Date of publication SEP-30, 2022, date of current version JUN-03, 2022.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.21.3.2687

Traffic-aware Routing with Software-defined Networks Using Reinforcement Learning and Fuzzy Logic

SHOHREH JAAFARI, MOHAMMAD NASSIRI, REZA MOHAMMADI
Computer Engineering Department, Bu-Ali Sina University, Iran
(e-mail: sh.jafari.en@gmail.com, m.nassiri@basu.ac.ir, r.mohammadi@basu.ac.ir

Corresponding author: Reza Mohammadi (e-mail: r.mohammadi@basu.ac.ir).

ABSTRACT In recent years, the idea of software-defined networks (SDNs) has been proposed for better network management. This architecture has succeeded in optimizing network management functions and increased the ability to synchronize network equipment. Currently, one of the major issues in this architecture is the routing of packets flowing in the network. The main aim in the routing of packets is to increase the quality of services. Enhancement of the quality and productivity of these networks will increase user satisfaction. To this end, the present study proposes a mechanism for selecting the best route from among several existing routes to direct a flow on such a network. The proposed method examines the network parameters including bandwidth, delay, and packet loss on each link of the route by using artificial intelligence algorithms and changes the parameters reducing network productivity by means of fuzzy logic. Our evaluations show that the proposed method can select routes with high productivity and increase the quality of services on the network. Receiving feedback and modifying the fuzzy membership functions related to each mentioned criterion can maintain the effect of these parameters on an acceptable level after which all transmissions tend towards the optimum. Given the use of reinforcement learning methods which underpin some of the routing methods in SDNs, the proposed idea may gradually contribute to the provision of optimized services on the network.

KEYWORDS software-defined network; reinforcement learning; optimization of routing; delay; packet loss; bandwidth.

I. INTRODUCTION

A computer network consists of many pieces of equipment including routers, switches, and firewalls, each functioning according to highly complicated protocols. In traditional networks, each one of the devices that act on different network layers may have been purchased from a different company, and the variety of their rules and structures can complicate network management actions [1]. For this reason, it is difficult to implement management policies in traditional networks and novel management ideas cannot be easily applied in a consistent and homogeneous manner, thereby necessitating the development of new types of networks to overcome these problems.

Software-defined Network (SDN) is an idea which was introduced to facilitate network management. This architecture makes it easier to manage the entire network and optimizes the programming of rules [2]. SDN involves a new architecture in which the control unit is separated from the data unit. The control unit pushes the rules required for the transmission of the existing flows into one or more flow tables in each switch and updates the tables if necessary [3]. In this architecture, OpenFlow communication protocol is used for connecting the data and control units. This protocol allows the controller to send the instructions to switches [4]. As the controller is responsible for management in SDN architecture, one of its tasks is routing. With increase in the number of users and traffic of networks as well as the ever-changing network topology, the SDN controller should be able to transfer the packets to the intended destination in the shortest time possible. Since the advent of SDN, researchers have been attempting to investigate this issue. Traffic engineering is a solution that has attracted the attention of many researchers. Given the importance of routing in the networks, various ideas and algorithms have so far been proposed. Following is a brief review of the work conducted in this field. Initial research introduced SDN and the tasks of the different parts of such networks. Feamster et al. refer to these networks as a new architecture where a central management unit called controller sends information to the data unit with the aim of improving network performance and decision-making concerning the quality of services (QoS) or security [5]. In

addition to explaining SDNs, Nunes et al. indicate that SDN dramatically facilitates network management and increases novelty through programming [6].

Moreover, a lot of research was conducted on large scale SDNs. For instance, Lin et al. proposed a comparative conscious routing in hierarchical multi-layer SDNs. Their architecture includes three controller levels, i.e., super, domain, and slave controllers. The slave controllers are responsible for collecting information for the domain controller. They can also run some of the control functions such as traffic reception control and flow control. The super controller that is connected to the domain controllers has full access to the switches and regulates the entire network performance. Furthermore, the slave controllers do not need extensive information from the network, which results in reducing the signal overload sent to the domain controllers [7]. One of the main functions of the SDN controller is to find the optimal route and therefore contribute to the enhancement of QoS during the transmission of packets [8]. To achieve the desirable performance, some researchers attempted to make use of artificial intelligence (AI) in SDN. Guo et al. devised a smart control mechanism for traffic optimization based on SDN and AI. Their findings suggest that using SDN and AI allows for a smarter traffic optimization. By incorporating AI technology, the SDN controller can provide a more flexible, precise, and dynamic method of traffic scheduling and map various flows onto more suitable routes according to traffic features, thus optimizing the use of network resources [9]. In fact, AI helps to increase the level of services and decrease delays in SDNs. Stampa et al. showed that, when AI is added to the network, the SDN controller can act automatically and present routing configurations, which reduces delays in the network. In their study, the controller is compatible with the traffic on the network and makes use of three signals, i.e., state, action, and reward. State denotes the bandwidth in each source and destination node pair; action refers to the weight of links; and reward is the average delay in the network. This method utilizes traffic matrix and random exploration technique to prevent a local minimum. It reduces network delay by means of one-step optimization [10]. As their results indicate, training the agent in this study was time-consuming. This is why we shall now review a study by Sendra which has reduced the time for training the agent.

