Q-learning based routing protocol for congestion avoidance

Daniel Godfrey
Chungnam National University

Beom Su Kim
Chungnam National University

Haoran Miao
Chungnam National University

Babar Shah
Zayed University, babar.shah@zu.ac.ae

Bashir Hayat
Institute of Management Sciences

See next page for additional authors

Computers, Materials & Continua
DOI:10.32604/cmc.2021.017475
Article

Q-Learning Based Routing Protocol for Congestion Avoidance

Daniel Godfrey[1], Beom-Su Kim[1], Haoran Miao[1], Babar Shah[2], Bashir Hayat[3], Imran Khan[4], Tae-Eung Sung[5] and Ki-Il Kim[1,*]

[1]Department of Computer Science and Engineering, Chungnam National University, Korea
[2]College of Technological Innovation, Zayed University, Abu Dhabi, UAE
[3]Institute of Management Sciences, Peshawar, Pakistan
[4]Department of Electrical Engineering, University of Engineering and Technology, Peshawar, Pakistan
[5]Department of Computer and Telecommunications Engineering, Yonsei University, Korea
[*]Corresponding Author: Ki-Il Kim. Email: kikim@cnu.ac.kr
Received: 31 January 2021; Accepted: 22 March 2021

Abstract: The end-to-end delay in a wired network is strongly dependent on congestion on intermediate nodes. Among lots of feasible approaches to avoid congestion efficiently, congestion-aware routing protocols tend to search for an uncongested path toward the destination through rule-based approaches in reactive/incident-driven and distributed methods. However, these previous approaches have a problem accommodating the changing network environments in autonomous and self-adaptive operations dynamically. To overcome this drawback, we present a new congestion-aware routing protocol based on a Q-learning algorithm in software-defined networks where logically centralized network operation enables intelligent control and management of network resources. In a proposed routing protocol, either one of uncongested neighboring nodes are randomly selected as next hop to distribute traffic load to multiple paths or Q-learning algorithm is applied to decide the next hop by modeling the state, Q-value, and reward function to set the desired path toward the destination. A new reward function that consists of a buffer occupancy, link reliability and hop count is considered. Moreover, look ahead algorithm is employed to update the Q-value with values within two hops simultaneously. This approach leads to a decision of the optimal next hop by taking congestion status in two hops into account, accordingly. Finally, the simulation results presented approximately 20% higher packet delivery ratio and 15% shorter end-to-end delay, compared to those with the existing scheme by avoiding congestion adaptively.

Keywords: Congestion-aware routing; reinforcement learning; Q-learning; Software defined networks

1 Introduction

Congestion in a network significantly increases the end-to-end delay. To prevent or remove congestion, many congestion control schemes have been proposed for current TCP/IP networks, where a strictly layered architecture and fully distributed algorithms are applied. Owing to layered restrictions, most of the current algorithms, including congestion control, have been implemented on the transport layer for an end-to-end connection.

As an alternative for current TCP/IP networks, software defined networks (SDN) have been studied to overcome the drawbacks of legacy networks (i.e., a lack of adaptability). In contrast to the typical combined model, the data and control plane are separated and operated in an SDN by centralizing the network intelligence and state logically. Therefore, it is feasible to execute or run new centralized algorithms over an SDN without regard to a layered architecture. Based on a centralized functionality, an intelligent algorithm such as a machine-learning algorithm used to solve the complexity can be gradually applied in an SDN. Specifically, machine learning (ML) algorithms have been applied to improve the network performance in the areas of network operation and management. ML techniques have recently been employed to deal with fundamental network problems, for example, traffic prediction, routing and classification, congestion control, resource and fault management, quality of service (QoS), quality of experience management, and network security [1–3].

Stemming from these observations, a new congestion-aware routing protocol for an SDN is presented herein. Unlike previous studies on end-to-end congestion control, our goal is to develop a routing protocol to manage congestion at the network layer. Thus, it is possible to control congestion in a hop-by-hop approach. In addition, it is extremely feasible to implement this type of protocol in an SDN. A new routing protocol is designed to search for an uncongested path with a Q-learning method known as reinforcement learning. We present a model for a routing protocol with Q-learning properties, which can be defined by the Q-value and reward function. With the Q-value and reward function, we can determine if the next-hop is a congested node. The reward function is characterized by a new buffer occupancy, retransmission ratio, and hop count parameters. Finally, we evaluate the performance of the proposed routing protocol through simulations.

The main contributions of this paper are as follows:

- An architecture that employs Q-learning for achieving efficient and intelligent congestion-aware routing in an SDN;
- A Q-learning based routing algorithm that considers a look-ahead algorithm to compute the Q-value;
- An extensive set of experiments with simulations and an analysis for the proposed routing protocol.

The rest of this paper is organized as follows. Following the introduction, we describe some previous state-of-the-art studies conducted in this area. The proposed scheme is explained and described. The simulation results are next given. Finally, some concluding remarks and areas of future study are presented.

2 Related Studies

In this section, we describe related studies on congestion-aware routing protocols in three parts. First, routing protocols used in an SDN are presented. Second, ML-based routing protocols used in an SDN are analyzed. Third, a congestion-aware routing protocol is detailed.

2.1 Routing in SDN

Zhang et al. [4] addressed the performance measurement of the routing protocol in an SDN in terms of a forwarding delay and convergence time for a failure as compared to a legacy protocol. They experimented and concluded that an SDN is beneficial in large-scale networks. In addition, the impact of a link failure in an SDN is less than that in legacy routing protocols. Thus, more robustness against a failure is achieved in an SDN by reducing the convergence time significantly. In terms of the performance evaluation of the routing protocol in an SDN, Gopi et al. [5] focused on the convergence time to recover a link or node failure with respect to the topology scale. Similar to the experiment results of a former study, a shorter convergence time is measured when a large-scale topology is assumed. Akin et al. [6] compared the routing protocol for an SDN with a static and dynamic link cost by implementing it on a Mininet emulator. Incorporating the use of a multi-criteria decision-making method (MCDM) in an SDN, Ali et al. [7] proposed the use of a hierarchical SDN control plane approach for an inter-domain collaboration and QoS class mapping to ensure the E2E quality-of-service for applications in heterogeneous networks with multiple domains of different QoS classes. In this study, the commonly used MCDM, known as TOPSIS, was applied at the controller module to select the most suitable QoS class for each domain in the E2E path. The findings of this study suggest that the use of a single controller with varying QoS classes could lead to a single point failure and E2E service delivery-related issues. For all the cases, it has been proved that the performance of a routing protocol in an SDN is mostly dependent on the accuracy of the network state information. Based on the mentioned study, it is reasonable to determine that the routing protocol in an SDN is more robust than a conventional routing protocol while requiring more accurate network state information. In addition to a performance evaluation, a new routing protocol for an SDN has been continuously studied.

First, centralized QoS routing protocols for an SDN were analyzed and compared in [8]. In addition to a description of outstanding features, the authors employ a novel four-dimensional evaluation framework for QoS routing protocols for a quantitative comparison in terms of the runtime and cost inefficiency. Despite a performance improvement in an SDN, the replacement cost from a legacy network to an SDN will be a major concern. To address this problem, a new QoS routing protocol for SDN hybrid networks was proposed by Lin et al. [9], whose proposed protocol, called simulated annealing based QoS-aware routing (SAQR), dynamically adjusts the weights of three QoS parameters, namely the delay, loss rate, and bandwidth, and achieves an improved delay performance exceeding 20%.

Second, a number of studies have proposed routing protocols in a specific SDN. Ji et al. [10] proposed an SDN-based geographic routing protocol for vehicular *ad hoc* networks. Unlike previous geographical routing protocols that use local information, a new protocol makes use of vehicle information, that is, the node location, vehicle density, and digital map, and computes the optimal path based on such information. In parallel with vehicular *ad hoc* networks, smart-city and IoT applications are regarded to be suitable for SDN infrastructure. To reduce the delay in an SDN, EL-Garoui et al. [11] proposed a new routing protocol based on an SDN by employing a machine learning algorithm as a prediction scheme. As for IoT, a new SDN-based routing was proposed by Shafique et al. [12]. The proposed scheme targets the balance between the cost for reconfiguration and the flow allocation in which multiple SDN controllers are assumed. In addition, heterogeneous network traffic is monitored to keep the networks balanced. As a special type of network, a disturbance-awareness routing algorithm [13] based on weather information has been proposed to minimize the network cost function as well as the cost of the risk function in an SDN. Each of the above-mentioned specific network types has its own approach to detect and deal with a link failure. To discuss a link failure and recovery schemes on SDN-based routing schemes, Ali et al. [14] presented a survey that highlights various link failure detection and recovery schemes, mechanisms, and their respective weaknesses in an SDN. In addition, a well-organized classification of link failure recovery approaches was presented based on a review of 49 papers. To combat congestion-related link recovery issues in routing, an introduction of proactive and reactive schemes was further mentioned for both single and multi-objective schemes.

