ORIGINAL RESEARCH ARTICLE UHD JOURNAL OF SCIENCE AND TECHNOLOGY

Adaptive Software-defined Network Controller for Multipath Routing based on Reduction of Time

Hemin Kamal[1,2], **Miran Taha**[2,3]

[1]*University of Human Development, College of Science and Technology, Department of Computer Science, Sulaymaniyah, Iraq,* [2]*College of Science, Department of computer, University of Sulaimani, Sulaymaniyah, Iraq,* [3]*Integrated Management Coastal Research Institute, Universitat Politécnica de Valencia, Valencia, Spain*

ABSTRACT

Software-defined network (SDN) is a new paradigm in the networking that makes a programmability and intelligence the networks. The main SDN characterize is separating network management (control plane) from the forwarding device (data plane). SDN logically centralizes the network with the programmable controller which collects global knowledge about the network. The SDNs can improve the performance of the routing packets in the networks because of agility and the ability to create a policy for a driven network. In the multipath routing, the SDNs controller is responsible to calculate the routes of optimum path and alternative path wherever a link is failed. However, a high delay time calculation of selecting optimum and alternative paths in multipath routing by the SDN controller is observed in the recent investigations. In this paper, we propose an efficient algorithm for SDN multipath routing controller. The mechanism of the proposed approach calculates the best path from the source to the destination which is based on using adaptive packet size and observing network link capacity. The proposed algorithm considers reducing delay time of the link handling when the flow traffic switches from the main path to the recovery path. As a result, this approach is compared to some state of the arts according to the delay time of choosing the best path and alternative paths in a given network topology. SDN based on the proposed algorithm consumed approximately 1 ms for selecting recovery routes. On the other hand, the proposed algorithm can be integrated to an SDN controller which provides better consolidation of transmission for sensitive applications as video streaming.

Index Terms: Adaptive algorithm, Multipath routing, Mininet, Software-defined network, Simulation

1. INTRODUCTION

The universal connectivity network that we use today is built on different network topology such as the Internet of Things, data center network, and mobile caller network provides pervasive global coverage on a scale [1]. In addition, the multipurpose network which uses different types of the data video stream, Voice over Internet Protocol, User Datagram Protocol, and Transmission Control Protocols flow of data is significantly more cost efficient than specialized or dedicated network solutions [2]. There is an amount of data through the network with the very difficult process of routing which is hugely important in networking. Data should be transferred continuously without any interruption which is one of the most performed functions of the network [3].

The modern network usually suffers link failures and topology changes. Link failure is a common phenomenon in computer networks that may occur anytime due to network configuration change, network device error, power outage, or many other reasons and results in the disruption of service distribution. This disruption results in unimaginable losses in critical networks [4]. Multipath routing protocols could be addressing

Access this article online

DOI:10.21928/uhdjst.v4n2y2020.pp107-116 | E-ISSN: 2521-4217 P-ISSN: 2521-4209

Corresponding author's e-mail: Hemin Kamal, University of Human Development, College of Science and Technology, Department of Computer Science, Sulaymaniyah, Iraq; College of Science, Department of computer, University of Sulaimani, Sulaymaniyah, Iraq. E-mail: hemin.kakahama@uhd.edu.iq

Received: 05-09-2020 Accepted: 05-11-2020 Published: 15-11-2020

this problem. Thus, efficient algorithm should be presented to handle link failures to keep the flow of the interrupted networks. For handling link failure, network systems keep redundant links for each path so that in case of a link failure, the network systems may calculate a new backup path and keep the system uninterrupted. Keeping redundant links may create loops in computer networks [5]. On the one hand, the most problem of multipath routing is selecting the optimal path on the network and divide the traffic amongst the paths [6]. On the other hand, to overcome this limitation of traditional networks in handling link failure of critical networks, one possible solution can be software-defined network (SDN) which is a new paradigm in computer networking. SDN makes computer networks programmable and more manageable by separating the control and forwarding components of the network where the centralized SDN controller programs and manages the network devices (e.g., switches and routers) by secured communication channel [7]. The controller detects the link failure by receiving messages from the data plane and it can handle link failure both proactively and reactively [8]. In the reactive approach, the SDN controller provides alternative paths on-demand in the SDN switches which is time consuming. In the proactive approach, the controller installs redundant paths in the switches initially and the switches handle link failures locally by directing the flow to the alternative path [9].

In this paper, we propose an algorithm for SDN controller which based on using adaptive packet to calculate multipath routing. The algorithm chooses the optimal path from some available paths on the network. The parameters of the proposed algorithm include the shortest/longest path, link capacity (bandwidth), and number of the hops. Whenever flow of main path is down, the controller switched over the traffic from main to backup path in a short period of time.

The structure of this paper is organized as follows. We review some related works in section 2 while section 3 presents an introduction to the SDN architecture and comparing with the traditional network architecture. Therefore, section 4 explains the problem statements. Section 5 presents the proposed algorithm for multipath routing based SDN. Section 6 explains the experiments and the test results. Finally, the conclusion and future works are given in section 7.

2. RELATED WORK

Delay calculating routes in the SDN controller are one of the main issues. There are numerous articles that investigated how the SDN controllers were decided to select the optimum path and alternative paths. However, the issue investigated in both academic and industrial areas. In this section, we review some related works in detail to explain the existing gaps regarding the delay time of calculating paths by the SDN controllers.

