Damascus university journal
V 42 (No):2 PP: 137-151

مجلة جامعة دمشق للعلوم الهندسية
المجلد 42 (العدد):2 الصفحات: 137-151

Designing and Evaluating Video-stream Routing Algorithm Based on Load-Balancing and Multipath Open-flow Over Software-Defined Network

Ahmad Zidane*[1] Talal Hammod[2]

*[1]. PhD student, Faculty of Mechanical and Electrical Engineering, Damascus University, Advanced Connections.
ah.zaidan88@Damascusuniversity.edu.sy

[2]. Professor, Mechanical and Electrical Engineering, Damascus University. talal64.hammoud@damascusuniversity.edu.sy

Received: 22/7/2023
Accepted: 26/11/2023

Abstract:

Multipath routing, i.e., routing with many paths, is an alternative to single-path routing, hence tending to fragment the network topology into a tree shape so that video packets take multiple paths through tree paths in the network topology. With multipath routing, it is expected that with many paths, the network load can be distributed in a balanced way to several existing paths or what is known as load-balancing, so as to increase the utility of the network. The implementation of multipath routing can be facilitated by the existence of a Software-Defined Network (SDN) paradigm that allows a centralized view of the network. For this reason, we design in this paper a load-balancing multipath routing system with OpenFlow protocol over SDN by using the Depth-First Search (DFS) algorithm to search for routes and utilizing the group actions feature on Open-v-Switch (OVS) to perform load-balancing. In this paper, we will show how our solution improves Quality of Experience (QOE) and Quality of Service (QoS) depending on (Delay, Throughput and rtt) parameters for high quality video streaming while remaining reactive to network congestion.

Keywords: Video-Streaming, Multipath Routing, Load-Balancing, Open-Flow protocol, DFS Algorithm, Software-Defined Network.

http://journal.damascusuniversity.edu.sy/index.php/index/index
ISSN (online): 2789-6854

تصميم وتقييم خوارزمية توجيه دفق الفيديو على أساس موازنة الحمل والتدفق المتعدد المسارات في الشبكة المعرفة برمجيّاً

أحمد زيدان*[1] طلال حمود[2]

*[1]. طالب دكتوراه، كلية الهندسة الميكانيكية والكهربائية، جامعة دمشق، اتصالات متقدمة.
ah.zaidan88@Damascusuniversity.edu.sy
[2]. أستاذ في كلية الهندسة الميكانيكية والكهربائية، جامعة دمشق
talal64.hammoud@damascusuniversity.edu.sy.

الملخص:

يُعدُّ التوجيه متعدد المسارات بديلاً عن التوجيه أحادي المسار في الشبكة، حيث يجري وفقه تجزئة بُنية الشبكة الكلّية إلى عدد شبكات جزئية تُسمى أشجار البث المتعدد. وفق هذه التجزئة، تأخذُ الرزم الفيديوية مسارات مختلفة تحددها شجرة البث في هيكل الشبكة نفسه. وهكذا، يُصبِحُ توزيع حمل الشبكة بشكل متوازن على عدة مسارات مُتاحة أمراً ممكناً، وهو ما يُعرَف بمصطلح موازنة الحمل التي تهدف إلى استغلال موارد الشبكة بشكل أكبر. مع اتخاذ نموذج الشبكة المُعرَّفة برمجياً، يمكن تسهيل تنفيذ إجرائية التوجيه متعدد المسارات، طالما أنّ هذا النموذج يسمح برؤية مركزية وشاملة للشبكة. لذلك، نقدم في هذه الورقة البحثية نظام توجيه متعدد المسارات لموازنة الحمل مع بروتوكول Open Flow عبر شبكة مُعرَّفة برمجياً باستخدام خوارزمية البحث المُعمَّق الأول (DFS) للبحث عن المسارات المطلوبة لتحقيق أداء موازنة الحمل. نوضح في هذه الورقة البحث كيف يعمل الحل المقترح على تحسين جودة التجربة وجودة الخدمة بالاعتماد على بارامترات محددة من أجل دفق فيديوي عالي الجودة مع الأخذ بعين الاعتبار التفاعل مع حالات الازدحام ضمن الشبكة.

الكلمات المفتاحية: الشبكات المعرّفة برمجيّاً – جودة الخدمة – التوجيه المتعدد المسارات – موازنة الحمل– بروتوكول التدفق المفتوح –خوارزمية البحث المعمّق الأول.

تاريخ الإيداع: 2023/7/22
تاريخ القبول: 2023/11/26

1-INTRODUCTION:

Conventional single-path routing algorithms, such as Dijkstra's procedure used with Open Shortest Path First (OSPF) algorithm, cannot fully utilize all existing paths in the network topology, or what can be called multipath capability. In spanning tree procedures like Dijkstra's, the network topology will always be reduced and truncated to form a tree. This will remove the multipath capability on topologies that have many redundant branches, causing a waste of network resources. On the other hand, with multipath routing, video traffic going to a destination will be divided into several paths for that destination. Therefore, multipath routing is proposed as an alternative to single-path routing in order to fully utilize the multipath capabilities of the network topology.

In addition, multipath routing can reduce network congestion by distributing video traffic to unused network resources, e.g., unused nodes or edges. Hence, in order to distribute video traffic evenly, load-balancing capabilities are required in designing our routing algorithm. Thus, with a multipath routing algorithm that has load-balancing capabilities, the resources of the network can be fully utilized, namely by distributing the load in a balanced manner on the whole network. The problem in building a network that has these capabilities is that the current network architecture is still highly distributed, where complex network devices are connected to each other without centralized control, making it very difficult to modify, customize, or update. Moreover, the capabilities of the network are highly dependent on vendors (such as Cisco) who build their devices without any standards, open interfaces, or open sources procedures, to update or optimize the network.

For this reason, one solution is to create a new network paradigm or concept called Software-Defined Networking (SDN). SDN is a concept that states a network with open standards so that anyone with the ability can innovate, update, or optimize the network. This is done by separating the control plane, namely the intelligence of a network that regulates how a packet is sent on a network, and the data plane, which is a function that carries out packet forwarding or sending activities based on the information provided by the control plane.

With SDN concept, the multipath routing problem can be more easily solved, where SDN can provide a centralized view of the network so that the network topology conditions can be described clearly. One embodiment of the SDN concept that has been widely used today is Open-Flow, which was first proposed by McKeown et al [1] as a standard open communication protocol/interface between control plane and data plane devices.

Therefore, in this paper, we design and implement multipath routing algorithm with load-balancing using OpenFlow protocol over SDN. Open-Flow protocol over SDN here is implemented and simulated in a Mininet virtual environment, namely a network emulator and on the Ryu SDN Controller, that is one of the common SDN Controller used in this research domain.