2.2 ML-Based Routing in SDN

Differing from the traditional model-driven approach for routing protocols, ML-based routing protocols can capture the growing complexity and adapt to network changes accordingly. However, the management of large-scale data for ML has been a challenge in the current distributed infrastructure. This is why an SDN based on a centralized entity is a suitable architecture for operating ML algorithms.

Before looking into the details, it is worth mentioning a comprehensive overview [15] for machine learning in an SDN. In this study, the authors provide a survey for machine learning algorithms feasible for an SDN. Following an ML outline, the authors have addressed the challenges and reviewed related studies in terms of several perspectives including a routing optimization. In addition, open issues and challenges for ML in an SDN are discussed. In addition to this survey, we categorize routing protocols based on the type of ML-algorithms and present their key features.

First, reinforcement learning (RL) to optimize the routing problems in an SDN is presented by C. Fang et al. [16]. The proposed RL model contributes to making decisions through interactions with the environment. A combination of RL and neural networks has been proposed for the routing algorithm. Another protocol called V-S routing (variable ϵ-Greedy function within SARSA-learning routing) is addressed by Yuan et al. [17]. The proposed algorithm takes the dynamic priority of the current state in an SDN to avoid a delay as well as improve the link transmission speed. Another scheme to utilize RL has been proposed to meet the QoS requirements. A new algorithm, called reinforcement learning and software-defined networking intelligent routing (RSIR) [18], utilizes RL to search for the best route for all flows with a link state metric (i.e., bandwidth, loss, and delay). To obtain an optimal path, the proposed algorithm finds the most-rewarding path for every pair of nodes in the network. The simulation results proved that an RSIR can avoid traffic concentration and congestion by applying different edge weights for mentioned metrics. Similar to the mentioned approaches, Hossain et al. [19] present an RL-driven QoS-aware routing algorithm that consists of both QoS monitoring for the delay, packet-loss rate, and RL-based intelligent routing decision-making (RIRD). During operation, if the RL agent selects the path having the lowest delay and packet-loss rate, it should obtain the highest reward value.

In addition to RL, a deep learning-based QoS routing protocol was proposed by Owusu et al. [20]. In this study, the authors mention the real-time application on the Internet and present a framework based on an SDN. A deep neural network is employed to classify the class of traffic and search for appropriate routes to meet the QoS demand. As a new ML framework, federated learning (FL) has recently attracted the interest researchers. As an example, Sacco et al. [21] merges network softwarization and FL to optimize routing decisions in an SDN. Their main contribution is a new path selection algorithm based on long short–term memory (LSTM) to predict the forthcoming traffic on a link based on history. In the case of a high traffic volume, a new path is selected to avoid high-loaded links and take the under-utilized ones. ML-based routing protocols can be deployed for a special objective. Pasca et al. [22] proposed an application-aware multipath flow routing framework called AMPS. The proposed scheme is composed of a dynamic prioritization of the flow, a path assignment based on priority, and Yen-K-shortest path algorithm to find the path. In addition to traffic, an energy-efficient routing protocol for an SDN called MER-SDN was suggested by Assefa et al. [23]. For energy efficiency, a principal component analysis (PCA) was suggested to reduce the feature size, along with a linear regression to train the model. In addition, an integer programming (IP) formulation for energy consumption as a function of the traffic amount and heuristics algorithm are presented.

2.3 Congestion-Aware Routing

A general congestion control scheme over the transport layer has a long convergence time under an end-to-end argument principle. Compared to the scheme used in the transport layer, an enhanced functionality in the network layer leads to a reduced convergence time. To identify and remove congestion proactively and reactively, diverse congestion-aware routing protocols have been studied, which we categorized into the underlying target networks.

First, some congestion-aware routing protocols have been proposed to prevent packet loss in wireless sensor networks [24]. In particular, if a lost packet contains important event or data information, it can affect the reliability of the system. To handle this situation appropriately, advanced congestion-aware routing (A-CAR) is a priority and congestion-aware routing protocol in wireless sensor networks. In ACAR, a differential routing policy depending on priority is applied. For a flow with a higher priority, an inside zone path is established, whereas another path is constructed outside a zone for a packet with lower priority. In addition, ACAR can provide mobility support by changing the routing zone accordingly. Unlike flat networks, Farsi et al. [25] proposed a new congestion-aware clustering and routing protocol to properly address congestion issues. Congestion is prevented by the load distribution of the cluster head node between members and the rotation of role changes in the cluster during every round. While taking limited energy as well as a real-time requirement into account, congestion-aware routing needs to cover the mentioned demands. El-Fouly et al. [26] presented the real-time energy-efficient traffic-aware approach (RTERTA) in industrial wireless sensor networks. In RTERTA, congestion can be avoided by utilizing underloaded nodes with a hop count to the sink node that is measured by the buffer occupancy in a node.

Second, unlike static wireless sensor networks, congestion-aware routing has been studied in dynamic networks, including vehicular *ad hoc* networks. Hung et al. [27] presented an intersection-based routing protocol called a data congestion-aware routing protocol (DCAR) that is suitable for urban environments. In DCAR, the amount of data and vehicular traffic are estimated. This value is used to construct a routing path. While establishing a path, a look-ahead algorithm for deciding the next intersection is also considered to avoid congestion. Congestion caused by a flooding broadcast was addressed by Liu et al. [28]. A novel congestion-aware GPCR routing protocol (CA-GPCR) utilizes a free buffer queue size and the distance between the next node and destination node and restricts the greedy forwarding procedure to avoid congestion. Simulation results show that the CA-CPCR protocol outperforms the existing protocol in terms of packet delivery ratio and delay caused by congestion. In addition, Keykhaie et al. [29] presented the congestion-aware and selfishness aware social routing protocol for use in a delay tolerant network. To distinguish congested and selfish nodes, both the buffer congestion and selfish behavior are measured and used to obtain the utility value. Depending on this value, a more suitable node is selected for message relaying.

Third, congestion-aware routing for an SDN is proposed. Attarha et al. [30] proposed a method to reroute a flow to avoid congestion in an SDN. To make a decision, link utilization is periodically measured and reported. A new flow is routed according to the network conditions. The controller predicts the congestion and calculates the amount of flow to be rerouted toward the backup paths. Another congestion-aware routing based on a rerouting path in an SDN was proposed by Cheng et al. [31]. In the proposed scheme, a flow along the congested route is detoured toward the local path and modeled by the LP. Finally, Ahmed et al. [32] addressed the congestion control and temperature-aware routing over SDN-based wireless body area networks. The authors presented the energy optimized congestion control based on temperature aware routing algorithm based on enhanced multi-objective spider monkey optimization. The proposed routing algorithm introduces the congestion queue length as a major factor in the routing cost model and combines it with other factors such as the residual energy, link reliability, and path loss.

As previously analyzed, an SDN is capable of implementing complicated algorithms such as ML in a central entity with topology information. In addition, congestion avoidance in the network layer not only can reduce the convergence time but also consequently adapt the network dynamics. However, despite the mentioned benefits, there is no ML-based congestion-aware routing protocol over an SDN. Furthermore, we take Q-learning, which is a model-free technique that does not require prior knowledge about the underlying reward resulting from taking specific action in a particular state. According to this property, Q-learning is suitable to handle dynamic network congestion properly. A typical operation of the Q-learning based congestion-aware routing protocol would appear as summarized below in Fig. 1.

