmission as shown in Figure 1.

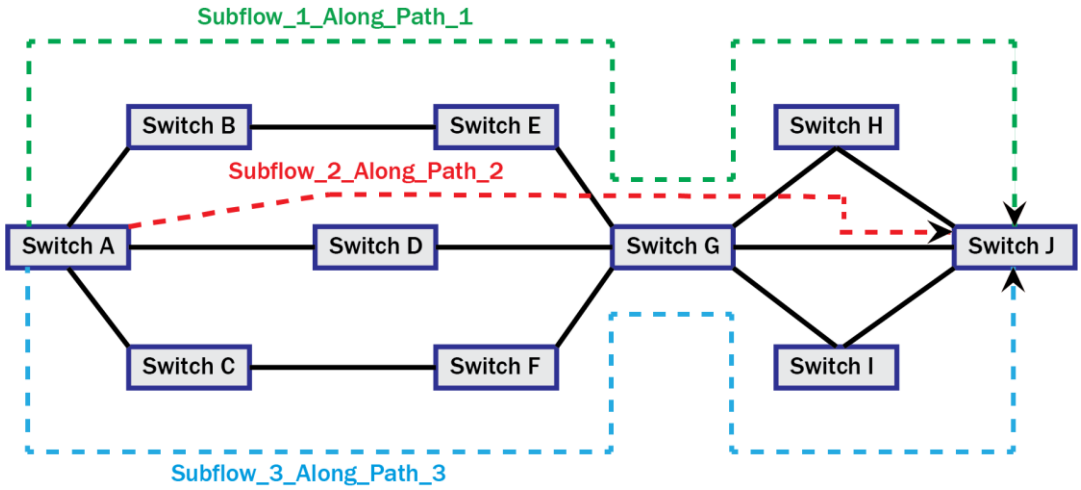

Figure 1. Example Transmission by Multipath Routing

There are three sub-flows that can be used to transfer data between node A and J. By using network topology that has multiple paths, SDN controller can choose among three paths that are available at the network to transmit data and the flow is split into three sub-flows. Multipath routing can be implemented on SDN architecture with OpenFlow protocol [21], which is regulated by Open Networking Foundation (ONF). Specifically, switches as data plane infrastructure will inform their flow and group tables to define how to forward data. Then the SDN controller establishes a Transport Layer Security (TLS) connection with every switch to install rules on its tables. Subsequently, the controller is capable for commanding and monitoring the data transmission.

Table 1. Example of Flow Table

match_fields	priority	instructions
in_port=1	priority=1	actions=output:3
eth_type=0x800, ipv4_src=”10.0.0.6”	priority=1	actions=flood
eth_type=0x800, ipv4_dst=”10.0.0.5”	priority=1	actions=group:87
	priority=0	actions=controller:6633

A flow table is composed of one or more flow entries. Table 1 gives an example that the first flow entry means a packet coming from port number one should be forwarded to port number three. The second flow entry tells the switch to broadcast a packet to all its ports if the packet has source IP address 10.0.0.6. The third flow entry asks the switch

to handle a packet by the group table with ID 87 if the packet's destination IP address is 10.0.0.5. The fourth flow entry has no match fields, which means that if none of the above flow entries can handle the packet, it should be sent to the controller whose port is 6633.

Table 2. Example of Group Table

group_identifier	group_type	action_buckets
group_id=87	select	bucket=weight:30, actions=output:3 bucket=weight:70, actions=output:4

The last flow entry on Table 1 will trigger an event of table missing, which makes switch send an OFPT_PACKET_IN message to the controller that contains the packet information. For resolving this case, the controller will generate new flow entry by using path searching algorithm and install the flow entry in switch by sending OFPT_PACKET_OUT and OFPT_FLOW_MOD messages [22]. Group table are used as additional processes from flow table illustrated in Tabel 2. The third flow entry from Tabel 1 asks the switch referring to Table 2 with group_id=87 to handle the packet forwarding. Furthermore, switch will pick the bucket actions based on the path weight for data transmission. In the example from Table 2, switch will forward packet through port 3 and 4 with probabilities 0,3 on port 3 and 0,7 on port 4. The number of ports for data transmission process depends on the available paths in the topology such as the Fat-Tree topology so that the multipath routing can be implemented [11][13][14].