Sendra et al. proposed a method of distribute routing on SDNs. They added the reinforcement learning algorithm using the Quagga set [11] to the OSPF routing protocol. This algorithm allows for changing the routing algorithms in the controller. The proposed routing algorithm was tested and compared with a traditional OSPF routing protocol based on an SDN topology. Using reinforcement learning in calculating the cost of routes and selecting optimal routes could decrease delay and jitter in these networks [12]. In fact, by smartly avoiding routes with excessive loss, a higher QoS was obtained. What was important in this study is the setting of the weights of the used parameters including available bandwidth (BW), delay, and packet loss. In fact, the weight of each parameter was set manually so that increase in any criterion would negatively affect the computation of the costs of links and all the criteria were not involved in the computation of the costs in equal proportions. For this reason, the selected routes did not maintain an acceptable level of productivity.

Given the problems discussed above concerning the optimization of routing in SDNs, we decided to develop a method for setting the coefficients of network parameters by using fuzzy membership functions. In the present study, after generating some traffic, we measure available BW, delay, and loss on the network and calculate the shortest route between a certain source and destination by means of routing algorithms. The route is selected with the aim of increasing network productivity. In this architecture, reinforcement learning enables the controller as an agent in the network to begin learning in order to find optimal routes.

The paper is organized as following. The second section discusses the fundamental concepts as well as the proposed architecture. It also explains how AI could be added to the intended scenario. Section 3 evaluates the method and compares it with other methods before concluding the discussion.

II. BACKGROUND CONCEPTS

A. REINFORCEMENT LEARNING

One of the fundamental concepts in the traffic engineering of SDNs is the optimization of routing as well as the finding of optimal routes on the network. In fact, finding a proper route between a source and a destination is perhaps among the central aims of traffic engineering [13]. Traffic engineering attempts to improve network management by identifying the different types of traffic in the network [14]. AI can greatly contribute to the optimization of routing in SDNs with the aim of accomplishing productivity. Today, use of AI algorithms in such architectures has resulted in a number of innovative solutions to the problem of routing. One of the AI techniques is reinforcement learning (RL) methods. Reinforcement learning includes three indicators, i.e., the agent, the state space, and the action space. The agent is an entity which begins learning through interaction with the environment and tries to maximize long-term reward by opting for the best action [15]. The long-term reward depends on the current reward of the selected action along with its future rewards. In this model, the agent is interacting with an ever-changing environment and attempts to fulfill the final goal as well as appropriate learning [15]. After an action, the agent receives either reward or penalty. When different actions have been done and the environment has been changed, the agent once again begins to learn and perform further actions. This cycle continues until the agent has achieved its final goal [16]. Fig. 1 illustrates the systemic model of reinforcement learning. In this model, the evaluation signals which are sent to the learning system may result in reinforcement or punishment. The interpreter that provides the learning agent with this information is at a higher level than the system. The training signals may be incomplete and affected by noise. Therefore, as the environment and the set of permitted actions are constantly changing, the learning agent must achieve its aim which is to maximize the rewards received during the learning time [17].

Figure 1. The systemic model of reinforcement learning

It should be noted that sometimes the agent may avoid small future rewards to receive bigger rewards in a more distant future [18]. Reinforcement learning is used to increase productivity in traffic engineering and packet routing in SDNs [19]. When reinforcement learning is embedded in an SDN, the controller is the agent and the network acts as the environment. Through reinforcement learning algorithms in an SDN we should select smarter routes with lower costs between the nodes. After receiving the feedback and inspecting the reward of actions, the learning agent sets the effect of distinct parameters on each route. Available delay, BW, and loss are among these parameters. There are numerous methods for calculating the effect of each of these parameters when computing the route cost. One such method is fuzzy logic.