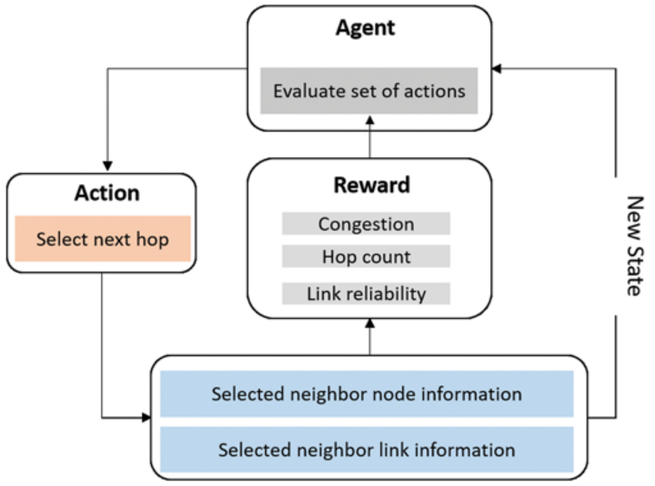

Figure 1: Typical Q-learning based congestion aware routing protocol flow

3 Q-Learning Based Congestion-Aware Routing in SDN

In this section, we propose a new Q-learning based congestion-aware routing (QCAR) in an SDN. Both the network architecture and details for a routing protocol are consequently described.

3.1 *Architecture and Component*

To implement QCAR over an SDN, the network architecture including the control plane, data plane, and an application plane is designed as shown in Fig. 2. The control plane collects raw data about the network status through periodical messages. The collected information is passed to the application plane. In the application plane, the Q-learning agent and algorithm compute the Q-values for the topology and the best route decision for the flow. This decision is sent to the control plane. Consequently, the control plane requests to update the forwarding table at the data plane.

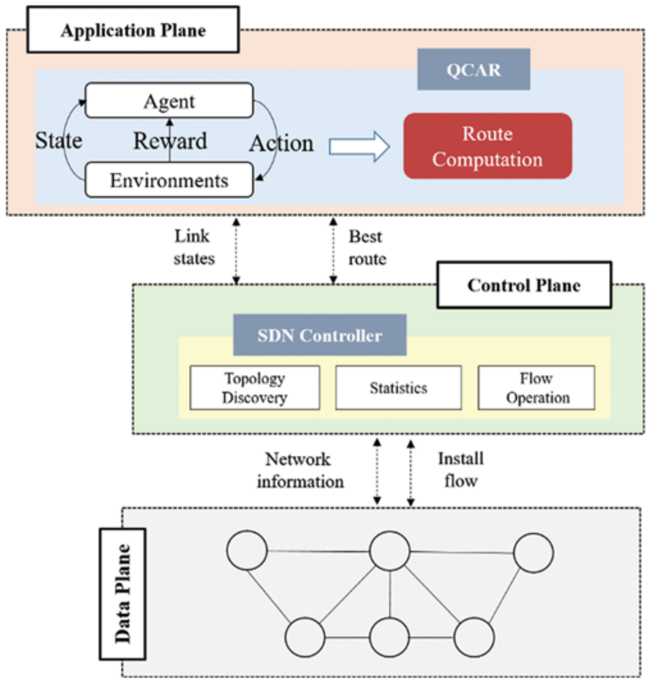

Figure 2: QCAR architecture

3.2 *QCAR Routing Protocol*

The QCAR protocol follows the Q-learning technique to define the routes to be followed by flows with source–destination pairs. Each step consists of selecting and performing an action, changing the state (i.e., moving from one to another), and receiving a reward. The updated Q-function value at time t is the underlying reward for the execution of action A_t while in state S_t, which provides an optimal reward R_t. Next, we provide details about the derived parameters for node and link states in an SDN, RL-agent, and RL-based routing algorithms.

3.2.1 *Node and Link States in SDN*

For the QCAR protocol, we define a set of parameters that indicate the node and link status to be used by the RL agent. For a node, say node i, the parameters are as follows: the queue length of node i (QL_i^t), the hop count to the destination (H_i^t), and retransmitted packet ratio ($RPR_{i,j}^t$) over a link between two adjacent nodes, i and j, at time t. Based on measured values, the parameters are computed as follows:

Queue length: To measure congestion level at arbitrary node i, we periodically estimate the buffer occupancy based on the queue length of node i. The queue length is computed at time t according to Eq. (1). Let QL_i^t be the sum of the queue length of node i and that of the node two hops ahead at time t. By taking into consideration the queue length of the node two hops away through the look-ahead algorithm, a large value is given to the node whose neighbors are already in a state of congestion. If there are at least two neighbors for node i, the minimum queue length at the neighbors is considered, where N_i denotes the set of neighbors of node i.

$$QL_i^t = QL_i^t + \min_{j \in N_i}(QL_j^t), \tag{1}$$

Retransmitted packet ratio: In addition to the congestion parameters of a node, the adjacent link reliability affects the congestion because the received packet remains at the buffer until a receiver successfully receives it. To measure the link reliability, we consider the retransmitted packet ratio, which counts for all retransmitted packets owing to propagation-related errors of a link. A link with a larger ratio of retransmitted packets is considered unsatisfying for the traffic demand and hence unreliable. To obtain the RPR of a link between nodes i and j during the past s seconds, we use the expression below:

$$RPR_{i,j}^t = \frac{Packets\_Retransmitted_{i,j}^{t-s}}{Packets\_Sent_{i,j}^{t-s}} (s < t), \tag{2}$$

$$RPR_{i,j}^t = RPR_{i,j}^t + \min_{j \in N_i}(RPR_{i,j}^t), \tag{3}$$

where $Packets\_Sent_i^{t-s}$ is the total number of packets sent by node i during the past s seconds from current time t, and $Packets\_Retrasmitted_j^{t-s}$ counts the total number of packets transmitted to neighbor j during the past t seconds. Similar to the queue length, the link reliability also employs a look-ahead algorithm, as given in Eq. (3).

3.2.2 RL-Agent

Typical RL-problems are usually referred to as discrete-time Markov decision problems owing to the modeling of their solution which is based on 4-tuples (S, A, P, R). Here, S is the finite set of states, A is the set of actions, P is the matrix of state transition probability, and R is the reward function for which the system is continuously looking to maximize. The environment for the RL agent to act on is composed of data packets flowing in a network from a given source to the desired destination. The presence of a given packet p at node i defines the state of that packet at time t as S_i^t. An action $A_{i,j}^t$ represents a decision made by the RL agent to forward the packet from node i to neighbor j as adopted by the policy (π_t) controlling state transition with a greedy exploration strategy at time t, as shown in Eq. (4). Upon this action being taken, the state of packet p will move from S_i^t to S_j^{t+1} and the reward associated with this action will be $R_{i,j}^{t+1}$.

$$\pi_t\left(S_i^t\right) \leftarrow \underset{a_n \in A(S_i)}{\operatorname{argmin}}\ Q\left(S_i^t, A_{i,n}^t\right) \tag{4}$$

This means that, instead of finding a path with the maximum reward, our proposed QCAR finds a path with the lowest costs by greedily selecting actions with the lowest rewards provided that all available neighbors have a level of congestion more than the predetermined threshold. In addition, for each state transition ($S_n \to S_{n+1}$), the Q-function value $Q_n\left(S_n, A_n\right)$ associates a reward function R, which is computed as shown in the following subsection, to estimate the cost of forwarding a packet toward that particular neighbor.

In Q-learning, the agent learning phase consists of a sequence of stages, called epochs (*0*, *1*, …, *n*…). During the *nth* epoch at time *t*, the RL-agent selects an action A_t on a packet p at a current state S_t and receives a reward R_t as it moves to the next state, S_{t+1}. The action-value $Q_{t+1}(S_t, A_t)$ is updated based on the following equation:

$$Q_{t+1}\left(S_t, A_t\right) = (1-\alpha) \times Q_t\left(S_t, A_t\right) + \alpha \times \left[R_t + \gamma \times \min_{a \in A} \left\{Q_t\left(s_{t+1}, a\right)\right\}\right] \tag{5}$$

where α is the learning rate that controls how fast the Q-table changes, and γ is the discount factor that determines the degree to which the agent considers the effect of the immediate rewards when estimating new Q-values. The initial Q-values, Q_O (S_0, A_0), for all the states and actions are initialized to zero before the RL-agent learning phase starts.