Sharma *et al*., 2011 [10], detected the decision time problem for multipath routing which are hard timeout and idle timeout while the links failed or topology changed. They proposed the fast link restoration mechanism on multipath routing architecture. The proposed mechanism based on the new Address Resolution Protocol (ARP) request and the controller able to restore the path to an alternative path within 12 ms regardless of any timeout of traffic flow. This mechanism decentralized because based on the ARP request on the local switch.

Sgambelluri *et al*., 2013 [11], determined the link failures in the SDN, designed the mechanism to enhance the OpenFlow architecture to protect the segment on the network. The novel mechanism provides an effective network by providing the different priorities of resource utilization for main path and backup path based on OpenFlow segment protection. The presented mechanism implemented the OSP scheme enabling fast recovery in networks composed of OpenFlow-based Ethernet switches. The OSP scheme does not require a full-state controller and, on failure, it does not involve the controller in the recovery process. The implementation results showed that the controller can decide to active the backup path on around 64 ms.

Dorsch *et al*., 2016 [12], outstanding the technical issue in link recovery time in dynamic topology. They suggested the technique for improving fault tolerant to link failure based on the SDN controller. The controller use a centralized approach for both failure detection and traffic recovery bidirectional forwarding detection and OpenFlow allow to saving the delay time on switching the connection. Bidirectional forwarding detection protocol for local link failure is detected between two switches, the drawback of this technique is the negotiation module needs to spend a time period in notifying the failure detection result, the proposed mechanism needs 4.5 ms to change the path to another path.

Lin *et al*., 2016 [13], detected the link failure and link conduction problem in datacenter network link congestion which occurs when a link is carrying too much data. A typical effect is packet loss which caused an actual reduction in network throughput. They proposed the mechanism for link handling for SDN enabled data center network for the traffic recovery. The SDN controller calculates the main and backup path based on a link weighting function. The experiment results showed the recovery time <100 ms. The

proposed mechanism designed for data center network and they did not address the scalability issue.

Dloryhu *et al.*, 2016 [14], focused on controller response time under normal consideration and presence the outage, which is generally impact by network latency. They presented the mechanism to deal with link fail handling issues. Depend on this mechanism, controller pre-establishes the multipath routing from the source to destination. When the current path downed on the SDN, the controller can switchover the traffic to another active path averagely in 40 ms.

Aldwyan and Sinnott, 2019 [15], investigated on the delay issue on multipath routing mechanism which raised a problem to sensitive application. They introduced the new approaches to improve the responsibility of the applications to provide latency aware of the failed link. They proposed an algorithm for path selection in the datacenter based on the SDN. Their work utilizes container technologies and microservice-based application architecture. The approach autonomously generates latency aware failover capabilities by providing deployment plans for micro services and their redundant placement across multiple datacenter networks, with the goal of minimizing the amount switchover time as per implementation result the link handling time reduced to 40 ms.

Hsieh and Wang, 2019 [16], determined the timeout issue for SDN multipath routing when the main link failed and the packet traffic not arrive to destination on time. The authors proposed an efficient mechanism to handle link failure detection and recovery. They explained an efficient mechanism to handle link failure detection and recovery. The proposed method based on a multicontroller in SDN architecture. The topology contains local and global controller to decide to find the best path by calculating the link cost weight based on switch controller propagation delay and load standard division. The experiment results in the implemented mechanism can handle the failed link only 7 ms of delay time.

Therefore, the proposed algorithm for the SDN controller is different from the aforementioned approaches, which uses adaptive packet size to select recovery routes in a given network topology further it reduces that the delay time is taken to select the paths compared to those recent works.

3. STATEMENT OF THE PROBLEM

Routing algorithms and protocols are one of the most important processes of selecting routes in static and dynamic networks. Although internet services utilized by end users, the users are suffered due to inefficient designation of routing algorithms and the quality of service parameters such as bandwidth, delay, packet loss, and jitter. Further, SDN controllers can be programmed to control the packets flows and this programmable approach used to provide multipath routing, however, the controller decided to specify a path to integrate the transmission. The decision process of multipath routing takes time, which called delay time, this time can be dynamic when the controller selects main path and recovery path for the routing packets. However, the problem becomes complex when the controller used insufficiently designed algorithms; thus, the delay time to retrieve the paths to end user takes longer time and end users unsatisfied with the service. For instance, one of the issues is calculating link capacity based on packet size and number of the packets, however, insufficient packets also introduced inaccurate path selection. It is important to take into consideration the number of parameters to design an efficient algorithm for the SDN controller. How the adaptive packet can be selected to test the capacity of the links and how long the sniffing of link capacity needed. To answer those questions, we design an adaptive algorithm for the SDN controller which can reduce the delay time of path selection.

