Efficient Multipath Routing Scheme for MPTCP-enable Software-Defined Networks

Khac Tuan Nguyen
Posts and Telecommunications Institute of Technology
Hanoi, Vietnam

Linh T. Nguyen
Posts and Telecommunications Institute of Technology
Hanoi, Vietnam
linhnt.B17VT215@stu.ptit.edu.vn

Huu Tien Vu
Posts and Telecommunications Institute of Technology
Hanoi, Vietnam
tienvh@ptit.edu.vn

Hai-Chau Le
Posts and Telecommunications Institute of Technology
Hanoi, Vietnam
chaulh@ptit.edu.vn

***Abstract*—In this paper, we study the multipath routing problem in MPTCP-enable software-defined networks. We have proposed an efficient multipath routing algorithm for creating bandwidth-abundant and flexible future software-defined core networks. Our proposed algorithm will figure out k-least-jointed shortest paths by using conventional k-shortest path algorithm, e.g., Yen's algorithm, incorporating with a greedy algorithm for the path selection. Thanks to that, the algorithm can help to exploit the use of MPTCP while reducing the bottleneck problem that may occur with shared links in the networks. In order to verify the efficiency of our proposal, we have implemented an MPTCP-enable software-defined testbed based on mininet, Ryu controller, and Kali Linux OS and perform numerical experiments to compare the network performance of our approach to traditional MPTCP as well as conventional TCP (without MPTCP). The obtained results confirmed that, by reducing the number of jointed links in the set of selected k-shortest paths, our solution outperforms the traditional MPTCP as well as that of conventional TCP networks.**

Keywords—Multipath TCP, software defined networking, multipath routing, k shortest path algorithm.

I. Introduction

Recently, the rapid growth of video/multimedia-centric services and the fast development of emerging next-generation network technologies including 5G, 6G have triggered and introduced new revenue potentials for Telcos, ISPs as well as Over-The-Top providers [1], [2]. They are expected to be able to deliver high video quality to the end users smoothly and cost-effectively to enhance the users' experience [3]. However, achieving good Quality of Experience (QoE) is still a challenging task because of many factors. Hence, great efforts from both academia and industry have been performed in order to optimize the video content delivery chain and enhance end users' QoE. The most common approaches are either based on network optimization or client driven adaptive video streaming [3].

In the network point of view, traditional TCP protocol that is employed widely at the present seems not to be able to meet the near future requirement. To cope with that, Multipath TCP (MPTCP) standardized by the IETF has recently emerged as a promising transport protocol capable of forwarding data traffic using multiple paths [4], [5], [6]. Key idea of MPTCP is to separate user's traffic into multiple paths to route in the network. However, one of the hardest challenges is how to perform routing for the multiple paths to bring optimal performance to the data flows [4]. Fortunately, Software-Defined Networking (SDN) has been emerged as one of the most promising network technologies that are capable of providing more programmability, automation and flexibility to the networks [7],[8][9]. SDN separates the control from the data plane of networking devices [7], [8]. In software-defined networks, the control plane becomes an external entity, called controller, while the data plane is composed of simplified networking devices, which carry out only forwarding packets along with a limited number of simple tasks. On the other hand, OpenFlow, which is the most notable protocol from the SDN's Southbound API, is utilized to provide a common interface for the controller interacting with its switches. The SDN controller has a global view of the overall network and is responsible for the management decisions. It is expected to offer a rich Northbound API to applications for flexibly and cost-effectively deploying new services as well as implementing network policies. Thank to that, the controller can calculate and perform routing to give a better route than the normal network system [10], [11], [12].

In this paper, we investigate MPTCP-enable software-defined networks for constructing next-generation bandwidth-abundant and low-latency networks. We propose an effective k-least-jointed shortest path routing scheme for exploiting the use of MPTCP while avoiding or reducing the bottleneck problem with shared links to improve the network performance. Our proposed multipath routing algorithm is able to find *k*-least-jointed shortest paths by using traditional *k*-shortest path algorithm incorporating with a greedy algorithm for the path selection. Based on that, the developed solution can help to exploit MPTCP while capable of limiting the bottleneck problem in the networks. We also implement a testbed and employ numerical experiments to evaluate and verify the performance efficiency of our proposal. It is demonstrated that, by reducing the number of jointed links in the set of selected *k*-shortest paths, our developed solution provides better performance than that of the traditional MPTCP approach as well as that of conventional TCP networks.

