electronics

Article

Dynamic Routing Using Fuzzy Logic for URLLC in 5G Networks Based on Software-Defined Networking

Yan-Jing Wu [1], Menq-Chyun Chen [2], Wen-Shyang Hwang [2,*] and Ming-Hua Cheng [3]

[1] Department of Information Technology and Communication, Shih Chien University, Kaohsiung Campus, Kaohsiung 84550, Taiwan; yanjing@g2.usc.edu.tw
[2] Department of Electrical Engineering, National Kaohsiung University of Science and Technology, Kaohsiung 807618, Taiwan; f110154128@nkust.edu.tw
[3] Department of Digital Media Design, Tzu-Hui Institute of Technology, Pingtung 926001, Taiwan; cmha6777@mail.tzuhui.edu.tw
* Correspondence: wshwang@nkust.edu.tw

Abstract: Software-defined networking (SDN) is an emerging networking technology with a central point, called the controller, on the control plane. This controller communicates with the application and data planes. In fifth-generation (5G) mobile wireless networks and beyond, specific levels of service quality are defined for different traffic types. Ultra-reliable low-latency communication (URLLC) is one of the key services in 5G. This paper presents a fuzzy logic (FL)-based dynamic routing (FLDR) mechanism with congestion avoidance for URLLC on SDN-based 5G networks. By periodically monitoring the network status and making forwarding decisions on the basis of fuzzy inference rules, the FLDR mechanism not only can reroute in real time, but also can cope with network status uncertainty owing to FL's fault tolerance capabilities. Three input parameters, normalized throughput, packet delay, and link utilization, were employed as crisp inputs to the FL control system because they had a more accurate correlation with the network performance measures we studied. The crisp output of the FL control system, i.e., path weight, and a predefined threshold of packet loss ratio on a path were applied to make routing decisions. We evaluated the performance of the proposed FLDR mechanism on the Mininet simulator by installing three additional modules, topology discovery, monitoring, and rerouting with FL, on the traditional control plane of SDN. The superiority of the proposed FLDR over the other existing FL-based routing schemes was demonstrated using three performance measures, system throughput, packet loss rate, and packet delay versus traffic load in the system.

Keywords: software-defined networking; URLLC; fuzzy logic; dynamic routing; monitoring period

Citation: Wu, Y.-J.; Chen, M.-C.; Hwang, W.-S.; Cheng, M.-H. Dynamic Routing Using Fuzzy Logic for URLLC in 5G Networks Based on Software-Defined Networking. *Electronics* **2024**, *13*, 3694. https://doi.org/10.3390/electronics13183694

Academic Editors: Javid Taheri, Ali Khoshkholghi and Andreas Johnsson

Received: 10 August 2024
Revised: 12 September 2024
Accepted: 16 September 2024
Published: 18 September 2024

1. Introduction

According to "Global Software-Defined Networking Market—Industry Trends and Forecast to 2028" that was published by Data Bridge Market Research in 2021 [1], the software-defined networking (SDN) market is expected to be valued at USD 112.95 billion by 2028 and to grow sustainably by a compound rate of approximately 20% from 2021 to 2028. SDN technology has become popular as an investment target for communication service providers because of its ability to flexibly and automatically manage internet infrastructure. It gives corporations more flexible network control capabilities. Furthermore, it has been used in various fields, such as healthcare, education, banking, and government. The adoption of network automation and virtualization is playing a key role in the growth of the SDN market. This growth is being boosted by declines in capital and operation expenditure, an increasing demand for cloud services, data-centre integration and server virtualization, and enterprise demand for mobility to increase their field service productivity. The rising profit being created by SDN is another major reason for the growth of the SDN market.

On the other hand, telecommunication operators worldwide have begun to deploy fifth-generation (5G) and beyond communication systems, which are revolutionary for mobile communications. Compared with fourth-generation (4G) communication technology, 5G and beyond systems are expected to enable communications with greater bandwidth, lower end-to-end latency, and more flexible and reliable internet connectivity [2]. For example, an ultra-reliable low-latency communication (URLLC) service is being implemented in industrial and Internet of Things applications, which require extremely low latency and high reliability. Such applications include autopilot systems, smart grids, virtual reality, and factory automation [3]. The development of 5G and beyond has resulted in an exponential increase in the volume of data flows due to network users employing numerous applications and various service features, and a higher volume of data flows means that network management and orchestration tasks are more complex. The automation of 5G network management can be supported by SDN, which provides operational intelligence through the separation of network control functions from data-plane devices. SDN and network function virtualization [4] are the concepts of network software and virtualization, which focus on solving and reconfiguring network complexity and enabling the efficient sharing of resources.

As shown in Figure 1, the SDN architecture consists of an application plane, a control plane, and a data plane. SDN is a new example of decoupling network management and forwarding. Developers can write various applications and specify certain rules on the application plane. For instance, the application plane can perform load balancing and monitoring while passing the load balance information to the control plane via the northbound application programming interface (API), and allows the controller to issue instructions. The control plane is the core of the SDN architecture. On the control plane, the controller provides centralized control by managing all forwarding rules and the network infrastructure is responsible only for forwarding. Through protocols such as OpenFlow [5], NetConf [6], and OVSDB [7], developers can communicate with the data plane via the southbound API. One of the challenges in SDN is traffic status and routing updates on the data plane. Because of changes in data flows, the diversion of traffic load, and node and link malfunctions, as well as other issues, the forwarding rules must be repeatedly amended on the data plane. How the routing update procedure can be improved remains an open research issue. Generally, a routing update can be divided into two stages: flow selection and flow table update. The aim of the flow selection stage is to select a group of flows that need to be rerouted. Then, in the flow table update stage, the selected flows are forwarded to the update path. Because the time required to make a routing update influences the scalability and flexibility of network control, it is considered a crucial factor affecting network performance. Conventionally, in SDN, the routing mechanism employed is Dijkstra's algorithm [8], which decides the shortest path on the basis of link cost. Routing mechanisms can be classified into static and dynamic mechanisms. The main difference between them is that when congestion or quality of service (QoS) degradation occurs, the paths are reconfigured in dynamic routing but not in static routing mechanisms. However, both types of mechanism have difficulty meeting the low-latency and low-error-rate demands of URLLC. When the paths cannot be reconfigured and link congestion occurs, packet loss or delay may occur. Static routing is clearly unable to satisfy the URLLC quality requirement of low latency and low error rate. Regarding dynamic routing, its suitability is more complicated because an optimal setting value for the monitoring period has not been defined. High-frequency monitoring means that the controller communicates extensively with the data plane. Although the network status can be obtained rapidly and managed, extensive communication results in a massive burden for the control channel. Conversely, when monitoring cycles are long, flow tables cannot adapt to a rapidly changing network status. After receiving the status information, the controller must analyze and reconfigure the paths and return new forwarding rules to the data plane if necessary. However, congestion still occurs before the controller completes these tasks. Although the degree of packet loss and packet delay is better in dynamic

routing than in static routing, the abilities of dynamic routing to satisfy the high-reliability and low-latency demands of URLLC remain questionable.