4. SDN ARCHITECTURE

In the traditional network, control plane routing protocols have been implemented together with data plane forwarding functions, monolithically, within a router as depicted in Fig. 1. In network communication, the data is transmitted from the source to the destination and vice versa. Generally, forwarding and routing processes are two important functions in the network layer [17]. The local process of the router executes the delivery of arrived packets from the input interface to the output interface. This is a simple hardware-based operation and takes a short period of time which can be finished on a few nanoseconds [18]. Another function is routing which is a network-wide process to provide end-to-end packet delivery from the source to destination. Routing process is more complex than forwarding and it takes a long time typically about the seconds, as well it implemented in software [19]. The routing algorithm calculates the paths from source to destination and it finds the best path to integrate the routing. Conventionally, a routing algorithm runs in each and every router, and both forwarding and routing functions. The routing algorithm function in one router communicates with the routing algorithm function in other routers to compute the values for its forwarding table. The forwarding table

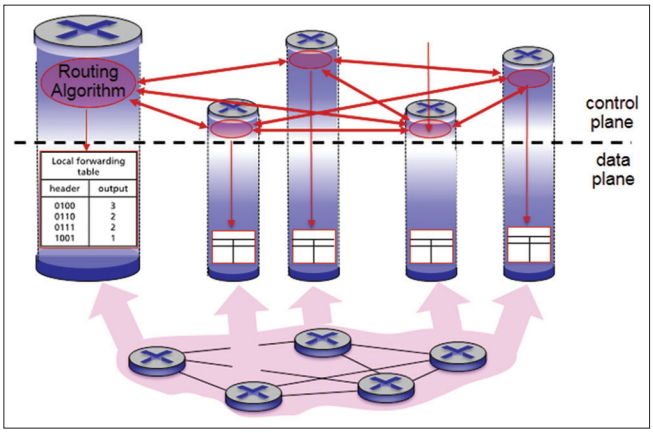

Fig. 1. Forwarding table in traditional networks.

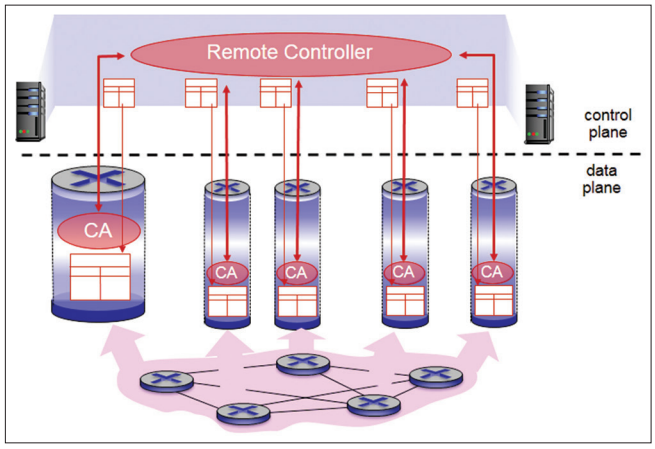

Fig. 2. Forwarding table in software-defined networks.

contains the values of the outgoing link interface to decide which packet should be forwarded [20]. SDN makes a clear separation between the data and control planes, implementing control plane functions in a separate "controller" service that is distinct, and remote, from the forwarding components of the routers it controls. The routing device performs forwarding only, while the remote controller computes and distributes forwarding tables [21]. The routers and the remote controller communicate by exchanging messages containing forwarding tables and other pieces of routing information. The control plane approach is at the heart of SDN, as shown in Fig. 2, where the network is "software defined" because the controller that computes forwarding tables and interacts with routers is implemented in software [22].

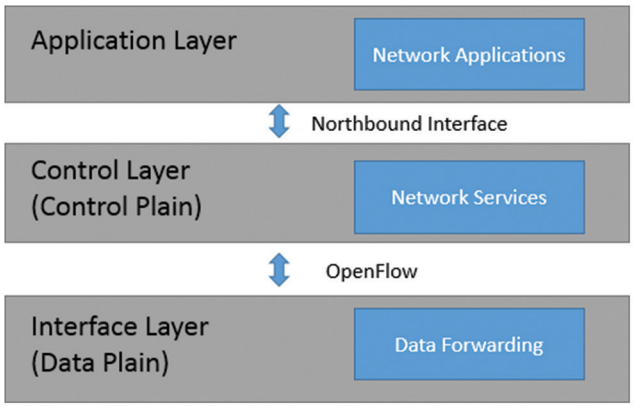

Fig. 3. Architecture of software-defined networks

The data plane also is a part of a network its functionality only forwards the network packets instructed by the controller. This data plane cannot take any forwarding decision [23]. The communication between the data plane and the control plane is performed by OpenFlow messages. The secured channel is used to connect the switches to the controller [24]. Fig.3 illustrates the general description of the SDN layered architecture. The architecture generally compromised three different layers, the button layer is the infrastructure layer that is where the network forwarding equipment, it relies on new layer the control layer to provide it with its configuration and its forwarding instructions, the middle layer or control layer is responsible for configuring that infrastructure layer it does that by receiving service requests from the third layer (the application layer), the control layer maps the service requests on to the infrastructure layer in the most optimal manner possible dynamically configuring that infrastructure layer, the third layer where cloud (internet) applications or management applications place their demands for the network on to the control layer. In SDN, each of these layers and the application program level interfaces between them is designed to be open and provides agility by logically centralizing the full configuration of the networks.

5. PROPOSED ALGORITHM

In this section, the description of the proposed algorithm, select the main and the backup multipaths, and choosing adaptive packet size to reduce delay time of the multipath routing, will explain in details. The decision of path selection in multipath routing is one of the most important parts for the SDN controller, the application layer multipath routing algorithm should be designed in SDN to provide adequate path selection with minimum cost. To provide best path selection, we propose an algorithm which uses important metrics to calculate the paths. The proposed mechanism for multipath routing describes the simple method to select a path set for multipath OpenFlow controller (OFC), which calculates the best main and backup path from source to

destination, it is based on topology discovery information from the controller, as shown in Fig. 1.