Table 3. Modified DFS Multipath Algorithm

```
1  DFS(G, s, d)
2      p, is array for storing paths
3      stack, is tuple as (s, [s])
4      while (stack is not empty) do
5          pop stack, save to node, p
6          for next (G[node] - p)
7              if next is d
8                  append to p
9              else
10                 push to stack
11         end for
12     end while
13 return p
14 end DFS()
```

In order to actualize the multipath routing concept, this research proposed a modification of the DFS algorithm so that it can find all paths in a multiple path network topology [18] shown in Table 3. DFS algorithm performs a path searching by checking the left node in depth then then the backtracking will be carried out if it has not found the destination node, but when it is found the searching process will be stopped. However, according to specified process only one path can be found. Path's searching modification is performed when the path from the source node to the destination node has been found, then the known link and node will be removed or marked from the network so that there is no re-checking on the same path and the searching continues until all nodes on the network have been checked for all paths available on the network. Because the DFS algorithm returns an unweighted list of paths [18], it is necessary to measure the paths weight. The calculation technique of paths weight is shown in Equation 1 and Equation 2.

$$nw[v] = \frac{\sum_{f \in Flow(v)} Bits(f)}{Capacity(v)} \quad (1)$$

$$ew[e] = \frac{\sum_{f \in Flow(e)} Bits(f)}{Bandwidth(e)} \quad (2)$$

Assume a graph $G = (V, E)$ derived from SDN topology, is weighted, directed, and connected. For a node $v \in V$ and an edge $e \in E$, let $Flow(v)$ and $Flow(e)$ denote the set of all the flows passing trhough v and e, respectively, let $Capacity(v)$ be the $capacity$ of v (i.e., the number of bits that v can process per second) and let $Bandwidth(e)$ be the bandwidth of e (i.e., the number of bits that e can transmit per second). Based on the previous equations, the total paths calculation can be determined as metric described in Equation 3.

$$pw(p) = \sum_{e \in E} ew(e) + \frac{rx\_bytes(e)}{rx\_packets(e)} + \frac{tx\_bytes(e)}{tx\_packets(e)} \quad (3)$$

$pw(p)$ expresses the path weight, $p\ (V, E)$ is a path that consists of several edges. The weight is calculated by adding edge weight in a path. As explained in the introduction, this research added port statistics as a parameter for the path calculation method which were the number of packets received and transmitted on specific edge. Therefore, the minimum distance of a path's calculation used a bucket weights formula shown in Equation 4.

$$bw(p) = \left(1 - \frac{pw(p)}{\sum_{i<n}^{i=0} pw(i)}\right) \times 10 \quad (4)$$

$bw(p)$ denotes the bucket weight from a path, which $pw(p)$ is path weight in previous step where lower cost is the better paths. Based on the context of OpenFlow group tables with bucket actions [22], the priority of choosing a bucket actions are selecting the highest bucket weight as explained in Table 2.

Furthermore, in term of the research experiment, the proposed method was emulated using Mininet that supported the OpenFlow protocol for SDN architecture [23]. It was one of the most popular tools used to design custom topologies by writing python scrip. Ryu introduced as the SDN controller during the experiment to manage switch with python programing language [24]. All of the emulations were performed on a $k = 4$ Fat-tree topology with 20 switches and 8 hosts [13] [14], then the bandwidth capacity was arranged at 1Gbps for each edges to simulate multipath routing in multiple path network scheme as shown Figure 2.