II. Proposed K-least-jointed Shortest Path Routing Algorithm for MPTCP-aware SDN Networks

Figure 1 shows a typical MPTCP-enable software-defined networks that we consider in this work. The software-defined network is capable of supporting multipath TCP for next generation ever-increasing bandwidth and QoS/QoE-ensured services. In the considered SDN system, MPTCP protocol works similarly to conventional ones. However, in our MPTCP-enable SDN network, sub-flow paths of each MPTCP connection are determined by SDN controller. The controller will dynamically calculate multiple paths for MPTCP connection based on current state of network information and control strategies applied in the network.

In fact, path calculation module plays an important role in network operations. Generally, in MPTCP-enable SDN networks, k-shortest path algorithms are applied to figure out multiple paths for assigning to sub-flows. Depending on the number of sub-flows utilized, an appropriate number of path candidates needs to be determined. Several conventional works have proved that efficient routing algorithm can help to greatly improve network performance [13]. The path selection is also relied on various network conditions, i.e., the applied traffic control strategy or the network information collected at the time of path calculation. Although using MPTCP with multiple paths for sub-flows greatly improves the network performance, the shortest paths may be not always the best choice and the selection of suitable multiple paths among found path candidates can also contribute significantly to network performance enhancement. However, to the best of our knowledge, up to now, conventional works simply applied k-shortest algorithm to find k paths and assign to sub-flows in load-balancing manner. This approach may encounter a bottleneck problem when two or more selected paths are sharing a same link and a traffic congestion may be occurred. As a result, it will affect the network operation and limit the performance of MPTCP in the network.

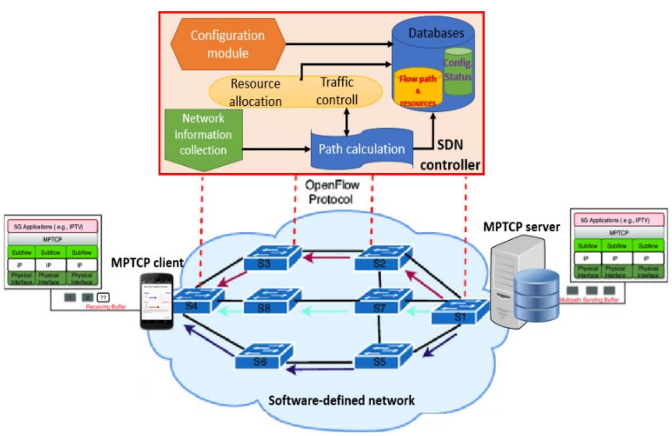


Figure 1. Typical MPTCP-enable software-defined network

To cope with that, we develop an effective multipath routing scheme that is able to reduce the number of joined links between the selected paths for exploiting the use of MPTCP more efficiently. Our idea is that, in order to find k shortest paths, we first find a set of path candidates with k-greater element number, say αk path candidates, where α is a pre-determined factor ($\alpha \geq 1$). We then select k paths among those found path candidates by using greedy algorithm to ensure that each selected path has a minimal correlation function of the joined link number in the relationship with the others. Consequently, our selected k paths can have a smaller number of jointed links among them. Hence, we call the proposed algorithm as k-least-jointed shortest path routing algorithm. Details of the proposed algorithm is described as following.

Algorithm: *k*-least-jointed shortest path*(t, s, d)*

```
Input:   ✓Network topology G(V, E)
         ✓Updated network information and configuration at the time t
         ✓Connection request R(t, s, d); t is the arrival time, (s, d) is the
          node pair
         ✓α: algorithm factor
         ✓k: number of shortest paths needed
Output:  k paths
Begin
    Calculate K := αk
    P_u is a path from s to u
    K_u is the number of found path from s to u
    B is a heap data structure containing paths
    P: set of shortest paths from s to t
    Q: set of k-least-jointed shortest paths from s to t
    Initialization:
    P := Ø
    Q := Ø
    K_u:=0 , for all u in V
    Insert path P_s = {s} into B with the cost of 0
    While B is not empty and K_t < K do
    Begin
        Let P_u be the shortest cost path in B with cost C
        B = B − {P_u}, K_u = K_u + 1
        if u = t then P = P U {P_u}
        if K_u ≤ K then
            For each vertex v adjacent to u do
            Begin
                Let P_v be a new path with cost C + w(u, v) formed
                 by concatenating edge (u, v) to path P_u
                Insert P_v into B
            Endfor
    Endwhile
    For i:=1 to k do
     Begin
        Find P_t a shortest path (in order of cost) that has the minimal
        correlation function of jointed link numbers with selected paths
        P = P U {P_u}
     Endfor
    Return Q
End
```