The SDN controller's algorithm chooses one best path as the main path, although two important metrics are considered to select the main path, respectively, included; availability of the links bandwidth and the paths length (number of hops). The mechanism calculates the link by sending series of adaptive packets with minimum cost. The controller's algorithm sets the main and backup path in the corresponding table of the individual OpenFlow switches. If there is no failure during the transfer, the incoming packet uses the main path to transfer flow to the destination. When the main path is failed, the OFC obtains notification from the OpenFlow switch of the failure and the controller automatically switches the flow to the backup path to transfer the packet to the destination. The description of proposed system algorithm is shown in Fig. 4.

To manipulate and develop multipath mechanism that can reduce network failure, the configured backup path traces the existence of the main path record in the flow table. Under normal conditions, the backup path stays in standby mode. When there is no any failure or any change of network circumstance, the timeout of selected backup path is automatically expired, and as soon as the recording is forcibly removed from the flow table, the backup path will be inactive mode and continue to run traffic, which guarantees the minimum network failed time.

The mechanism implemented using the following algorithm. The proposed algorithm used depth first search to achieve the feature of finding the multipath between two nodes.

After finding all the paths between source and destination, depth first search path finding algorithm explores possible vertices in a graph by finding the deepest vertex in the graph

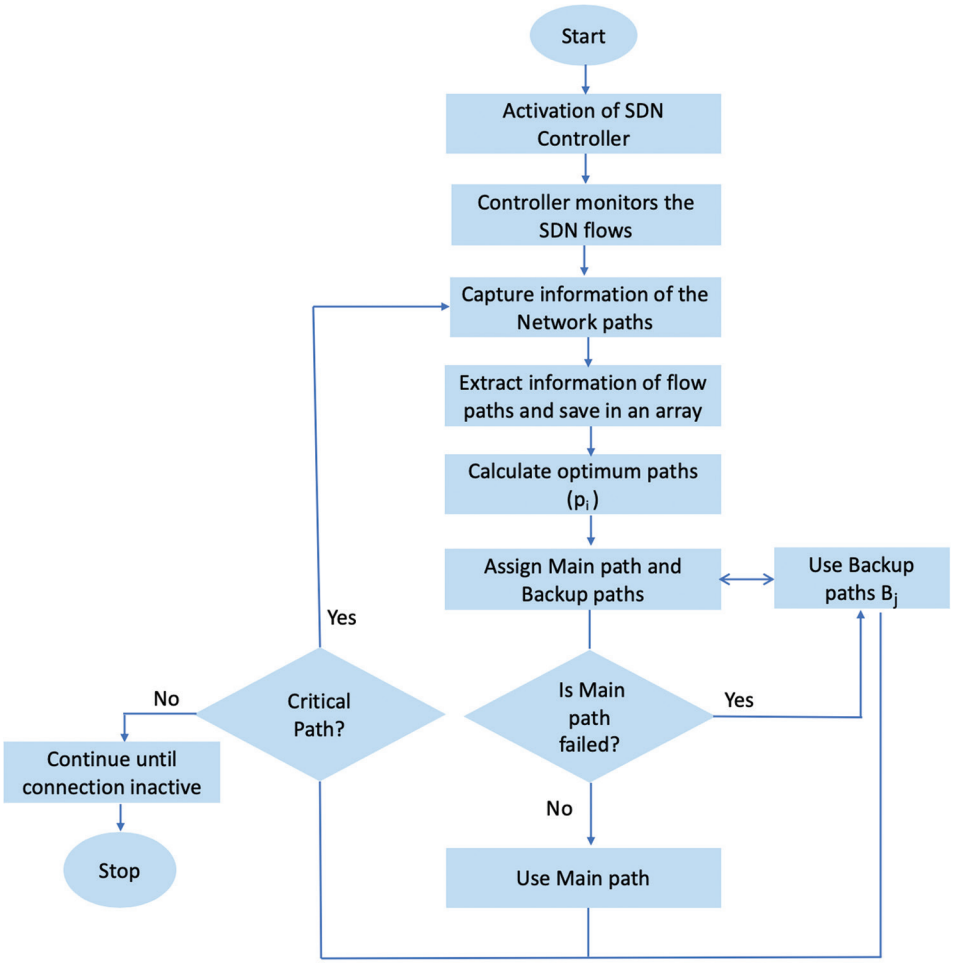

Fig. 4. Description of proposed algorithm function.

first before it backtracks to find other possible vertices using a stack also evaluate those paths to find the best path for the transmission. This part is rather simple first calculate the link cost between two nodes, logically a packet might be having more delay in passing throw lower link capacity than higher link. Respectively, it will take less time in crossing a higher bandwidth link than a lower bandwidth link. The proposed algorithm uses this logic to calculate the cost. Higher link capacity has a lower cost. Lower link capacity has a higher cost. Proposed algorithm uses this formula to calculate the cost as explained in Equation (1).

$$Cost = \left(\frac{Refereces\ link\ capcity}{Interface\ link\ capacity\ in\ bps}\right) \quad (1)$$