3.2.3 Reward Function

The reward function used by the RL-agent is based on three measured parameters. The reward is proportional to the queue length, retransmitted packet ratio, and hop count, as defined in Eq. (6). To normalize the mentioned parameters, Q_{max} and H_{max} were introduced and denote the maximum queue length and maximum allowed hop count, respectively. In addition, Eq. (6) is applied along with the respective tuning weights ω_1, ω_2, and $\omega_3 \in [0, 1]$, where $\omega_1 + \omega_2 + \omega_3 = 1$.

$$R = \omega_1 \cdot \left(\frac{QL_i^t}{Q_{max}}\right) + \omega_2 \cdot \left(LR_{i,j}^t\right) + \omega_3 \cdot \left(\frac{H_i^t}{H_{max}}\right) \tag{6}$$

3.3 QCAR Routing Decision

The general process of the proposed congestion-aware based routing protocol is explained in Algorithm 1, which provides a brief explanation of how the different layers work together to find a better path for all pairs of nodes at the data plane. First, the RL-agent at the application plane is provided with processed link state information from the control layer and given inputs (i.e., the learning rate, discount factor, network size, training epochs, all (*src, dst*) pairs, network graph, and weights $(\omega_1, \omega_2, \omega_3)$). From the given inputs, the RL agent is expected to continuously compute and update the best paths for all pairs of nodes in a given network.

Algorithm 1: Q-value Update

```
Input: All (src, dst), Network graph, Link states
Output: $Q_i^{t+1}\left(S_i^t, A_i^t\right)$.
1   Initialize $\alpha$, $\gamma$, $\omega_1, \omega_2, \omega_3$, Q: A × S → R, initialized with 0
2   For each (src, dst) pair do
3       current_state = src
4       While current state is not the destination do
5           $R_{t+1} \leftarrow R(S_t, A_t)$ with Eq. (6)
6           Update Q-table
8           $S_t \leftarrow S_{t+1}$
9       End While
10 End For
```

Algorithm 2: Selecting Next Hop at Node i

```
1   If ( $NID$(i) == $NID$(dst))
2       Deliver a packet to upper layer
3   Else
4       Flag ← False
5       $NH_i =$
6       For all nodes in $N_i$
7           If ($QL_i^t <$ Threshold)
8               $NH_i = NH_i \cup NID(i)$
9               Flag ← True
10          End If
10      End For
11      If (Flag == True)
12          Select a next hop in $NH_i$ randomly
13      Else
14          Take Q-table and find a path with lowest Q-value
15      End If
16 End If
```

The algorithm execution processes to find the optimum paths for all pairs of nodes start by initializing the Q-values of the Q-table to zeros (Line 1). For a given packet at the source node, the first exploration epoch starts by initializing the state of a packet at the *src* node (Line 1), and from that state selects one action (A_t) among all possible actions from the current state (Line 2). With the selection of this action, it considers moving to the next state (S_{t+1}) (Line 8). Using Eq. (4), the minimum Q-value for this next state is obtained based on all possible actions (Line 6), followed by setting the next state as the current state (Line 8). The state transition loop continues until the current state is equal to the final state (i.e., the packet reaches the *dst* node) (Line 4). Once the final goal is reached, the training epoch ends and a new one starts until they have all run (Line 2). Based on the computed Q-values, the RL-agent computes the optimal routes to forward data packets between the given *src-dst* pairs and forwards them to the flow control module at the control plane.

The routing algorithm of QCAR is described in Algorithm 2. Initially, if a node is a destination by comparing the node identifier, a packet is passed to the upper layer. Otherwise, a node chooses the next hop for a packet. Choosing the next hop is dependent on the neighbor node's congestion level. From Lines 5 to 10, we construct the new neighbors' subset (NH_i) of N_i with the only node whose queue length is less than the predetermined threshold. After building $NH_{i,}$, a node performs two different operations. The former is to select a next-hop among the NH_i, set randomly to prevent node congestion by distributing packets along with multiple nodes, whereas the latter is to set the next hop as the node along the path with the lowest Q-value. These actions are shown between Lines 11 and 15. That is, when the congestion levels of multiple neighbors are acceptable, the next hop is randomly selected among them. Otherwise, the best route through QL is chosen and set as the next hop for a given packet.

4 Performance Evaluation

This section presents an evaluation of the proposed QCAR protocol through simulations based on the network simulator ns-3. First, we illustrate our simulation settings followed by a discussion on the impacts of different settings of the Q-learning-related parameters. In addition, the influence of the data flow rate, the number of traffic sources, the node density, and the maximum buffer size on the system performance will be discussed and ultimately compared between the performance of our proposed QCAR, the shortest path based on Dijkstra's algorithm, and the traditional Q-learning without a look-ahead, which is represented as QL in the figures. We present the performance comparisons with two performance parameters: packet delivery ratio and end-to-end delay.

4.1 Simulation Settings

To verify the routing mechanism based on QCAR, we deployed network topologies with nodes uniformly distributed. To observe how well the proposed approach reacts to different congestion levels, we perform several simulations with different data flow rates, a varied number of traffic sources, varied node densities, and the maximum buffer size. To avoid the formation of long routes between the given source and destination nodes, we limit the formation of the route length to the maximum of only 4 hops. The rate error model with a byte unit is applied to cause packet corruption. According to the probability, a packet is discarded if a byte is corrupted. To best estimate the obtained results, we run each scenario 10 times with different seed values and obtain the averaged results. For the specific configuration of the parameters, see Tab. 1 below.

Table 1: Simulation parameters

Parameter	Value or range
Simulation time (s)	100
Traffic types	UDP
Packet size	1 Kb
Link types	Point-to-point
Ethernet type	IEEE 802.3.
Queue type	FIFO
Route update interval	2 s
Learning rate (α)	0.8
Probability for error model	0.0001–0.004

4.2 Impact of Q-Learning Related Parameters on QCAR

In the QCAR proposed approach, the link-state associated with each node is periodically updated based on the look-ahead method to determine the potential next-hop(s). The updated link states offer information necessary for next-hop selection such as the available queue size and measured link reliability on that particular node. The degree by which the information of the potential neighbor is considered important when selecting the next-hop depends on the discount factor parameter, which ranges between zero and 1. The closer it gets to 1, the higher the impact will be, and vice-versa. In addition, we discuss the impacts of different weight settings $(\omega_1, \omega_2, \omega_3)$

that determines which of the three metrics (available buffer size ratio, link reliability, and hop count) is dominant when computing the routes to the destination node.

As previously mentioned, the weights are added to comprehensively minimize the effects of the available buffer size, link reliability, and path length in the route selection. We randomly select the ratios and run through the simulations to find a single weight set that gives the best results. We categorize the three sets under different cases with each showing the effect of setting one of the parameters as dominant over the others. In Case 1 ($\omega_1 : \omega_2 : \omega_3 =$ 2:1:7) the parameter hop count is placed as the most dominant, whereby the shortest path to the destination is the most favored. Case 2 ($\omega_1 : \omega_2 : \omega_3 =$ 2:7:1) favors the formation of a path based on the reliability of the links. Finally, in Case 3 with the ratio of ($\omega_1 : \omega_2 : \omega_3 =$ 7:1:2), the nodes prefer the selection of next hops based on the degree of packet congestion. According to the simulation results shown in Figs. 3 and 4, assigning a relatively larger weight value to the congestion metric causes more data packets to be delivered at an acceptable increased delay with the QCAR approach. Case 3 shows a better trade-off between the parameters by allowing the nodes to prefer the selection of less congested and shorter routes as much as possible. In Figs. 3 and 4 we use a single traffic source by sending packets at a rate of 20 packets per second, which is expected to cause a buffer overflow after some time on certain nodes because the maximum buffer size is only 10 packets.