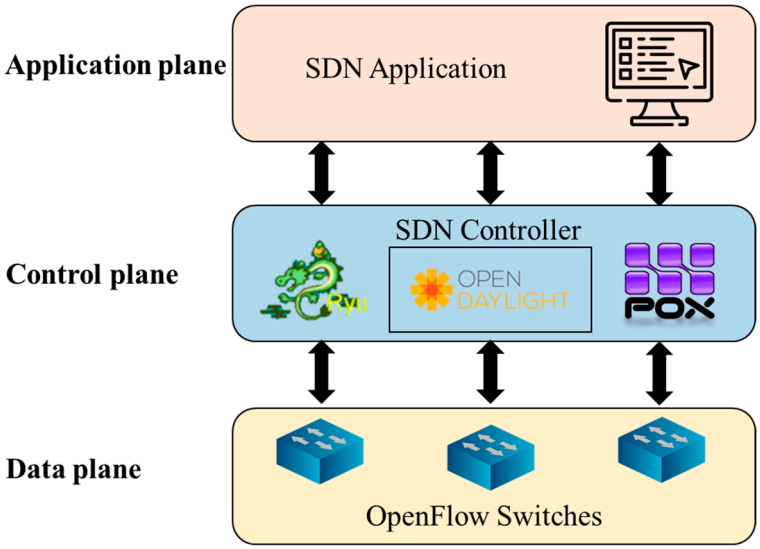

Figure 1. SDN architecture.

Many dynamic routing mechanisms with emphasis on network flexibility and intelligence have been proposed [9]. The evolution of network-related technologies has been aided by the maturity of machine learning (ML), which has ushered in many revolutionary changes. The application of ML has provided new possibilities for improving routing mechanisms. Favourable routing strategies have been developed by using ML algorithms to learn from network conditions and changes in SDN. ML-based algorithms need to train a large volume of network status data [10–16], while fuzzy logic (FL) does not require a large amount of advance data training and labelling and is better at coping with uncertainty and ambiguity in network status. Several researchers have employed FL to design dynamic routing schemes for SDN [17–20]. FL can take various network performance measures (e.g., throughput, delay, packet loss rate, and link usage) into account, and it enables the development of real-time routing strategies based on fuzzy inference rules. Therefore, FL has been widely used in dynamic routing optimization. This paper proposes an FL-based dynamic routing (FLDR) mechanism for URLLC with congestion avoidance in SDN-based 5G networks. The proposed FLDR mechanism uses the network status data that the control plane regularly receives from the data plane and employs an FL control system to select a path on the basis of fuzzy inference rules and a predefined threshold of packet loss ratio for URLLC flows. Three input parameters, normalized throughput, packet delay, and link utilization are employed as crisp inputs to the FL control system because they have a more accurate correlation with the network performance measures we are interested in. The crisp output of the FL control system, i.e., path weight, and a predefined threshold of packet loss ratio on a path are applied to make routing decisions. It is demonstrated via simulations that FLDR can update flow tables in a timely manner, effectively enhance the forwarding efficiency of SDN switches, and improve the overall system throughput, packet delay, and packet loss rate.

The remainder of this paper is organized as follows. Section 2 explores the related literature. Section 3 describes the framework and operating procedure of the proposed FLDR scheme. Section 4 analyzes the simulation results. Finally, the concluding remarks are presented in Section 5.

2. Related Works

As summarized in Table 1, static routing is compared with dynamic routing that includes ML-based and FL-based algorithms in terms of algorithm complexity, computational

cost, and response to variation. Although static routing is unable to adapt to a changing network status given various network conditions, load variation, and quality requirements, both the complexity and computational cost are relatively low. Because ML-based algorithms need to train a large volume of network status data, the uncertainty in accuracy and the high cost of computing equipment accompany a large amount of labelled data and a consistent training process in ML. Compared with ML-based routing, using FL is less costly to implement and has more intuitive design rules.

Table 1. Comparison of routing mechanisms.

Characteristics	Static Routing	ML-Based Dynamic Routing	FL-Based Dynamic Routing
Algorithm complexity	Low	High	Moderate
Computational cost	Low	High	Moderate
Response to variation	Slow	Moderate	Fast

In the most recent decade, several dynamic routing mechanisms for SDN have been proposed. Generally, the architecture of SDN with dynamic routing resembles that proposed by Kao et. al. in [21], which comprises a topology detection module, a monitor module, and a reroute module. Through the topology detection module, the controller uses the link layer discovery protocol (LLDP) to explore the topological environment of the data plane and visualize the underlying network topology. The reroute module then uses Dijkstra's algorithm to configure the initial path and employs a dynamic routing algorithm to perform a timely reroute. Finally, the controller uses the monitor module to periodically monitor the status of each switch on the data plane. According to the research, the length of the monitoring period has a significant effect on the overall network performance. A three-second monitoring period is considered sufficient to achieve a relatively favourable balance between control overload and network adaptation. Furthermore, in order to support the QoS demands of high reliability and low latency, when the link utilization of a certain path is greater than 80%, the reroute module is triggered to reconfigure the path. Recently, FL has been used in many fields such as automotive systems, domestic goods, environment control, decision-making support systems, etc. Specifically, FL has been successfully applied at the reroute module for dynamic routing because it can provide relatively reliable outcomes at lower computational cost and greater speeds compared with ML-based approaches. As illustrated in Figure 2, the FL control system [22,23] considers network status data as the crisp inputs to the fuzzifier, and the fuzzifier produces the fuzzy inputs. The fuzzy outputs of the inference engine, which converts the fuzzy inputs according to the fuzzy rules, are then input to the defuzzifier to yield the crisp outputs. Finally, these crisp outputs of the FL control system are used for updating the flow tables to fulfil the QoS of data flows and ensure high overall network performance. The correctness of the FL inference engine depends on the choice of input parameters, membership functions (MFs), fuzzy rules, and rule base size.

Several studies have proposed dynamic routing algorithms involving FL for SDN [24–28]. Two studies [25,26] proposed an FL algorithm for load balancing among servers in SDN. When traffic load continually increases, limited server resources may be an obstacle to high QoS. Therefore, load balancing is essential for the effective management of service requests and the even distribution of traffic load to application servers. Guoyan et. al. exploited the way that control and forwarding are separated in the architecture of SDN [25]. The load state of the virtual server is evaluated through FL. The SDN controller for the whole network, owing to its centralized control capability, is then used to monitor virtual server information in real time and schedule virtual server tasks. The scheme proposed in [26] considers server status data, including CPU usage, memory usage, response time, and throughput. Through a series of simulation tests, the importance of load balancing to network performance was demonstrated, especially when handling an increasing amount of traffic. This study can be

used as a reference by SDN researchers and practitioners on how network performance can be optimized by considering load balancing in SDN.

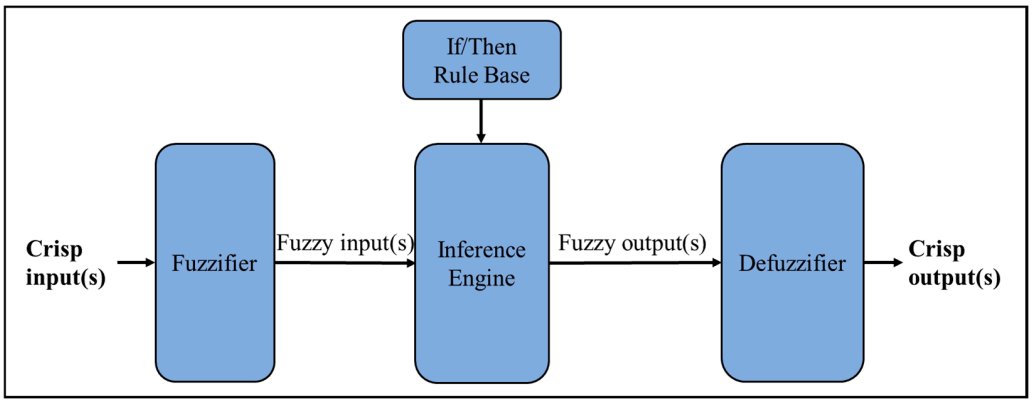

Figure 2. Mamdani's fuzzy logic control system.