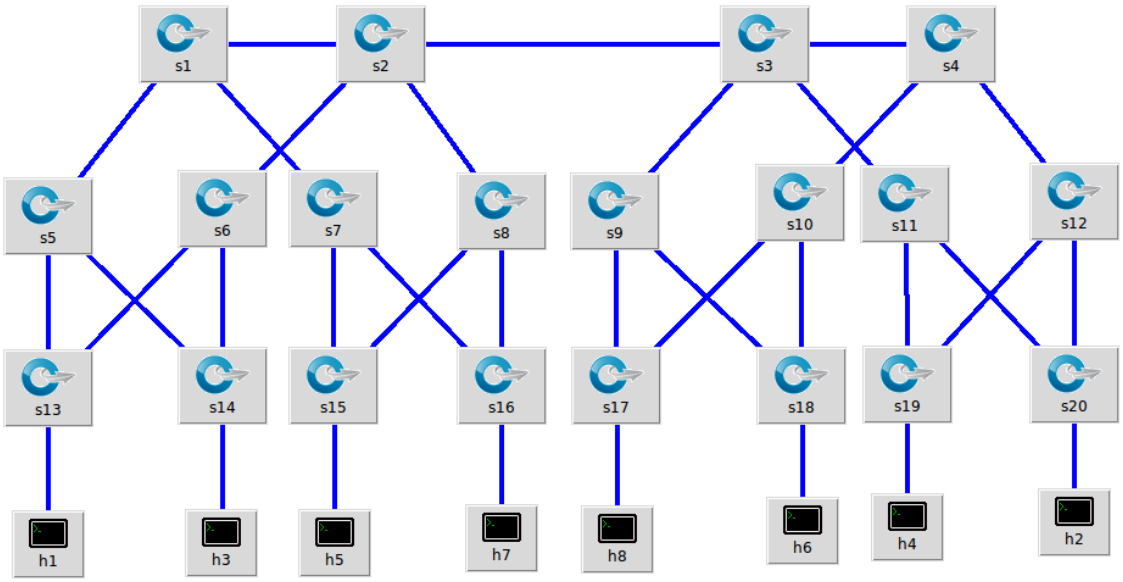

Figure 2. Fat-tree Topology on Mininet

In term of the research experiment, there were several scenarios for evaluating the performance. The first scenario was to show different path used during data transmission by sending packets using ping command from h5 to h6. The second scenario was performed to measure the execution time needed for algorithm to find all of the available paths on the multiple path network by pinging from h5 to h6 with 10 times iteration. In term of QoS evaluation, the list of evaluated parameters were throughput, jitter, and packet loss [20]. Iperf [25] was used as a testing tools where h5 deployed as a server and h6 as a client to simulate data transmission using UDP traffic [26] within 120 seconds over 10 parallel client connection and traffics loads variation between 20 until 100 Mbit. Iperf also utilized to measure the performance results on the parameter of jitter and packet loss. Moreover, Wireshark [27] also deployed as a traffic monitoring tools for displaying the result of throughput.

3. Results and Discussion

Based on the multipath routing experiment, depicted from the first scenario, the results showed that the path was used for data transmission generated from the proposed algorithm was slightly different with another algorithm as shown in Table 4.

Table 4. Path Searching Results on Fat-Tree Topology

	Modified DFS Multipath	DFS Multipath	Dijkstra Multipath
Path Searching Results	[15, 8, 2, 9, 18], [15, 8, 3, 9, 18], [15, 7, 1, 2, 9, 18]	[15, 8, 3, 9, 18], [15, 8, 2, 9, 18], [15, 8, 3, 4, 10, 18]	[15, 8, 2, 9, 18], [15, 8, 3, 9, 18], [15, 7, 1, 2, 9, 18]

The first and second paths used in Modified DFS Multipath and DFS Multipath were the same but it was different in priority. A significant difference found on the third path used between Modified DFS Multipath with the path of (15, 7, 1, 2, 9, 18) and DFS Multipath with the path of (15, 8, 3, 4, 10, 18). While the paths used by Dijkstra Multipath were all the same as Modified DFS Multipath. Overall, all of the algorithm could implement multipath routing using three optional paths for data transmission.

The next experiment was calculating the time for a multipath routing algorithm to find all of the paths on the Fat-tree network topology based on the second scenario as illustrated in Table 5.

Table 5. Execution Time Results on Fat-Tree Topology

Iteration	Modified DFS Multipath (ms)	DFS Multipath (ms)	Dijkstra Multipath (ms)
1	0,072	0,096	0,893
2	0,096	0,094	0,892
3	0,098	0,097	0,894
4	0,100	0,072	0,895
5	0,100	0,071	0,891
6	0,098	0,068	0,893
7	0,073	0,097	0,934
8	0,071	0,068	0,902
9	0,098	0,098	0,927
10	0,097	0,097	0,893
Average	0.0903	0,0858	0,901

Based on the results shown in Table 5, the average execution time needed between modified DFS multipath algorithm and DFS multipath algorithm did not fluctuate too much. However, the DFS multipath algorithm had a faster execution time than the other algorithms.

3.1 Throughput Results

As shown in Table 6, the average throughput results informs that the modified DFS multipath algorithm has higher value than the other routing algorithms with 4820 Kbps based on various traffic load while DFS multipath is at 4695 Kbps and Dijkstra multipath is at 4424 Kbps.

Table 6. Throughput Results

Traffic Load (Mbit)	Modified DFS Multipath (Kbps)	DFS Multipath (Kbps)	Dijkstra Multipath (Kbps)
20	3722	6124	3583
40	6119	5239	4328
60	4846	2816	4758
80	5409	4147	4883
100	4008	5151	4571

3.2 Jitter Results

In term of jitter, based on results depicted from Table 7, the modified DFS multipath is better the other routing algorithms with jitter average results at 0.2576 ms followed by Dijkstra multipath at 0.2714 ms and DFS multipath at 0.4176 ms.

