


Article

Route Path Selection Optimization Scheme Based Link Quality Estimation and Critical Switch Awareness for Software Defined Networks

Babangida Isyaku ^{1,2,*}, Kamalrulnizam Abu Bakar ¹, Mohd Soperi Mohd Zahid ³, Eman H. Alkhamash ⁴, Faisal Saeed ⁵ and Fuad A. Ghaleb ^{1,*}

¹ Department of Computer Science, Universiti Teknologi Malaysia, Johor Bahru 81310, Malaysia; knizam@utm.my

² Department of Mathematics and Computer Science, Sule Lamido University, Kafin Hausa P.M.B 048, Nigeria

³ Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Seri Iskandar 32610, Malaysia; msoperi.mzahid@utp.edu.my

⁴ Department of Computer Science, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia; eman.kms@tu.edu.sa

⁵ College of Computer Science and Engineering, Taibah University, Medina 42353, Saudi Arabia; alsamet.faisal@gmail.com

* Correspondence: isyaku@graduate.utm.my (B.I.); abdulgaleel@utm.my (F.A.G.)

Abstract: Software-defined network (SDN) is a new paradigm that decouples the control plane and data plane. This offered a more flexible way to efficiently manage the network. However, the increasing number of traffics due to the proliferation of the Internet of Things (IoT) devices also increase the number of flow arrival which in turn causes flow rules to change more often, and similarly, path setup requests increased. These events required route path computation activities to take place immediately to cope with the new network changes. Searching for an optimal route might be costly in terms of the time required to calculate a new path and update the corresponding switches. However, the current path selection schemes considered only single routing metrics either link or switch operation. Incorporating link quality and switch's role during path selection decisions have not been considered. This paper proposed Route Path Selection Optimization (RPSO) with multi-constraint. RPSO introduced joint parameters based on link and switches such as Link Latency (LL), Link Delivery Ratio (LDR), and Critical Switch Frequency Score (CWFscore). These metrics encourage path selection with better link quality and a minimal number of critical switches. The experimental results show that the proposed scheme reduced path stretch by 37%, path setup latency by 73% thereby improving throughput by 55.73%, and packet delivery ratio by 12.5% compared to the baseline work.

Keywords: SDN; OpenFlow; restoration; path selection; link quality; critical switch



Citation: Isyaku, B.; Bakar, K.A.; Mohd Zahid, M.S.; Alkhamash, E.H.; Saeed, F.; Ghaleb, F.A. Route Path Selection Optimization Scheme Based Link Quality Estimation and Critical Switch Awareness for Software Defined Networks. *Appl. Sci.* **2021**, *11*, 9100. <https://doi.org/10.3390/app11199100>

Academic Editors: Eusebi Calle and José Luis Marzo

Received: 9 September 2021

Accepted: 27 September 2021

Published: 29 September 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

SDN decouples the control plane (networking logic) from the data plane (forwarding logic) [1]. The SDN controllers acted as a central network intelligent while the data plane is released from control function and focus on forwarding packets to the next hop. This way, the SDN controller maintained the global network knowledge and extract network information from the data plane at real time. Standard communication interface is used to separate the control plane from the data plane [2]. The early stage of SDN commences with some standard protocols which include Forwarding and Control Elements (ForCES) and Protocol-Oblivious Forwarding (POF) [2]. However, this standard protocol required modification of forwarding devices to support flow tables update operation [3]. Therefore, OpenFlow standard emerged to support the switch flow table without any modification but required controller intervention for further actions. Although, OpenState was developed

as an extended OpenFlow 1.3 version with goal of delegating some control logic to the switches [4]. In other words, switches can handle packets locally without frequent controller intervention. However, OpenState has not been considered as future extended version of OpenFlow by the standard organization [5]. As a result, OpenFlow is considered the most popular southbound interface standardized by the Open Network Foundation [6] to control the behavior of OpenFlow switch. This way, OpenFlow uses Link Layer Discovery Protocol (LLDP) message to discover switches and links between SDN-Switches [7]. Afterward, the SDN controller defines flows and computes a path to redirect flows to their desired destination, and instructs the OpenFlow switches to install the corresponding rules to guide the routing process [8]. Usually, path selection decision occurs when there is a new network event or change in network status. For instance, network traffic rules have to be frequently updated on average every 1.5 to 5 s due to traffic variabilities exhibited by flows [9]. Similarly, the arrival of every new flow requires switches to request the controller for a new path. In either case, the route path computation activities need to take place immediately to converge with the new network event or network changes.

The SDN controller used either a restoration or protection approach to compute a path and update the switch flow table [10]. In the former, upon arrival of a new flow, switch consult the controller to compute the new path to reroute flows to their destination. In contrast, the latter provisions path routing rules in advance before occurrences of the event. Therefore, it is obvious that protection may have better forwarding performance compared to restoration. However, the main concern is at the significant cost of Ternary Content Addressable Memory (TCAM) in the switch flow table. In other words, the flow table is mostly implemented with high-speed and expensive memory called TCAM. Thus, TCAM is affordable with limited size and capacity typically supporting around 2000 forwarding entries [11]. Hence, restoration offers more flexibility in dealing with frequent network changes but at the cost of time latency to compute a new path more often. In other words, searching for an optimal route might be costly in terms of the time required to calculate a new path and update the corresponding switches flow table [12]. The update operation is proportional to the number of flow rules in the switch flow table. Switches with a high load of flow tables tend to have higher update operation which may prolong the routing convergence time that is sensitive to some applications such as VoIP (Voice over IP). Traffic from delay-sensitive application flows must reach the intended destination without incurring significant flow setup latency typically within 25 ms [13].

The existing route path selection considers a single metric during path selection but does not optimize the switch update operation mainly caused by poor link quality and higher update operation in the switch flow table. Several path selection works have been proposed but they focus on the shortest distance [12], link latency [14,15], link bandwidth [16] as routing metrics for route path selection. These types of metrics are insufficient and not appropriate to meet the challenging Service-Level Agreement (SLA) of delay-sensitive applications. Such flows have a stringent requirement for rerouting, for example, the routing of flows is required to be completed within 25 ms [13]. Moreover, the existing routing schemes overlook a set of critical switches during route path selection decisions. Such a set of switches have many flow table entries more often, which further augment the convergence time, and may cause update operation to yield a non-optimum path. Furthermore, frequent network changes with flows variations cause undesirable network performances in term of longer path stretch which lead to higher path setup latency, and longer switch flow table space consumption.

This paper proposed an optimized route path selection scheme by incorporating new routing metrics considering link and switch parameters. The contributions of this paper can be summarized as follows.

- Devising a model to estimate the link quality based on link latency and link delivery ratio. The restoration approach is embedded with this model to help in the selection of a route for every newly arrived flow.

- Proposing a scheme to detect critical switches among a set of candidate paths. Link quality value and a minimum number of critical switches are used in the computation of an optimized route.

The rest of the paper is organized as follows: Section 2 presents the various route path selection solutions. The design of the proposed solution is described in Section 3. Experimental setup and performance evaluation are explained in Section 4. Finally, Section 5 concludes the paper and suggests future works.