2- THEORETICAL FRAMEWORK:

2-1: Open-Flow protocol:

Open-Flow is a standard open communication protocol between the data plane and the SDN control plane [2]. In contrast to conventional networks, where the data plane and control plane functions are on one switch/router device, a network installed by Open-Flow allows the data plane and control plane functions to be on separate devices, making it easier to extend or add functions to a network. Switches in Open-flow point of view are only data plane devices that only perform packet forwarding functions based on a set of criteria that selects packets and a set of actions that are applied to the packet. The unit of the set of criteria and their actions is what is referred to as flow. Meanwhile, the control plane device in Open-flow can be a server that provides logic and intelligence from the network or what is known as a Controller or Network Operating System (NOS), e.g., Ryu Controller [3].

2-2- Mininet Emulator:

Mininet is a network emulator that builds a network consisting of virtual hosts, switches, controllers and links [4]. Mininet is a system for rapid prototyping of very large networks on limited resources, such as a laptop [5]. Mininet uses process-based virtualization to run multiple hosts and switches on a single operating system kernel [4]. Mininet utilizes

the Open-v-Switch (OVS) virtual software switch to build an Open-Flow

SDN topology. Mininet will be used in this paper to simulate SDN networks with various topologies in order to evaluate the performance of the proposed multipath routing algorithm.

2-3-Ryu controller as a choice:

Ryu is a component-based framework for building control and network management applications [6]. Application development using Ryu can be done using the python language or by sending JSON messages via the available API in the network. Ryu supports various tools for network management and evaluation including in Open-Flow such as, netconf, ofconfig, iperf… etc. The need for Ryu as an Open-flow controller is because Ryu supports Open-flow versions 1.0 to 1.5, providing group actions that can be used for multipath routing. Ryu is here used as the SDN Controller and as a framework for developing a multipath routing system in Open-Flow SDN.

2-4- Open-v-Switch (OVS):

Open-v-switch (OVS) is a production quality virtual multilayer switch licensed under the open-source Apache 2.0 license [7]. OVS is designed to enable massive network automation by extending programmatically, but also supports several standard management interfaces and protocols such as Open-flow, NetFlow, sFlow, IPFIX, RSPAN, CLI, LACP, 802.1ag [7]. OVS version 2.5.1 and above has supported OpenFlow up to version 1.5 so that it provides a group-actions feature to perform load-balancing techniques on switches of the SDN.

3- RELATED WORKS:

Several studies have been conducted to produce a multipath routing protocol with load balancing on SDN [8]. Lei et al [9] performed multipath routing on an SDN-based data center network by developing an adaptive worst-fit multipath routing (AWMR) algorithm that determines several paths based on available bandwidth and required bandwidth of a flow. Ramdhani et al [10] perform multipath routing using the DFS (Depth First Search) algorithm to search for multiple paths and load-balancing based on switch load, but load-balancing is still carried out at the SDN controller so that one match flow on the switch still passes one output only and cannot be said to be multipath. A similar thing is done by Naqvi et al [11] where multipath routing for unicast traffic is carried out load-balancing based on the TCP/UDP port of a flow. In our paper, load-balancing for multipath routing is carried out at the switch level, so that video streaming flow of a host can pass several paths in the network topology.

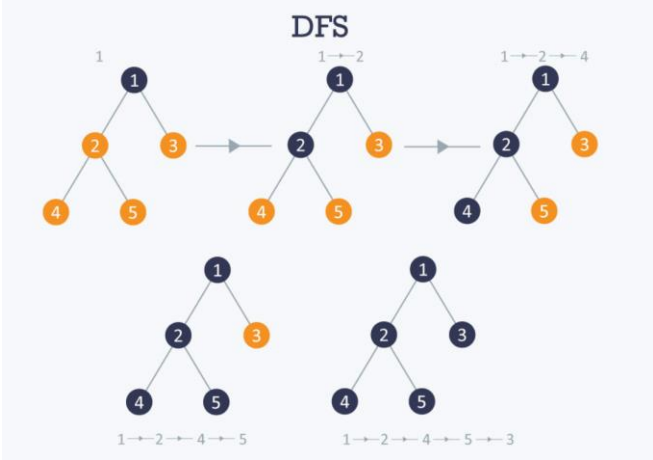
4- DESIGN METHODOLOGY:

4-1-Path Search:

In this paper, the DFS algorithm is modified to search the path. The DFS algorithm is a path-finding method that is carried out using a stack data structure that can store routes, backtracking, then can expand again even though one path has been found to a destination, so that it can search for all paths that can be passed to a certain node. In other words, the DFS algorithm can be modified to search the entire path from one node to another which can be described as pseudocode in Figure 1.A, and Figure 1.B

```
DFS-iterative (G, s):   //Where G is graph and s is source vertex
  let S be stack
  S.push( s )            //Inserting s in stack
  mark s as visited.
  while ( S is not empty):
      //Pop a vertex from stack to visit next
      v  =  S.top( )
      S.pop( )
      //Push all the neighbours of v in stack that are not visited
      for all neighbours w of v in Graph G:
          if w is not visited :
                  S.push( w )
                  mark w as visited
DFS-recursive(G, s):
    mark s as visited
    for all neighbours w of s in Graph G:
        if w is not visited:
            DFS-recursive(G, w)
```

1.A Pseudocode of DFS.



1.B an example how DFS works

Figure (1) DFS Algorithm.

The time complexity of the DFS algorithm is represented in the form "O(V+E)", where (V) is the number of nodes, and (E) is the number of edges. The reason behind this is that we visit each node only once by function (DFS), and every node we

visit we will visit connected nodes, therefore, we will explore every edge within the graph twice. Suppose the edge is between nodes n1 and n2, for the first time, we detect that node n1 is connected to node n2, while in the second time, we detect that node n2 is connected to node n1.

4-2- Metrics of Path Scoring:

Because the DFS algorithm assumes the topology as an unweighted graph, a mechanism is needed to evaluate the paths that have been found. Paths generated by the DFS algorithm are not always optimal, so it is necessary to sort out the suitable paths to be included in our multipath routing scheme.