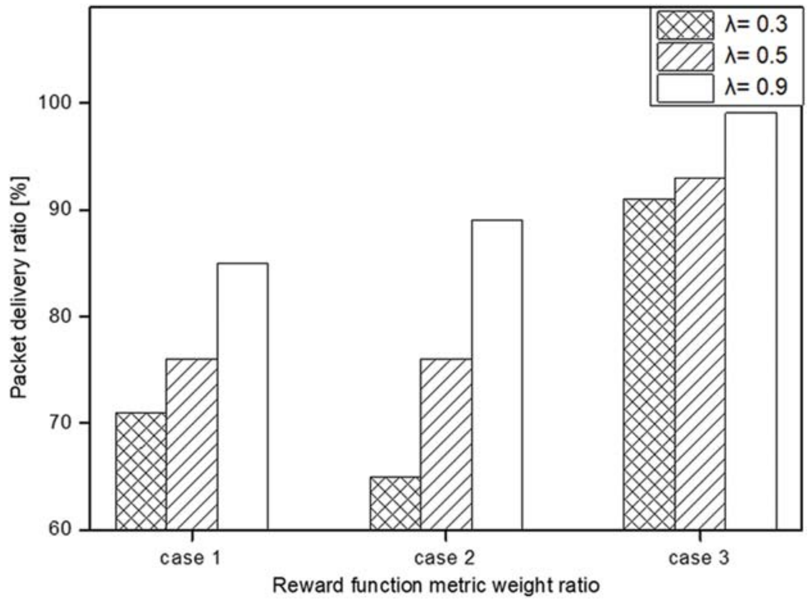

Figure 3: Packet delivery ratio *vs.* weight ratio

The results suggest that a large discount factor has a better impact on the performance of the QCAR algorithm because it allows nodes to give higher priority to neighbors whose neighbors are less congested and closer to the destination node. In addition, we studied the impacts of different learning rates on both the QL and QCAR approaches and present the results in Figs. 5 and 6 below. The learning rate parameter determines how fast nodes update the routing table based on newly computed route information. The higher the learning rate is, the faster the nodes tend to find the optimal route information and vice-versa. However, this reaches its limit as the value approaches 1. At this moment, the nodes will almost always use the newly computed path without considering the effectiveness of the currently used path, which in some cases is better than the newly computed path.

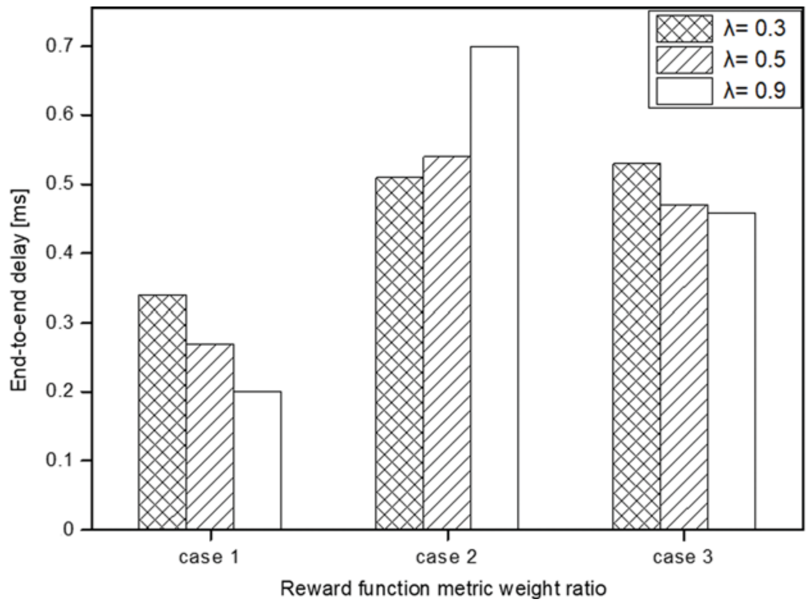

Figure 4: End-to-end delay *vs.* weight ratio

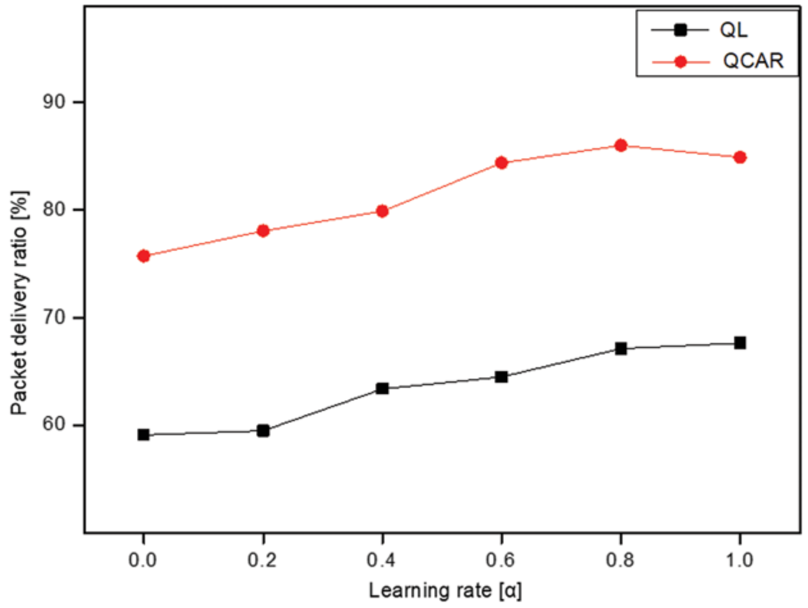

Figure 5: Packet delivery ratio *vs.* learning rate

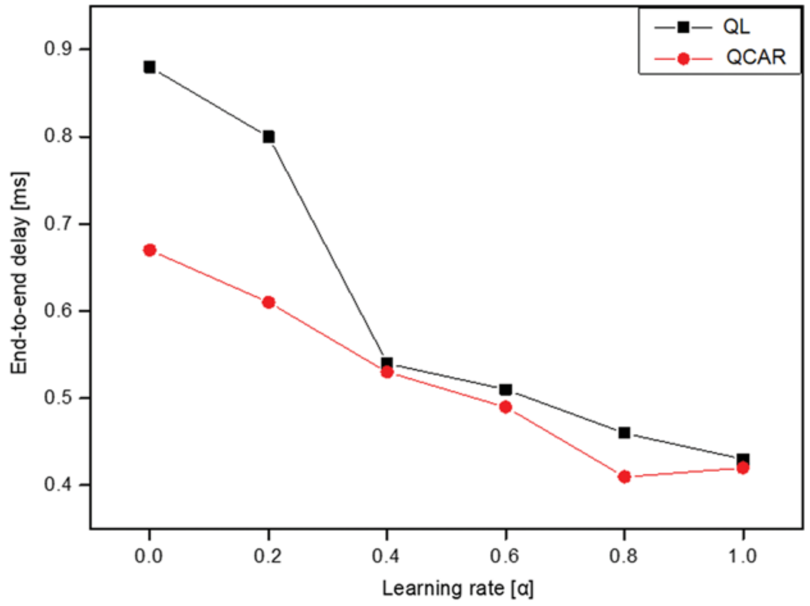

Figure 6: End-to-end delay *vs*. learning rate

4.3 Effect of Data Flow Rate

To learn the effectiveness of the proposed QCAR algorithm on the formation of the shorter less-congested paths, we conducted some simulations at different data flow rates. In this particular set of experiments, a single source node was allowed to send packets at different rates of 5, 10, 15, 20, and 25 packets per second toward a single destination. As can be seen from Fig. 7 below, at low data rates, the network has sufficient resources to forward all data packets to the destination. Regarding the packet delivery ratio, for all three approaches, almost all data packets were successfully delivered. Meanwhile, in Fig. 8, the shortest-path algorithm performs better in terms of delay because packets are delivered through a shorter and less congested path.

With a gradual increase of the data flow rate, the performance of the shortest-path approach falls sharply owing to the congestion experienced at the selected short path. Meanwhile, the QL and QCAR approaches adapt better to the increased packet flow rate, thereby avoiding paths with congested neighbors and hence a relatively increased delivery ratio. Our proposed QCAR approach exhibits a better performance compared to the typical Q-learning-based approach by delivering approximately 10% more packets with a slightly reduced delivery delay. This is because the selection of next-hops considers the future possible consequences that could happen 2-hops away if the current action is taken. Simply stated, the QCAR allows for the selection of neighbor nodes that may currently be seen as congested but are soon to be potential next hops, unlike with the QL method. As shown in Fig. 8, the delivery delay for both the QL and QCAR approaches increases in proportional to the increase in the data flow rate. This is caused by the tendency of nodes to create longer routes as they try to find less congested next hops. Regardless, the QCAR approach exhibits a shorter delivery delay by 10% compared to that of the traditional Q-learning. All approaches exhibit a sharp increase in delay when the data flow rate is more than 10 packets per second because the maximum buffer size set for this experiment was 10 packets. Hence, it is at this rate when some nodes tend to experience congestion owing to a buffer overflow, upon which our proposed QCAR method reacts accordingly through the random route selection algorithm, which prevents congestion at the intermediate node.