2. Related Works

Several studies were proposed to select a path to route flows to destination based on certain routing metrics as summarized in Table 1. Deploying a Distributed Routing Fabric was among the state of art routing techniques [17]. Other schemes are [18–20]. A Quagga [18] scheme provides an implementation of TCP/IP routing protocols. Similarly, RouteFlow [19] is another technique that implements an IP level control plane on top of an OpenFlow network, allowing the underlying devices to act as IP routers under different possible arrangements. However, the proliferation of IoT devices increases the arrival of flows and flows tend to have different Quality of Service (QoS) requirements. Thus, meeting the QoS demand for different flows may be quite a challenging task [21]. Although the restoration approach has more flexibility in managing real-time flows. Hence, it cannot meet carrier-grade requirements in a large-scale network serving many flows [22]. In contrast, protection may have better forwarding performance, but at the cost of flowtable space. As such, it is challenging to find the best routing path considering the trade-off between flow table operation cost and path cost [23]. Astaneh et al. [24,25] presented a local restoration scheme with a minimal number of flow operations for disaster scenarios. The scheme minimizes the cost of flow table update operations and also the number of hops based on a threshold value. Malik et al. [12] proposed the selection of the best path by examining the existence of shared links between candidate paths since shared links demonstrate the ability to reduce flow table rule consumption. The path with a maximum number of shared links is chosen. However, the chance of overflow and imbalance of network load increases when the utilization ratio of switches becomes high. To balance the network load, Yu et al. [26] presented a path selection method based on significant nodes, and flow prediction using Deep Neural Networks (DNNs) and Q-learning techniques. The use of DNN is quite effective for searching optimal path-based constraints. However, it requires statistical history data which may be ineffective in SDN due to memory constraints [9].

Hu et al. [15] considered latency and packet loss to design path selection. Delay-sensitive flows are routed through a path with minimal delay, other flows are routed via a path with lower packet loss. Similarly, Alnajim and Salehi [27] presented a QoS-aware path selection algorithm to speed up rerouting of time-sensitive flows for real-time applications, which spare the bottleneck links causing the infeasible scheduling. However, the work in [15,27] did not consider the switch load during the path selection decision. Link latency and the ratio of the link delivery should also be considered to guarantee the quality of the link selected. Thus, QoS-driven and SDN-assisted MPTCP path selection scheme (QSMPS) for high-quality transmission service was proposed in [28]. QSMPS utilizes a scalable SDN-assisted approach to monitor and analyze network status information. QSMPS calculates the optimal number of sub-flows and distributes them to the least differential delay paths accordingly. However, an optimal path based on residual bandwidth may not always guarantee the requirements of different flows. Other flows may prefer minimal setup latency and link quality that reduces frequent link changes due to topology changes. Consequently, Saha et al. [29] presented a greedy heuristic scheme based on Yen's k -shortest path algorithm to compute optimal routing paths while considering QoS requirements for each flow. Multiple metrics are jointly considered (packet loss, delay, and bandwidth). An integer linear programming is utilized to solve the multi-constraints optimal QoS-aware route. However, an ILP based solution may lead to slow convergence in large-scale networks with frequent topology changes. [9].

Chooprateep et al. [16] introduced Video Path Selection Algorithm (VPSA) for multimedia flows. VPSA uses the previous and current link bandwidth usage data collected by the controller to calculate a suitable path. The arrival of video flows is rejected or accepted based on the existence of a suitable path. Similarly, Rangkutiy et al. [10] leverage the concept of monitoring to pull the statistical data and calculate the port load. Flows are routed through a less loaded path. However, frequent polling of previous and latest link bandwidth usage or port in [10,16] may significantly affect topology learning and discovering time, which in turn increases the path setup latency [7]. The flow setup latency was measured by (Khalili et al., 2018). The author used a special packet issued by the controller to switches to measure the round trip time of packets back to the controller. This way the time is appended to each path to obtain the total path setup latency. Ravuri et al. [30] extend in the work in [14] considering distributed controllers. However, both schemes may experience higher flow table operation time thereby affecting path setup switching time. An effective method to minimize the total setup switching time of all paths in the network was presented in Gotani et al. [31,32]. The authors considered path route selection based on path switching latency in heterogeneous networks, where switches had different specifications. This way, several paths are explored and the path with the set of switches having the shortest processing time is considered. However, the solution may not give the optimal performance in a large-scale network. Thus, Perner and Carle [33] investigate different objective functions for the optimization model with respect to their impact on network performance such as link utilization and latency. This way path routing was devised based on certain constraints to meet the demand of the network performance. Again, Integer Linear Programming (ILP) model may slow down network convergence of large-scale networks [9].

Table 1. Summary of the related works.

| Schemes | Aim | Algorithm/ Scheme | Link Latency/ Delay | Switch Update Operation | Limitation |
|--------------------------|---|--|------------------------|----------------------------|---|
| (Astaneh et al. [24,25]) | To minimized rule update operation | Optimization model minimizes operation on the path | X | ✓ | An optimization scheme may take time to converge in large-scale networks |
| Malik et al. [9,12,34] | To devise a path selection scheme with a minimum number of entries | Reliable pathfinder and path selection criteria | X | ✓ | Overlook critical switches, which may affect path setup time |
| Yu et al. [26] | To optimized data transmission through the optimal path | Deep neural network | X | ✓ | Overlook quality of links during route selection decision |
| Khalili et al. [14] | To reduce latency and the variance of the flow setup | Aggregated flow setup mechanism | ✓ | X | Path switching time and number of flow rules may affect path setup convergence time |
| Hu et al. [15] | To joint QoS-specific factors to select an optimal path for data delivery | Path selection model (PSM) | ✓ | X | Increased packet processing time due to high number of rules in critical switches |
| Alnajim et al. [27] | Routing time-sensitive flows based on QoS requirement | QoS-aware path selection algorithm | ✓ | X | Considering the link latency only can't guarantee the quality of a link |
| Saha et al. [29] | Path selection to maximize the overall network performance | ILP multi-constraint QoS aware routing | ✓ | X | Slow convergence in large scale networks due to frequent changes in network |

Table 1. Cont.

| Schemes | Aim | Algorithm/ Scheme | Link Latency/ Delay | Switch Update Operation | Limitation |
|-----------------------|---|----------------------------------|------------------------|----------------------------|--|
| Chooprateep [16] | Path selection aimed at accepting large video request | Video path selection algorithm | X | X | Discovery of previous and current link bandwidth result in trade-off b/w topo discovery period and learning which result in path setup latency |
| Rangkuty et al. [10] | To prevent congestion or avoid link overload and achieve load balancing | Path selection mechanism (PSM) | ✓ | X | |
| Gotani et al. [31,32] | minimizing the total path switching time of all paths in the network | The path switching time model | X | ✓ | The solution was tested in small network and frequent network changes can affect the quality of links on a selected path |
| Pemer et al. [33] | Investigate the impact of link utilization and latency on network performance | ILP Model | ✓ | X | ILP solution required frequent trigger when a network change |
| [28] | path selection for high-quality transmission service | QoS-driven path selection scheme | ✓ | X | Bandwidth based path selection cannot guarantee optimal path setup latency |

From Table 1, none of the existing works have considered both link quality and switch update operations. In a dynamically changing network, link quality is critically affected, resulting in frequent link failure in the network. Therefore, it is paramount to consider both link quality and switch resource parameters during routing path selection decisions. This is a non-trivial problem because link quality and switch resources provide information on how well the network can handle additional demand as the network evolves. To this end, this paper proposed an optimized route selection scheme-based link quality estimation and critical switch awareness.