B. Fuzzy Logic

Fuzzy logic is a many-valued logic in which truth values may be any real number in the range from 0 to 1, inclusive of 0 and 1. The term 'fuzzy' means inexact and ambiguous. Fuzzy logic has come to be used as a way of representing data that are not exact enough [20]. It is a kind of mathematical theory used for modeling and controlling indeterminacy problems [21]. Today, fuzzy logic helps with a wide range of decision-making problems and often produces the best decisions based on the input values. This logic is based on the human process of decision-making [22]. Fuzzy logic architecture, which is depicted in Fig. 2, consists of four components: fuzzifier module, fuzzy rule base, inference engine, and defuzzifier module. In a fuzzy process, a set of input data is collected and converted to a fuzzy set using fuzzy language variables, fuzzy language expressions, and membership functions. After fuzzification, an inference is made based on a set of rules. Finally, the defuzzifier module illustrates the fuzzy output as a clear output by using membership functions [23].

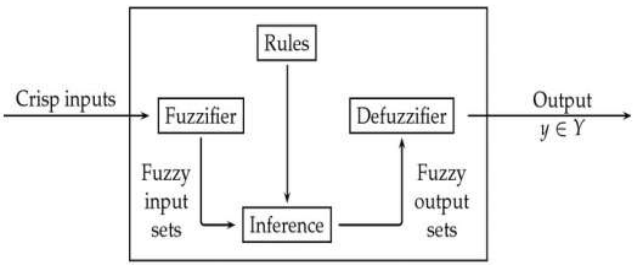

Figure 1. Fuzzy logic architecture

III. THE PROPOSED METHOD

The problem with traditional networks is that the routes selected for packet transmission could not maintain acceptable levels of network parameters including available BW, delay, and packet loss. In addition, the criteria collected from the network do not bear equal effects on the weight of links, which may increase delay and loss in the network and reduce its productivity. For this reason, in the present study, routing is done based on QoS and the routes with high levels of productivity are selected.

Fig. 3 illustrates the proposed method for creating a route between a source and a destination. First, the intended topology is defined on an SDN and then the reinforcement learning algorithm and the fuzzy logic function are implemented on the controller. Traffic is produced and exchanged among the network nodes using different methods of traffic generation. First, all the switches on the network collect information about available delay, BW, and packet loss for every link and send it to the controller. Next, the controller receives these three parameters and passes them through to the fuzzy logic function. The fuzzy logic function calculates the fuzzy value of each link. To do this, we use three triangle fuzzy membership function with three different values Min, Medium and Max. The values which we have considered for these functions are 0, Max/2 and Max. In fact, Max is the maximum value of each measure. Afterwards, using the routing algorithms, the shortest route between a source and a destination is determined by means of the fuzzy value of each of the links. When routing is finished, the determined route is set up between the nodes and the generated traffic is transmitted through it. Several seconds after the execution of the scenario, the controller elicits feedback through the network switches and inspects the status of the network links in terms of available BW, delay, and loss. If the values of the mentioned parameters on each link of the network routes exceeds the specified thresholds, the route in question will receive a penalty; otherwise, it will be rewarded. In the end, the controller sets the parameters of the link of each route by use of fuzzy logic and repeats the routing process.

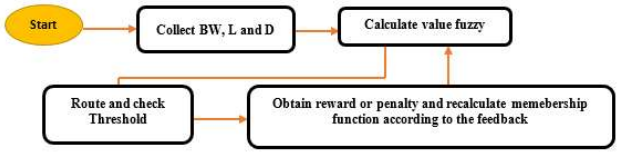

Figure 2. The algorithm for computing the route using reinforcement learning and fuzzy logic

Algorithm 1 shows the method used in this study. To perform this experiment, first the network switches collect the information about available BW, delay, and packet loss from the entire network links and pass it over to the SDN controller (line 1). Next, the controller calculates the three parameters for each link and gives the values to the fuzzy logic function. The function has three inputs and one output. The fuzzy function gets the three inputs and calculates the fuzzy number of each link using a defuzzifier function. In the end, we will have a fuzzy number for each link in the topology (lines from 2 to 4). Subsequently, using the Dijkstra algorithm, the shortest route between a source and destination is calculated based on the fuzzy numbers. In fact, the Dijkstra algorithm begins the routing process based on the fuzzy numbers obtained in the previous step. After calculating the shortest route between two hosts, the route is set up and video traffic is exchanged between the hosts (line 5 and 6).

10 seconds after the execution of the program, the controller receives reports from the network switches and examines the values of the parameters at each link on the route (line 7). It should be mentioned that each of the parameters has a threshold values which must not be exceeded during the execution of the algorithm. For example, in this study, the threshold value is 80 bps for available BW, 200 ms for delay, and 5% for packet loss. If the value of any of the parameters has not exceeded the defined threshold, the route will be rewarded; otherwise, the route receives a penalty (lines 8 to 13). In the end, the fuzzy membership functions are refreshed based on the reward or penalty of the previous step (line 14). This cycle will continue until better routes are selected for transmission of traffic over the network. One of the main achievements of this policy is that

choosing best route for traffic flows leads to less congestion. This is because of forwarding traffics to the links with more available bandwidth which reduces the probability of congestion.