Reference (link capacity) bandwidth was defined as arbitrary value in SDN which define as Fast Ethernet uses 100 Mbps (10^8) bandwidth as reference bandwidth. With this bandwidth, our equation would be (Cost = 10^8/interface bandwidth in bps). For instance, if the link capacity is 10 Mbps, the link cost is (10) and think capacity is 100 Mbps, the link cost equal 1, then adds the link cost of the path to get the cost and get the optimal according to the cost of the path. Considering the OSPF, calculating the interface cost is an indication to compute maximum number of the packets across a certain interface. The cost of an interface is inversely proportional to the bandwidth of that interface. A higher bandwidth indicates a lower cost. There is more overhead (higher cost) and time delays. After that, the controller defined property of packets for every switch to execute specific actions such as forward or drop the packets. After getting all available paths, controller adds them into the SDN switch. The state of the selection path algorithm configuration is external to OpenFlow. The proposed algorithm based on bucket weights which provide equal load sharing. When a specified port or link goes down instead of the packet dropping, controller decides to switchover the traffic to backup path. This process reduces the disconnection time of a downed link or switch. Equation (2) explains the weight calculation for the proposed algorithm.

$$weight = \left(\left(1 - \frac{pw}{\sum pw}\right) * 10\right) \quad (2)$$

Where *Weight* represents the bucket weight and *pw* is cost of path. Furthermore, detail of the Pseudo code is explained in Algorithm 1. The link failure handling with SDN is an emerging concept, different fast failover mechanisms have been proposed as explained in the state of the art. We propose

ALGORITHM 1: Multipath routing algorithm in SDN

```
Input: Host x, y = h1,h2,h3,...h_n, Data transfer between Host x and
Host y
Output:
1: SDN_C → SDN Controller
2: N → Switch Number (SrcSwitch, DesSwitch)
3:Cost → 0
4:Datapath →Path[]
5: While SDN_C do:
6:For j in N do
7: j.setFirstSwitch (SrcSwitch)
8: While (nextSwitch! = DesSwitch)
9:For i in range(len(path) - 1):
10:Cost += get_link_cost(path[i], path[i+1])//calculate link cost
11:Return cost
12:End for
13: nextSwitch= Random(N)
14: If (nextSwitch in datapath)
15: nextSwitch = Random(N)
16: Else
17: j.append(nextSwitch)
18: End if
19:End while
22:Path. append(j)
23:End for
24: OptPath = getLestCost(datapath) //Evaluate the link cost and
    choose best path
26: End while
27:ReturnOptPath
```

adaptive failover mechanism for the controller of SDN to make it more efficient and applicable when a delay of few milliseconds can be challenged. The proposed mechanism for handling path failures and restoration involves the controller calculates all available paths from the source to the destination and installs the flow entries on the switches. The controller contains information about the complete topology. In case of any path or link failure, a switch informs the controller. When the controller is notified about a path or link change, controller checks each calculated path P whether it is affected by the link change or not. If any path is affected, the transition rate adaptively decreases by the controller, further, the controller changes the current link to the backup link considering the minimum rate. It also checks that if the flow adds to the entries in OpenFlow switches for older path. If so, then, the controller removes all flow entries from OpenFlow switches for older path and adds in the switches for the new path. The connection of hosts might be recovered in short time and controller overloading also reduced.

6.EXPERIMENTS AND TEST RESULTS

6.1.Testbed Description

We used different tools for conducting our experiments, creating the experimental network, implementing multipath

routing mechanisms, observing the network's behavior, and calculating the delay. We install the SDN environment using virtualization install Ubuntu 18.04 which is one of newest Ubuntu version to implement SDN scenarios [25]. Mininet has been used for implementing the research in SDN and OpenFlow [26]. We can run unmodified code on virtual hardware on a simple PC using Mininet to simulate the SDN and make a centralized network with centralized controller which can use both command line and API [27]. For writing our scripts, we use the API provided by Ryu SDN controller for multipath routing. Furthermore, ping tool is used to observe the path of the packets. The ping is common and quick tool to measurement the latency and how long it takes one packet to get from source to destination [28]. This time measured by the local clock in the pinging device, from the sending the request to get reply. The interval time between each packet is 1 ms and the default packet size is 56 byte with 8 byte from the packet header. In addition, we use VLC application to generate the video streaming which is very sensitive traffic through the network [29]. To apply the tests, we use a workstation with these characteristics; HP workstation, processor: 2, 4 GHZ Core i7, Memory: 8 GB, graphics card: Nvidia 1 GB, and Hard disk: 500 GB.

6.2.Experiments

To achieve our goal of implementing the multipath routing and reduce the link handling time in SDN environment using OpenFlow protocol and flow tables, we implement the proposed algorithm as described in Section 5. Therefore, we apply different experiments to evaluate the proposed algorithm, we demonstrate that our system performs using proof of concept tests to manifest the controller which can find the best path from some available path on the network and the controller can handle the link failed in shortest time without any packet lost. Therefore, to show accuracy and efficiency of the proposed algorithm in the test results, we provide a video streaming test, the videos are streamed over VLC application from source to destination using multipath routing. The received video compares to the original one to realize that how link handling is affected on the quality of the videos.

6.2.1. Network topology

The scenario tested for the multipath network topology. We implemented the SDN using python script with the APIs of Mininet and Ryu to create the data plane and the controller. The network topology consists of two hosts' h1 and h2 with the IP address 10.0.0.1 and 10.0.0.2, respectively. The network topology provided six switches such such as s1, s2, s3, s4, s5, s6, s7, and s8. The switches are enabled with OpenFlow version 1.3 and one remote SDN controller c0. There are seven available paths between the hosts. The fast path capacity is10, 20, 30, 40, 50, 60, and 70 Mbps. This simple scenario is used to observe that if the presented solution is working correctly, as well as to observe the effect of the parameter values on the proposed algorithm. In addition, Fig. 5 depicts the general description of the proposed network topology to find and calculate the link handling time.