3. Design of the Proposed Solution

The proposed Route Path Selection Optimization (RPSO) scheme selects a path with optimized links quality and switches resources from source to destination in SDN. Quality links are selected considering multi-constraints (link latency, link delivery ratio), and a minimal set of critical switches are considered from the switches point of view. Figure 1 presents the architecture view of the proposed RPSO. The network topology maintained by the SDN-Controller is entirely dependent on OpenFlow Discovery Protocol (OFDP) [7]. Any falsification in network topology can directly affect the SDN controller services. Therefore, the SDN controller uses a Link-Layer Discovery Protocol (LLDP) message to discover the network forwarding element such as switches, links, and host at time t_i . In this regard, the SDN controller generates a *Packet-Out event* to send LLDP advertisements to each active switch. In return, the switch will send the link information by encapsulating the LLDP message through *the Packet-In event*. Similarly, the arrival of new flows or table-mis entry, switches generate *Packet-In event* to consult the SDN controller for further action. In this study, the controller generates special crafted *Eco_Ping* (ECP) packets through special Internet Protocol (IP) and Medium Access Control (MAC) address from the controller and back while measuring the amount of time it took to do so. This way, the ethernet frame was broadcasted with a payload containing the port number and a timestamp of the packet's creation time toward adjacent switches S_1 via *Packet-Out*. Conversely, entry-miss will occur for the ethernet type at S_2 and ECP will be sent back to the controller through *the Packet-In event*. This way, RPSO retrieves the packet and deduce how long it took to complete the journey by subtracting the received time from sending timestamp. The RPSO scheme consists of two main phases (i) an initial routing path set-up which includes topology discovery, multi-link constrained aware routing phase, and (ii) critical switch detection. The subsequent subsection describes the details of each phase.

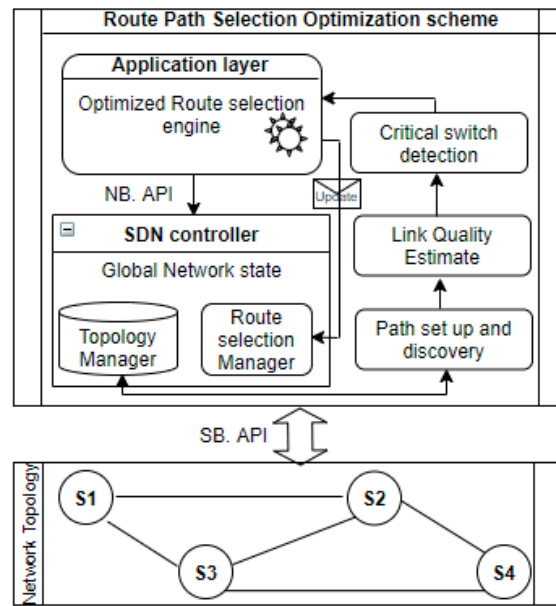


Figure 1. Architecture of the Route path selection Optimized scheme.

3.1. Initial Network Discovery and Path Setup

Initially, at time t_0 , the port numbers of switches and links connecting them are discovered through the LLDP packet. Once this network information is discovered at t_1 , the list of adjacent switches connected to each switch are formed and stored in a local data structure implemented in Algorithm 1 (line 1–4). Upon topology discovery, all the adjacent switches are obtained. For each switch, S_i a path routing table is maintained with a list of adjacent switches and all possible sets of P_i from source S_i to destination S_j as demonstrated in Algorithm 1 (line 5–7). For each adjacent switch, S_i to S_j the Link Latency (LL) and Link Delivery Ratio (LDR) is estimated and stored on each link as shown in Algorithm 1 (line 7–9). To calculate the LL and LDR , the controller creates a Special Ping Packet (SPP) and sends it to S_i -containing sequence no and time stamp. Consequently, S_i sends the SPP to S_j while S_j will forward the SPP back to the controller. These special packets are measured through the probe message at the discovery stage which is transmitted using a dedicated link before actual data transmission.

The total LL time can be summarized as a combination of four parameters: Controller SPP generation time (T_{Pc}), controller to switch SPP creation time (T_{Pc-si}), S_i to S_j SPP forwarding time ($T_{P,si-sj}$), S_j to controller SPP forwarding time ($T_{P,sj-c}$). The LL for $L_{i,j}$ is the summation of a lower and upper bound of the four parameters as shown in Equation (1).

$$LL = \sum_{S_i}^{S_j} \left(\frac{T_{Pc} + T_{P,S-si} + T_{P,si-sj} + T_{P,sj-c}}{2} - C \right) + 1 \tag{1}$$

The LDR is defined as the ratio of received SPP to the sent SPP on both directions of a particular link as shown in Equation (2). Thus, the routing of delay-sensitive flows required the design of realistic metrics that incorporate link quality to meet up the application demand via a path with lower latency and a set of switches incurring lightweight update operation. As such, the total path latency PL is obtained by the summation of the LL on the set of links connecting source switches S_i link to the last link connecting destination switch S_j and the LDR as presented in Equation (3).

$$LDR = \frac{\text{No. of sample } SPP_{Ack}}{\text{No. of Sample } SPP_{sent}} \tag{2}$$

$$PL = \sum_{src}^{dst} (\min(LL)) + (\min(LDR)) \quad (3)$$

Arrivals of new flow will trigger the scheme to compute a set of $k = \{P_1, P_2, \dots, P_n\}$ such that $P_i \in k$ is based on the summation of lower and upper bound from source switches to destination which in turn give the PL . The values of LL and LDR are assigned to each P_i (Algorithm 1, line 19–24) and the updated set k is returned. The set k of P_i satisfying QoS requirement with minimum PL are considered as candidate paths which are used in the critical switches detection phase to choose the P_i with min set of switches.

Algorithm 1: Initial discovery and routing path set

```

1  Get switches  $S_i$ 
2  Get links  $i, j$ 
3  Get hosts
4  Set discovery time  $t$ 
5  //Initial discovery
6  If  $T_{Discovery} = 1$  then
7      | Get adjacent  $S_i \rightarrow S_j$ 
8      | Set  $L_{i,j} \leftarrow LL: i, j \in L$ 
9      | Set  $L_{i,j} \leftarrow LDR: i, j \in L$ 
10     | else
11     | | Repeat line 1–4
12     | end
13  End
14  //Compute the value of  $LL(L_{i,j})$ ,  $LDR(L_{i,j})$ , for set of  $k\{P_1, P_2, P_n\}$ ;  $P_i \in k$ 
15   $LL = \sum_{S_i}^{S_j} \left( \frac{T_{Pc} + T_{P,S-S_i} + T_{P,S_i-S_j} + T_{P,S_j-C} - C}{2} + 1 \right)$ 
16   $LDR = \frac{\text{No.of sample } SPP_{Ack}}{\text{No.of Sample } SPP_{sent}}$ 
17   $PL = \sum_{src}^{dst} (\min(LL)) + (\min(LDR))$ 
18  //Evaluate the cost and append to each  $P_i$ 
19  If  $S_i = \text{Src switch}$ , then
20      | Get the dst switch  $S_j$ 
21      | Compute  $k$  shortest path  $S_i$  to  $S_j$ 
22      |  $k[P_i][S_i \text{ to } S_j] = LL + LDR$ 
23      | Sum the lower and upper bound  $PL$ 
24  end

```

3.2. Critical Switch Detection and Expected Load

The criticality of switches is measured by the number of rules operations on each switch. It reflects the amount of traffic load between different $s-t$ pairs passing through the switches. A switch serving an intermediary node ($N_\Psi \in N$) for n number of shortest paths between $s-t$ is considered more critical than ordinary ($N_o \in N$) switches. This way, N_Ψ are heavily loaded with a large number of routing rules while N_o tend to have a small number of rules compared to N_Ψ . The former and the latter are detected in two ways with the respect to the connectivity and expected traffic load. Regarding connectivity, the

number of shortest paths k is returned from the initial discovery and routing path sets are used. Using k , SCBR checks how many P_i of these paths passed through N_Ψ . Therefore, the criticality of the C_{SCBR} node, N_Ψ of switch N uses the betweenness centrality (NBC) formula, $k(s, t | N_\Psi) / k(s, t)$ presented in equation 4. NBC is a concept that reflects the importance of node/switch about others in the network. The total expected traffic load of $\sigma(N_\Psi)$ on the switch N_Ψ is determined by the summation of rules $\Sigma_{\mathcal{R}}(N_\Psi)$ for the flow demand whose passed through switch N_Ψ . It is assumed that all switches originate the same amount of traffic.