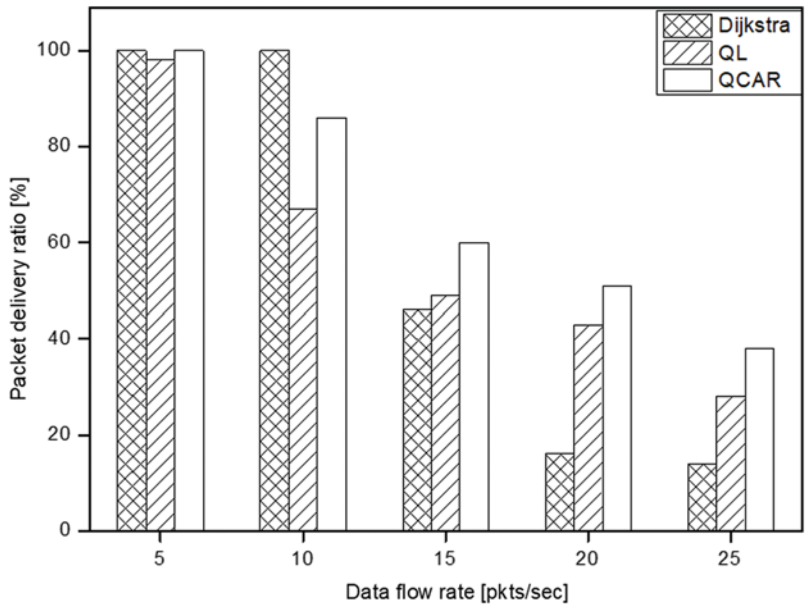

Figure 7: Packet delivery ratio *vs.* data rate

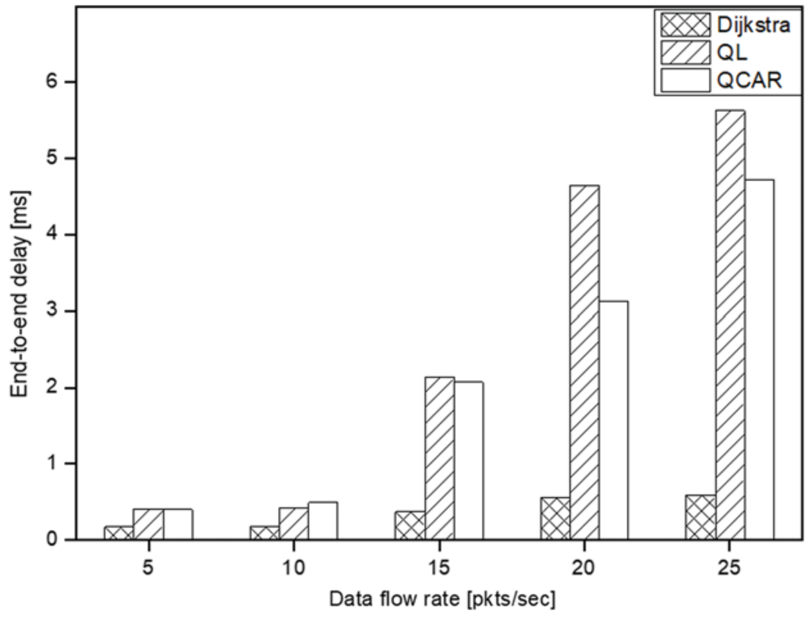

Figure 8: End-to-end delay *vs.* data rate

4.4 Effects of Varied Number of Traffic Source Nodes

In this section, we discuss the impacts of using a varied number of traffic source nodes on all three approaches discussed. We limit the maximum buffer size to 10 packets, in a network of 10 nodes, and observe how the different approaches react to varied traffic sources of 1, 3, 5, and 7 nodes. In this set of experiments, the intermediate nodes are subjected to the reception of data packets from different sources directed toward different destinations at some point in the simulation time. We expect our proposed QCAR to react better than the shortest-path and the QL approach because nodes use the look-ahead method to detect possible consequences of selecting a node as its next hop.

To conduct the experiments, each source node is allowed to send data packets at a constant flow rate of 10 packets per second towards a given destination. To create varied congestion levels on nodes, each link connecting two nodes is given a different bandwidth. As can be seen in Fig. 9, with a single traffic source, most of the data packets are successfully delivered to their respective destinations within a short time for all schemes because the paths are not congested. As the number of sources of the traffic nodes increases, some intermediate nodes start to experience congestion caused by a traffic burst. The shortest path approach experiences a sharp decline in delivery ratio caused by a buffer overflow because the nodes use fixed routes to forward the data packets.

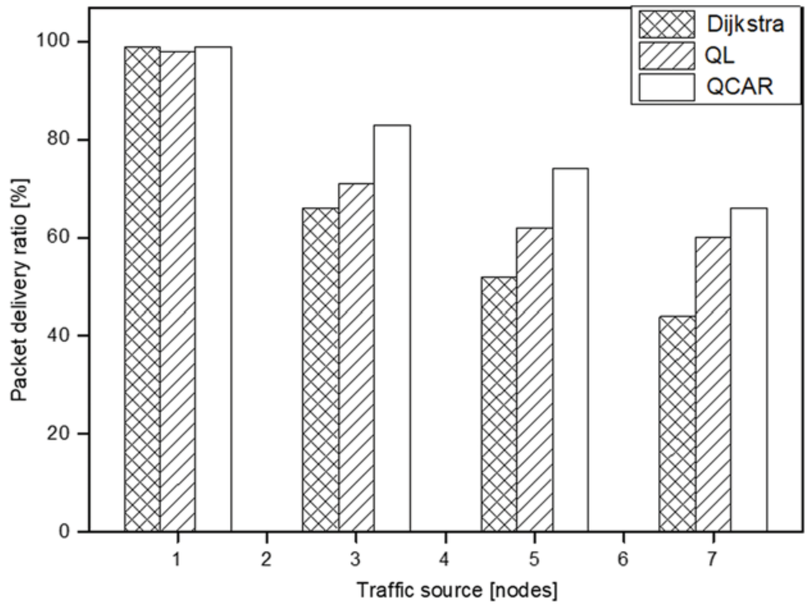

Figure 9: Packet delivery ratio *vs.* source nodes

Compared to the QL approach and the shortest-path, our proposed scheme can deliver more data packets regardless of the increased traffic flow owing to its ability to distribute traffic by a random selection of next-hops among the nodes with low congestion levels. In addition, periodic updates of the route information allow for a temporary rest of the busy routes, thereby allowing buffered packet forwarding without any loss. The QL mechanism is unable to do this better than QCAR because a node will continue to forward data packets toward a neighbor as long as it can accept data packets without considering what will happen shortly thereafter. The QCAR mechanism delivers approximately 13% more data packets than the QL and 19% more than the shortest-path approach at an acceptable increased delay (see Figs. 9 and 10) caused by the tendency of nodes routing packets toward longer routes compared to the shortest-path approach.