6.2.2. Multipath routing based on video streaming

In this experiment, we apply video stream test on the implemented SDN topology not only to evaluate network performance of different network services but also to determine the network path which the controller detects the traffic of the links throughout the network. In this test, we use VLC application to provide video streaming from h1 as a video stream server and sending the video stream (broadcast) to the network and h2 receives the video stream as a client. The result of this experiment showed that the link handling time of streamed video traffic from the main path to backup path do not make any effect on the video stream service or frame frozen from client side and the video streaming could be transferred through the network without any delay or interruption. Therefore, in the another test, the video stream is transmitted over the network, the main link is down manually and monitoring the flow traffic to observe that the controller's activity, the controller flows the traffic from the fast path to slow path (compared with each other) without any disruption in the video streaming. Fig. 6 explains the change of the traffic strategy path from the proposed multipath network topology.

6.2.3. Link handling time

In this scenario, we test with ICMP protocol between h1 and h2 with different packet size (32, 64, and 128 bit) (8-bit header in each packet and the remaining are datagram) and time interval (0.5, 1, and 2 ms). The experiment results are stored in text file. In this experiment, the main path is downed in the special timestamp and we compare the time in this time stamp. The result presented in Table 1, in the first test,

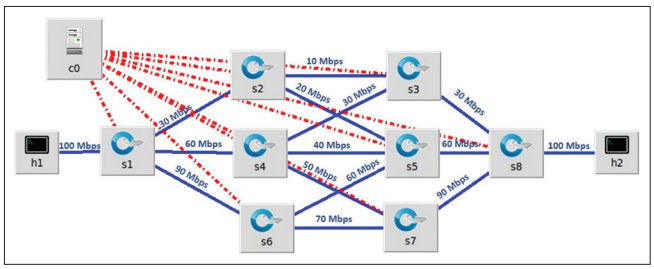

Fig. 5. Multipath network topology.

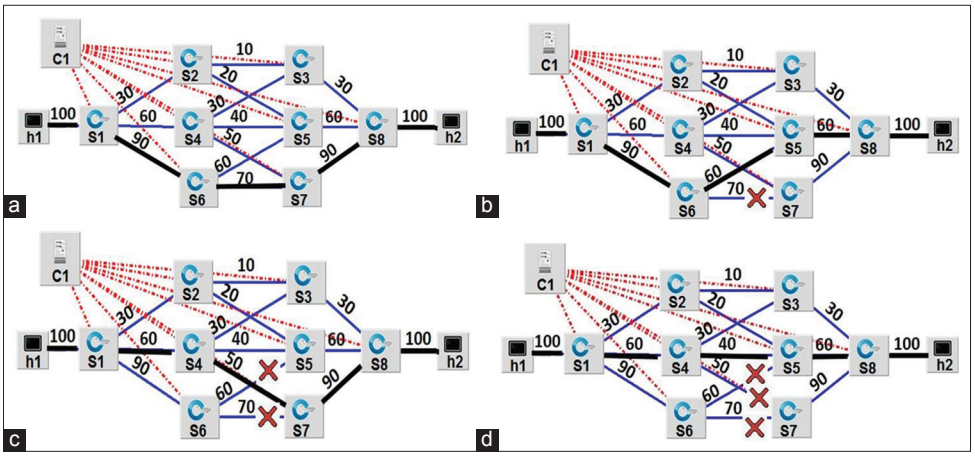

Fig. 6. Path failing and handling for the proposed network topology. (a) Flow over S1-S6-S7-S8, (b) flow over S1-S6-S5-S8, (c) flow over S1-S4-S7-S8, (d) flow over S1-S4-S5-S8.

TABLE 1: Relation between packet size and link handling time

Test No.	Packet size (Byte)	Interval(s)	Number of packets	Link handling time (ms)
1	32	0.5	20	0.82
2	32	1 (Default)	20	0.87
3	32	2	20	1.23
4	64 (Default)	0.5	20	1.14
5	64 (Default)	1 (Default)	20	1.28
6	64 (Default)	2	20	1.11
7	128	0.5	20	1.23
8	128	1 (Default)	20	1.26
9	128	2	20	2

the obtained result is 0.82 ms and I is a minimum delay time of link handling also the packet size for this test was 32 bit and time interval between two packets is 0.5 ms, which is the optimal data transfer for multipath routing.

The experiments results showed that the link handling time according to proposed algorithm approximately 1 ms which the proposed algorithm provides minimum link handling time compare to the papers mentioned on related work. Fig.7 illustrates the comparison the link handling time between proposed and related works algorithms.