First, we take a metric for assessing a path. For this reason, calculations for metrics or link costs in the OSPF algorithm; $\mathrm{Cost_{OSPF}}$; can be referred to as in the following equation (1) [1]:

$$\mathrm{Cost_{OSPF}} = \frac{\mathrm{BW_{OSPF}}}{\mathrm{BW_{link}}} \qquad (1)$$

Based on the formula (1), $\mathrm{BW_{OSPF}}$ states the reference bandwidth used by the OFPS protocol, which is for example 100 Mbps, while BW_{link} represents the link bandwidth of a pair of connected routers/switches. However, the formula (1) does not take into account the video traffic load experienced by the switch, so it is less able to adapt to network conditions. However, OpenFlow SDN can overcome this problem by presenting the ability to provide useful statistics on the flow of packets on a switch. Research conducted by [12,13] has taken advantage of this capability by proposing an extension to Dijkstra's algorithm by adding node weight and edge weight assessment metrics to the route search, which are defined in the following equations (2) and (3) respectively:

$$\mathrm{Node}_{weight}(\mathrm{n}) = \frac{\sum_{f \in F(n)} \boldsymbol{Bits(f)}}{\boldsymbol{C(n)}} \quad (2)$$

$$\mathrm{Edge_{weight}}(\mathrm{n}) = \frac{\sum_{\mathrm{f}\in \mathrm{F(e)}} \mathrm{Bits(f)}}{\mathrm{BW(e)}} \quad (3)$$

Here, according to the literature, it is assumed that a graph $G = (N, E)$ which is weighted, directed, and connected can be derived from the SDN topology. For a node $\mathrm{n} \in N$ and an edge $\boldsymbol{e} \in E$, the flow $F(\mathrm{n})$ and $\mathrm{F(e)}$ denote the set of flows passing through n and e, whereas $C(\mathrm{n})$ is the capacity of n (the number of bits the node n can process per second) and $\mathrm{BW}(e)$ is the bandwidth of the edge $\boldsymbol{e}$ (the number of bits that can be transmitted through $\boldsymbol{e}$ per second), whereas $\mathrm{Bits(f)}$ the number of bits contained in the flow f.

However, the weakness of these equations (2) and (3) is that if the video traffic on the network comes from zero, then the resulting cost is also zero so that all paths with zero video traffic will have the same value. For this, it is possible not to take into account the hop count of the topology the and link bandwidth of the path. In addition, in the context of this paper since we are using OVS, the software-based switch, the capacity of the switch is very dependent on the hardware used, thus making it difficult to calculate the capacity of a switch. Therefore, we instead propose a combination of these formulas with the OSPF cost formula, which is as in the following equation (4):

$$\mathrm{Edge_{weight}}(\mathrm{e}) = \frac{\mathrm{BW_{OSPF}}}{\mathrm{BW(e)} - \sum_{\mathrm{f}\in \mathrm{F(e)}} \mathrm{Bits(f)}} \qquad (4)$$

With this equation, the hop count and link bandwidth can be calculated as well as the video traffic load from an edge without taking into account the overall traffic effect for a switch relative to the switch capacity so that heavy video traffic loads from other links of the switch cannot be avoided. From all of this, an evaluation metric of a path can be generated as the total edge weight of a path, which is as in the following equation (5).

$$\boldsymbol{Path}_{weight}(\boldsymbol{p}) = \sum_{\boldsymbol{e}\in E} \boldsymbol{Edge}_{weight}(\boldsymbol{e}) \quad (5)$$

Based on the equation (4), the weight of a path $\boldsymbol{p}$ (path weight) is expressed by $\boldsymbol{Path}_{weight}(\boldsymbol{p})$ where $p(N_0, E_0)$ is a path consisting of some determined set of nodes represented by N_0 and some determined set of edges represented by E_0. The path weights are obtained here by adding up all node weights $\mathrm{Node}_{weight}(\boldsymbol{e})$ and edge weights $\mathrm{Edge_{weight}}(\boldsymbol{e})$ on a path. From that, it can be stated that the best path is the path with the lowest path weight $\mathrm{Path}_{weight}(\mathrm{p})$, i.e.,

The path is best ⇔ The path weight is lowest

4-3- Path Selection:

Based on the definition of the path assessment metric that has been used, it can be calculated some of the best paths from all the paths that have been found. What needs to be considered here is the number of paths selected and the weight of the paths selected for the load-balancing process. First, the entire path, then the paths with the lowest $\mathrm{Path}_{weight}(\mathrm{p})$ values are selected for use in our routing multipath scheme. The number of paths used

by the weight $\boldsymbol{w}$ is calculated to find the percentage of the $\text{Path}_{\text{weight}}$ of each path $\boldsymbol{p}$ with the total Path_{weight} of the paths, which can be shown in the following equation (6):

$$W(p) = \left(1 - \frac{\text{Path}_{\text{weight}}(p)}{\sum_{p \in \text{Set of all paths found}} \text{Path}_{\text{weight}}(p)}\right) \quad (6)$$

Basically, $\frac{\text{Path}_{\text{weight}}(p)}{\sum_{p \in \text{Set of all paths found}} \text{Path}_{\text{weight}}(p)}$ finds the ratio of the path weight of p with the total path weight of the available paths. To put it simply, when it comes to path weight or cost where ideally, we want to use the shortest path first, then lower is better. While in the context of buckets in OpenFlow Group tables, the priority of choosing a bucket is with the highest bucket weight, hence higher is better. By using the formula (6), we can expect that the lower the path weight, then the higher the bucket weight W(p).

To put all in a clearer direction, we present here a numerical example with a simple topology, as following. Let's consider the topology showed in FIGURE 2.

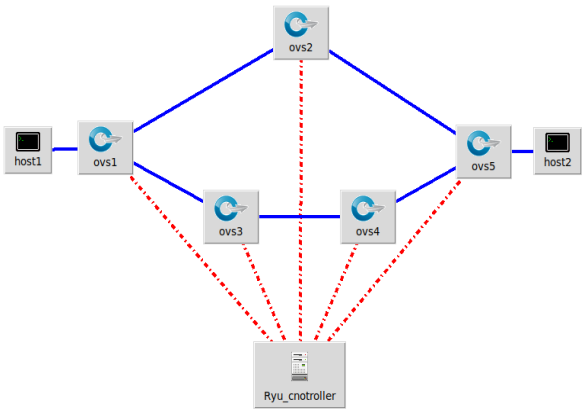


FIGURE (2)TOPOLOGY EXAMPLE TO CLARIFYING BUCKET WEIGHT CALCULATION

we can calculate (by using the equation (5)) the path weight for the two paths available, assuming every link/edge is assigned a weight of 1 (This means that: $\text{Edge}_{\text{weight}}(e_{ij}) = \text{Node}_{\text{weight}}(\text{ovsi}) \rightarrow \text{Node}_{\text{weight}}(\text{ovsj}) = 1,$ where $i, j \in \{1,2,3,4,5\}$with $i \neq j$):

$\text{Path}_{\text{weight}}1 = \text{Edge}_{\text{weight}}(e_{12}) + \text{Edge}_{\text{weight}}(e_{25}) = 1 + 1 = 2$

$\text{Path}_{\text{weight}}2 = \text{Edge}_{\text{weight}}(e_{13}) + \text{Edge}_{\text{weight}}(e_{34}) + \text{Edge}_{\text{weight}}(e_{45}) = 1 + 1 + 1 = 3$

$W(p1) = (1 - \frac{2}{2+3}) = 0.6$

$W(p2) = (1 - \frac{3}{2+3}) = 0.4$

So as expected, the shorter path (lower Path_{weight}), is assigned a higher bucket weight w(p).

4-4- Introducing Load-Balancing:

For our load-balancing process, packet forwarding decisions are made by the switch without having to consult the SDN controller. This is necessary because if the switch has to consult directly with the SDN controller, then the packet forwarding process will take a very long time and will greatly impact the packet delivery throughput. Therefore, a load-balancing mechanism is needed that is fast but still pays attention to the weight of the shipping line.