$$C_{SCBR}(N) = \sum_{(s,t) \in N, s \neq t} \frac{k(s, t | N)}{k(s, t)}, \sum \mathcal{R}(N) \tag{4}$$

The traffic load on switch $\sigma(N_o)$ is obtained similarly with the critical switch N_Ψ . Equation (5) represents the traffic load on both switches. As the C_{load} represents the switches traffic load, the initial switch traffic load is estimated first and then incremented at run time as the traffic flows evolve. The initial estimate of the C_{load} , also referred to as network early-stage demand $d_e(s, t)$, is obtained by generating ping packets between some hosts at time t during the network booting time which in turn is also used during the network discovery stage. The procedure is demonstrated in Algorithm 2, lines 1–11.

$$C_{load} = \sum (\sigma(N_\Psi), \sigma(N_o)) \tag{5}$$

Obviously, generating equal traffic across all the switches may easily increase the rule update operations especially to critical switches. It is also important to note in computing C_{load} , the total number of flows rules augment as the network size increase. In practice, this issue can be addressed by computing only a small subset of switches forming the path of route flow to destination. Therefore, computing C_{load} does not involve an exhaustive computation of all the summations of C_{load} presented in Equation (5) for the whole network. This way, SCBR focus on the given source to destination pairs switches on the path where the route path selection is restricted to the k -shortest path candidate for each demand pair $d_e(s, t)$, with $k = 5$. Since the role of switches differs, and similar roles apply among the switches N_Ψ , the list of the topmost critical switches N_Ψ is also derived from Equation (6). Critical switches were restricted to the topmost critical switches. For instance, it is assumed that the switches frequency in each of the list of paths follows Zipf’s distribution with $i = 1$, and i to (φ) .

$$\varphi = \sum_{i=1}^n i = \frac{n}{2} \tag{6}$$

RPSO focuses on routing flows through a path with a lightweight rule update operation. In other words, the rule update operation is proportional to the number of rules on the switch and critical switches exhibit such behavior. For this reason, it is important to explore paths together with their set of critical switches. As such, C_{SCBR} obtained the path with the least critical switches. Each critical switch N_Ψ is set with a flag value $N_\Psi flag = 1$. Therefore, the study obtains the summation $\sum N_\Psi, flag$ in each list of k paths, which in turn served as critical switch frequency score (CWF_{score}). In this way, for each switch S_i the value of $C_{SCBR}(N)$ and C_{load} are retrieved and append the corresponding S_i flag as shown in Algorithm 2 line (12–16). Similarly, for each P_i in P , such that $P_i \in P$, the summation of S_i value is appended to the corresponded P_i . This way, three case scenarios are derived, Algorithm 2 lines (17–19) signifies the procedure. If CWF_{score} in P_i is less than T , then P_i is considered, and the status is updated to P_{low} . However, the CWF_{score} is greater than T P_i is set to P_{high} . Interestingly, there may be some variation, or more than one P_i may have the same score. In this case, the P_i with a minimum number of hop count or distance is considered as the final path, Algorithm 2 demonstrates the procedure line (20–32). In practice, shorter paths typically tend to consume fewer network resources compare to

longer paths, and hence the path which best minimizes the CWF_{score} is one among the k shortest candidate path. SCBR performs route path selection using online (reactive).

Algorithm 2: Critical switch detection and load

```

1  Get distance
2  Get set of  $k$   $\{P_1, P_2, P_n\}$ 
3  //Compute NBC
4   $S_i \leftarrow C_{SCBR}(N) = \sum_{(s,t) \in N, s \neq t} \frac{k(s,t|N)}{k(s,t)}, \sum \mathcal{R}(N)$ 
5   $S_i \leftarrow C_{load} = \sum(\sigma(N_\psi), \sigma(N_o))$ 
6  If  $S_i$   $C_{SCBR} > N$  &  $C_{load} > N$ , then
7  |   Set  $S_i \leftarrow N_\psi$ 
8  |   else
9  |   |   Set  $S_i \leftarrow N_o$ 
10 |   End
11 end
12 For  $S_i$  in  $P_i$ 
13 |   Get  $C_{SCBR}$ 
14 |   Get  $C_{load}$ 
15 |   Set  $S_i^{flag} \leftarrow (C_{SCBR} \cup C_{load})$ 
16 end
17 For each  $P_i$  in  $P$ :
18 |   Get  $CWF_{score}$ 
19 |   Append  $P_i(CWF_{score})$ 
20 |   If  $CWF_{score}$  in  $P_i < T$ , then
21 |   |    $P_i\_status = low$ 
22 |   |   Set  $P_i \leftarrow P_{low}$ 
23 |   end
24 |   If  $CWF_{score}$  in  $P_i > T$ , then
25 |   |    $P_i\_status = high$ 
26 |   |   Set  $P_i \leftarrow P_{high}$ 
27 |   End
28 |   If  $CWF_{score}$  in  $P_i < T$  &  $P_i > I$ , then
29 |   |   Select  $\min(P)$ 
30 |   End
31 end
32 Return  $P$ 

```

3.3. Route Path Selection with Link Quality and Critical Switch Aware

RPSO is responsible for routing a flow from source to desire destination based on two conditions: Arrival of new flows and change in network state. This way, upon arrival of new flows Algorithm 1 is called to return the k set of candidate paths (Algorithm 3 lines 1–4). Afterward, the set of P_i that made it to the list is forwarded to the critical switch detection phase. For each P_i in P call Algorithm 2 and get the P_i with $\min(CWF_{score})$ (Algorithm 3

lines 5–7). If P_i CWF_{score} is equal to P_{low} , the set of corresponding rules are installed using the command $F_{install}$, and flow routing will commence immediately (Algorithm 3 lines 8–12). However, when there is more than one P_i with equal CWF_{score} the P_i with minimum distance (number of hop count) is considered as the final P . Similarly, the associated rules are installed $F_{install}$ (Algorithm 3 line 13–16). However, when the network operation state changes because of dense flow arrival or timeout. To avoid frequent expiration of flow rules, an advance timeout setting is can be found at [35]. The affected path is obtained, and the set of old rules are removed and replaced with new rules. This way, data forwarding continues as demonstrated in Algorithm 3 (lines 17–22).