6.2.4. Video stream test experiment

In this test experiment, we develop the testbed to be ready for streaming videos. Purpose of this test is to show time of path recovery between the proposed mechanism and traditional mechanism. We use 40 s of BigBuckBunny [30], with x.264 tool [31], the video is encoded to SD quality resolution further average of the streamed video bitrate approximately reaches to 1000 kb/s, the bitrate of the

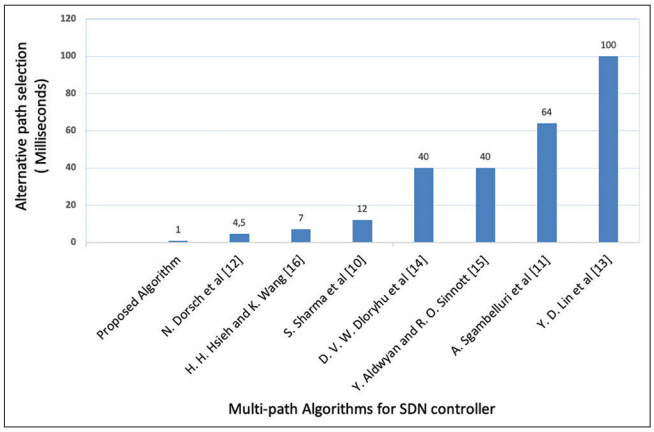

Fig. 7. Comparing the link handling time on proposed and related works algorithm.

video is dynamic which means the bitrate can be less or high than 1000 kb/s that is depended on the size of each i-frame. Therefore, time of path recovery of multipath routing is observed for streaming video between the sender and the receiver. When a path is failed, the recovery paths are used. To save the result experiment, we monitor the network throughput and bitrate of the streamed video, the information is captured in the entrance of receiver side and the information extracted for analyzing. As shown in Fig. 8, when the process initiated, the video is streamed to the receiver (end user), the received video at client side is received through VLC application. In this experiment, the current path is failed at the 10th s of video time, the programmed SDN controller is taken optimum path to integrate the transmission. The process through using proposed algorithm took 3 s, it means that the video is frozen 3 s at the client side also this value can be changed

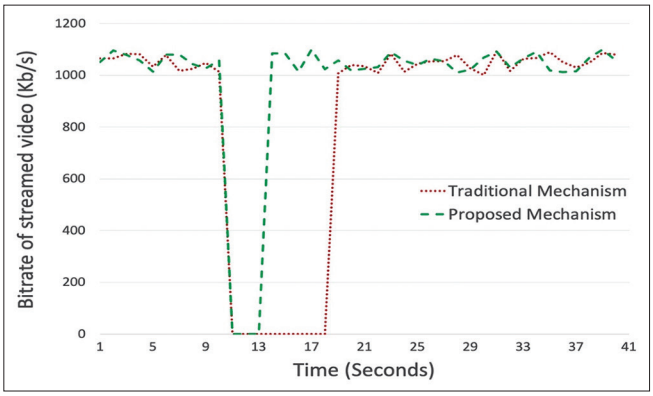

Fig. 8. Comparing path recovery time in proposed and traditional mechanisms.

according to the network conditions, however, in the traditional approach, the process takes 8 s, which means video is frozen for 8 s and the client is waited for a long time until the buffer's client application received entire packets. This test presents that using previous programmable SDN controller (traditional) for multipath routing makes end users unsatisfactory with sensitive applications. On the other hand, the proposed mechanism can provide better result and less time consumption for path recovering.

7. CONCLUSION AND FUTURE WORK

In this paper, we proposed a multipath routing algorithm for the programmable SDN Ryu controller using Mininet network emulator. The proposed algorithm is based on important metrics such as adaptive packet size and observing the network state. The SDN controller based on proposed algorithm decided to switch the flow traffic from the main path to optimal backup path when the main link is down. The proposed algorithm provided the efficient result for selecting routes for critical network infrastructures as well as for any network systems intolerant of delays. As a result, the experimental results showed that the link handling time approximately equal to 1 ms which is the best result presented than other recent researches as mentioned in the related works.

On the other hand, the proposed mechanism also provided better quality of experience for the sensitive applications like video streaming.

In future work, we plan to apply the reinforcement learning approach to the proposed algorithm to deploy an accurate decision of selecting recovery path.

8. ACKNOWLEDGMENTS

This work has done at the University of Sulaimani and Human Development University.

9. DECLARATION OF CONFLICTING INTEREST

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

REFERENCES

[1] M. Amiri, A. Sobhani, H. Al Osman and S. Shirmohammadi. "SDN-enabled game-aware routing for cloud gaming datacenter network". *IEEE Access*, vol. 5, no. c, pp. 18633-18645, 2017.

[2] M. Taha, L. Garcia, J. M. Jimenez and J. Lloret. "SDN-Based Throughput Allocation in Wireless Networks for Heterogeneous Adaptive Video Streaming Applications". In: *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 963-968, 2017.

[3] P. Faizian, M. A. Mollah, Z. Tong, X. Yuan and M. Lang. "A Comparative Study of SDN and Adaptive Routing on Dragonfly Networks". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis 2017*, 2017.

[4] S. N. Hertiana, Hendrawan and A. Kurniawan. "A Joint Approach to Multipath Routing and Rate Adaptation for Congestion Control in Open Flow Software Defined Network". In: *2015 1st International Conference on Wireless and Telematics (ICWT)*, pp. 1-6, 2016.

[5] W. Jiawei, Q. Xiuquan and N. Guoshun. "Dynamic and adaptive multi-path routing algorithm based on software-defined network". *International Journal of Distributed Sensor Networks*, vol. 14, no. 10, pp. 1-10, 2018.

[6] R. Baruah, P. Meher and A. K. Pradhan. *Efficient VLSI Implementation of CORDIC-Based Multiplier Architecture*. Springer, Singapore, 2019.