Algorithm 1. RLTE

```
M ← collect D and L and BW from links
for src < Number of nodes do
 for des < Number of nodes do
  F[src][des] ← Fuzzy(M_D[src][des], M_L [src][des],M_BW [src][des],P_d[src][des], P_l[src][des],P_BW[src][des])
  a ← dijkstra(F [src] [des], src, des)
  If (a.findpath) then The path is installed end
  M ← collect D and L and BW from links
  If (M_D >= TH-Delay) then F-D[s][d] ← true end
  If (M_L >= TH-Loss) then F-L[s][d] ← true end
  If (M_BW >= TH-BW) then F-BW[s][d] ← true end
  If (F-D [s] [d] == true || F-L [s] [d] == true || F-BW [s] [d] == true) then Calculate Penalty for the link
      else  Calculate Rewards for the link
  end
  Update(P_D[src][des], P_L[src][des], P_BW [src][des])
 end
end
```

IV. PERFORMANCE EVALUATION

To design the scenario, we considered a system consisting of a number of switches and hosts. Fig. 4 shows the topology used in this study. Traffic was produced and exchanged among several selected node pairs using different methods of traffic generation. These flows were used to increase the load of network traffic. The hosts connected to s1 and s23 nodes were the intended source and destination, and video traffic was exchanged between these two hosts. The flow between these two nodes is the flow which is investigated in this study. The aim is to find an optimal path and increase productivity in transmission of traffic between the two hosts. In this paper, a single-path method has been used, i.e., the traffic is transmitted between the source and destination only through a single route.

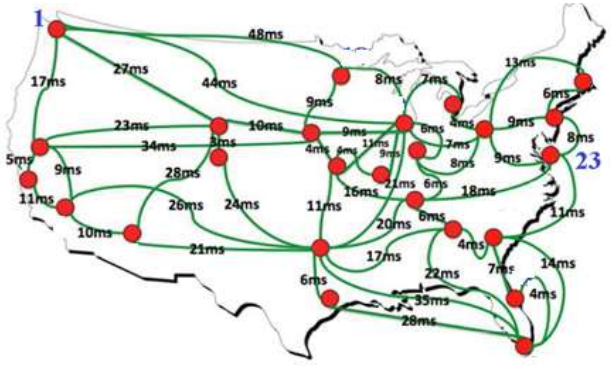

Figure 4. The topology of Sprint GIP network in North America [24]

To avoid using expensive hardware, we constructed our scenario by means of Mininet emulator. Mininet was developed by Kaur et al. [25] as a tool for designing virtual SDNs so as to avoid the use of expensive specialized hardware equipment having tested the performance of Mininet during the transmission of multimedia traffic over an SDN. Jimenez et al. concluded that it was a powerful tool that could be used for experiments on SDNs [25]. This was optimized by Jimenez et al. in [26]. They extended the range of experiments and confirmed that Mininet is a reliable SDN emulator.

The scenario was implemented in Mininet and the fuzzy logic and reinforcement learning were executed in the controller. In each execution, video traffic loads of 5, 7, 10, 13, and 15 Mbps were exchanged between h1 and h23 nodes as the intended source and destination. The simulation ran for 300 s and the results were recorded. In the presentation of our results, Sendra refers to the older method in which the coefficients of the parameters were set manually. In the proposed solution, fuzzy logic has been added to this model and the parameters are updated using fuzzy membership functions. The proposed method is called RLTE (Reinforcement Learning Traffic Engineering for SDN). In this study, we shall compare RLTE with Sendra and OSPF. We evaluate the performance of the three methods in terms of some important QoS metrics as follows:

- Packet loss ratio: It is the fraction of received packets by the target to the total sent packets by the senders.
- End-to-end delay: It is the time between sending a packet in the source and receiving it at the destination.
- Received throughput: It is the volume of received traffic by the target nodes during a second and expresses by Bps (Bit per Second).

In the end, to examine the average of the obtained results in RLTE and Sendra, we executed each method 10 times with different traffic intensities and recorded the results.

Fig. 5 shows the end-to-end delay in the three methods. The horizontal axis represents the traffic transmitted between the source and destination, and the vertical axis represents the delay in milliseconds. Each of the numbers shown in the graph is the average of the results of multiple executions. As can be seen, the traditional OSPF method has more delay than the other methods. This might be explained by the fact that routes in OSPF are selected using the Dijkstra algorithm and the existing network parameters do not affect this selection; therefore, the parameters are not examined for this purpose. According to the results, the delay in RLTE is less than in Sendra. To clarify this point, Figure 6 compares the delay values of the two methods.