Algorithm 3: RPSO

```

1  If new flow  $f$  arrived, then
2  |   Call algorithm 1
3  |   Get set of  $k$  { $P$ }
4  End
5  For each  $P_i$  in  $P$ 
6  |   Call algorithm 2
7  |   Get  $min$  ( $P_i$  CWFscore)
8  |   If  $P_i == P_{low}$  then
9  |   |    $F_{install}$ 
10 |   |   Route  $f$ 
11 |   End
12 End
13   If  $P_i < T$  &  $P_i > 1$ , then
14   |   Select  $min$  ( $P_{distance}$ )
15   |    $F_{install}$ 
16   |   Route  $f$ 
17   end
18 If network state change, then
19 |   Get  $P_i$  with expire rules
20 |   For each  $P_i$ 
21 |   |    $F_{removed}$  R: R e old rules
22 |   |    $F_{install}$  new R
23 |   End
24 end

```

4. Experimental Results and Performance Evaluation

The simulation is run on a machine with 3.60 GHz and 16 GB RAM equipped with a Ryu SDN controller, and Mininet version 2.2.2. The study considered two (2) real network topologies: ATT (25 nodes, 58 edges) and Spain (30 nodes, 110 edges). These topologies are created by writing a script in the Mininet emulator. Each switch is deployed using a software switch (OpenVswitch) in Mininet running on virtual machines. Typically, the OpenFlow switch is constrained with limited capacity and OpenVswitch can support many entries. The study modifies OpenVswitch to support the limited as in real OpenFlow switch. The study also assumes traffic in ATT and Spain network topologies follows the Poison (interval, packet size) distribution. The D-ITG Srivastava et al. [36] utility is used

to generate the traffic based on the Poisson model. Random traffic flows demand pairs correspond to a source and destination in the network were generated in Mininet.

The simulation results were reported, and the performance of the proposed scheme is evaluated against its counterpart benchmarking work under different scenarios. The results of the proposed solution are compared based on the following metrics: path stretch, path setup time, Throughput, packet delivery ratio, and flow table occupancy. Its effectiveness and improvement were also discussed and analyzed in the following subsections.

4.1. Average Path Setup Latency

Path setup time defined the installation time to set up the full path to route flows from source to destination. The time of path installation increased as the path length augment and decreased when the path is short. It can be observed initially from Figure 2, FSL is having a slightly higher time compared to RPSO while RPF time is high. Since the first ping usually takes a longer time than the subsequent ones because of the need to consult the controller to compute the first path of each and upload the corresponding switch flow table to guide the routing process. The study focuses on the first path setup involving a set of critical switches. In other words, higher time was observed on the set of such switches because of the roles they played. As such, RPSO has reduced the setup time by 73% and 93% compared to FSL and RPF, respectively. This implies that RPSO results in a lower number in path installation time, and the merit is attributed to the better load-balancing along with the switches on the path. Because faster installation flow rules are proportional to the number of rules in the set of switches. The benchmarking works FSL and RPF overlook critical switches and treat all switches equally, which is not always the case.

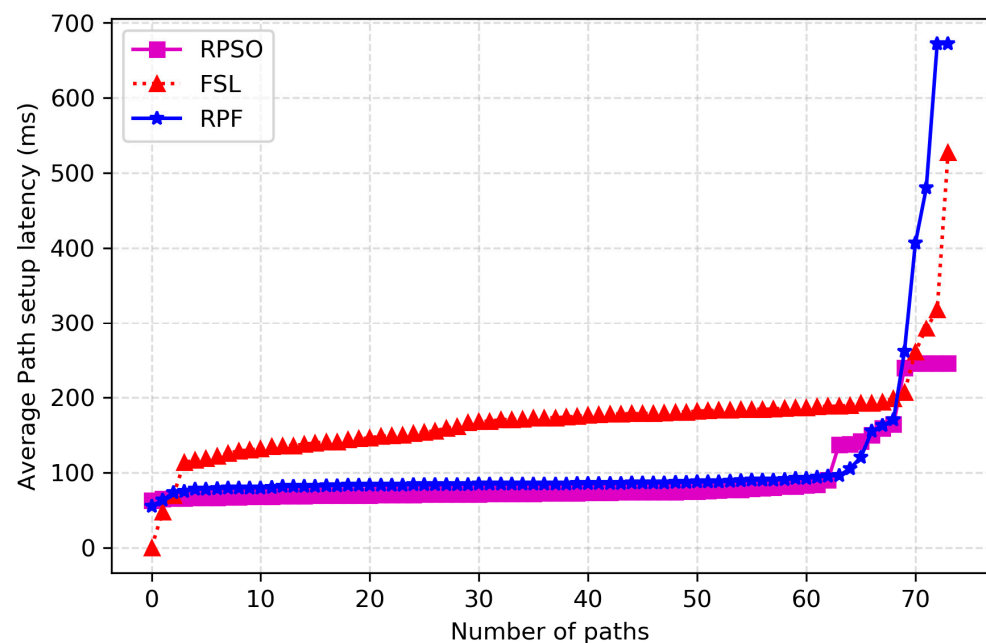


Figure 2. Average Path setup latency.

4.2. Throughput

Throughput defined the average successfully data packets received at the destination through communication link over a period of second. The throughput is measure after a change in network events such as the increase in flow arrival or failure. Figure 3 demonstrates a performance comparison between the proposed solution RPSO and counterpart RRT and FRT. RPSO achieves 29.54% and 55.73% improved throughput performance as compared to FSL and RPF respectively. The merit is attributed to the adopted methodology (link quality, minimal critical switches) of the RPSO scheme makes a more informed decision regarding path selection. Tilwari et al. [37], stress the need for reliable link quality for

data transmission. Poor link quality affects data transmission and declines in throughput. Therefore, RPSO showed better link quality estimation and thereby ensuring a high number of critical data packets reach the intended destination. This achievement is consistent even when the traffic flows arrival increased. However, the counterpart shown declined in the throughput, when traffic flow arrival increases with respect to time, a change of behavior was observed which in turn reduced the throughput performance on FSL and RPF.

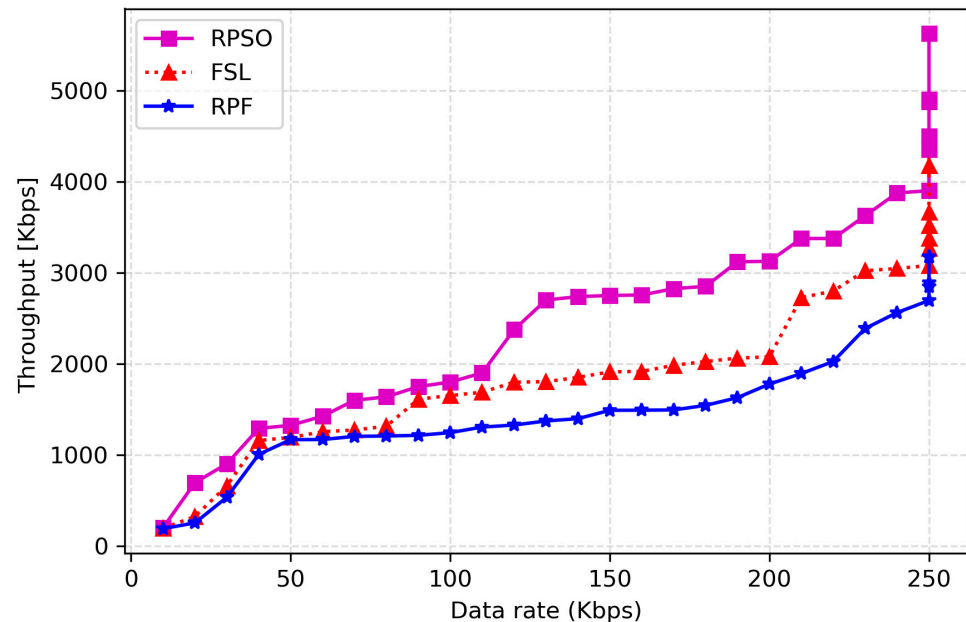


Figure 3. Average Throughput.