Moravejosharieh et. al. proposed an FL control system in which packet loss rate and link delay are employed as input parameters to the FL control system to provide higher QoS for various traffic flows in SDN [27]. In their FL-enabled SDN (FLE-SDN), all communication paths between source and destination nodes are constantly collected for the calculation of path weights. The most suitable path for a specific service flow is selected in accordance with the respective QoS requirement. That is, a service flow with a lower QoS requirement is redirected to the path with a smaller weight so that flows with higher QoS requirements are more likely to be satisfied. The FLE-SDN controller can redirect a service flow to a path with a higher weight when the required QoS for that service has not been met. Although the packet loss rate is an important network performance metric, it only represents the number of lost packets but not the throughput of a data flow. In a heavy load environment, the packet loss rate for a high-rate data flow may increase, but this does not mean that the overall throughput has deteriorated. In a data-centre network, the network is usually constantly expanding. The characteristics of traffic in the network become complex and difficult to manage, with network congestion therefore being likely to occur. To solve the problem of low network throughput caused by congestion, Xu et. al. proposed an SDN-based traffic scheduling algorithm that exploits FL (named the FL-SDN algorithm) [28]. In FL-SDN, link utilization and hop count are considered as input parameters for the FL control system, the candidate path set of accessible paths between source and destination hosts is calculated, and the path with the highest comprehensive score is then found and employed in the scheduling of traffic. The number of hops represents the number of switches that a path passes through. However, paths with the same hop count but completely different performance may exist. For instance, a path with a smaller hop count may have a longer delay.

Excluding the packet loss rate and hop count as mentioned above, we consider the packet delay, link utilization, and normalized throughput of each candidate path as input parameters for the FL control system in our proposed FLDR mechanism for URLLC on SDN-based 5G networks. FLDR considers packet delay instead of link delay because the former is more accurate to evaluate the delay of data packets than the latter. The main contribution of this paper is that the proposed FLDR scheme considers one more input parameter, normalized throughput, and a more accurate input parameter, packet delay, for the FL control system for dynamic routing. In addition, FLDR introduces a predefined threshold of packet loss ratio to support the QoS requirement of URLLC flows. As a result, FLDR not only can effectively support the quality requirement of URLLC, but also can improve the overall network performance, such as system throughput, packet delay, and packet loss rate, for comparison with the other existing FL-based dynamic routing schemes.

3. Proposed FLDR

3.1. FLDR Framework

As illustrated in Figure 3, a dynamic routing mechanism using FL is introduced into the reroute module on the control plane to update the forwarding rules in response to new traffic requirements or changes by referring to the system architecture of dynamic routing in SDN [21]. FLDR ensures the timeliness and accuracy of network forwarding while preventing the unnecessary wastage of network resources and network congestion. In principle, the FLDR mechanism is a method for dynamically updating flow tables in SDN. To enhance the network performance and flexibility, forwarding rules are dynamically added or modified in accordance with actual network status. The FLDR mechanism is divided into two parts—network resource collection and FL control.

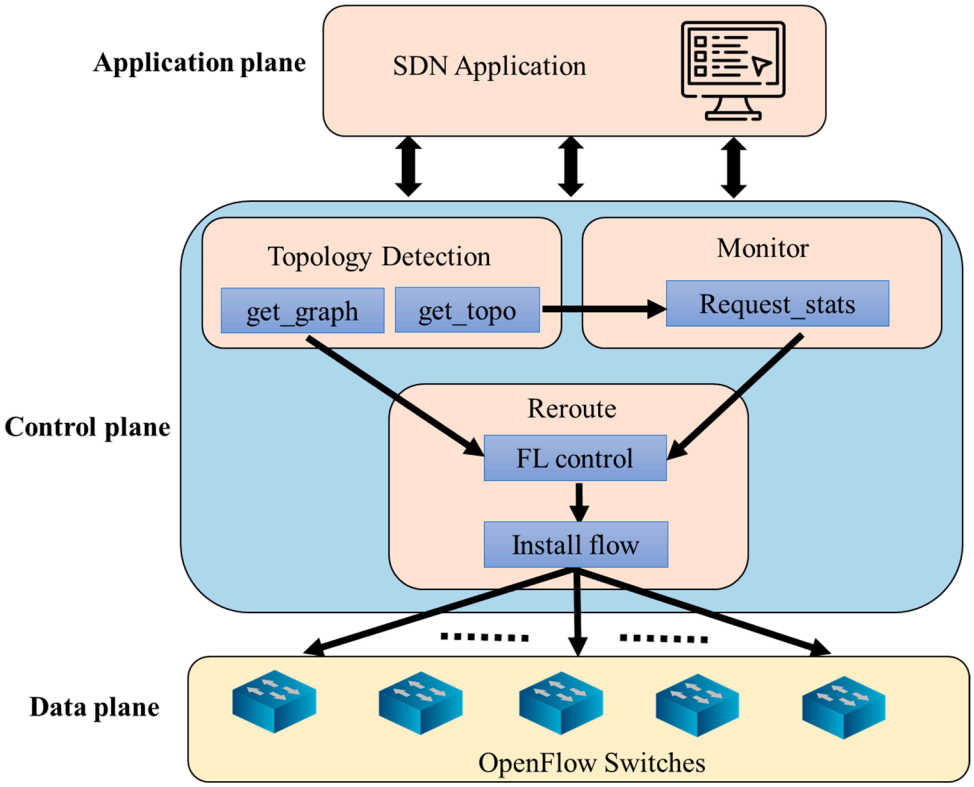

Figure 3. FLDR framework.

3.1.1. Network Resource Collection

Network resource collection involves collecting state information about each node and link in the network topology to enable subsequent path selection and decision making. The control plane sends instructions to the switches on the data plane through the OpenFlow protocol to monitor the network status and manage the entire network. The following are network resource data:

(a) Node load: the load of every switch, including CPU usage, memory usage, and port utilization.
(b) Network traffic: information about network traffic, such as traffic size, traffic type, and traffic throughput.
(c) Network topology: information about network topology, including connections between nodes and the bandwidth of each connection.
(d) Error and congestion information: information about errors and congestion in the network, such as packet loss rate and delay.

3.1.2. FL Control

The collected data on network resources are delivered to the FL control system, and FL is employed to evaluate the feasibility and performance of different paths and select suitable paths. Referring to Figure 2, which illustrates Mamdani's fuzzy logic control system, the normalized throughput, link utilization, and packet delay in FLDR are input parameters for the fuzzifier. The throughput is the amount of data transmitted along a path per unit time, which is a crucial indicator for routing and resource scheduling. The normalized throughput is the throughput divided by the link bandwidth. The link utilization is the degree of usage of a link in the network. High link utilization can lead to congestion, whereas low link utilization is a waste of the link bandwidth. Both high and low link utilization affect the performance of the entire network, such as load balancing and packet loss. The packet delay is the time needed for a packet to travel from source to destination via a path in the network. Avoiding long delay is important in many applications, especially those that require instant interaction and data transmission, such as URLLC. As to the three input parameters, the normalized throughput and link utilization are scalar, while the packet delay is in milliseconds. The FL control system of FLDR has the following three processing stages:

(a) Fuzzification

The fuzzifier passes each crisp input through a membership function (MF) to generate a corresponding fuzzy input. The MF of a fuzzy set is a generalization of the indicator function in classical sets. In other words, it represents the degree of truth as an extension of valuation. Thus, the y-axis of an MF has no units and varies from zero to one. Each crisp input variable therefore has a corresponding MF that operates independently from all other MFs. The normalized throughput, link utilization, and packet delay are the crisp inputs of FLDR and are denoted as t, u, and d, respectively. The most commonly used MFs are the triangular, trapezoidal, bell, and Gaussian MFs. However, the shape of the MF is not usually as important as the number and positions of the curves. Three to seven curves are usually suitable for transferring all possible input values [22]. As illustrated in Figure 4, in FLDR, the normalized throughput (ranging from 0% to 100%) is divided into three levels, represented by three overlapping trigonometric functions. When t $\leq$ 20, the MF is fully the bad function. When the t is between 20 and 90, the MF is proportionally the bad, mid, and good function simultaneously. And, when t $\geq$ 90, the MF is fully the good function. As illustrated in Figure 5, the MF of the link utilization (also ranging from 0% to 100%) is similar to that of the normalized throughput. When u $\leq$ 20, the MF is fully the low function. When u is between 20 and 80, the MF is proportionally the low, mid, and high function. When u $\geq$ 80, the MF is fully the high function. As illustrated in Figure 6, the MF of the packet delay (in milliseconds) is designed for the low-latency service quality guarantee for URLLC. When d $\leq$ 1, the MF is the low function. Otherwise, it is the high function. The MFs used for the normalized throughput, link utilization, and packet delay are expressed by Equations (1)–(3), respectively. For example, the normalized throughput, link utilization, packet delay for a certain path are 85%, 70%, and 0.5 ms, respectively. The fuzzy control system in FLDR converts them into fuzzy inputs, 0.125mid + 0.25$good$, 0.33mid + 0.25$high$, and 0.5low, respectively, obtained from Equations (1) to (3).

$$f(t) = \begin{cases} \frac{30-t}{30-0} * bad, \ t \leq 20 \\ \frac{\max\{0,\ 30-t\}}{30-0} * bad + \frac{t-20}{50-20} * mid, \ 20 < t < 50 \\ \frac{90-t}{90-50} * mid + \frac{\max\{0, t-80\}}{100-80} * good, \ 50 \leq t < 90 \\ \frac{t-80}{100-80} * good, \ t \geq 90 \end{cases} \quad (1)$$

$$f(u) = \begin{cases} \frac{40-u}{40-0} * low, \ u \leq 20 \\ \frac{\max\{0,\ 40-u\}}{40-0} * low + \frac{u-20}{50-20} * mid, 20 < u < 50 \\ \frac{80-u}{80-50} * mid + \frac{\max\{0,\ u-60\}}{100-60} * high, \ 50 \leq u < 80 \\ \frac{u-60}{100-60} * high, \ u \geq 80 \end{cases} \quad (2)$$

$$f(d) = \begin{cases} \frac{1-d}{1-0} * low, \ d \leq 1 \\ \frac{d-1}{10-1} * high, \ d > 1 \end{cases} \quad (3)$$

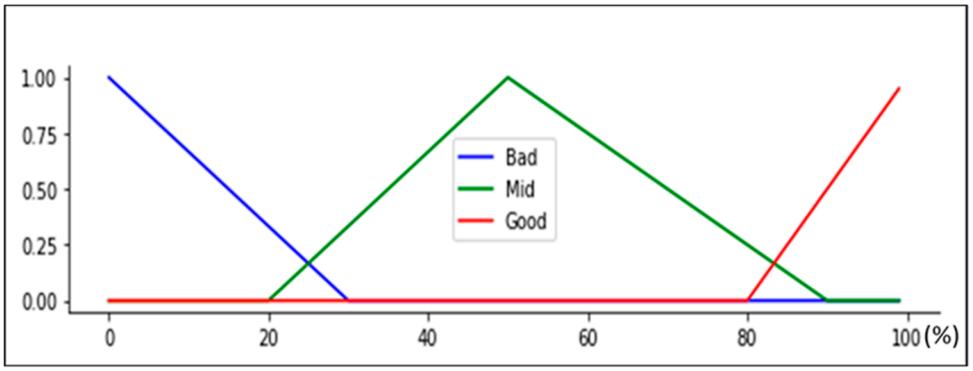

Figure 4. The MF of normalized throughput.

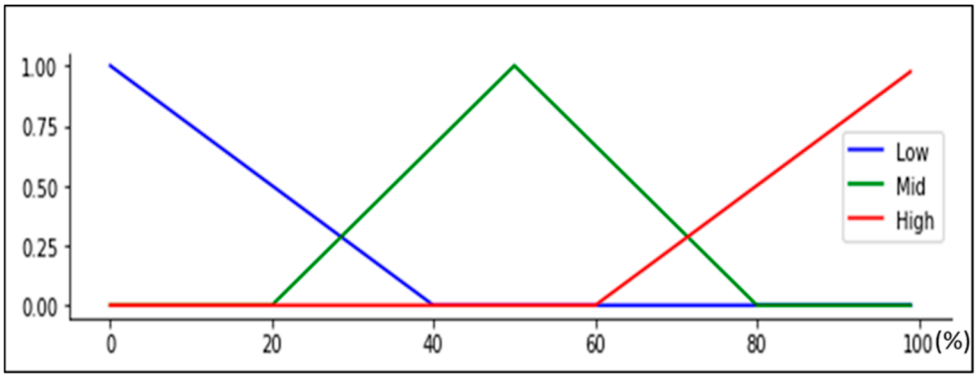

Figure 5. The MF of link utilization.

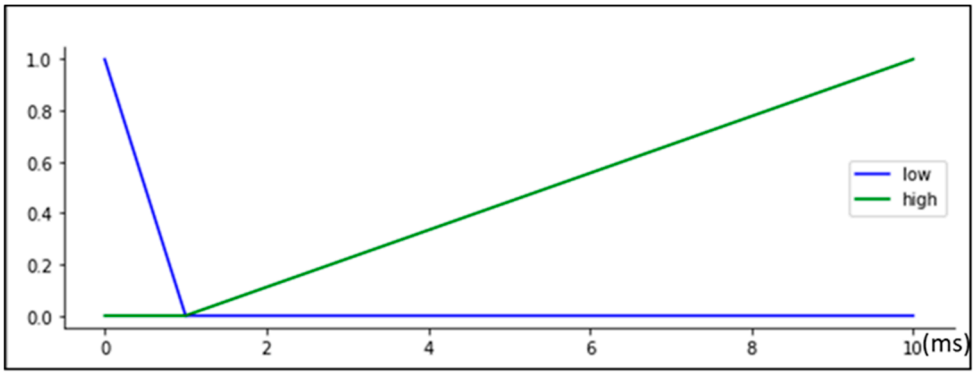

Figure 6. The MF of packet delay.

(b) Inference Engine and Rule Base

The crisp inputs are converted into fuzzy inputs through fuzzification. The inference engine then generates linguistic outputs according to the rule base. As detailed in Table 2, because the three fuzzy inputs (normalized throughput, packet delay, and link utilization) have fuzzy levels of 3, 2, and 3, respectively, all possible fuzzy outputs are obtained using the *AND* (the intersection of antecedents) operator, and a total of 18 rules are generated [23]. The inference engine counts good throughput as 2 points, moderate throughput as 1 point, and bad throughput as 0 points. Low and high delay are counted as 1 and 0 points, respectively. High link utilization is counted as 0 points, moderate link utilization is counted as 1 point, and low link utilization is counted as 2 points. Therefore, each inference

rule belongs to a set of total scores, {0, 1, 2, 3, 4, 5}, which have one-to-one correspondence with a set of six linguistic values, namely {NB, NM, NS, PS, PM, PB}. The MF of the linguistic path weight is plotted in Figure 7, which is used to obtain the numerical value of the path weight at the processing stage of defuzzification.

Table 2. Rule base for inference engine.