III. Performance Evaluation and Discussion

A. Experimental Setup

In this section, we implement an SDN network that enables MPTCP for high-speed, low latency and reliable services, i.e., video streaming or large capacity file transmission. We evaluate the performance of the network employing our proposed k-least-jointed shortest path routing scheme and compare to that of traditional MPTCP with k-shortest path scenario [13], [14] and the case of conventional TCP. For realizing the testbed, we use mininet [15] for creating an SDN environment with OpenvSwitches and external SDN controller based on Ryu [16] and two virtual machines (VM) playing as MPTCP client and server. Figure 2 shows our experimental testbed that consists of nine OpenFlow-enabled switches (called as OvS#i where #i is the switch index) controlled by a Ryu-based controller to support a pair of MPTCP client-server

VMs running Kali Linux 2020 [3] with the kernel version of *4.19.126.mptcp* (4-core CPUs and 8 GB RAM).

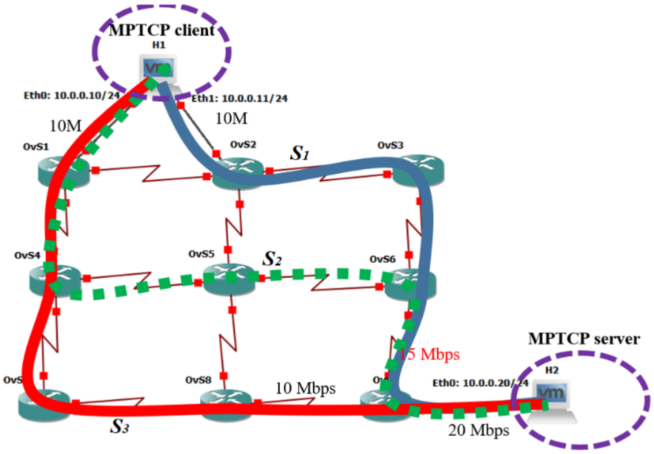


Figure 2. Experimental testbed

In the experimental testbed, the IP addresses of MPTCP client (denoted as H1) and server (H2) are set at 10.0.0.11/24 and 10.0.0.10/24 respectively. The IP address of the SDN controller is 127.0.0.1. The bandwidth of the link from OvS9 to the MPTCP server, H2, is assumed to be 20 Mbps, that of the link connecting OvS9 to OvS6 is 15 Mbps and the bandwidth of other remaining links is 10 Mbps. For simplicity, the link delay (link distance) will not be considered. We also assume that the number of *k*-shortest paths used in our systems is 2. The tested OpenFlow version is 1.3. We test with two service scenarios that are video streaming and large video file transferring.

B. *System Operation Verification*

We have tested our system to verify its operations with MPTCP service. Figure 3 shows the flow graph of a MPTCP connection establishment with 2 routes. It is confirmed that the MPTCP connection works successfully in our testbed. The MPTCP establishment for two streams (sub-flows) includes four steps: (1) is the handshaking for establishing the MPTCP connection of the first sub-flow, (2) is the way that MPTCP adds another sub-flow, (3) is the communication session of the MPTCP protocol using two sub-flows, and finally, (4) is the disconnection process.