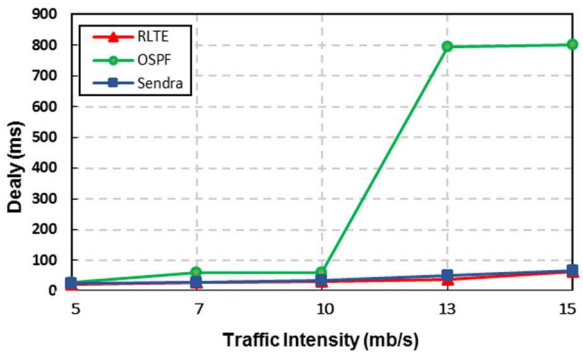

Figure 5. Comparison of the end-to-end delay in the three methods

As shown in Fig. 6, the delay values of both methods are relatively close to each other but the delay of RLTE is slightly

less than that of Sendra in all the traffic intensities transmitted over the network. In RLTE, the network parameters are set according to the fuzzy numbers; that is, route selection is influenced by all the parameters together, not by a single parameter having a low or high value. For this reason, packets are transmitted with less delay.

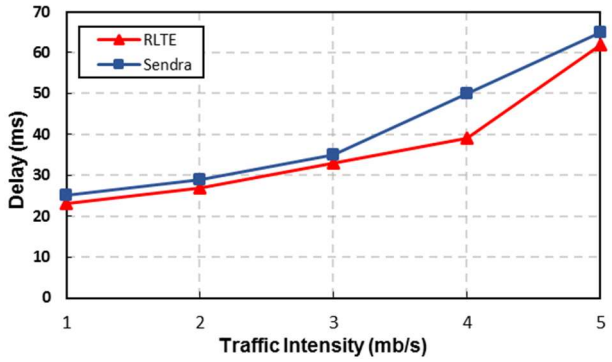

Figure 6. Comparison of the end-to-end delay between RLTE and Sendra

For a closer inspection of the delay in different times in the three methods, the end-to-end delay was specifically analyzed for the traffic of 7 Mbps. The results are illustrated in Fig. 7. As can be observed, the delay of OSPF protocol is about 60 ms, which has remarkably increased towards the end of the simulation. As the parameters are not involved in calculating the route in Sendra, this method shows more delay than RLTE. Due to using reinforcement learning and fuzzy logic, RLTE had less delay during the simulation.

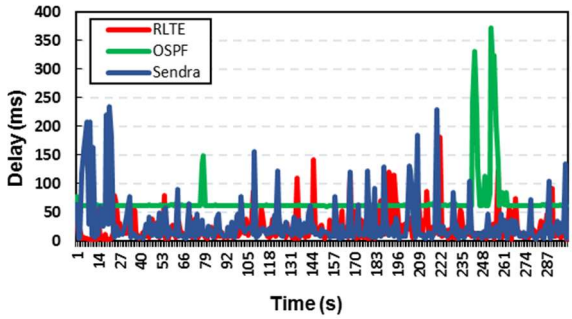

Figure 7. Comparison of the end-to-end delay for the traffic of 7 Mbps in the three methods

To examine the average of results as well as the minimum and maximum of each scenario, each method was executed 10 times and the recorded results are presented below. Fig. 8 shows the average delay in RLTE and Sendra. Due to the involvement of network parameters in RLTE, the delay values in this method had smaller standard deviation values than in Sendra, and the overall delay of RLTE in larger traffic intensities was less than that of Sendra. Also, the larger the traffic load on the network, the closer the RLTE results were to the average result. As the traditional OSPF method has very long delays (around 2000 ms) in some traffic intensities, it is not illustrated in the graph of the results.

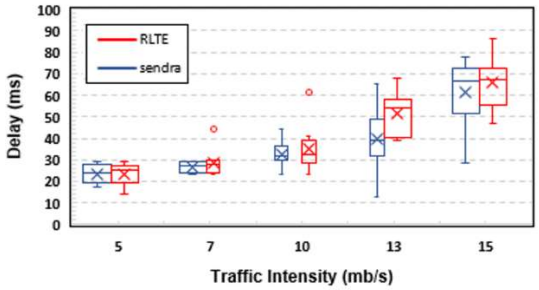

Figure 8. Comparison of the average delay between RLTE and Sendra

Fig. 9 shows the packet loss in the three methods. The horizontal axis represents the traffic transmitted between the source and destination, and the vertical axis represents the loss in percentage. As mentioned earlier, the OSPF protocol does not take into account the network parameters in route selection and therefore increases packet loss. As shown in Fig. 9, loss in OSPF is as high as 0.4. However, it is much less in RLTE and Sendra. Fig. 10 shows the value of loss for the two methods.