Table 7. Jitter Results

Traffic Load (Mbit)	Modified DFS Multipath (ms)	DFS Multipath (ms)	Dijkstra Multipath (ms)
20	0.287	0.622	0.008
40	0.008	0.568	0.326

60	0.236	0.009	0.047
80	0.048	0.566	0.543
100	0.709	0.323	0.433

3.3 Packet Loss Results

The last experiment scenario for calculating the packet loss is shown in Table 8. The average packet loss results indicated that DFS multipath was accumulating the packet loss result at 0.6%, followed by the modified DFS multipath at 0.74%, and Dijkstra multipath at 3.76%. The results described that the modified DFS multipath algorithm was better than Dijkstra multipath algorithm.

Table 8. Packet Loss Results

Traffic Load (Mbit)	Modified DFS Multipath (%)	DFS Multipath (%)	Dijkstra Multipath (%)
20	0.78	0.84	4.3
40	0.74	0.85	3.3
60	0.85	0.36	3.7
80	0.76	0.39	3.5
100	0.60	0.56	4.0

In term of the comprehensive results compared to the previous similar research in [14], the simulated Dijkstra Multipath algorithm using both bandwidth and link capacity metrics produced less effective results compared to the proposed modified DFS. The notable differences was pointed in all evaluated variables (Throughput, Jitter, and Packet loss). The results might occur because the routing metric utilized in modified DFS was complex and covered wide-ranging variable including the switch's bandwith usage, link capacity, and port's statistic.

4. Conclusion

Based on the experiment in this paper, it can be concluded that the propose method has better value in term of the QoS parameters. This circumstance indicated that the addition of port statistics routing metric did not affect the network performance. It also can be deduced that the modified DFS multipath algorithm was better than Dijkstra multipath algorithm. Overall, the implementation of multipath routing concept in SDN with all algorithms has been highlighted with the good results indicating that the algorithms could provide multipath data transmission process.

Notation

n	: the number of finding paths
$nw[v]$	: the node weight value on vertex
$ew[e]$	: the edge weight value on edges
$pw(p)$	: the path weight value on paths
$bw(p)$	: the bucket weight value on paths
$Flow_{(v)}$	: the number flows data of vertex
$Flow_{(e)}$	: the number flows data of edges
$Capacity_{(v)}$	: the number bits per second can be process on vertex
$Bandwidth_{(e)}$	: the number bits per second can be transmit on edges

Acknowledgement

The author would like to express profound gratitude for the Informatics laboratory of Universitas Muhammadiyah Malang to support this research.

References

[1] Lopes, F. A., Santos, M., Fidalgo, R., & Fernandes, S. (2016). A Software Engineering Prespective on SDN Programmability. *IEEE Communications Surveys and Tutorials, 18*(2), 1255-1272. https://doi.org/10.1109/COMST.2015.2501026

[2] Rowshanrad, S., Namvarasl, S., Abdi, V., Hajizadeh, M., & Keshtgary, M. (2014). A survey on SDN, the future of networking. *Journal of Advanced Computer Science & Technology, 3(2),* 232. https://doi.org/10.14419/jacst.v3i2.3754

[3] Collaguazo Jaramillo, A., Alcivar, R., Pesantez, J., & Ponguillo, R. (2019). Cost Effective test-bed for Comparison of SDN Network and Traditional Network. *2018 IEEE 37th International Performance Computing and Communications Conference, IPCCC 2018,* 1–2. https://doi.org/10.1109/PCCC.2018.8711223

[4] Jany, M. H. R., Islam, N., Khondoker, R., & Habib, M. A. (2018). Performance analysis of OpenFlow based software defined wired and wireless network. *20th International Conference of Computer and Information Technology, ICCIT 2017, 2018-January,* 1–6. https://doi.org/10.1109/ICCITECHN.2017.8281814