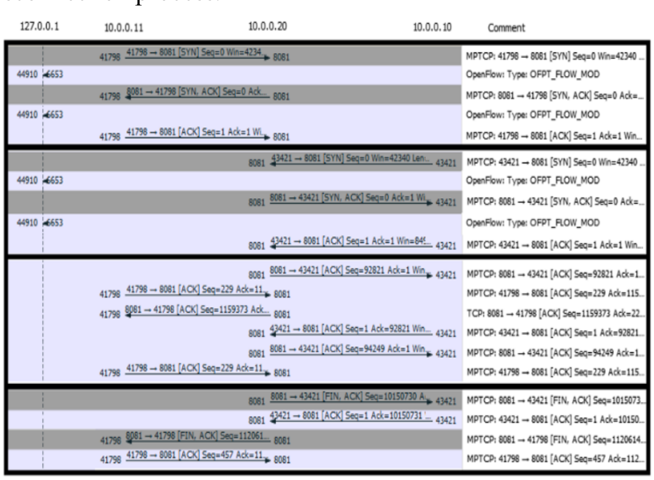
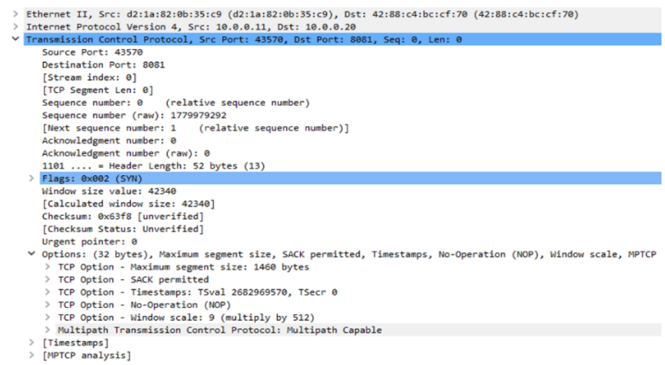
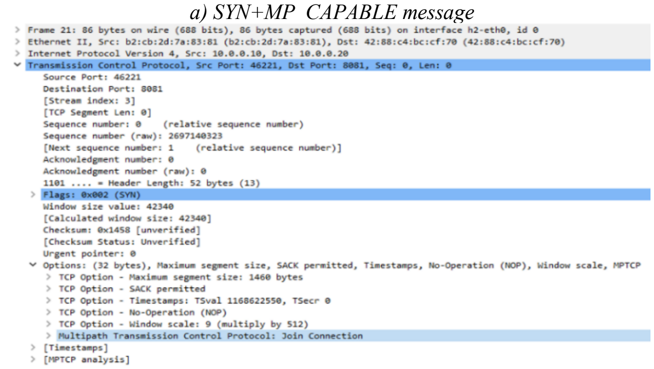


Figure 3. Flowgraph of MPTCP connection establishment

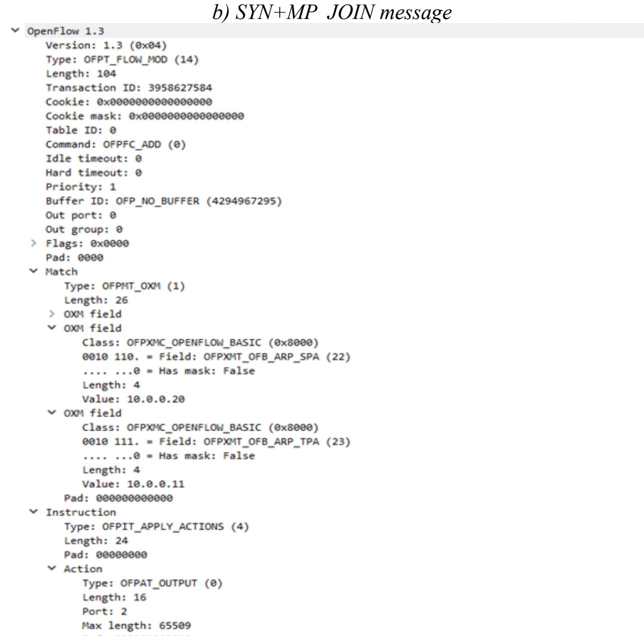
Furthermore, details of major OpenFlow messages for the MPTCP connection are listed in Figure 4; Figure 4.a shows the message for the process of establishing the connection of subflow1 by the MPTCP client (on the network interface with the IP address of 10.0.0.11); Figure 4.b is the message requested to add the second sub-flow by the MPTCP client (on the network interface with the IP address of 10.0.0.10) and, Figure 4.c is the OpenFlow MOD message from the Controller that specifies the port forward for OpenvSwitches when setting up the first sub-flow.