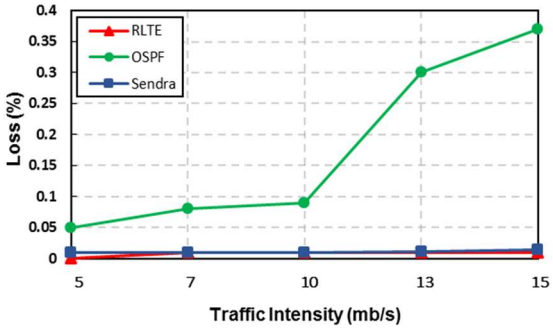

Figure 9. Comparison of packet loss in the three methods

Fig. 10 shows the average loss in RLTE and Sendra. According to the results presented in Fig. 10, the higher the transmitted traffic, the less the packets lost in RLTE; however, loss grows at an increasing rate in Sendra. The reason is that, in each execution of the routing protocol in RLTE, loss is considered as a factor in computing the route and set via fuzzy membership functions in relation to the other parameters. In Sendra, however, the coefficient of loss is set manually and is not adjusted to the conditions of network links.

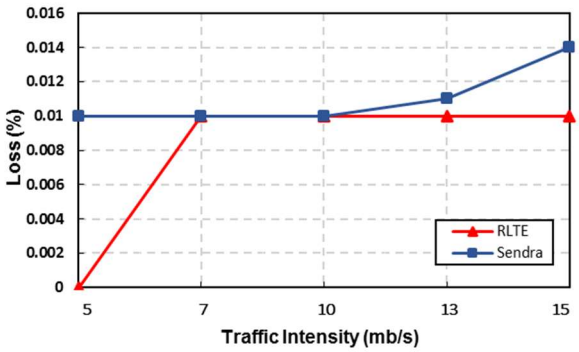

Figure 10. Comparison of the packet loss between RLTE and Sendra

Fig. 11 shows the packet loss in 10 executions of the three methods. As shown in previous parts, the average loss for all

traffic intensities in OSPF is greater than in the other two methods. As the network parameters in RLTE are set dynamically according to the conditions of the network, the loss in RLTE is expected to be less than in Sendra. As Fig. 11 shows, the average loss in RLTE has even been recorded as zero for some of the traffic loads. Also, the higher the traffic intensity on the network, the better the performance of RLTE in comparison with Sendra. Thus, it can be inferred that the proposed method has sought to minimize loss and provide appropriate conditions for packet transmission.

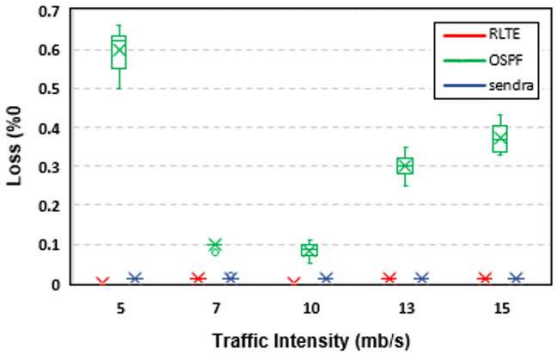

Figure 11. Average packet loss in the three methods

Fig. 12 compares the jitter values in RLTE and Sendra. It can be observed that, as the traffic rate increases on the network, RLTE has less jitter than Sendra. In RLTE, after computing the route, packets arrive at the destination from more optimal routes and in shorter intervals.