4.3. Packet Delivery Ratio

PDR refers to the ratio of successfully received packets to the total transmitted averaged over all flows. This plays a key role in evaluating the efficiency of the transmission deployed in the network. Because PDR increases when there is a higher number of packets dropped because of a change of network event such as link failure. If the selected link has poor transmission quality the packets drop increased. Therefore, PDR is measure with respect to a different time of data transmission and the resulting plot is presented compared to RPF and FSL in Figure 4. Increases in PDR indicate the improved performance of the system, and RPSO exhibits an increase with stable behavior thereby achieving 100%. The performance gain is due to accurate link quality estimation and rerouting of packets through a good link quality channel. Therefore, packets transmission may not be affected even when there is a change in the network. While the RPF shows a 90% PDR due to rerouting of the packet without considering link quality. Therefore, RPSO outperforms RPF with better PDR by 12.5%. The comparative study demonstrates that the proposed scheme is more efficient in terms of the analyzed metric.

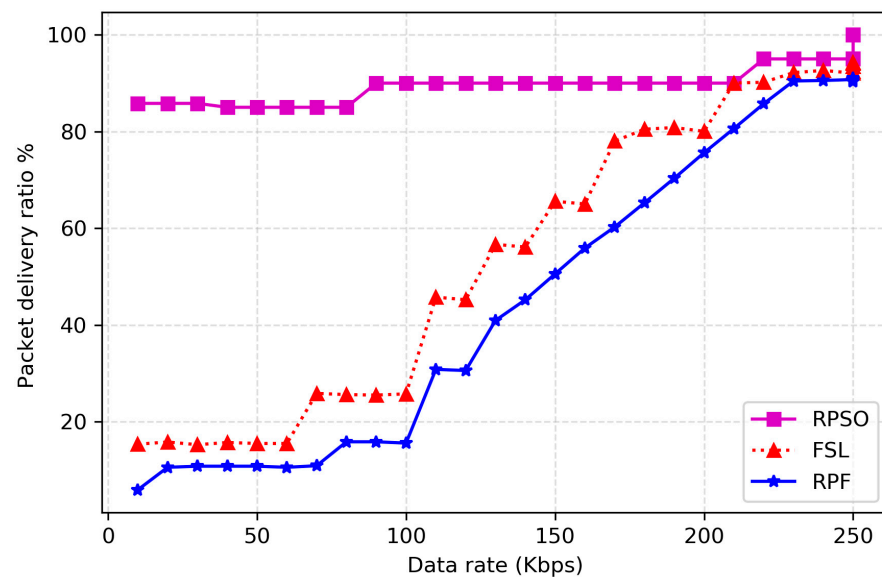


Figure 4. Average Packet Delivery Ratio.

4.4. Path Stretch

Route path stretch is defined as the differences between the length of the longest path used for transmitted flows to the length of the best possible shortest path for those flows. Figure 5 shows the average route path stretch (RPS) for ATT network topology. RPSO outperforms FLS and RPF for all the number of flows. The curve shows that it can lead to a route path up to 37% smaller than the two-benchmarking works FLS, and RPF for 67 flows, respectively. To further evaluate the performance of the proposed solution, the size of the network was increased using different topologies. As expected, a similar trend was observed in Spain's topology as shown in Figure 6, where RPSO performs better than its counterpart by reducing path stretch 12% and 16% compared to FSL, and RPF, respectively. Intuitively, this implies that the computed RPS reflects one of the design objectives of RPSO to select a path with minimum critical switches and length. Conversely, the benchmarking works overlook RPS and focus on other features such as shared edges and path latency only, respectively. Therefore, the proposed solution has a better performance compared to the existing solutions.

4.5. Flow Table Occupancy Rate

Flow table occupancy rate refers to the average number of installed forwarding rules in the switch flowtable to guide the routing process. The RPSO showed a small occupancy rate in the first study with ATT topology as shown in Figure 7. In this regard, FSL has a higher number of rules because of the unoptimized route which results in a longer path, and a longer path led to a large number of rules installation [33]. RPF showed a better occupancy rate than FSL. This is because RPF enjoyed share links which contribute to the lower rule's installation. However, the prior study by Malik and Fréin [9] indicates that the network size contributes to many sets of paths which in turn increase rule consumption. Toward, this goal, the performance of the RPSO was also tested on a topology (Spain) with relatively high density, average node degree 4, as it tends to have large rules consumption. This way the traffic generation was increased which in turn required corresponding routing rules in the flow table. Interestingly, RPSO is always lower while a higher number of rules can be seen in the RPF and FSL. This will not only save more space but can also speed up flow rules updating time and routing convergence time Lei et al. [38]. Therefore, this demonstrates that RPSO reduces the number of additional rules by 99% against RPF and FSL, respectively.

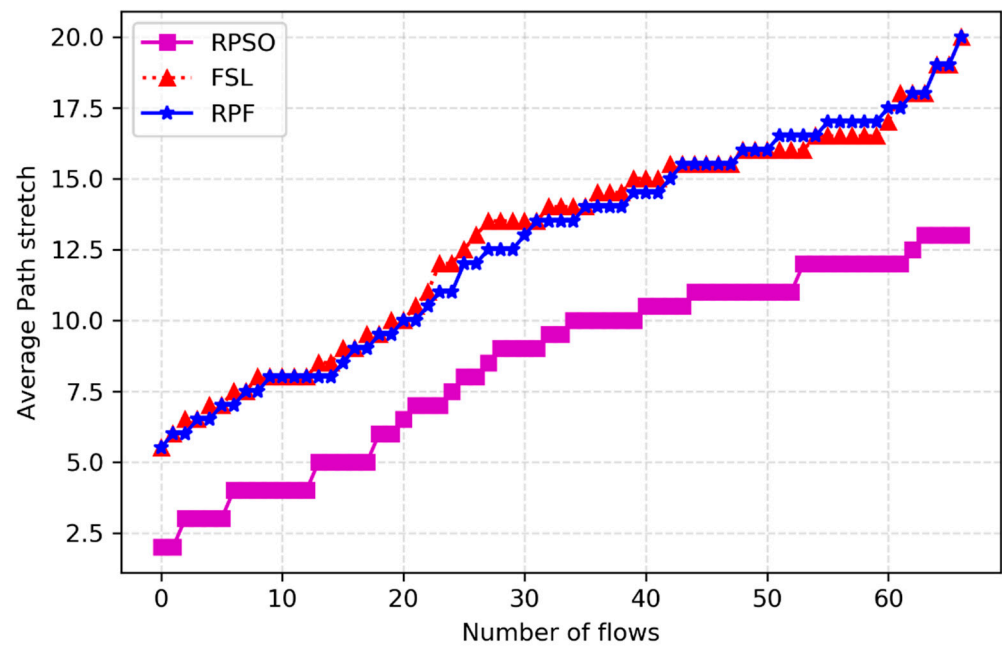


Figure 5. Average Path stretch using ATT topology.