a) SYN+MP CAPABLE message



b) SYN+MP JOIN message

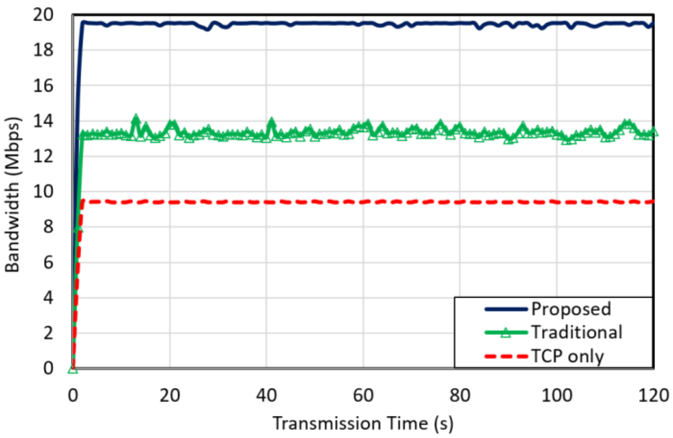


c) OpenFlow_MOD message

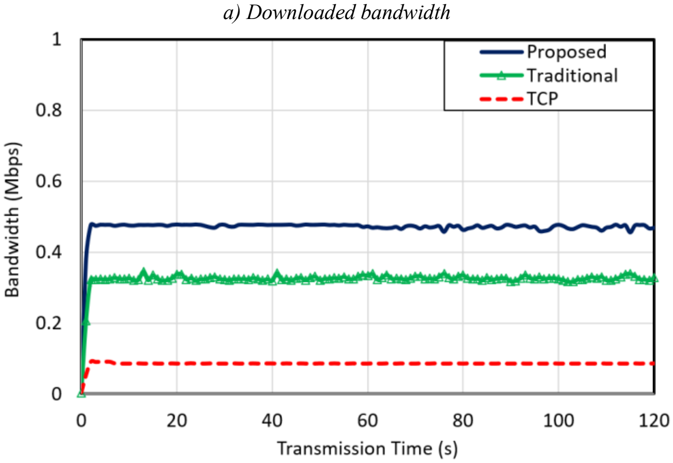
Figure 4. Typical SDN messages for multipath TCP connection

C. *Experimental Results and Discussion*

In this part, we have evaluated the network performance of three comparable SDN network scenarios that are: 1) MPTCP-ware SDN network with our developed routing schemes (so called *Proposed*), 2) an equipvalent MPTCP enable SDN network with traditional *k*-shortest path algorithm (denoted as *Traditional*) and 3) conventional network without MPTCP (then being called *TCP only*) respectively. All three comparable network scenarios are tested under the same network conditions and the same applications including video streaming and large capacity file transferring.



a) Downloaded bandwidth



b) Uploaded bandwidth

Figure 5. Bandwidth comparison

Firstly, we have tested video streaming service with/without MPTCP on our MPTCP-ware software-defined network. Video streaming service is assumed as ultra-high quality with adaptive streaming rates. We have conducted the test 3 times and the average results are then plotted. For each test, we have streamed the video in 120 seconds. Figure 5 illustrates the downloaded and uploaded bandwidths of the video streaming connections for the three comparing cases. The results show that, by using two sub-flows with two separate routes for carrying out the communication session between the client and the server, both MPTCP-enable network scenarios (*Proposed* and *Traditional*) outperform that of conventional TCP network (without MPTCP - *TCP only*) and offer higher bandwidth. It is also verified that our proposed approach provides the best performance, much higher that *Traditional* scenario and as almost twice as that without MPTCP. The reason is that *Traditional* scenario has encontered a bottleneck on the link between OvS6 and OvS9 due to none consideration of the jointed paths while our proposed algorithm can be able to reduce the number of jointed links in the set of selected *k*-shortest paths.

Thank to that, the network utilizing our proposed routing algorithm also has smoother bandwidth graphs because it can avoid traffic congestion on the jointed link too. Figure 6 clearly show the impact of link congestion on the network performance. In the Traditional scenario, two sub-flows conflicted together and caused the bumpy graph lines. It means that, our proposed solution can not only exploit MPTCP but also help to avoid or reduce the bottleneck problem in jointed links of *k*-shortest paths. In Figure 6, the bandwidths of both sub-flows are almost equal thank to the use of load balancing scheme.