For load-balancing, we make use of the load-balancing mechanism provided by open source-codes of the OVS, which implements group tables and bucket actions from the OpenFlow version 1.3 (see specification [14]). In the version of OVS version 2.5.1 used in this paper, load-balancing is done by hashing the layer 2 to layer 4 headers of the video packet such as the MAC address, IP address, and TCP port, which then the hashing result is an integer multiplied by the weight of the path that has been set. From that, the path with the highest bucket result is taken between the hash multiplication and the weight.

5: Simulation Results and Evaluation

5-1: Topologies for Path-finding Scenarios

Path-finding experiments now are carried out on several topologies with dependent paths to assess the success of the multipath routing that are implemented here to be evaluated. There are several topological scenarios that are used, including having paths that have the same cost and paths with different costs, as shown in Figures 3.A, 3.B, 3.C, 3.D. Each topology consists of 2 hosts, h1 and h2, each of which is at the end of its multipath topology. All of these topologies use switches with uniform link bandwidth, which is 100 Mbps, and a reference bandwidth of 1 Gbps for cost calculations.

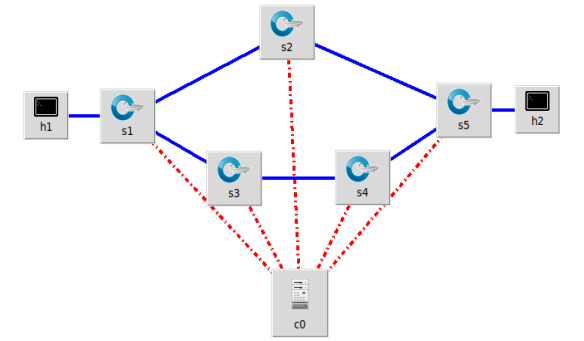


FIGURE (3.A) - TOPOLOGY 1

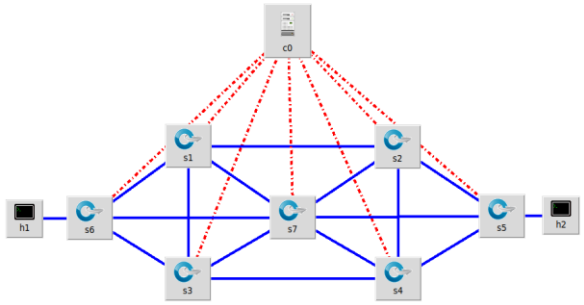


FIGURE (3.B) TOPOLOGY 2

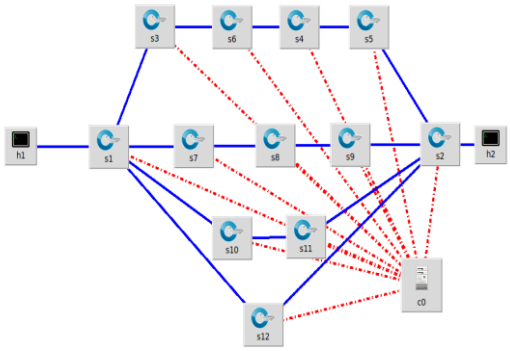


FIGURE (3.C) TOPOLOGY 3

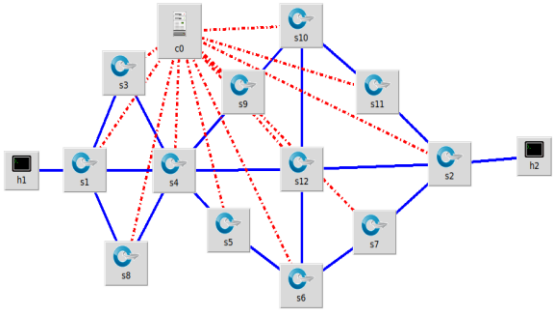


FIGURE (3.D) TOPOLOGY 4

Moreover, there may be topological scenarios where multipath routing can be applied to real-world networks. One of them is the Internet2 Network topology we could use here, which generally represents the backbone network infrastructure in the United States, replacing the Abilene Backbone Network [15]. Internet2 Network can be mapped and modeled by using Mininet as shown in Figure 4, where the link delay of each pair between two nodes of this topology is taken in [ms] as [15] measured and shown in Figure 4.

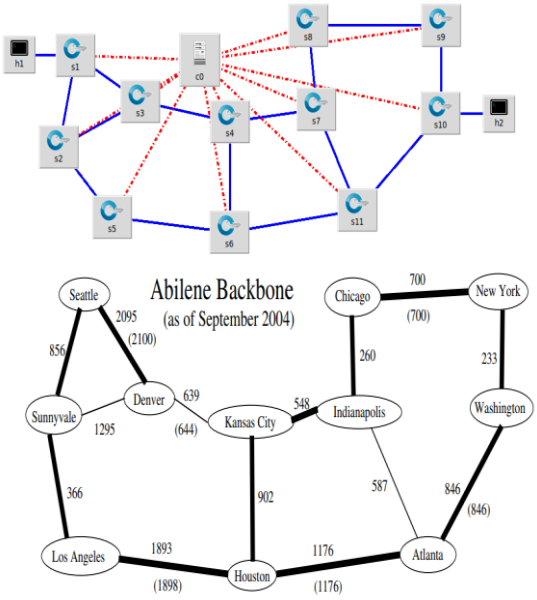


FIGURE (4) INTERNET2 TOPOLOGY

5-2- Path-finding Test Results:

In assessing the multipath routing algorithm, we had developed, we observe the number of paths found and measure the response time of the routing carried out. To get the response time value, we use the ping command form h1 to h2. Response Time (i.e., rtt parameters) is taken from the latency of sending the first ping packet. The results are shown in Table 1, considering MAX_PATHS =3. This test was carried out on our laptop computer with CPU Inter Core i7 specifications, 8 GB RAM, and on the Ubuntu 20.04 operating system running on VMWare Workstation 16. In Figure 5, one can see an example of system output in routing on Figure 3.A, where the system can find all available paths between h2 and h1.

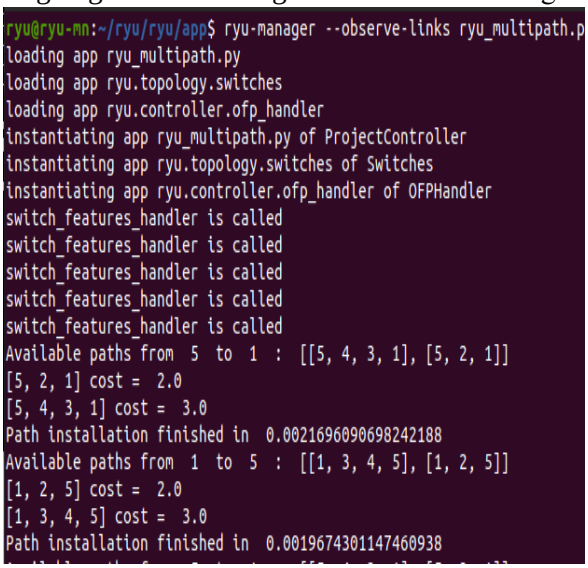


FIGURE (5) ALL AVAILABLE PATHS BETWEEN H2 AND H1, ACCORDING TO TOPOLOGY1.

