Dynamic Network Provisioning with Reinforcement Learning based on Link Stability

1st Hong-Nam Quach
Department of Artificial Intelligence Convergence
Chonnam National University
Gwangju, South Korea
quachhongnam1995@gmail.com

2nd Sungwoong Yoem
Department of Artificial Intelligence Convergence
Chonnam National University
Gwangju, South Korea
yeomsw0421@gmail.com

3rd Kyungbaek Kim
Department of Artificial Intelligence Convergence
Chonnam National University
Gwangju, South Korea
kyungbaekkim@gmail.com

***Abstract*— Recently, with rising attention and widespread awareness of 5G technology, the rapid growth of mobile devices and various network infrastructures and services emerge. As means to provide responsive services and a guaranteed QoS level to individual demands while maintaining resource constraints, it is necessary to consider various factors affecting network service performance and dynamic network provisioning. In this paper, a Reinforcement Learning-based routing algorithm is proposed, which uses the information related to link stability to make routing decisions, called Reinforcement learning-based Routing with Link Stability (RRLS). To evaluate this algorithm, we applied the RRLS algorithm on a dynamic network provisioning framework and compared it to the RRLS algorithm and Dijkstra's algorithm. The result shows that the proposed algorithm performed better than Dijkstra's algorithm and shows that the proposed approach is an appealing solution for dynamic network provisioning routing.**

Keywords*— SDN, Network provisioning, Machine Learning, Reinforcement Learning, Location-aware network, QoS demand constraints, Link-stability.

I. INTRODUCTION

Recently, with the rapid development of network infrastructures and services and the increased demand for network QoS, dynamic network provisioning has been made available to users to assist network service providers in providing more flexible and personalized network services to their customers. However, because of the harmful effects of the Covid-19 pandemic, people are unable to leave their homes, increasing the pressure on customers to join their advanced networks, despite the fact that their network resources and infrastructure are limited. As a result, they must address the unique requirements contained in user requests. This fact has prompted the question of how to more efficiently leverage limited network resources to support personalized network services and respond to a variety of user-specific requests with varying constraints and a guarantee of QoS while also considering the locations and durations of a user request. To address this issue, network service providers have been looking for a framework that can make the best use of their available network resources while also accommodating as many specific requirements as possible.

Additionally, Software-defined Networking (SDN) is well-known for its numerous benefits, including decoupling the network control block from the underlying routers, switching to a logically centralized controller, and programmability of the network. Apart from simplifying policy enforcement, network configuration, and evolution, this separation of the control and data planes enables dynamic control and management of packet forwarding and processing in switches, which is expected to simplify network management and improve network capacity utilization delay-and-loss performance [1].

A Reinforcement Learning-based approach for routing decisions in the SDN environment is proposed in this paper, dubbed Reinforcement learning-based Routing with Link Stability (RRLS). The result demonstrates that the RRLS algorithm can discover, learn, and utilize potential routing routes efficiently, even when network traffic changes dynamically.

II. RELATED WORK

Recently, pathfinding in a location-aware network has been the subject of many studies in recent years. For example, a method of best pathfinding using Location-aware AODV (Ad-hoc On-demand Distance Vector) for MANET (Mobile Ad-hoc NETwork) is suggested in [2]. This paper used multiple parameters such as node-ID, timestamp, GPS, bandwidth, RTT, packet loss ratio, and others to modify an existing protocol called AODV based on location to find the best path among multi-path routing protocols for MANET. On the other hand, the authors in [3], [4] proposed online algorithms with an auxiliary graph for unicast and multicast requests, including a bandwidth constraint and maximized network throughput. However, these studies do not consider the location-specific information of user requests. For example, due to Covid-19, everyone can not go out; the universities want to organize online classes via video conference.

Furthermore, there are many students with different location want to access class. As a result, they would require a high-quality network for this type of lecture. In this case, the requirements locations are needed to consideration by the network providers to configuring a network slice with guaranteed performance without network parameters. To address the issue above, papers[5]-[6] mentioned an SDN-based architecture that addresses customers' requirements, including locations and QoS levels, and dynamically implements a suitable network service. Following a user request, the proposed system will map the corresponding switches in network resources, construct the shortest paths between these devices, and ensure the required QoS level. Also, how to measure and calculate to have a correct route between selected switches, on the other hand, becomes a critical problem. To overcome this problem, .an approach to accepting customers' requests (Spatio-Temporal QoS requests), including position, use time, and QoS, is adopted by Huu-Duy et al. [7].