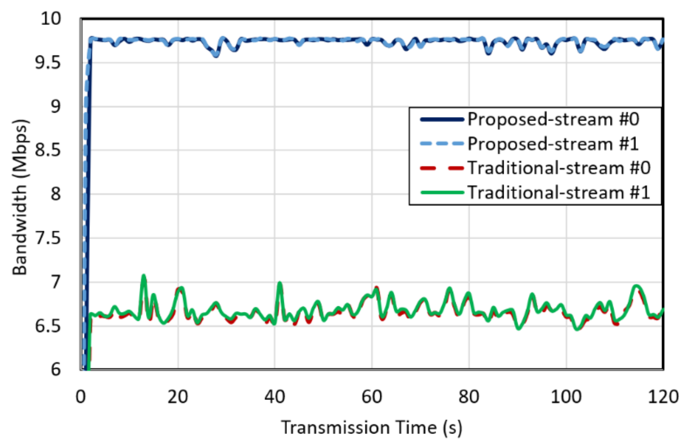


Figure 6. Downloaded bandwidths of each MPTCP sub-flow

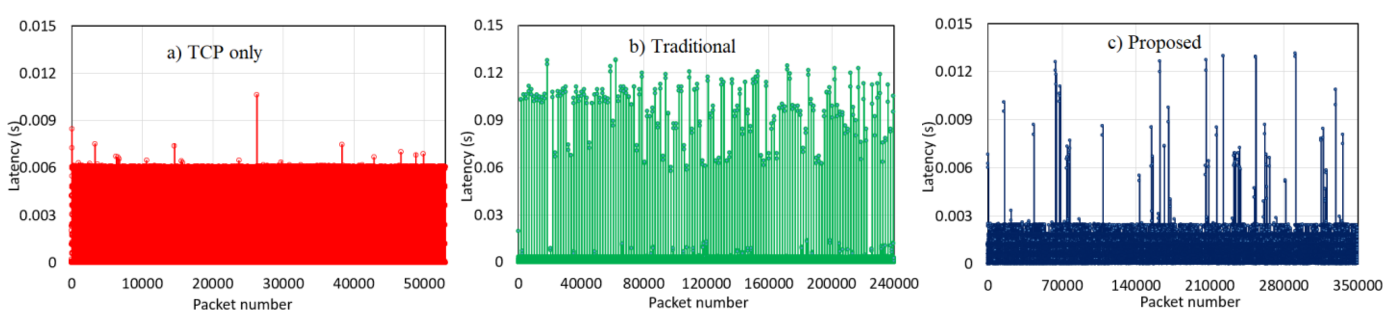


Figure 7. Latency comparison

Moreover, we also evaluated the latency, one of the most important parameters affecting the quality of experience of video streaming services, of the video streaming service connections in the tested software-defined network scenarios. Figure 7 demonstrates the latency comparison of three network scenarios. Actually, here, the results are measured in the ideal environment with the assumption that the transmission delay is ignored, the attained latency of all scenarios is relatively low. The obtained results demonstrated that the Traditional scenario suffers the worst latency due to the great variation of latency when coping with the traffic congestion in the jointed link. Thank to wisely dealing with jointed links while selecting the *k*-shortest paths from the set of path candidates, our proposed approach is capable of reducing the latency. In addition, the latency of MPTCP packets is significantly lower than that of TCP packets because MPTCP tends to divide user datagrams into shorter messages.

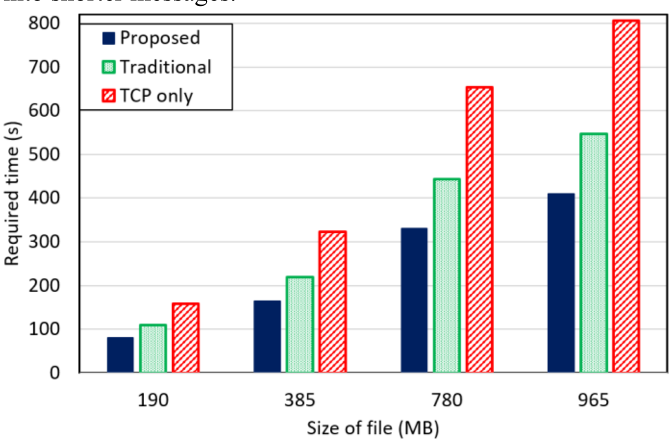


Figure 8. Transferring time comparison

Finally, in order to emphasize the performance of our proposed routing scheme for MPTCP-enable SDN networks, we estimated the transferring time of large capacity video files. We tested with 4 different file sizes and for each file, the transferring time is measured 5 times and the average value is calculated and plotted in Figure 8. One more time, the experimental results confirm the best performance of our developed scheme comparing to others'. It shows that, using MPTCP can help to reduce transferring time greatly and applying efficient routing scheme will help to save more time.