[5] Lee, K., Kwon, B., Kang, J., Heo, S., & Lee, S. (2017). Optimal Flow Rate Control for SDN-Based Naval Systems. *IEEE Transactions on Aerospace and Electronic Systems, 53(6),* 2690–2705. https://doi.org/10.1109/TAES.2017.2711679
[6] Varyani, N., Zhang, Z. L., & Dai, D. (2020). QROUTE: An Efficient Quality of Service (QoS) Routing Scheme for Software-Defined Overlay Networks. *IEEE Access, 8,* 104109–104126. https://doi.org/10.1109/ACCESS.2020.2995558
[7] Kim, T., & Choi, S. (2017). Load Balancing of Distributed Datastore in OpenDaylight Controller Cluster. *IEEE Transactions on Network and Service Management. vol. 16, no. 1,* 72-83. https://doi.org/10.1109/TNSM.2019.2891592
[8] Vaghani, R., & Lung, C. H. (2016). Investigation of data forwarding schemes for network resiliency in POX Software defined networking controller. *IET Wireless Sensor Systems, 6(4),* 130–137. https://doi.org/10.1049/iet-wss.2014.0107
[9] Xu, Y., Cello, M., Wang, I., Walid, A., Wilfong, G., Wen, C. H., … Chao, H. J. (2019). Dynamic Switch Migration in Distributed Software-Defined Networks to Achieve Controller Load Balance. *IEEE Journal on Selected Areas in Communications, 37(3),* 515–529. https://doi.org/10.1109/JSAC.2019.2894237
[10] Jiang, J. R., Huang, H. W., Liao, J. H., & Chen, S. Y. (2014). Extending Dijkstra's shortest path algorithm for software defined networking. *APNOMS 2014 - 16th Asia-Pacific Network Operations and Management Symposium*. https://doi.org/10.1109/APNOMS.2014.6996609
[11] Lei, Y. C., Wang, K., & Hsu, Y. H. (2015). Multipath routing in SDN-based Data Center Networks. *2015 European Conference on Networks and Communications, EuCNC 2015,* 365–369. https://doi.org/10.1109/EuCNC.2015.7194100
[12] Ramdhani, M., Hertiana, S. and Dirgantara, B., (2016). Multipath routing with load balancing and admission control in Software-Defined Networking (SDN). *2016 4th International Conference on Information and Communication Technology (ICoICT)*. https://doi.org/10.1109/ICoICT.2016.7571949
[13] Jo, E., Pan, D., Liu, J., & Butler, L. (2015). A simulation and emulation study of SDN-based multipath routing for fat-tree data center networks. *Proceedings - Winter Simulation Conference, 2015-January,* 3072–3083. https://doi.org/10.1109/WSC.2014.7020145
[14] Alghadhban, A., & Shihada, B. (2016). Energy efficient SDN commodity switch based practical flow forwarding method. *Proceedings of the NOMS 2016 – 2016 IEEE/IFIP Network Operations and Management Symposium, (Noms),* 784 – 788. https://doi.org/10.1109/NOMS.2016.7502899
[15] Fu, M., & Wu, F. (2017). Investigation of multipath routing algorithms in software defined networking. *Proceedings - 2017 International Conference on Green Informatics, ICGI 2017,* 269–273. https://doi.org/10.1109/ICGI.2017.21
[16] Pang, J., Xu, G., & Fu, X. (2017). SDN-Based Data Center Networking With Collaboration of Multipath TCP and Segment Routing. *IEEE Access, 5,* 9764–9773. https://doi.org/10.1109/ACCESS.2017.2700867
[17] Rhamdani, F., Suwastika, N. A., & Nugroho, M. A. (2018). Equal-cost multipath routing in data center network based on software defined network. *2018 6th International Conference on Information and Communication Technology, ICoICT 2018, 0(c),* 222–226. https://doi.org/10.1109/ICoICT.2018.8528730
[18] Hossen, M. S., Rahman, M. H., Al-Mustanjid, M., Shakil Nobin, M. A., & Habib, M. A. (2019). Enhancing quality of service in SDN based on multipath routing optimization with DFS. *2019 International Conference on Sustainable Technologies for Industry 4.0, STI 2019, 0,* 1–5. https://doi.org/10.1109/STI47673.2019.9068057
[19] Naing, M. T., Khaing, T. T., & Maw, A. H. (2019). Evaluation of TCP and UDP Traffic over Software-Defined Networking. *2019 International Conference on Advanced Information Technologies, ICAIT 2019,* 7–12. https://doi.org/10.1109/AITC.2019.8921086
[20] Binsahaq, A., Sheltami, T. R., & Salah, K. (2019). A Survey on Autonomic Provisioning and Management of QoS in SDN Networks. *IEEE Access, 7,* 73384–73435. https://doi.org/10.1109/ACCESS.2019.2919957
[21] Open Networking Foundation
[22] Wang, Y. C., & You, S. Y. (2018). An Efficient Route Management Framework for Load Balance and Overhead Reduction in SDN-Based Data Center Networks. *IEEE Transactions on Network and Service Management, 15*(4), 1422–1434. https://doi.org/10.1109/TNSM.2018.2872054
[23] Mininet
[24] Ryu
[25] Iperf
[26] Wang, M. H., Chen, L. W., Chi, P. W., & Lei, C. L. (2017). SDUDP: A Reliable UDP-Based Transmission Protocol over SDN. *IEEE Access, 5,* 5904–5916. https://doi.org/10.1109/ACCESS.2017.2693376
[27] Wireshark.