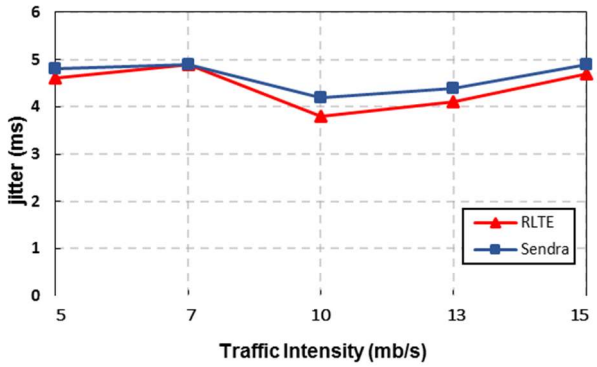

Figure 12. Comparison of the jitter between RLTE and Sendra

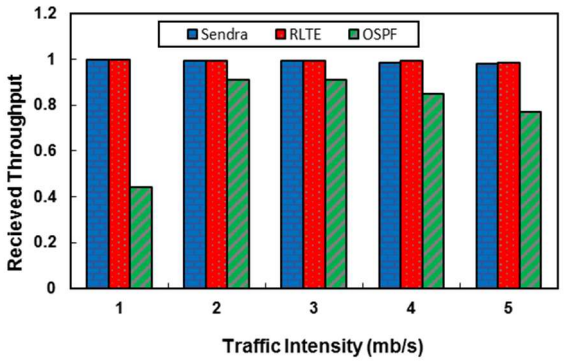

Figure 13. Comparison of the received throughput in the three methods

V. CONCLUSIONS

In this study we proposed a smart routing method for SDNs. For this purpose, we analyzed the construction method of SDN topology which runs the routing protocol in a distributed manner. To improve routing, we implemented a smart algorithm based on reinforcement learning in the controller. The network parameters were involved in the routing process by means of fuzzy logic. The proposed routing algorithm was tested and compared with an older method and a traditional OSPF routing protocol based on an SDN topology. The results show that the proposed routing method works well and obtains better QoS features than the traditional method. In addition, the obtained jitter value is better than the values offered by traditional routing and the Sendra method. Thus, better quality and QoS values become possible by smartly avoiding routes with high loss values. In general, the Sendra method could optimize the traditional OSPF protocol and lead to better results. We added fuzzy logic to the proposed method. The obtained results under saturated conditions show that as network traffic becomes heavier, RLTE performs better than the other two methods and optimizes the functionality of routing protocols.

References

[1] H.-N. Quach, S. Yoem, and K. Kim, "Survey on reinforcement learning based efficient routing in SDN," Proceedings of the The 9th International Conference on Smart Media and Applications SMA'2020, September 2020, pp. 196–200. https://doi.org/10.1145/3426020.3426072.

[2] M. Vafaei, A. Khademzadeh, and M. A. Pourmina, "A new QoS adaptive multi-path routing for video streaming in urban VANETs integrating ant colony optimization algorithm and fuzzy logic," *Wireless Personal Communications*, vol. 118, pp. 2539-2572, 2021. https://doi.org/10.1007/s11277-021-08142-7.

[3] Q. Fu, E. Sun, K. Meng, M. Li, and Y. Zhang, "Deep Q-learning for routing schemes in SDN-based data center networks," IEEE Access, vol. 8, pp. 103491-103499, 2020. https://doi.org/10.1109/ACCESS.2020.2995511.

[4] S. Torkzadeh, H. Soltanizadeh, and A. A. Orouji, "Energy-aware routing considering load balancing for SDN: a minimum graph-based ant colony optimization," *Cluster Computing*, vol. 24, issue 3, pp. 2293-2312, 2021. https://doi.org/10.1007/s10586-021-03263-x.

[5] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87-98, 2014. https://doi.org/10.1145/2602204.2602219.

[6] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617-1634, 2014. https://doi.org/10.1109/SURV.2014.012214.00180.

[7] S.-C. Lin, I. F. Akyildiz, P. Wang, and M. Luo, "QoS-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach," *Proceedings of the 2016 IEEE International Conference on Services Computing (SCC)*, 2016, pp. 25-33. https://doi.org/10.1109/SCC.2016.12.

[8] I. F. Akyildiz, P. Wang, and S.-C. Lin, "SoftAir: A software defined networking architecture for 5G wireless systems," *Computer Networks*, vol. 85, pp. 1-18, 2015. https://doi.org/10.1016/j.comnet.2015.05.007.

[9] A. Guo and C. Yuan, "Network intelligent control and traffic optimization based on SDN and artificial intelligence," *Electronics*, vol. 10, no. 6, article 700, 2021. https://doi.org/10.3390/electronics10060700.

[10] G. Stampa, M. Arias, D. Sánchez-Charles, V. Muntés-Mulero, and A. Cabellos, "A deep-reinforcement learning approach for software-defined networking routing optimization," arXiv preprint arXiv:1709.07080, 2017.

[11] P. Jakma and D. Lamparter, "Introduction to the quagga routing suite," *IEEE Network*, vol. 28, no. 2, pp. 42-48, 2014. https://doi.org/10.1109/MNET.2014.6786612.