Input			Output	
Throughput	**Packet Delay**	**Link Utilization**	**Score**	**Linguistic Value**
good	low	low	5	PB
good	low	mid	4	PM
good	low	high	3	PS
good	high	low	4	PM
good	high	mid	3	PS
good	high	high	2	NS
mid	low	low	4	PM
mid	low	mid	3	PS
mid	low	high	2	NS
mid	high	low	3	PS
mid	high	mid	2	NS
mid	high	high	1	NM
bad	low	low	3	PS
bad	low	mid	2	NS
bad	low	high	1	NM
bad	high	low	2	NS
bad	high	mid	1	NM
bad	high	high	0	NB

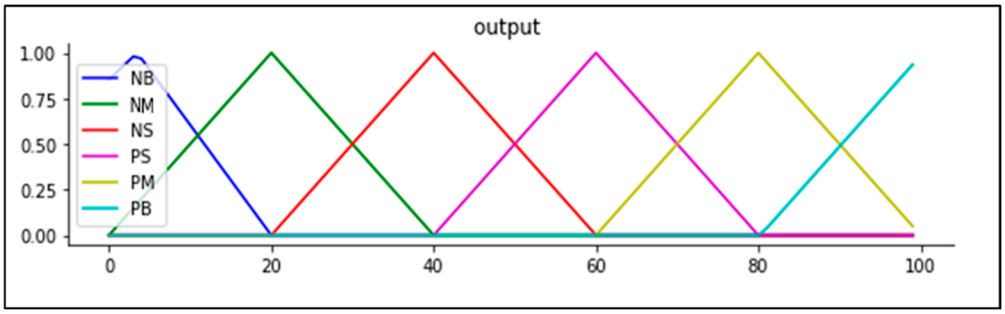

Figure 7. The MF of linguistic path weight.

(c) Defuzzification

After a linguistic output has been generated by the inference engine on the basis of the rule base, the defuzzifier converts the linguistic output into a crisp output. The centre of gravity method [29] is the most commonly used approach for defuzzification. This method calculates the centre point of all overlapping areas that are obtained from the fuzzy outputs for path weight in the corresponding membership function, as plotted in Figure 7, in which the range of the crisp output for the path weight is [0, 100].

3.2. Pseudocode of FLDR

The algorithm of FLDR is listed in Table 3. When the system is started, initialization is first performed, which involves the loading of network topology information, the configuration of the SDN controller, the setting of the monitoring period, and the installation of path selection rules. The system continually monitors the network status, including link utilization, path throughput, and packet delay, and inputs the related network status information to the FL control system to obtain the weight of every path. When a new data flow arises, it is assigned to the path with the highest weight. If the path weight for an

existing data flow is larger than or equal to 80, no routing update is made. If the current path weight is less than 80 and other paths have a weight greater than or equal to 80, the flow table is updated by selecting the path with the highest weight for an URLLC flow. If no other paths have a weight larger than or equal to 80 but other paths have a weight larger than the original path, one path with a packet loss rate of less than δ is selected as the path to be updated to support the high reliability of URLLC. Finally, if no other paths have a weight larger than the original path and a packet loss rate of less than δ, the original path is not updated. According to the report for the actual average packet loss rate of a large international internet service provider [30], δ is approximately 3% to 6%.

Table 3. The algorithm of FLDR.

Input:thethroughput, packet delay, and link utilization of each path collected from the data plane **Output:the optimal route according to the weight of each path**

//do for each path
Collect network status data, throughput, packet delay, and link utilization.
Calculate the weight of each path using FL.
//do for a new flow
If (a flow is new)
Find the path with the greatest weight.
End if
//do for an existing flow and at least one of candidate paths with weight $\geq$ 80
If (the current path weight $\geq$ 80)
Send data packets along the original path.
Else if (the weight of other paths $\geq$ 80)
Reroute an URLLC flow to the path with the greatest weight.
End if
//do for an existing flow and no one of candidate paths with weight $\geq$ 80
If (the weight of another path $\geq$ the original path weight)
If (packet loss ratio $\leq$ δ)
Reroute an URLLC flow to the path with the greatest weight.
Else
Packets are forwarded to the host along the original path.
End if
Else
Packets are forwarded to the host along the original path.
End if

4. Performance Evaluation

4.1. Simulation Settings

The network topology used to perform simulations is shown in Figure 8. It contains 27 paths in total, three source hosts (denoted as H1, H2, and H3), and one destination host (denoted as H4). Additionally, eight OpenFlow switches are denoted as S1 to S8, where S1 is the ingress switch, S8 is the egress switch, and the others are intermediate switches in the transport network of an SDN-based 5G network. The bandwidth of each link is set to 100 Mbps. As summarized in Table 4, the well-known Mininet simulator [31] is used to construct the SDN environment, and the well-established Ryu controller [32] is employed as the SDN controller. The monitoring period of the monitor module is set to 1 s, and four routing mechanisms (static, FLDR, FLE-SDN, and FL-SDN) take turns to operate at the reroute module for comparison. The control and data planes communicate through the OpenFlow protocol (version 1.3). The queue size of each switch is set to 100 packets and the packet generation tool [33], named Iperf, is used to generate UDP (User Datagram Protocol) data packets. The length of the simulation time is 10 s. In the simulation, three data flows are generated successively and they are initially transmitted along one of the shortest paths, S1–S2–S5–S8. The source and destination hosts of these three data flows are H1 and H4, H2 and H4, and H3 and H4, respectively. Flow 1 is generated at the beginning

of the simulation, Flow 2 is generated at the third second, and Flow 3 is generated at the sixth second. Different bit rates of background traffic are transmitted along links S2–S5, S3–S6, and S4–S7 at 15, 30, and 45, respectively, to make light-, medium-, and heavy-load links. We set δ in the proposed FLDR to 5%. Under different data flow rates (i.e., 15, 30, or 45 Mbps), the proposed FLDR mechanism is compared with the static, FLE-SDN, and FL-SDN routing mechanisms in terms of its system throughput, packet loss rate, and packet delay, as plotted in Figures 9–17.

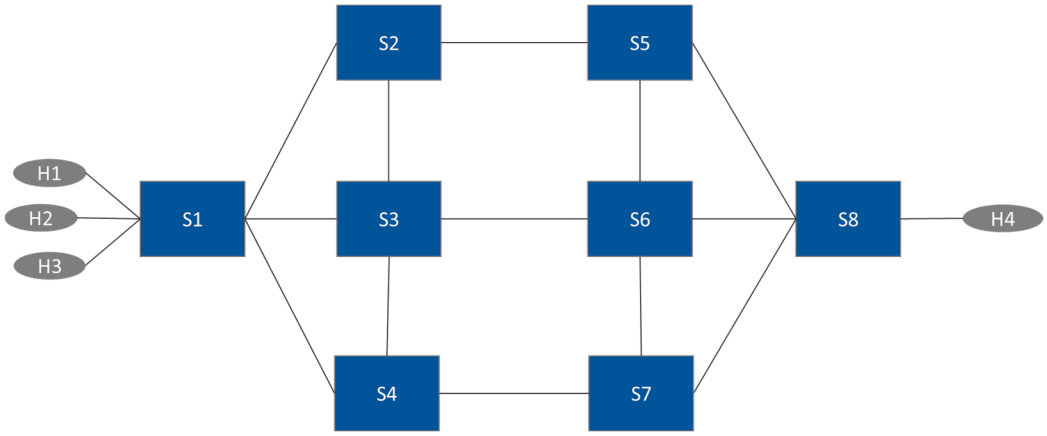

Figure 8. Simulation topology.

Table 4. Parameter settings for simulation.