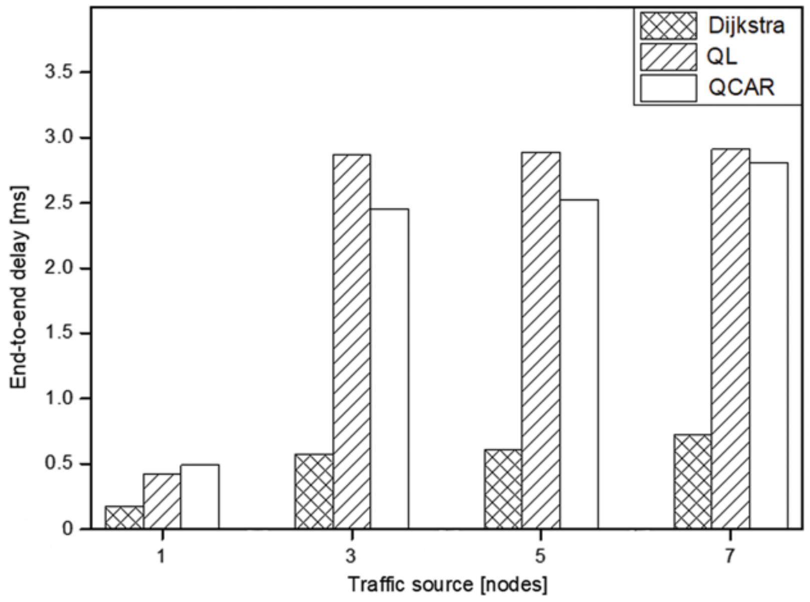

Figure 10: End-to-end delay *vs.* source nodes

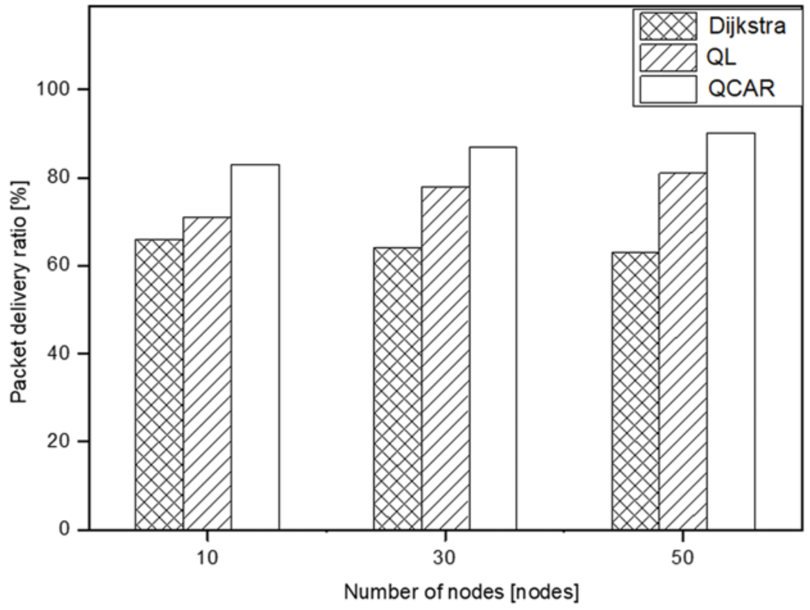

Figure 11: Packet delivery ratio *vs.* number of nodes

4.5 Effect of Varied Number of Nodes

To observe the impact of increasing the number of nodes, we created three different topologies with 10, 30 and 50 nodes to represent small, intermediate, and large node density topologies, respectively. Here, we use three traffic source nodes, each sending data packets at the rate of 10 packets per second during the entire simulation time toward different destinations. Similar to the previous set of experiments, we limit the maximum buffer size to 10 packets only and present the simulation results in Figs. 11 and 12 below to reflect the behaviors shown by the three approaches. As can be seen from Fig. 11, the shortest path approach exhibits almost a similar tendency by mostly maintaining the amount of data packets delivered for all varied node densities. This is because the shortest-path approach chooses the same short routes regardless of the presence of other nodes. However, the QL and QCAR approaches react differently. Both show a linear increase in the packet delivery ratio. This is caused by the presence of multiple neighbors, which offers additional options to forward data packets without experiencing congestion.

At some point during the simulation, some intermediate nodes experiencing congestion perform better with the QCAR approach because doing so guarantees better routing decisions by considering nodes up to two hops away. This offers more options to forward data packets compared to the previous QL approach. The QCAR approach sends more data packets at an increased ratio of almost 7% and nearly 20% compared to the QL and shortest-path, respectively. Similar to the previous scenarios, the QL and QCAR approach tend to increase in terms of delivery delay owing to the tendency of nodes selecting longer routes to forward data packets, as shown in Fig. 12.

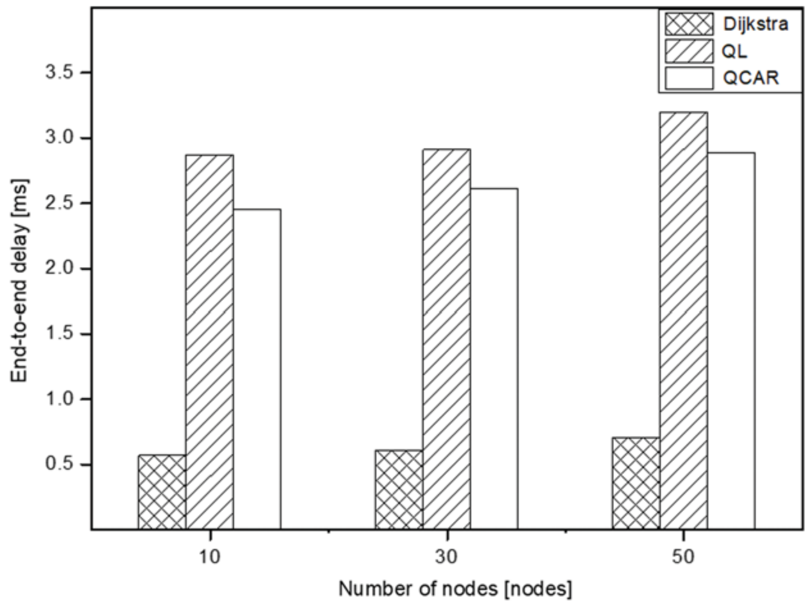

Figure 12: End-to-end delay *vs*. number of nodes

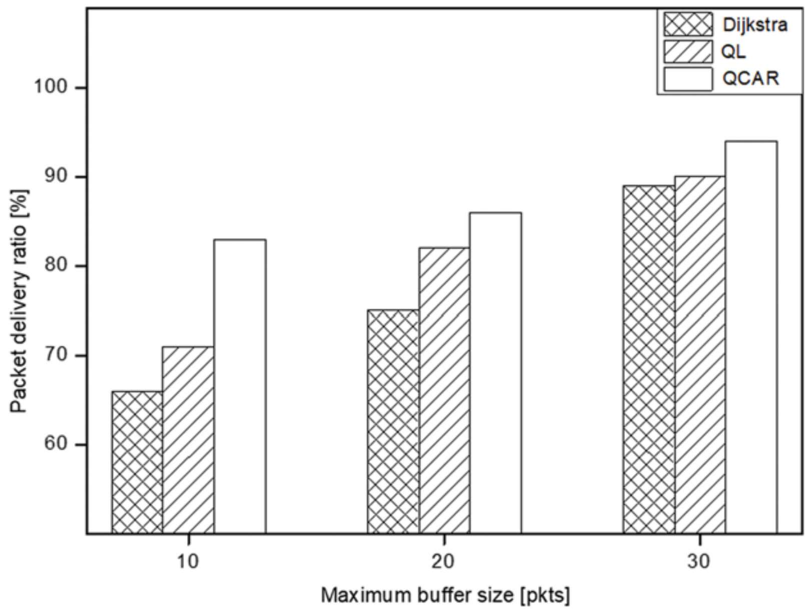

Figure 13: Packet delivery ratio *vs.* buffer size

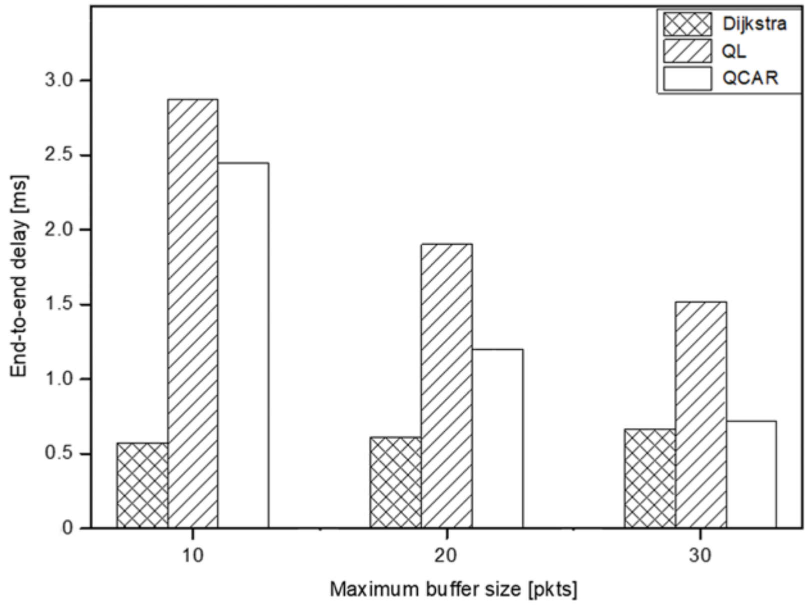

Figure 14: End-to-end delay *vs.* buffer size

4.6 Effect of Varied Maximum Buffer Size

In this set of experiments, we observe the impact of varying the maximum buffer size of the nodes. We set a network of 30 nodes with three traffic sources, all generating packets at a rate of 10 packets per second. It is expected that the packet delivery ratio should increase proportionally to the increase in buffer size. As shown in Fig. 13, all approaches exhibit a relative linear increase in packet delivery ratio and reduced delivery delay as the buffer size increases. With our proposed approach, increasing the buffer size means the nodes tend to have a relatively larger subset table of nodes with congestion levels lower than a predetermined threshold (see Algorithm 2). A larger table for nodes actively participating in the routing means that the nodes have increased options to choose the next hops with far less congestion. The QCAR approach performs better by delivering data packets nearly 10% and 5% smaller (10 packets at maximum) and larger (30 packets at maximum) with a relatively shorter delay compared to the traditional Q-learning approach, as shown in Fig. 14.