[12] S. Sendra, A. Rego, J. Lloret, J. M. Jimenez, and O. Romero, "Including artificial intelligence in a routing protocol using software defined networks," *Proceedings of the 2017 IEEE International Conference on*

Communications Workshops (ICC Workshops), 2017, pp. 670-674. https://doi.org/10.1109/ICCW.2017.7962735.
[13] H. Yang et al., "Time-aware software defined networking for OpenFlow-based datacenter optical networks," Netw. Protoc. Algorithms, vol. 6, no. 4, pp. 77-91, 2014. https://doi.org/10.5296/npa.v6i4.5922.
[14] N. Wang, K. H. Ho, G. Pavlou, and M. Howarth, "An overview of routing optimization for internet traffic engineering," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 1, pp. 36-56, 2008. https://doi.org/10.1109/COMST.2008.4483669.
[15] M. K. Awad, M. H. H. Ahmed, A. F. Almutairi, and I. Ahmad, "Machine learning-based multipath routing for software defined networks," *Journal of Network and Systems Management*, vol. 29, no. 2, pp. 1-30, 2021. https://doi.org/10.1007/s10922-020-09583-4.
[16] J. Xie et al., "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 393-430, 2018. https://doi.org/10.1109/COMST.2018.2866942.
[17] L. El-Garoui, S. Pierre, and S. Chamberland, "A new SDN-based routing protocol for improving delay in smart city environments," *Smart Cities*, vol. 3, no. 3, pp. 1004-1021, 2020. https://doi.org/10.3390/smartcities3030050.
[18] S. Peng-hao, L. Ju-long, S. Juan, and H. Yu-xiang, "An intelligent routing technology based on deep reinforcement learning," *Acta Electonica Sinica*, vol. 48, no. 11, p. 2170, 2020.
[19] H. A. Al-Rawi, M. A. Ng, and K.-L. A. Yau, "Application of reinforcement learning to routing in distributed wireless networks: a review," *Artificial Intelligence Review*, vol. 43, no. 3, pp. 381-416, 2015. https://doi.org/10.1007/s10462-012-9383-6.
[20] T. J. Ross, *Fuzzy Logic with Engineering Applications*, John Wiley & Sons, 2005.
[21] H. El Alami and A. Najid, "SEFP: A new routing approach using fuzzy logic for clustered heterogeneous wireless sensor networks," *International Journal on Smart Sensing & Intelligent Systems*, vol. 8, no. 4, pp. 2286-2306, 2015. https://doi.org/10.21307/ijssis-2017-854.
[22] L. Zhao, Z. Bi, M. Lin, A. Hawbani, J. Shi, and Y. Guan, "An intelligent fuzzy-based routing scheme for software-defined vehicular networks," *Computer Networks*, vol. 187, p. 107837, 2021. https://doi.org/10.1016/j.comnet.2021.107837.
[23] R. Mohammadi, R. Javidan, and A. Jalili, "Fuzzy depth based routing protocol for underwater acoustic wireless sensor networks," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 7, no. 1, pp. 81-86, 2015.
[24] S.-C. Lin, P. Wang, and M. Luo, "Jointly optimized QoS-aware virtualization and routing in software defined networks," *Computer Networks*, vol. 96, pp. 69-78, 2016. https://doi.org/10.1016/j.comnet.2015.08.003.
[25] K. Kaur, J. Singh, and N. S. Ghumman, "Mininet as software defined networking testing platform," *Proceedings of the International Conference on Communication, Computing & Systems (ICCCS)*, 2014, pp. 139-42.
[26] J. M. Jimenez, O. Romero, A. Rego, A. Dilendra, and J. Lloret, "Performance study of a software defined network emulator," *Proceedings of the Twelfth Advanced International Conference on Telecommunications (AICT 2016)*, 2016, pp. 17-22.

***SHOHREH JAAFARI** received her BSc degree in Computer Software Engineering from Hamedan University of Technology. She received his MSc degree in Computer Software Engineering from Bu-Ali Sina University, Hamedan, in 2020. She is currently working toward a research assistant at the Department of Computer Engineering in Bu-Ali Sina University, Hamedan. Her research interests include SDN, the Internet of Things and IoT-fog networks.*

***MOHAMMAD NASSIRI** is currently an Associate Professor at Bu-Ali Sina University. After having graduated from Sharif University of Technology (Iran), he obtained his Ph.D. in computer science from Grenoble INP (France) in 2008. He was a visiting researcher at the Universite Grenoble Alpes (France) during summer 2019. His research interests mainly concern improving the performance of various wireless technologies, namely wireless LANs, wireless sensor networks, underwater networks, and interconnecting technologies for the Internet of Things.*

***REZA MOHAMMADI** is an Assistant Professor in Computer Engineering at Bu-Ali Sina University since 2018. He received his MSc and Ph.D. degrees in Computer Networking from Shiraz University of Technology in 2013 and 2017, respectively. His PhD thesis was concerned with traffic engineering in Software Defined Networks (SDN). He has several publications in international conferences and journals regarding Underwater Wireless Sensor Networks (UWSNs) and Software Defined Networks (SDNs). His major fields of interest are SDN, heuristic algorithms, SDN security, Underwater Wireless Sensor Networks, Ad hoc Networks, Underground Wireless Sensor Networks, Internet of Things (IoT) and IoUT.*