For the Internet2 Network topology scenario, Figure 4, a variation is made of the maximum number of paths taken, namely by setting the MAX_PATHS variable. First, continuous changes are made from values 2 to 5 of this variable, then our experiments are carried out if MAX_PATHS is infinite. The results are shown in Table 2.

TABLE (1)Route search test results for the four topologies considered.

Topology Scenario	The total number of paths found	rtt [ms]	
		Mean	Standard Deviation
Figure 3.A	2	7.509	12.844
Figure 3.B	2	8.761	19.422
Figure 3.C	3	10.507	23.275
Figure 3.D	3	11.593	16.791

TABLE (2) Route search test results for the Internet2 Topology.

MAX_PATHS	The total number of paths found	Response Time [ms]	
		Mean	Standard Deviation
2	2	30.054	33.164
3	3	29.224	29.715
4	4	29.263	31.280
∞	16	Too long	Undetermined

Based on the results of the path search test that has been carried out, the system has succeeded in performing a multipath path search in all topologies. Topology 1 and Topology 1 have two paths between h1 and h2, while Topology 3 and Topology 4 have three paths between h1 and h2. Looking at the results of the response time test, the mean (average value) rtt increases with the addition of switches to the topology for each case considered, that is: mean(rtt1)<mean(rtt2)<mean(rtt3)<mean(rtt4).

For topologies with much many possible paths with predetermined delay realistic links, as is the case for Internet2 Topology (Figure 4), it is necessary to control the maximum number of paths included in our multipath scheme. This may be done through determine the parameter MAX_PATHS. On Figure 4, when there is no maximum limit on the number of paths in the topology (MAX_PATHS=∞), which is 16 paths, ping packets will always be dropped because there are too many path choices, making the switch take too long to process the packet. But, by limiting the number of paths with the MAX_PATHS variable in the simulation, the system can provide a response time of 30.054 ms for 2 paths, 29.224 ms for 3 paths, 29.263 ms for 4 paths, as Table 2 shows.

5-3- Throughput and Delays Performance (TCP/IP Test):

Throughput states the number of bits processed on a link per unit time, while delay is the delay in delivery time experienced on the network. Throughput and delay measurements are carried out here to find out whether the addition of paths that do not have the same weight will affect network performance in the means of QoS, comparing with the single path algorithm (see Table 3). To measure throughput, the iperf tools is used between h1 and h2 of each topology. Here, h1 acts as a video server by using the command iperf -s as Figure 6.A shows.

where throughput testing was carried out for 60 seconds for each topology of Figures 3.A, 3.B, 3.C, 3D. Then the delay test is carried out by pinging for 60 seconds from h2 to h1. Ping action, for delay measurement purposes, is done after routing is done, where we put MAX_PATHS = 3.

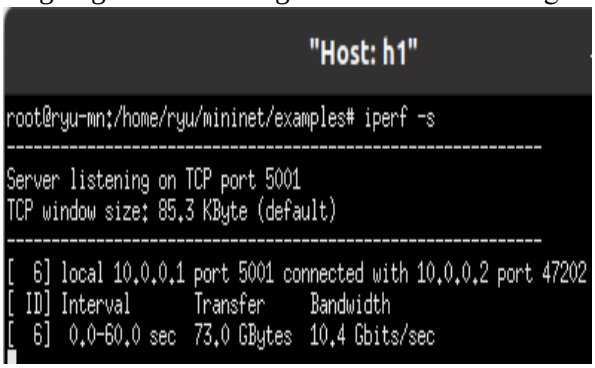


FIGURE (6.A) SETTING UP H1 AS SERVER.

and h2 acts as a video client by using the command iperf -c 10.0.0.1 -t 60, as Figure 6.B shows.

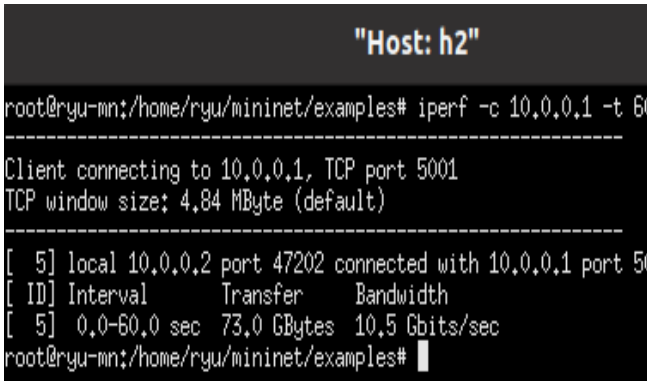


FIGURE (6.B) - SETTING UP H2 AS CLIENT.

TABLE (3)TCP/IP test results for the four topologies considered.

Topology Scenario	Throughput and Delay			
	Throughput [Gbps]		Delay [ms]	
	Multipath	Single-Path	Multipath	Single-Path
Figure 3.A	10.5	11.9	0.405	0.825
Figure 3.B	9.13	10.1	0.212	0.712
Figure 3.C	11.5	13.1	0.206	0.562
Figure 3.D	10.2	11.8	0.199	0.623

Based on the results of throughput and delay testing, there is no significant change in network performance when there are additional lines or there are paths that have different costs in the topology. When compared between 3.A and 3.B which both have 2 possible paths between h1 and h2, it can be seen that the difference in average delay is not too significant, even 3.A which has a shorter path has a higher delay, but 3.B has an average lower average throughput. This also happens when comparing 3.C with 3.D which both have 3 possible paths, where the network performance is not significantly different if there are paths with different weights. In addition, the difference in the number of paths here may not have a very significant effect on network performance, where as a whole 3.A, 3.B, 3.C, and 3.D which have 2 or 3 number of paths do not change so much the average values of throughput and delay. From Table 3, We also note that less bandwidth was consumed in the network using the multipath algorithm, compared to the single path algorithm, and that the delay is less in the case of the multipath algorithm than in the case of the single path algorithm, because the transmission of packets in the case of multipath is done in parallel. As we found out when transmitting a video broadcast between h1 and h2, the more delay in the arrival of the stream packets was evident when watching the video at h2, when applying single-path routing compared to load-balanced-multi-path routing.

5-4: Load-Balancing Performance

Load-balancing testing is carried out here by observing the transmission packet counter (byte counter) from switches with outgoing traffic or traffic coming out of each port contained in our multipath scheme. This test is carried out to assess whether the system has successfully distributed the load to several existing paths. For this reason, the iperf tool is used again, which is run with an interval of 30 seconds from h1 to h2, for each topology. This is done by using the commands, iperf -s and iperf -c 10.0.0.1 -t 30 -i 1. This will also show the time-varying of bandwidth used on all paths considered in multipath mode, and this is shown in Figure 7, where the topology that is considered here is that of the Figure 3.A. The same settings are taken for the other topology scenarios.