Parameter	Value
Simulator	Mininet 2.5
SDN controller	Ryu Controller
SDN protocol	OpenFlow V1.3
Packet generation tool	Iperf
Traffic type	UDP
Bit rate per flow	15 M, 30 M, 45 M
Link bandwidth	100 Mbps
Queue size	100 packets
Monitoring period	1 s
Routing mechanism	Static, FLDR, FLE-SDN, FL-SDN
Number of data flows	3 (H1–H4, H2–H4, H3–H4)
δ of FLDR	5%

4.2. Results and Discussions

Given a data flow rate of 15 Mbps, the system throughput for each of the four routing mechanisms is illustrated in Figure 9. In this low-load scenario, congestion is unlikely to occur because the bandwidth of each link is 100 Mbps. The system throughputs for the four routing mechanisms are almost the same. They all increase from 15 to 30 Mbps and then from 30 to 45 Mbps at the third and sixth seconds, respectively. This is because Flows 2 and 3 are generated at these two time points, respectively. The trends for the packet delay and packet loss rate (Figures 10 and 11, respectively) are similar to that for the system throughput.

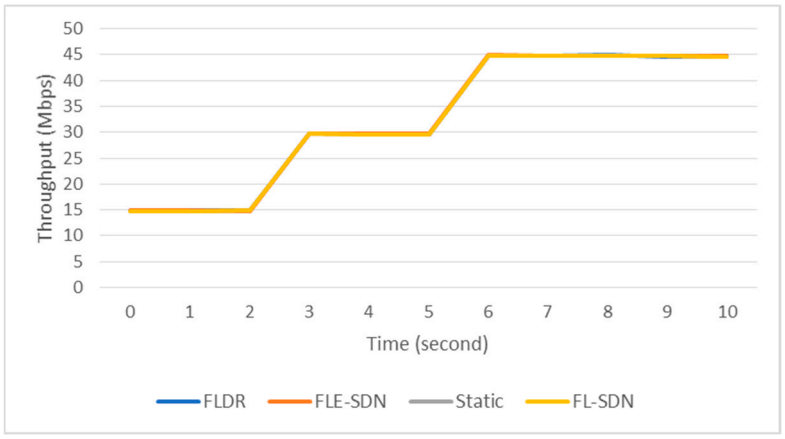

Figure 9. System throughput for 15 Mbps flow rate.

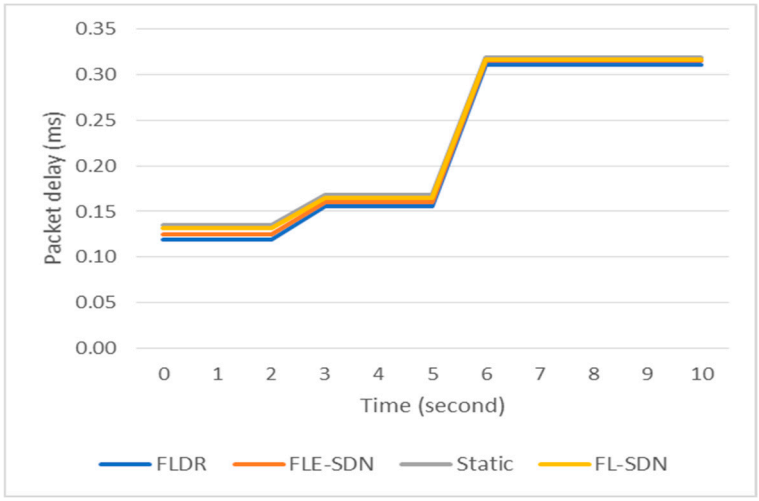

Figure 10. Packet delay for 15 Mbps flow rate.

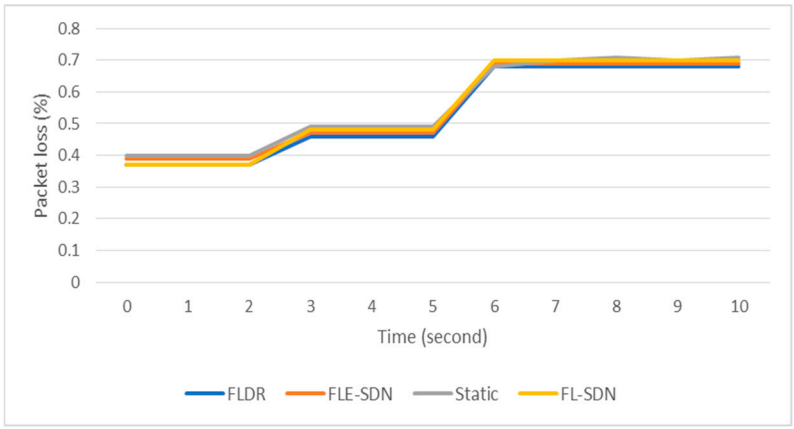

Figure 11. Packet loss rate for 15 Mbps flow rate.

Under a data flow rate of 30 Mbps, similar variation is observed (Figure 12 vs. Figure 9). Only when static routing is employed does the system throughput not reach 90 Mbps at the 7th second. This is because no rerouting is performed in static routing. For the other three routing schemes, from the sixth second onward, data flows are dynamically redirected to the other paths to alleviate link congestion. The packet delay and packet loss rate, respectively, are illustrated in Figures 13 and 14. The four routing mechanisms are found to

yield different results for the packet delay. Owing to its dynamic routing, FLDR performs best, followed by FLE-SDN and FL-SDN, whereas static routing results in the longest packet delay because no rerouting is performed except for link disruption. The reason for this is that the link utilization becomes higher than 80% at the sixth second, and FLDR considers a packet delay of less than 1 millisecond to be a low delay. Therefore, when FLDR is used, the packet delay is considerably better from the seventh second onward, and the 1-millisecond delay guarantee is provided for URLLC. Additionally, the packet delay for FLE-SDN is shorter than that for FL-SDN because the input parameters of the FL control system in FL-SDN are the hop count and bandwidth usage of links, which make FL-SDN more effective for large-scale network environments. Instead of the hop count, FLE-SDN considers the link delay as the input parameter, which has a more accurate correlation with the packet delay than FL-SDN. A similar phenomenon can be identified in Figure 14 for the packet loss rate.

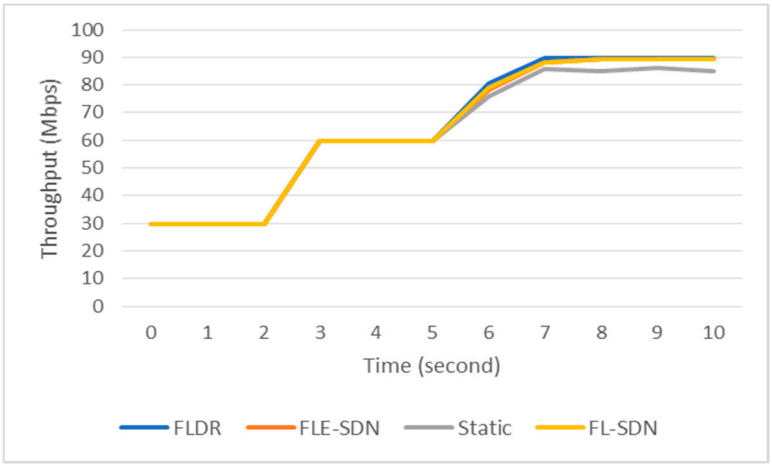

Figure 12. System throughput for 30 Mbps flow rate.