5 Conclusion

In this paper, we proposed a new congestion-aware routing protocol based on Q-learning over an SDN architecture. Topology information and the periodical measured value for congestion are used to compute the Q-value and make the best route to avoid a congestion. The performance evaluation reveals that QCAR outperforms the existing scheme by more than 15% in terms of packet delivery ratio and reduced end-to-end delay at a high traffic rate, large network density, and varied buffer size. In addition to the selection of the best route, a load balance along the multiple paths can contribute to congestion avoidance and stabilize the network performance. Based on this research, load balancing with a Q-value for each path and an intelligent next-hop selection instead of a random selection will be studied and evaluated.

Funding Statement: This work was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-01343, Training Key Talents in Industrial Convergence Security) and Research Cluster Project, R20143, by Zayed University Research Office.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar *et al.,* "A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 16, pp. 147, 2018.

[2] D. Cote, "Using machine learning in communication networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 10, no. 10, pp. D100–109, 2018.

[3] Y. Sun, M. Peng, Y. Zhou, Y. Huang and S. Mao, "Application of machine learning in wireless networks: Key techniques and open issues," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3072–3108, 2019.

[4] H. Zhang and J. Yan, "Performance of SDN routing in comparison with legacy routing protocols," in *Proc. Int. Conf. on Cyber-Enabled Distributed Computing and Knowledge Discovery*, Xian, China, 2015.

[5] D. Gopi, S. Cheng and R. Huck, "Comparative analysis of SDN and conventional networks using routing protocols," in *Proc. Int. Conf. on Computer, Information and Telecommunication Systems*, Dalian, China, 2017.

[6] E. Akin and T. Korkmaz, "Comparison of routing algorithms with static and dynamic link cost in software defined networking (SDN)," *IEEE Access*, vol. 7, pp. 148629–148644, 2019.

[7] J. Ali and B. H. Roh, "An effective hierarchical control plane for software-defined networks leveraging TOPSIS for end-to-end QoS class-mapping," *IEEE Access*, vol. 8, pp. 88990–89006, 2020.

[8] J. W. Guck, A. V. Bemten, M. Reisslein and W. Kellerer, "Unicast QoS routing algorithms for SDN: A comprehensive survey and performance evaluation," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 388–415, 2018.

[9] C. Lin, K. Wang and G. Deng, “A QoS-aware routing in SDN hybrid networks,” *Procedia Computer Science*, vol. 110, pp. 242–249, 2017.

[10] X. Ji, H. Yu and W. Fu, “SDGR: An SDN-based geographic routing protocol for VANET,” in *Proc. IEEE Int. Conf. on Internet of Things, Green Computing and Communications, Cyber, Physical and Social Computing and Smart Data*, Chengdu, China, 2016.

[11] 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.

[12] A. Shafique, G. Cao, M. Aslam, M. Asad and D. Ye, “Application-aware SDN-based iterative reconfigurable routing protocol for Internet of Things (IoT),” *Sensors*, vol. 20, no. 12, pp. 3521, 2020.

[13] A. Abdo, K. Maamoun, C. D’Amours and H. T. Mouftah, “Proactive disturbance-aware routing within software-defined networking,” in *Proc. Int. Symp. on Networks, Computers and Communications*, Montreal, QC, Canada, 2020.

[14] J. Ali, G. M. Lee, B. H. Roh, D. K. Ryu and G. Park, “Software-defined networking approaches for link failure recovery: A survey,” *Sustainability*, vol. 12, no. 10, pp. 4255, 2020.

[15] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu *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, 2019.

[16] C. Fang, C. Cheng, Z. Tang and C. Li, “Research on routing algorithm based on reinforcement learning in SDN,” in *Proc. Int. Conf. on Data Mining, Communications and Information Technology*, Beijing, China, 2019.

[17] Z. Yuan, P. Zhou, S. Wang and X. Zhang, “Research on routing optimization of SDN network using reinforcement learning method,” in *Proc. Int. Conf. on Safety Produce Informatization*, Chongqing, China, 2019.

[18] D. M. Casas-Velasco, O. M. C. Rendon and N. L. S. da Fonseca, “Intelligent routing based on reinforcement learning for software-defined networking,” *IEEE Transactions on Network and Service Management*, vol. 8, no. 1, pp. 870–881, Early Access, 2020.

[19] M. B. Hossain and J. Wei, “Reinforcement learning-driven QoS-aware intelligent routing for software-defined networks,” in *Proc. IEEE Global Conf. on Signal and Information Processing*, Ottawa, ON, Canada, 2020.

[20] A. I. Owusu and A. Nayak, “A framework for QoS-based routing in SDNs using deep learning,” in *Proc. Int. Symp. on Networks, Computers and Communications*, Montreal, QC, Canada, 2020.

[21] A. Sacco, F. Esposito and G. Marchetto, “A federated learning approach to routing in challenged SDN-enabled edge networks,” in *Proc. IEEE Conf. on Network Softwarization*, Ghent, Belgium, 2020.

[22] S. T. V. Pasca, S. S. P. Kodali and K. Kataoka, “AMPS: Application aware multipath flow routing using machine learning in SDN,” in *Proc. National Conf. on Communications*, Chennai, India, 2017.

[23] B. G. Assefa and O. Ozkasap, “MER-SDN: Machine learning framework for traffic aware energy efficient routing in SDN,” in *Proc. IEEE Int. Conf. on Dependable, Autonomic and Secure Computing, Pervasive Intelligence and Computing, Big Data Intelligence and Computing and Cyber Science and Technology Congress*, Athens, Greece, 2018.

[24] V. Sonar and S. Sirsikar, “Prioritized congestion aware routing protocol in distributed sensor network,” in *Proc. Int. Conf.on Electronic Systems, Signal Processing and Computing Technologies*, Nagpur, India, 2014.

[25] M. Farsi, M. Badawy, M. Moustafa, H. A. Ali and Y. Abdulazeem, “A congestion-aware clustering and routing (CCR) protocol for mitigating congestion in WSN,” *IEEE Access*, vol. 7, pp. 105402–105419, 2019.

[26] F. H. El-Fouly and R. A. Ramadan, “Real-time energy-efficient reliable traffic aware routing for industrial wireless sensor networks,” *IEEE Access*, vol. 8, pp. 58130–58145, 2020.

[27] Y. Hung and T. Huang, “Data congestion-aware routing for vehicular ad hoc networks in urban environments,” in *Proc. Int. Conf. on Wireless Communications, Networking and Mobile Computing*, Beijing, China, 2014.

[28] X. Liu, B. Hu, Z. Wei and Z. Zhu, "A congestion-aware GPCR routing protocol for vehicular ad-hoc network in urban scenarios," in *Proc. IEEE Int. Conf. on Communication Software and Networks*, Guangzhou, China, 2017.

[29] S. Keykhaie and M. Rostaei, "Congestion-and selfishness-aware social routing in delay tolerant networks," in *Proc. Int. Conf. on Computer and Knowledge Engineering*, Mashhad, Iran, 2017.

[30] S. Attarha, K. H. Hosseiny, G. Mirjalily and K. Mizanian, "A load balanced congestion aware routing mechanism for software defined networks," in *Proc. Iranian Conf. on Electrical Engineering*, Tehran, Iran, 2017.

[31] Z. Cheng, X. Zhang, Y. Li, S. Yu, R. Lin *et al.,* "Congestion-aware local reroute for fast failure recovery in software-defined networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 9, no. 11, pp. 934–944, 2017.

[32] O. Ahmed, F. Ren, A. Hawbani and Y. Al-Sharabi, "Energy optimized congestion control-based temperature aware routing algorithm for software defined wireless body area networks," *IEEE Access*, vol. 8, pp. 41085–41099, 2020.