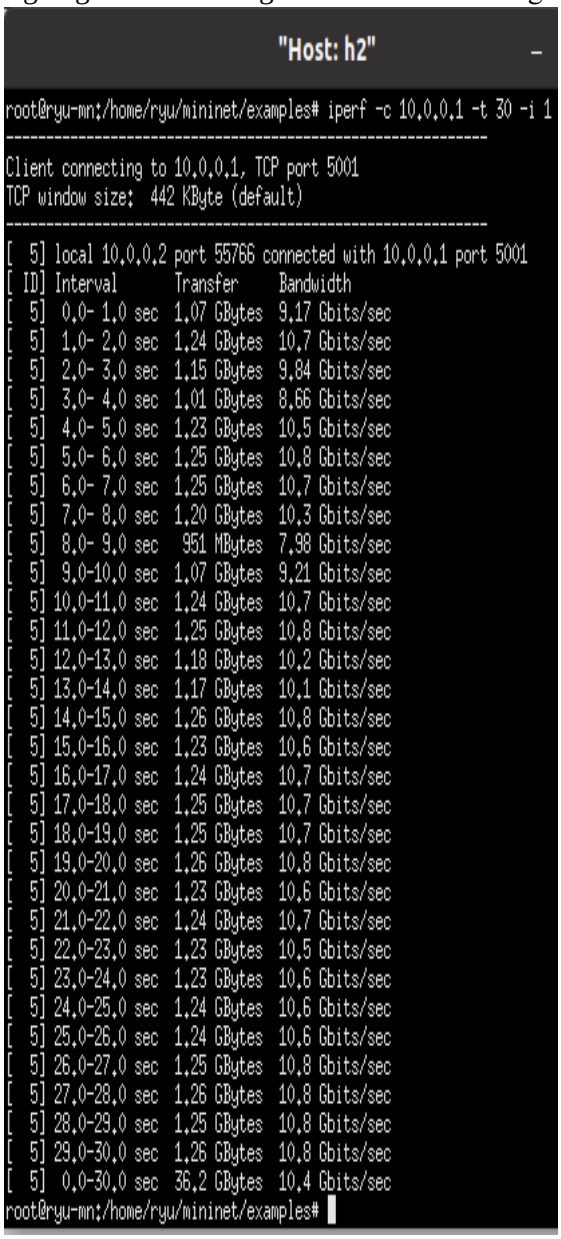


FIGURE (7) SETTING UP H2 AS CLIENT, IN ORDER TO STUDY LOAD-BALANCING AND TIME-VARYING OF BANDWIDTH.

One can note from Figure 7 that the consumed bandwidth is almost constant around the value 10.5 Gbps without any sudden changes that negatively affect the transmission QoS. Now we check if the total packets sent from the outgoing ports of s5 of topology of Figure 3.A, by using this command:
sudo ovs-ofctl -O OpenFlow13 dump-ports s5
The result of this is shown in Figure 8.

FIGURE (8) RESULT OF THE GROUP ACTION RULE APPLYING AT THE OUTPUT PORTS OF OVS NAMED S5 OF TOPOLOGY OF THE FIGURE 3.A.

Figure 8 previously shows that the group action is processed by the OpenFlow 1.3 protocol along with the output ports of OVS named s5 of the Figure 3.A, i.e., all the ports of s5 are in use for the purpose of multipath routing. The details of that group action can be shown by using this command:
sudo ovs-ofctl -O OpenFlow13 dump-groups s5
The result of this command is shown in Figure 9.

FIGURE (9) THE GROUP ACTION RULE APPLYING AT THE OUTPUT PORTS OF OVS NAMED S5 OF TOPOLOGY OF THE FIGURE 3.A.

From Figure 9, we get a bucket weight result corresponding with that obtained in the example proposed in the III. Path Selection section $\boldsymbol{W(p1)} = 0.6$, where the {bucket=weight:6} (the path $\boldsymbol{p1}$ via port s5-eth3, after multiplying by 10 in the Ryu Code) and $W(p2) = 0.4$, where the {bucket=weight:4} (the path $p1$ via port s5-eth1, after multiplying by 10 in the Ryu Code). This confirms that the load-balancing is carried out here by the network system at switch s. Various information that can be obtained here is the number of transmission and reception of packets on each port in the switch, and the group action that determines which ports are used in multipath along

with their weights. For this, we examine how the packets are sent/received using through those ports. To visualize the current state of the topology of the Figure 3.A, the traffic of the ports of s5 is shown in Figure 10.

FIGURE (10) MULTIPATH ROUTING THROUGH THE OVS NAMED S5 OF TOPOLOGY OF THE FIGURE 3.A.

From Figure 10, we note, according to the flow rules implemented by switch s5 that has been learned from Ryu controller, that host h2 via port "s5-eth2" of switch s5 should receive packets from host h1, while the packets from h2 should be sent via switch s5 via ports "s5 -eth1" and "s5-eth3". We can easily see this by looking at the tx (uplink) packets of port 's5-eth1' and 's5-eth3', and rx (receive) packets for port 's5-eth2'. The green and purple arrows indicate the mechanism of routing the incoming and outgoing flows through the s5 switch, by the multi-path method assumed in this research, where the ports "s5-eth1", "s5-eth2", "s5-eth3" in this switch are determined based on the paths discovered between the hosts h1, h2 that want to communicate with each other over a network SDN that is programmatically controlled according to the Open Flows 1.3 protocol. We note from Figure 8 that 1,793,426 packets were sent from switch s5 through port "s5-eth1", and that 2,456,555 packets were sent from switch s5 through its port "s5-eth2", and that 886,130 packets were sent from it through its port "s5-eth3", and that 803,962 packets were received by the s5 switch via port "s5-eth1", and that 2,673,195 packets were received through port "s5-eth2", and that 1,655,781 parcels were received through port "s5-eth3". This corresponds to the use of two paths to achieve the flow of video traffic through the s5 switch and not just through one path of the network as the traditional situation of single-path routing. Thus, the multi-path installation could represent a good principle in routing traffic in the SDN network. In the following we verify the load balancing ratios between these two paths, i.e., $\frac{W(p1)}{W(p2)}$ the ratio actually resulting from the simulation of communication between h1 and h2.

Now we suppose that a number of clients, 4 ones, (four participants in a video conference, or let's say four YouTube users) are connected through node s5 via our SDN with server h1, considering each one of the network topologies assuming in the Figures 3.A, 3.B, 3.C, 3.D, and 4. This could be simulated by using the following command at the host h2, where P indicates "Participants".

iperf -c 10.0.0.1 -P 4

We begin with the topology 3.A, so, we obtain the result via the following Figure 11.