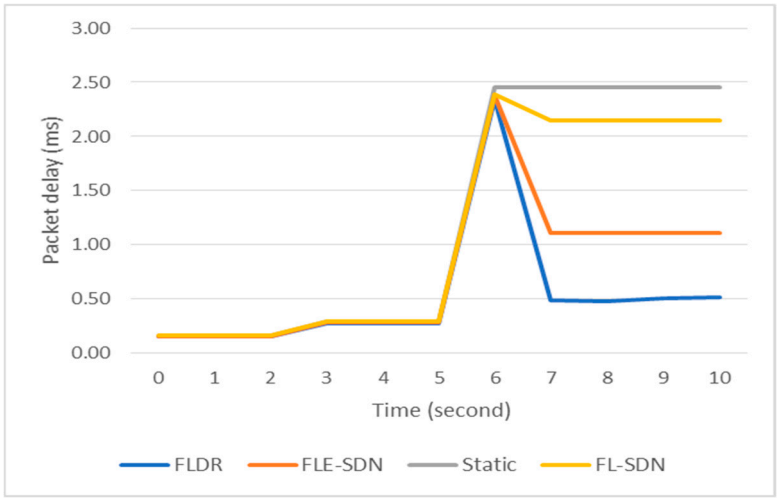

Figure 13. Packet delay for 30 Mbps flow rate.

When the data flow rate is set to 45 Mbps, the first obvious difference is observed at the third second, because the link utilization of the initial path, S1–S2–S5–S8, becomes higher than 80% and link congestion occurs from the third second onward. However, as illustrated in Figures 15–17, the performance is improved at the fourth second regardless of what measure is used because in the three dynamic routing schemes, FLDR, FLE-SDN, and FL-SDN, the forwarding rules are updated after a one-second monitoring period. It is noteworthy that all of the performance measures in static routing (grey curves) have

no improvement, since static routing performs rerouting only in case of link disruption. Similarly, the packet delay and loss rate (Figures 16 and 17, respectively) increase again at the sixth second because Flow 3 is generated then. Nevertheless, the two performance measures for the three dynamic routing schemes decrease again after one-second monitoring period due to a second reroute. However, compared with the other two FL-based routing schemes, FLDR can obtain the best outcomes even in the circumstances of a heavy load, as FLDR considers one more parameter than the other two FL-based routing schemes. Furthermore, the input parameters used in FLDR have a more accurate correlation with the performance measures.

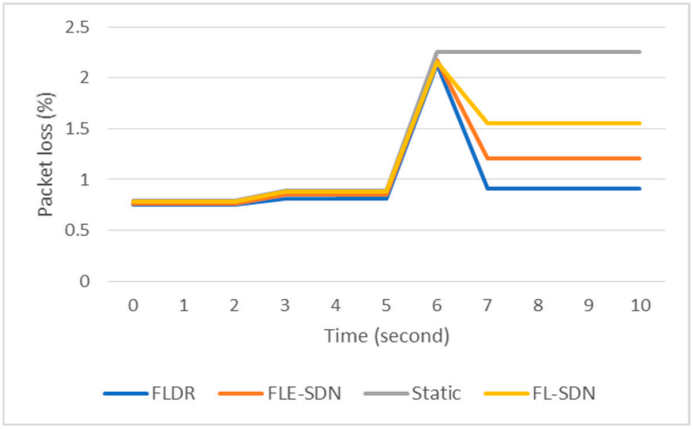

Figure 14. Packet loss rate for 30 Mbps flow rate.

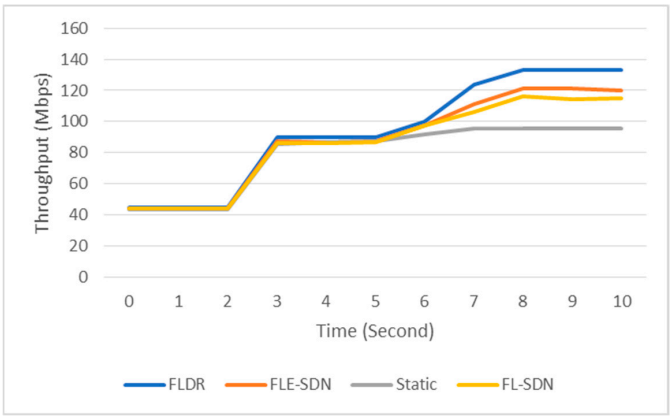

Figure 15. System throughput for 45 Mbps flow rate.

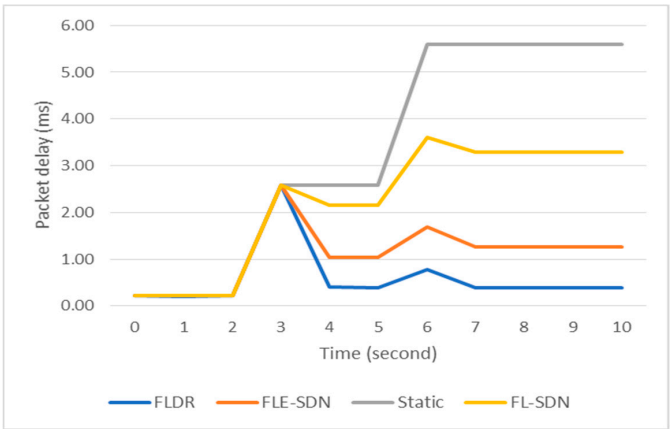

Figure 16. Packet delay for 45 Mbps flow rate.

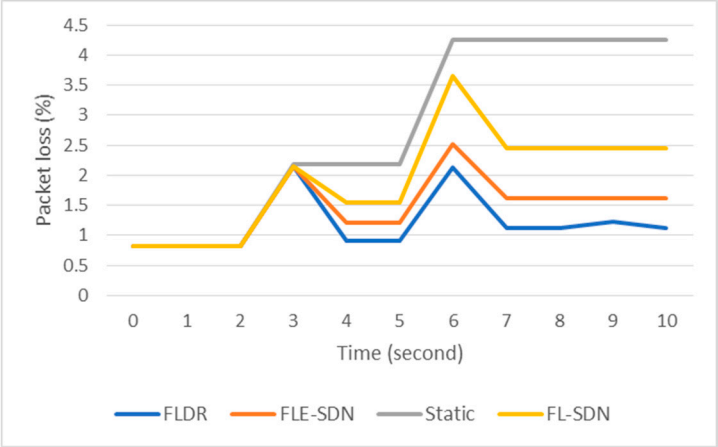

Figure 17. Packet loss rate for 45 Mbps flow rate.

5. Conclusions

This paper proposes an FLDR mechanism with congestion avoidance for URLLC on SDN-based 5G networks. FL control is employed at the reroute module on the control plane in the SDN architecture, and the normalized throughput, link utilization, and packet delay are employed as input parameters to the FL control system. The three input parameters considered in FLDR have a more accurate correlation with the performance measures we are interested in, including system throughput, packet loss rate, and packet delay, than the other existing FL-based dynamic routing schemes. We define the MFs and the rule base of inference engine used in FL and introduce one more condition, the pre-defined threshold of packet loss ratio, for the routing update of URLLC flows. FL-based rerouting can cope with changes and uncertainty in the network environment over time. A series of simulation tests are performed to evaluate the performance of FLDR under various data-load situations. The simulation results demonstrate the outstanding performance of FLDR in terms of the system throughput, packet delay, and packet loss rate. Compared with other FL-based routing algorithms and static routing, FLDR can better cope with instantaneous changes in network traffic and environment in the existing SDN architecture and has superior network forwarding and routing control capabilities. In addition to URLLC, other traffic types exist in 5G networks. In the future, simultaneously taking into account more performance measures such as jitter and energy consumption, we are going to design an FLDR algorithm that can provide QoS guarantee for the other traffic types on SDN-based 5G and beyond communication networks.

Author Contributions: Conceptualization, Y.-J.W. and W.-S.H.; methodology, Y.-J.W. and M.-C.C.; software, M.-C.C. and M.-H.C.; validation, M.-C.C. and M.-H.C.; formal analysis, Y.-J.W. and M.-C.C.; investigation, Y.-J.W., M.-C.C. and M.-H.C.; resources, Y.-J.W. and W.-S.H.; data curation, M.-C.C.; writing—original draft preparation, Y.-J.W.; writing—review and editing, Y.-J.W. and W.-S.H.; visualization, Y.-J.W. and M.-C.C.; supervision, Y.-J.W. and W.-S.H.; project administration, Y.-J.W. and W.-S.H.; funding acquisition, Y.-J.W. and W.-S.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Science and Technology Council in Taiwan under grant number: NSTC 112-2221-E-992-032, NSTC 113-2221-E-158-001. And the APC was funded by NSTC 113-2221-E-992-001.