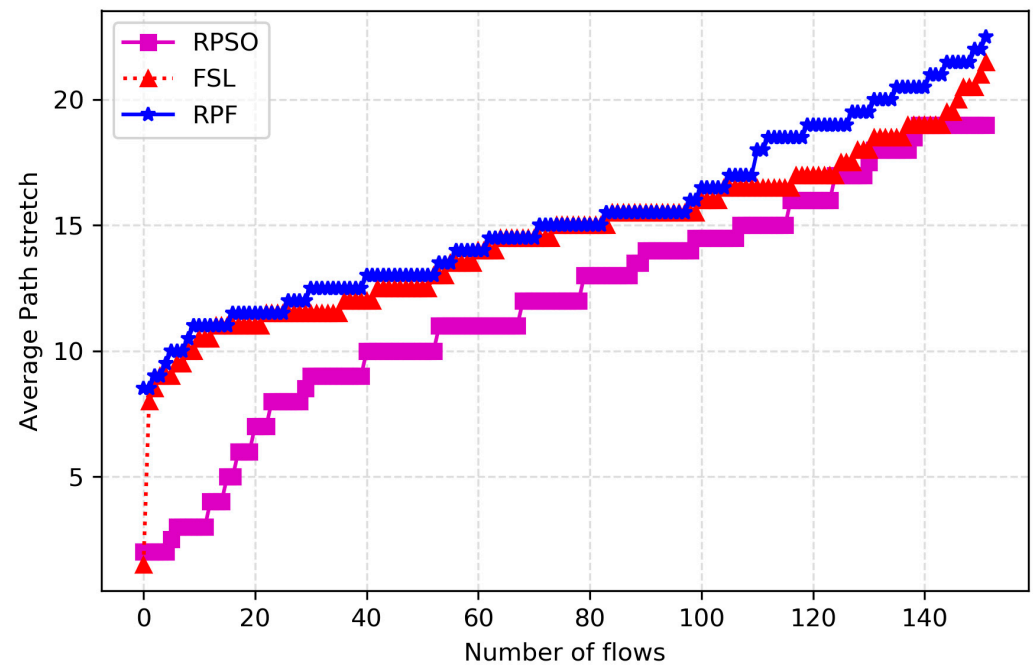


Figure 6. Average Path stretch using Spain topology.

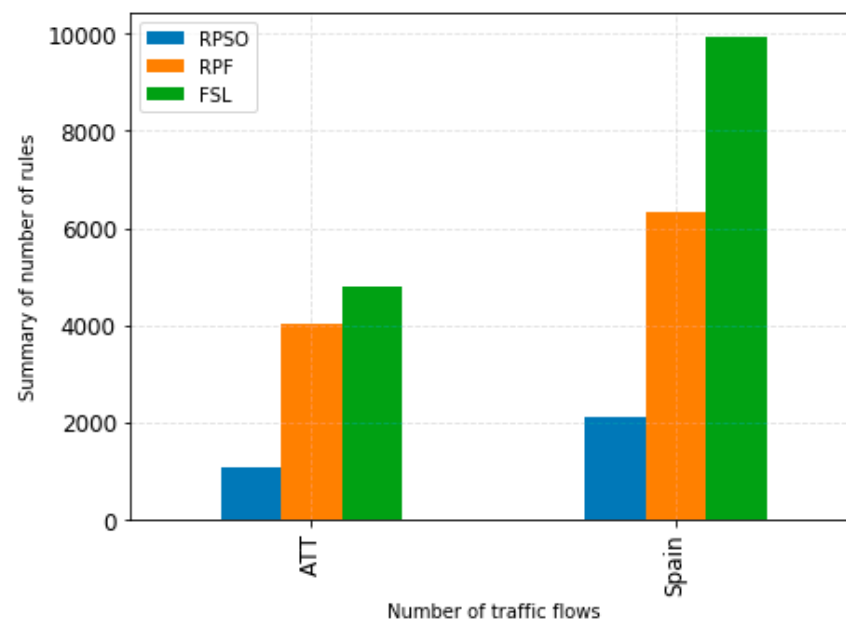


Figure 7. Comparison of the number of rules between ATT and Spain topology.

5. Conclusions

This paper proposed route path selection with Link quality and critical switch awareness while improving forwarding performance. Path selection processes are introduced and route traffic flows through better quality links. Link quality estimation has been introduced and minimal Critical switch-based routing has been devised. Critical switch frequency score (CWFscore) metric which encourages path selection with better link quality and the minimal number of critical switches. Based on the experimental result, the proposed solution has reduced path stretch by 37%, path setup latency by 73% thereby improving throughput by 55.73%, and packet delivery ratio by 12.5% compared to the baseline work. In the future, RPSO will incorporate congestion management for link failure recovery at the SDN data plane. Moreover, it would be interesting to devise an intelligent flow-based end-to-end routing to classify flows based on their size and prioritize them. This way, higher priority flows shall be rerouted through an optimized path with multi-Quality of Service parameter based on a protection approach. Lower priority flows take the shortest path based on a restoration approach while minimizing the time and space gap between restoration and protection.

Author Contributions: Conceptualization, B.I. and F.A.G.; methodology, B.I. and K.A.B.; writing—original draft preparation, B.I.; writing—review and editing, M.S.M.Z., K.A.B. and F.A.G.; supervision, E.H.A. and F.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Taif University Researchers Supporting Project number (TURSP-2020/292) Taif University, Taif, Saudi Arabia.

Data Availability Statement: Not Applicable.