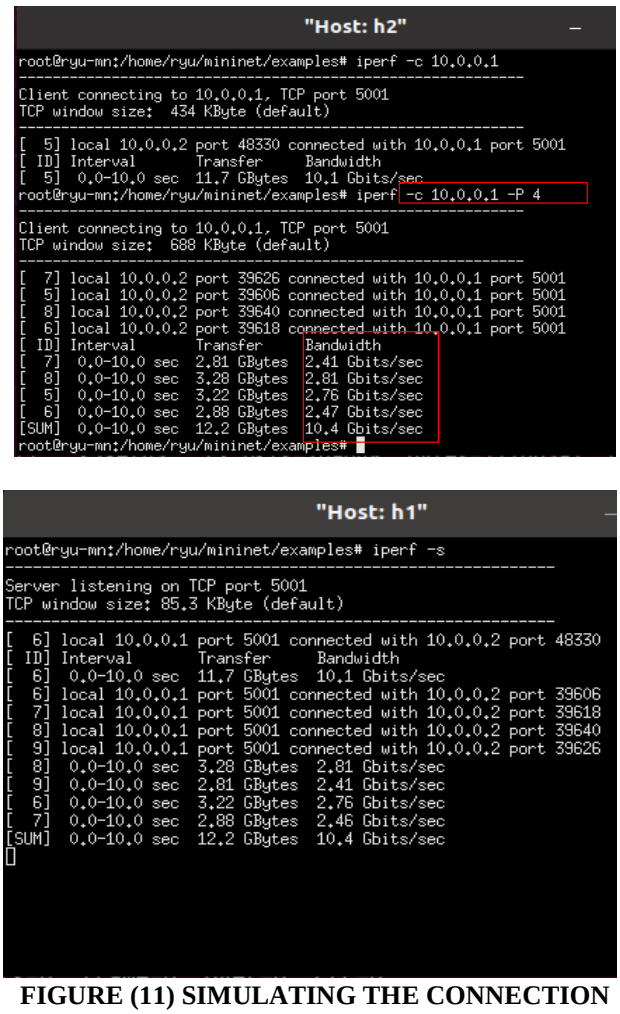


FIGURE (11) SIMULATING THE CONNECTION BETWEEN 4 PARTICIPANTS VIA SWITCH S5 WITH THE SERVER H1.

Here, four clients each connect to s5, with a connection speed of about 2.5 Gbps. To view the number of packets transiting the s5 switch according

to this connection, and to view the ports used in this switch to route and direct packets to and from the host server h1. So, we use the following command again, and Figure 12 shows the results.

sudo ovs-ofctl -O OpenFlow13 dump-ports s5

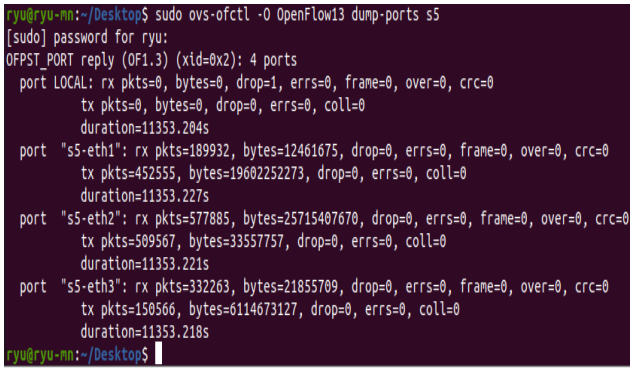


FIGURE (12) SIMULATING THE CONNECTION BETWEEN 4 PARTICIPANTS VIA SWITCH S5 WITH THE SERVER H1.

Here we have the number of packets received from port "s5-eth2" in this test (four clients connected to s5) equals 577885, and in another test (one client connected to s5) equals 286630, so the difference is 286,630. The number of packets sent from port "s5-eth1" in this test (four clients connected to s5) is 452555, and in another test (one client connected to s5) equals 295410, so the difference is 295,410. The number of packets sent from port "s5-eth3" in this test (four clients connected to s5) is 150566, and in another test (one client connected to s5) is 8822, so the difference is 141,744.

Now, we can see how packets routed through the s5 switch are load balanced (i.e., the flow of video traffic is flowing over more than one path in the network and the flow through one path is balanced with the flow of the same video content over the other path), where the load balancing ratio (LB_R) can be calculated between ports "s5-eth1" and "s5-eth3" according to the criterion suggested here, which should be close to the theoretical ratio suggested in equation (6), we found:

$$\text{LB\_R}_{Topology3.A} = \frac{141744}{295410} \approx 0.5 \cong \frac{\text{W(p1)}}{\text{W(p2)}} = \frac{0.4}{0.6} = 4{:}6 \cong 0.6$$

where $p1$ corresponds to the path s5 → s2 → s1, and p2 corresponds to the path s5 → s4 → s3 → s1. Comparing this to our theoretical bucket weight ratio of 4:6, it seems to be Close enough.

We literally carry out the same previous steps for the rest of the network scenarios Figures 3.B, 3.C, 3.D, and 4, and we get Table 4, where we put MAX_PATHS = 2, for each, so that the theoretical can be 4:6 for each topology, i.e., there are two available paths whose costs are 2 and 3 as the example shown previously.

TABLE (4) Simulated Load-Balanced-Ratio obtained from the rest of Topology scenarios.

Topology Scenario	LB_R (Theoretical)	LB_R (Simulated)
Figure 3.B	4:6	0.52
Figure 3.C	4:6	0.46
Figure 3.D	4:6	0.56
Figure 4	4:6	0.48

Table 4 shows the rounding ratio of packet transmission on each port. At T1, the load-balancing result of the packet transmission is 3:1. Still far from the weight allocation ratio of each port, which is 0.5. This is also experienced in other topologies, where the load-balancing results are approximately consistent with the allocation scheme of bucket weights considered for each port of each OVS connected to the considered hosts h1 and h2. Thus, we can say that the routing in the network is load-balanced in some sense.

We conducted simulated video transmission tests via the RTP/MPEG transport protocol using VLC media player software under the UBUNTU operating system to measure the quality of service and the video content streaming process. We used a short video clip file and using the VLC software, which allows performance to be studied almost realistically in real time on the mininet platform. Data packets are transferred between the virtual h1,h2 as a Server and h2 as a Client on the operating system, which are virtual hosts configured according to special settings of the VLC software on each one of them, where it is possible to simulate a real video transfer between two computers connected to the network and observe the QoE resulted and the resulting performance completely clearly when adopting multi-path routing based on the load balancing principle compared to single-path routing.