Data Availability Statement: The data used to support the findings of this study are available from the corresponding author upon request.

Acknowledgments: We thank the anonymous reviewers for their constructive comments, which helped improve the quality of this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Global Software-Defined Networking Market-Industry Trends and Forecast to 2028. Available online: https://www.databridgemarketresearch.com/news/global-sdn-market (accessed on 10 July 2024).
2. Pokhrel, S.R.; Ding, J.; Park, J.; Park, O.-S.; Choi, J. Towards Enabling Critical mMTC: A Review of URLLC Within mMTC. *IEEE Access* **2020**, *8*, 131796–131813. [CrossRef]
3. Kamboj, P.; Pal, S.; Bera, S.; Misra, S. QoS-Aware Multipath Routing in Software-Defined Networks. *IEEE Trans. Netw. Sci. Eng.* **2023**, *10*, 723–732. [CrossRef]
4. Yousaf, F.Z.; Bredel, M.; Schaller, S.; Schneider, F. NFV and SDN-Key Technology Enablers for 5G Networks. *IEEE J. Sel. Areas Commun.* **2017**, *35*, 2468–2478. [CrossRef]
5. OpenFlow Switch Specification. Available online: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf (accessed on 10 July 2024).
6. *RFC 6241, Network Configuration Protocol (NETCONF)*; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2011.
7. Open vSwitch. Available online: https://docs.openvswitch.org/en/latest/ref/ovsdb.7/ (accessed on 10 July 2024).
8. How to Find Shortest Paths from Source to all Vertices Using Dijkstra's Algorithm. Available online: https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/ (accessed on 10 July 2024).
9. Jane, J.B.; Ganesh, E.N. A Review On Big Data with Machine Learning and Fuzzy Logic for Better Decision Making. *Int. J. Sci. Technol. Res.* **2019**, *8*, 1122–1125.
10. Amin, R.; Rojas, E.; Aqdus, A.; Ramzan, S.; Casillas-Perez, D.; Arco, J.M. A Survey on Machine Learning Techniques for Routing Optimization in SDN. *IEEE Access* **2021**, *9*, 104582–104611. [CrossRef]
11. Wu, Y.J.; Hwang, P.C.; Hwang, W.S.; Cheng, M.H. Artificial Intelligence Enabled Routing in Software Defined Networking. *Appl. Sci.* **2020**, *10*, 6564. [CrossRef]
12. Sendra, S.; Rego, A.; Lloret, J.; Jimenez, J.M.; Romero, O. Including Artificial Intelligence in a Routing Protocol Using Software Defined Networks. In Proceedings of the IEEE International Conference on Communications Workshops, Paris, France, 21–25 May 2017; pp. 670–674.
13. Casas-Velasco, D.M.; Rendon, O.M.C.; da Fonseca, N.L.S. DRSIR: A Deep Reinforcement Learning Approach for Routing in Software-Defined Networking. *IEEE Trans. Netw. Serv. Manag.* **2022**, *19*, 4807–4820. [CrossRef]
14. Xu, J.; Wang, Y.; Zhang, B.; Ma, J. A Graph Reinforcement Learning Based SDN Routing Path Selection for Optimizing Long-term Revenue. *Future Gener. Comput. Syst.* **2024**, *150*, 412–423. [CrossRef]
15. Desgeorges, L.; Georges, J.-P.; Divoux, T. Detection of Anomalies of a Non-Deterministic Software-Defined Networking Control. *Comput. Secur.* **2023**, *129*, 103228. [CrossRef]
16. Fu, Q.; Sun, E.; Meng, K.; Li, M.; Zhang, Y. Deep Q-Learning for Routing Schemes in SDN-Based Data Center Networks. *IEEE Access* **2020**, *8*, 103491–103499. [CrossRef]
17. Zadeh, L.A. Fuzzy sets. *Inf. Control* **1965**, *8*, 338–353. [CrossRef]
18. Klir, G.J.; Folger, T.A. *Fuzzy Sets, Uncertainty, and Information*, 1st ed.; Prentice Hall: Englewood Cliffs, NJ, USA, 1988.
19. Kamble, A.J.; Rewaskar, R.P. Soft Computing—Fuzzy Logic: An Overview. *Int. J. Fuzzy Math. Arch.* **2020**, *18*, 45–52. [CrossRef]
20. Pedrycz, W. *Fuzzy Control and Fuzzy Systems*, 2nd ed.; Research Studies Press Ltd.: Taunton, UK, 1993.
21. Kao, M.T.; Huang, B.X.; Kao, S.J.; Tseng, H.W. An Effective Routing Mechanism for Link Congestion Avoidance in Software-Defined Networking. In Proceedings of the 2016 International Computer Symposium (ICS), Chiayi, Taiwan, 15–17 December 2016; pp. 154–158.
22. Hájek, P. *Metamathematics of Fuzzy Logic*; Kluwer Academic Publishers: Dordrecht, The Netherlands; Boston, MA, USA; London, UK, 1998.
23. Izquierdo, S.; Izquierdo, L.R. Mamdani Fuzzy Systems for Modelling and Simulation: A Critical Assessment. *J. Artif. Soc. Soc. Simul.* **2018**, *21*, 2. [CrossRef]
24. Finogeev, A.; Deev, M.; Parygin, D.; Finogeev, A. Intelligent SDN Architecture with Fuzzy Neural Network and Blockchain for Monitoring Critical Events. *Appl. Artif. Intell.* **2022**, *36*, 2145634. [CrossRef]
25. Li, G.; Wang, X.; Zhang, Z.; Chen, Y.; Liu, S. A Scalable Load Balancing Scheme for Software-Defined Datacenter Networks based on Fuzzy Logic. *Int. J. Perform. Eng.* **2019**, *15*, 2217–2227.
26. Prakoso, I.A.; Hertiana, S.N.; Dewanta, F. Analysis of Fuzzy Logic Algorithm for Load Balancing in SDN. In Proceedings of the 2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), Yogyakarta, Indonesia, 16–17 December 2021; pp. 401–406.
27. Moravejosharieh, A.; Ahmadi, K.; Ahmad, S. A Fuzzy Logic Approach to Increase Quality of Service in Software Defined Networking. In Proceedings of the 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Greater Noida, India, 12–13 October 2018; pp. 68–73.
28. Xu, Y.; Wu, M.; Yao, G. An Effective Routing Mechanism Based on Fuzzy Logic for Software-Defined Data Center Networks. In Proceedings of the 2020 IEEE 6th International Conference on Computer and Communications (ICCC), Chengdu, China, 11–14 December 2020; pp. 1793–1798.
29. Abbas, A. Fuzzy Logic Control in Support of Autonomous Navigation of Humanitarian De-mining Robots. In *Using Robots in Hazardous Environments*; Elsevier B. V.: Amsterdam, The Netherlands, 2011; pp. 453–475.

30. What Is Packet Loss: The Invisible Enemy of Network Performance. Available online: https://obkio.com/blog/what-is-packet-loss (accessed on 10 July 2024).
31. Mininet. Available online: https://github.com/mininet/mininet (accessed on 10 July 2024).
32. Ryu Controller. Available online: https://ryu-sdn.org/index.html (accessed on 10 July 2024).
33. Iperf. Available online: https://iperf.fr (accessed on 10 July 2024).