Acknowledgments: The Authors would like to acknowledge Taif University Researchers Supporting Project number (TURSP-2020/292) Taif University, Taif, Saudi Arabia.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Isyaku, B.; Soperi, M.; Zahid, M.; Kamat, B.M. Software Defined Networking Flow Table Management of OpenFlow Switches Performance and Security Challenges: A Survey. *Future Internet* **2020**, *12*, 147. [[CrossRef](#)]
2. Alsaeedi, M.; Mohamad, M.M.; Al-roubaiey, A.A. Toward Adaptive and Scalable OpenFlow-SDN Flow Control: A Survey. *IEEE Access* **2019**, *7*, 107346–107379. [[CrossRef](#)]
3. Kreutz, D.; Ramos, F.M.V.; Verissimo, P.; Rothenberg, C.E.; Azodolmolky, S.; Member, S. Uhlig, Software-Defined Networking: A Comprehensive Survey. *Proc. IEEE* **2014**, *103*, 14–76. [[CrossRef](#)]
4. Bianchi, G.; Bonola, M.; Capone, A. Cascone, OpenState: Programming Platform-independent Stateful OpenFlow Applications Inside the Switch. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 44–51. [[CrossRef](#)]
5. Malik, A.; Aziz, B.; Al-haj, A.; Adda, M. Software-Defined Networks: A Walkthrough Fault Tolerance. *PeerJ Preprints* **2019**, e27624v1.
6. *SDN Architecture Overview*; Onf.: Palo Alto, CA, USA, 2013; pp. 1–5.
7. Wazirali, R.; Ahmad, R. SDN-OpenFlow Topology Discovery: An Overview of Performance Issues. *Appl. Sci.* **2021**, *11*, 6999. [[CrossRef](#)]
8. Qian, S.; Zhang, Q.; Tizghadam, A.; Park, B.; Bannazadeh, R.; Boutaba, A. Leon-garcia, TCAM space-efficient routing in a software defined network. *Comput. Netw.* **2017**, *125*, 26–40. [[CrossRef](#)]
9. Malik, A.; de Fréin, R. Rapid Restoration Techniques for Software-Defined Networks. *Appl. Sci.* **2020**, *10*, 3411. [[CrossRef](#)]
10. Rangkutiy, M.F.; Al-hooti, M.H.A. Path Selection in Software Defined Network Data Plane using Least Loaded Path. In Proceedings of the 2020 International Conference on Advanced Computer Science and Information Systems (ICACSIS), Depok, Indonesia, 17–18 October 2020; pp. 135–140.
11. Li, R.; Wang, X. A Tale of Two (Flow) Tables: Demystifying Rule Caching in OpenFlow Switches. In Proceedings of the 48th International Conference on Parallel Processing, Kyoto, Japan, 5–6 August 2019.
12. Malik, A.; Aziz, B.; Bader-el-den, M. Finding Most Reliable Paths For Software Defined Networks. In Proceedings of the 13th International Wireless Communications and Mobile Computing Conference, Valencia, Spain, 26–30 June 2017; pp. 1309–1314.
13. Qiu, K.; Member, S.; Yuan, J.; Zhao, J.; Wang, X.; Secci, S.; Member, S.; Fu, X.; Member, S. FastRule: Efficient Flow Entry Updates for TCAM-Based OpenFlow Switches. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 484–498. [[CrossRef](#)]
14. Khalili, R.; Despotovic, Z.; Hecker, A. Flow Setup Latency in SDN Networks. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 2631–2639. [[CrossRef](#)]
15. Hu, C.; Hsu, C.; Khuukhenbaatar, S.; Dashdorj, Y.; Dong, Y. Path Selection with Joint Latency and Packet Loss for Edge Computing in SDN. In Proceedings of the 20th Asia-Pacific Network Operations and Management Symposium, Matsue, Japan, 18–20 September 2019; pp. 2019–2022.
16. Chooprateep, A. Video Path Selection for Traffic Engineering in SDN. In Proceedings of the 11th International Conference on Information Technology and Electrical Engineering (ICITEE), Pattaya, Thailand, 10–11 October 2019; pp. 1–6.
17. Stringer, J.P.; Fu, Q.; Lorier, C.; Nelson, R.; Rothenberg, C.E. Cardigan: Deploying a distributed routing fabric. In Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, Hong Kong, China, 16 August 2013; pp. 169–170. [[CrossRef](#)]
18. Nascimento, M.R.; Rothenberg, C.E.; Salvador, M.R.; Magalhães, M.F. QuagFlow: Partnering Quagga with OpenFlow. *Comput. Commun. Rev.* **2010**, *40*, 441–442. [[CrossRef](#)]
19. Rothenberg, C.E.; Nascimento, M.R.; Salvador, M.R.; Corrêa, C.N.A.; de Lucena, S.C.; Raszuk, R. Revisiting routing control platforms with the eyes and muscles of software-defined networking. In Proceedings of the 1st Workshop on Hot Topics in Software Defined Networks, Helsinki, Finland, 13 August 2012; pp. 13–18. [[CrossRef](#)]
20. Sharma, S.; Staessens, D.; Colle, D.; Pickavet, M.; Demeester, P. Automatic configuration of routing control platforms in OpenFlow networks. *Comput. Commun. Rev.* **2013**, *43*, 491–492. [[CrossRef](#)]
21. Sharma, S.; Staessens, D.; Colle, D.; Palma, D.; Goncalves, J.; Figueiredo, R.; Morris, D.; Pickavet, M.; Demeester, P. Implementing quality of service for the software defined networking enabled future internet. In Proceedings of the 2014 3rd European Workshop on Software Defined Networks, EWSDN, Budapest, Hungary, 1–3 September 2014; pp. 49–54. [[CrossRef](#)]
22. Sharma, S.; Staessens, D.; Colle, D.; Pickavet, M.; Demeester, P. OpenFlow: Meeting carrier-grade recovery requirements OpenFlow: Meeting carrier-grade recovery requirements. *Comput. Commun.* **2013**, *36*, 656–665. [[CrossRef](#)]
23. Cheng, Z.; Zhang, X.; Li, Y.; Yu, S.; Lin, R.; He, L. Congestion-Aware Local Reroute for Fast Failure Recovery in Software-Defined Networks. *J. Opt. Commun. Netw.* **2017**, *9*, 934. [[CrossRef](#)]
24. Astaneh, S.A.; Heydari, S.S. Optimization of SDN Flow Operations in Multi-Failure Restoration Scenarios. *IEEE Trans. Netw. Serv. Manag.* **2016**, *13*, 421–432. [[CrossRef](#)]
25. Astaneh, S.; Heydari, S.S. Multi-Failure Restoration with Minimal Flow Operations in Software Defined Networks. In Proceedings of the 2015 11th International Conference on the Design of Reliable Communication Networks (DRCN), Kansas City, MO, USA, 24–27 March 2015; pp. 263–266.
26. Yu, C.; Zhao, Z.; Zhou, Y.; Zhang, H. Intelligent Optimizing Scheme for Load Balancing in Software Defined Networks. In Proceedings of the 2017 IEEE 85th Vehicular Technology Conference (VTC Spring), Sydney, NSW, Australia, 4–7 June 2017.
27. Alnajim, A.; Salehi, S. Incremental Path-Selection and Scheduling for Time-Sensitive Networks. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 4–9.

28. Gao, K.; Xu, C.; Qin, J.; Yang, S.; Zhong, L.; Muntean, G. QoS-driven Path Selection for MPTCP: A Scalable SDN-assisted Approach. In Proceedings of the 2019 IEEE Wireless Communications and Networking Conference (WCNC), Marrakesh, Morocco, 15–18 April 2019; pp. 1–6.
29. Saha, N.; Bera, S.; Misra, S. Sway: Traffic-Aware QoS Routing in Software-Defined IoT. *IEEE Trans. Emerg. Top. Comput.* **2021**, *9*, 390–401.
30. Ravuri, H.K.; Da, B.; Clemm, A.; Wauters, T.; de Turck, F. An Experimental Evaluation of Flow Setup Latency in Distributed Software Defined Networks. In Proceedings of the 2019 IEEE Conference on Network Softwarization (NetSoft), Paris, France, 24–28 June 2019; pp. 432–437.
31. Gotani, K.; Takahira, H.; Hata, M.; Guillen, L.; Izumi, S. Design of an SDN Control Method Considering the Path Switching Time under Disaster Situations. In Proceedings of the 5th International Conference on Information and Communication Technologies for Disaster Management, Sendai, Japan, 4–7 December 2018; pp. 18–21.
32. Gotani, K.; Takahira, H.; Hata, M.; Guillen, L.; Izumi, S.; Abe, T. Overview, OpenFlow Based Information Flow Control Considering Route Switching Cost. In Proceedings of the 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Milwaukee, WI, USA, 15–19 July 2019; pp. 2019–2022. [[CrossRef](#)]
33. Perner, C.; Carle, G. Comparison of Optimization Goals for Resilient Routing. In Proceedings of the 17th IEEE International Conference on Communications Workshops, ICC Workshops 2019, Shanghai, China, 20–24 May 2019; pp. 1–6.
34. Malik, A.L.I.; Aziz, B.; Ke, C.; Liu, H.A.N.; Adda, M.O. Virtual Topology Partitioning Towards an Efficient Failure Recovery of Software Defined Networks. In Proceedings of the 16th International Conference on Machine Learning and Cybernetics, ICMLC 2017, Ningbo, China, 9–12 July 2017.
35. Isyaku, B.; Bakar, K.A.; Soperi, M.; Zahid, M. Adaptive and Hybrid Idle—Hard Timeout Allocation and Flow Eviction Mechanism Considering Traffic Characteristics. *Electronics* **2020**, *9*, 1983. [[CrossRef](#)]
36. Srivastava, S.; Anmulwar, S.; Sapkal, A.M. Comparative Study of Various Traffic Generator Tools. In Proceedings of the 2014 Recent Advances in Engineering and Computational Sciences (RAECS), Chandigarh, India, 6–8 March 2014; pp. 6–8.
37. Algorithm, Q.; Tilwari, V.; Dimiyati, K.; Hindia, M.H.D.N.; Fattouh, A. Mobility, Residual Energy, and Link Quality Aware Multipath Routing in MANETs with Q-learning algorithm. *Appl. Sci.* **2019**, *9*, 1582.
38. Lei, Y.-C.; Wang, K.; Hsu, Y.-H. Multipath Routing in SDN-Based Data Center Networks. In Proceedings of the 2015 European Conference on Networks and Communications (EuCNC), Paris, France, 29 June–2 July 2015; pp. 365–369.