6: CONCLUSION AND FUTURE WORK:

We have implemented a multipath routing system with load-balancing on OpenFlow 1.3 SDN by

adopting the DFS algorithm for route finding and utilizing the group actions feature in OVS to perform load-balancing. In this paper, a way of calculating edge weight or link cost has also been proposed to calculate the weight or cost of a path by taking into account the load of a link. These weights can be used in path selections to sort out the best paths in multipath routing and as a reference for load-balancing. we used the "select" group action type, and created "buckets" in a group action, then calculated bucket weights from the path weight. Packet load-balancing was done by hashing packet headers down to the TCP headers, multiplying it with the bucket weights, and then selecting the bucket with the highest "score" (Best Paths).

The simulation results for this system are that the system has been able to perform route searches for several paths in a topology with acceptable average response times. In addition, taking into account throughput and delay in the network, there is no significant degradation of network QoS performance when there are additional paths with different weights in all considered topologies. For the test results of the load-balancing, it can be said that the distribution of the transmission load is approximately in accordance with the theoretical bucket weights allocated to each port considered for multipath routing of each OVS connected to the tested hosts.

In this paper, we implemented our algorithms programmatically on an SDN controller, by using Ryu controller type, in order to conduct experiments on a virtual network created by the Mininet platform. This enables us to confirm the current results using OVSs, considering multiple paths for video transmission instead of a single path produced improvements in the quality of the image transmitted in the video and achieving the principle of load balancing on the OVS-type switches connected to the virtual network. The next step as a future work will be to achieve this on a real network if we have solid equipment for both the Ryu controller and OVS-type switches on which the SDN concept is implemented.

The General Telecommunications Authority in Syria could benefit from these future prospects to conduct what is necessary with the aim of implementing some field experiments in the context of SDN, and providing the necessary support to provide the appropriate requirements of hardware equipment to connect a simplified SDN network that supports video transmission between several users who are geographically separated and connected to the network via OVS-type switches. This can be achieved by sending a television video broadcast in cooperation with the Ministry of Information between users for pre-defined purposes. Thus, obtaining more realistic results for the multi-path routing algorithm. Thus, it is possible to achieve an initial research structure that conducts real experiments on the concept of SDN, and lays the foundation for more advanced technical work to keep pace with current developments in this fie

قائمة بأهم الاختصارات والمصطلحات

المصطلح باللغة الإنكليزية	الاختصار	المصطلح باللغة العربية
Adaptive Worst-fit Multipath Routing	AWMR	التوجيه المتعدد المسارات الأكثر ملاءمةً للتكيف
Application Programming Interface	API	واجهة برمجة تطبيق
Bandwidth	BW	عرض الحزمة
Depth-First Search	DFS	البحث عن العمق الأول
Internet Protocol	IP	بروتوكول الإنترنت
Media Access Control	MAC	التحكم بالنفاذ للوسط
Network Operating System	NOS	نظام التشغيل الشبكي
Open Shortest Path First	OSPF	أول أقصر طريق مفتوح
Open-v(virtual)-Switch	OVS	مبدل افتراضي مفتوح المصدر
Quality of Service	QoS	جودة الخدمة
Round-trip time	RTT	وقت الجولة
Software Defined Networks	SDN	الشبكات المعرفة بالبرمجيات
Transmission Control Protocol	TCP	بروتوكول التحكم بالإرسال
Unicast	UN	البث الأحادي
User Datagram Protocol	UDP	بروتوكول حزم بيانات المُستخدِم

FUNDING: THIS RESEARCH IS FUNDED BY DAMASCUS UNIVERSITY-FUNDER NO (501100020595).

References:

[1] McKeown et al, "OpenFlow: enabling innovation in campus networks", ACM SIGCOMM Computer Communication Review, Volume 38, Issue 2, April 2008, pp.69–74 https://doi.org/10.1145/1355734.1355746

[2] https://opennetworking.org/sdn-resources/customer-case studies/openflow/, [Online] Access on November 2022.

[3] C. Prabha, A. Goel and J. Singh, "A Survey on SDN Controller Evolution: A Brief Review," 2022 7th International Conference on Communication and Electronics Systems (ICCES), 2022, pp. 569-575, doi: 10.1109/ICCES54183.2022.9835810.

[4] Darijo Raca, Meghana Salian, Ahmed H. Zahran, "Enabling scalable emulation of differentiated services in mininet", MMSys '22: Proceedings of the 13th ACM Multimedia Systems Conference, June 2022 pp.240–245 https://doi.org/10.1145/3524273.3532893.

[5] Gupta, N.; Maashi, M.S.; Tanwar, S.; Badotra, S.; Aljebreen, M.; Bharany, S. A Comparative Study of Software Defined Networking Controllers Using Mininet. Electronics 2022, 11, 2715. https://doi.org/10.3390/electronics11172715.

[6] Bhardwaj, S., Panda, S.N. Performance Evaluation Using RYU SDN Controller in Software-Defined Networking Environment. Wireless Pers Commun 122, 701–723 (2022). https://doi.org/10.1007/s11277-021-08920-3.

[7] B. Pfaff, (2015), "The Design and Implementation of Open vSwitch", [Online]. Available at: https://benpfaff.org/papers/ovs.pdf (Accessed: 3 November 2022).

[8] 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

[9] Yi-Chih Lei, Kuochen Wang and Yi-Huai Hsu, "Multipath routing in SDN-based Data Center Networks," 2015 European Conference on Networks and Communications (EuCNC), 2015, pp. 365-369, doi: 10.1109/EuCNC.2015.7194100.

[10] M. F. Ramdhani, S. N. Hertiana and B. Dirgantara, "Multipath routing with load balancing and admission control in Software-Defined Networking (SDN)," 2016 4th International Conference on Information and Communication Technology (ICoICT), 2016, pp. 1-6, doi: 10.1109/ICoICT.2016.7571949.

[11] H. A. Naqvi, S. N. Hertiana and R. M. Negara, "Enabling multipath routing for unicast traffic in Ethernet network," 2015 3rd International Conference on Information and Communication Technology (ICoICT), 2015, pp. 245-250, doi: 10.1109/ICoICT.2015.7231430.

[12] M. Abdelghany, H., W. Zaki, F. i M. Ashour, M. (2022). Modified Dijkstra Shortest Path Algorithm for SD Networks. International journal of electrical and computer engineering systems, 13 (3), 203-208. https://doi.org/10.32985/ijeces.13.3.5

[13] J. -R. Jiang, H. -W. Huang, J. -H. Liao and S. -Y. Chen, "Extending Dijkstra's shortest path algorithm for software defined networking," The 16th Asia-Pacific Network Operations and Management Symposium, 2014, pp. 1-4, doi: 10.1109/APNOMS.2014.6996609.

[14] OpenFlow Switch Specification, Version 1.3.0 (Wire Protocol 0x04) June 25, 2012, ONF TS-006, URL:https://opennetworking.org/wp-content /uploads/2014/10/openflow-spec-v1.3.0 .pdf

[15] Zheng H, Lua EK, Pias M, Griffin TG (2005) Internet routing policies and round-trip-times. In: International workshop on passive and active network measurement, pp 236–250. https://doi.org/10.1007/978-3-540-31966-5_19.