[7] F. Rhamdani, N. A. Suwastika and M. A. Nugroho. "Equal-cost Multipath Routing in Data Center Network Based on Software Defined Network". In: *2018 6th International Conference on Information and Communication Technology (ICoICT)*, pp. 222-226, 2018.

[8] M. F. Ramdhani, S. N. Hertiana and B. Dirgantara. "Multipath Routing with Load Balancing and Admission Control in Software-Defined Networking (SDN)". Vol. 4. In: *2016 4th International Conference on Information and Communication Technology (ICoICT)*, pp. 4-9, 2016.

[9] R. Wang, S. Mangiante, A. Davy, L. Shi and B. Jennings. "QoS-aware Multipathing in Datacenters using Effective Bandwidth Estimation and SDN". In: *2016 12th International Conference on Network and Service Management (CNSM)*, pp. 342-347, 2017.

[10] S. Sharma, D. Staessens, D. Colle, M. Pickavet and P. Demeester. *Enabling Fast Failure Recovery in OpenFlow Networks*. pp. 164-171, 2011.

[11] A. Sgambelluri, A. Giorgetti, F. Cugini, F. Paolucci and P. Castoldi. "Open flow-based segment protection in Ethernet networks".

Journal of Optical Communications and Networking, vol. 5, no. 9, pp. 1066-1075, 2013.

[12] N. Dorsch, F. Kurtz, F. Girke and C. Wietfeld. Enhanced Fast Failover for Software-Defined Smart Grid Communication Networks". In: *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1-6, 2016.

[13] Y. D. Lin, H. Y. Teng, C. R. Hsu, C. C. Liao and Y. C. Lai. "Fast Failover and Switchover for Link Failures and Congestion in Software Defined Networks". In: *2016 IEEE International Conference on Communications (ICC)*, 2016.

[14] R. H. Hwang and Y. C. Tang. "*Fast Failover Mechanism for SDN-Enabled Data Centers*." International Computer Symposium, Chiayi, Taiwan, pp. 171-6, 2016.

[15] Y. Aldwyan and R. O. Sinnott. "Latency-aware failover strategies for containerized web applications in distributed clouds". *Future Generation Computing Systems*, vol. 101, pp. 1081-1095, 2019.

[16] H. H. Hsieh and K. Wang. "A Simulated Annealing-based Efficient Failover Mechanism for Hierarchical SDN Controllers". In: *IEEE Region 10 Annual International Conference, Proceedings/ TENCON*, pp. 1483-1488, 2019.

[17] F. Y. Okay, S. Ozdemir and M. Demirci. "SDN-Based Data Forwarding in Fog-Enabled Smart Grids". In: *2019 1st Global Power, Energy and Communication Conference (GPECOM)*, pp. 62-67, 2019.

[18] A. Tariq, R. A. Rehman and B. S. Kim. "Forwarding strategies in NDN-based wireless networks: A survey". *IEEE Communications Surveys and Tutorials*, vol. 22, no. 1, pp. 68-95, 2020.

[19] Y. Zhang, Z. Xia, A. Afanasyev and L. Zhang. A note on routing scalability in named data networking." In: *2019 IEEE International Conference on Communications Workshops (ICC Workshops*), pp. 1-6, 2019.

[20] A. Jayaraman. *Comparative Study of Virtual Machine Software Packages with Real Operating System*, 2012.

[21] J. F. Kurose, R. Rose. *Computer Networking a Top-Down Approach.* 7th ed. Pearson, United Kingdom, 2017.

[22] J. Ali, S. Lee and B. H. Roh. "Performance Analysis of POX and Ryu with Different SDN Topologies". *ACM International Conference Proceedings Series*, pp. 244-249, 2018.

[23] Y. Shi, Y. Cao, J. Liu, and N. Kato. *A Cross-Domain SDN Architecture for Multi-Layered Space-Terrestrial Integrated Networks*. Vol. 33. IEEE Network, Piscataway, pp. 29-35, 2019.

[24] M. Alsaeedi, M. M. Mohamad and A. A. Al-Roubaiey. *Toward Adaptive and Scalable Open Flow-SDN Flow Control: A Survey*. Vol. 7. *IEEE Access*, Piscataway, pp. 107346-107379, 2019.

[25] F. Keti and S. Askar. "Emulation of Software Defined Networks Using Mininet in Different Simulation Environments". In: *2015 6th International Conference on Intelligent Systems, Modelling and Simulation*, pp. 205-210, 2015.

[26] S. Lee, J. Ali, and B. H. Roh. "Performance Comparison of Software Defined Networking Simulators for Tactical Network: Mininet vs. OPNET". In: *2019 International Conference on Computing, Networking and Communications (ICNC)*, pp. 197-202, 2019.

[27] C. Fernandez and J. L. Muñoz. *Software Defined Networking (SDN) with Open Flow 1.3, Open v Switch and Ryu*, pp. 183, 2016.

[28] Z. H. Zhang, W. Chu and S. Y. Huang. "The Ping-Pong Tunable Delay Line in a Super-Resilient Delay-Locked Loop". In: *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pp. 90-91, 2019.

[29] M. Taha, J. Lloret, A. Canovas and L. Garcia. "Survey of transportation of adaptive multimedia streaming service in internet". Network Protocols and Algorithms, vol. 9, no. 1-2, pp. 85, 2017.

[30] Available from: https://www.peach.blender.org. [Last accessed 2020 Jun 01].

[31] Available from: https://www.videolan.org/developers/x264.html. [Last accessed on 2020 Jun 20].