IV. CONCLUSION

In this paper, we have proposed an effective multipath routing algorithm for MPTCP-enable software-defined networks to create bandwidth-abundant and flexible future core networks. Our proposed algorithm is able to find *k*-least-jointed shortest paths by using *k*-shortest path algorithm incorporating with a greedy strategy for the path selection. It can exploit the advantage of MPTCP while being able to reduce the bottleneck problem in the networks. Numerical experiments have been employed to verified the performance of our proposal. It is confirmed that, by reducing the number of jointed links in the set of selected *k*-shortest paths, our solution outperforms the traditional MPTCP approach as well as that of conventional TCP networks.

REFERENCES

[1] J. Wu, C. Yuen, B. Cheng, M. Wang, and J. Chen, "Streaming High-Quality Mobile Video with Multipath TCP in Heterogeneous Wireless Networks," *IEEE Trans. Mob. Comput.*, vol. 15, no. 9, pp. 2345–2361, 2016, doi: 10.1109/TMC.2015.2497238.

[2] A. A. Barakabitze, I. H. Mkwawa, L. Sun, and E. Ifeachor, "QualitySDN: Improving Video Quality using MPTCP and Segment Routing in SDN/NFV," *2018 4th IEEE Conf. Netw. Softwarization Work. NetSoft 2018*, pp. 10–18, 2018, doi: 10.1109/NETSOFT.2018.8459917.

[3] "Kali OS." https://www.kali.org/.

[4] A. A. Barakabitze, L. Sun, I. H. Mkwawa, and E. Ifeachor, "A Novel QoE-Centric SDN-Based Multipath Routing Approach for Multimedia Services over 5G Networks," *IEEE Int. Conf. Commun.*, vol. 2018-May, pp. 1–7, 2018, doi: 10.1109/ICC.2018.8422617.

[5] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "RFC 6182, Architectural Guidelines for Multipath TCP Development," *IETF Secr. RFC 6182*, no. 6182, pp. 1–29, 2011.

[6] "MultiPath TCP - Linux Kernel implementation : Main - Home Page browse," *Www.Multipath-Tcp.Org*. 2020.

[7] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, 2015, doi: 10.1109/JPROC.2014.2371999.

[8] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "OpenFlow: Enabling Innovation in Campus NetworksSoftware-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[9] P. Rezende, S. Kianpisheh, R. Glitho, and E. Madeira, "An SDN-Based Framework for Routing Multi-Streams Transport Traffic over Multipath Networks," *IEEE Int. Conf. Commun.*, vol. 2019-May, pp. 1–6, 2019, doi: 10.1109/ICC.2019.8762061.

[10] K. D. Joshi and K. Kataoka, "SFO: SubFlow Optimizer for MPTCP in SDN," *26th Int. Telecommun. Networks Appl. Conf. ITNAC 2016*, pp. 173–178, 2017, doi: 10.1109/ATNAC.2016.7878804.

[11] H. Nam, D. Calin, and H. Schulzrinne, "Towards dynamic MPTCP Path control using SDN," *IEEE NETSOFT 2016 - 2016 IEEE NetSoft Conf. Work. Software-Defined Infrastruct. Networks, Clouds, IoT Serv.*, pp. 286–294, 2016, doi: 10.1109/NETSOFT.2016.7502424.

[12] N. Kukreja, G. Maier, R. Alvizu, and A. Pattavina, "SDN based automated testbed for evaluating multipath TCP," *2016 IEEE Int. Conf. Commun. Work. ICC 2016*, pp. 718–723, 2016, doi: 10.1109/ICCW.2016.7503872.

[13] S. Zannettou, M. Sirivianos, and F. Papadopoulos, "Exploiting path diversity in datacenters using MPTCP-aware SDN," *Proc. - IEEE Symp. Comput. Commun.*, vol. 2016-Augus, pp. 539–546, 2016, doi: 10.1109/ISCC.2016.7543794.

[14] A. Hussein, I. H. Elhajj, A. Chehab, and A. Kayssi, "SDN for MPTCP: An enhanced architecture for large data transfers in datacenters," *IEEE Int. Conf. Commun.*, 2017, doi: 10.1109/ICC.2017.7996653.

[15] "Mininet website." http://mininet.org/.

[16] "Ryu controller." https://ryu-sdn.org/.