Still, these studies have not examined how to troubleshoot problems that arise unexpectedly on the network, such as node congestions, damaged connection devices, broken devices, or natural disasters. The Internet itself is a complex network that is constantly changing its condition. Thus, when the congestion occurs, the network providers must quickly calculate and find alternate routes that impact the customer. Machine Learning (ML) methods are now becoming more commonly used in a variety of fields. Machine learning has been used to address the impasse issues in network operation and management [8]-[9], and RL is a technique that allows agents to constantly explore their environment without prior experience, become acquainted with the entire climate after many training cycles, and eventually make the best decision [10]. As a result, it is a good fit for dealing with network management issues. In this paper, an RRLS algorithm is proposed, which uses link-state information to identify, build and, make the routing decision.

III. RL-BASED ROUTING WITH LINK STABILITY

The Q-routing is a variation of the Q-learning algorithm. Q-routing algorithm trains an agent to interact with the environment. Agents take action on network situations called states. First, the agent identifies an episode composed of steps to convert the Q-table composed of states and activities into an optimal state. Then, an action is chosen, the state is transformed, and the best policy is set to approximate the optimal Q-value via the best reward, according to steps outlined in this episode. Therefore, this Q-routing algorithm can be used to search for an optimal network. In this paper, we propose an optimal routing scheme considering link stability. In the next, we defined details on the RRLS agent, the RRLS algorithm.

A. RRLS Agent

In this paper, we propose a technique to search for the optimal paths based on reinforcement learning in consideration of link-state pieces of information. Switches configured in the SDN (Software-Defined Networking) data plane are represented as states, and these states are managed in the state space. The topology composed of this state space is transferred to the agent as a graph corresponding to the switch topology of the data plane. The adjacent switch of the corresponding switch corresponds to the adjoining state of the state. Action converts the current state into a state. Beyond the agent and the environment, there are three primary elements of an RL system: the reward, the policy, and the exploration with exploitation.

- *The Reward:* The reward is used to the best route options based on calculates link stability using the three-parameter collected from customers requirements, such as requirement bandwidth (b_{rq}), delay requests (d_{rq}), packet-loss requests (l_{rq}) and then expresses it as equation (1), and $\bar{b}_{rq}, \bar{d}_{rq}, \bar{l}_{rq}$ are the normalization value of bandwidth, delay, and packet loss, respectively.

$$R = \theta_1 * \bar{b}_{rq} + \theta_2 * (1 - \bar{d}_{rq}) + \theta_3 * (1 - \bar{l}_{rq}) \quad (1)$$

  The reward is proportional to bandwidth requests, and the opposite is proportional to delay and packet-drop requirements. These parameters also are the metrics of link stability. It could be explained that the link stability metric is a positive in the link state [11]. The values θ_1, θ_2 and, $\theta_3 \in [0, 1]$ values are parameters representing weights for the matrix for calculating the compensation. The weight is expressed as in equation (2).

$$\theta_1 + \theta_2 + \theta_3 = 1, \theta_1, \theta_2, \theta_3 \in [0, 1] \quad (2)$$

  In equation (1), bandwidth, delays, and packet-loss rate requirements are normalized, respectively. Since each parameter is composed of different units in the agent's learning process, the Min-Max method normalizes each feature [12]. An example of normalization is shown in equation (3).

$$\bar{x}_i = \frac{x_i - \min(X)}{\max(X) - \min(X)}, \; x_i \in X \quad (3)$$

  Equation (3) is used to calculate each normalized value (requests of bandwidth, delay, and packet loss). In equation (3), the collection of values used to normalize represents X, and $\bar{x}_i$ is the value normalized.

- *The Policy:* In the Q-routing method, the policy is built to maximize the reward value. Therefore, the agent learns to avoid connection with high latency and drop rate as well as to prefer links with a large available bandwidth when selecting a route. By visiting all state-action pairs, the agent estimates the optimum $Q_{t+1}(S_t, A_t)$. Then updates the Q-value in the Q-table, which is used to find the best routes for a node pair. When the agent is in a state S_t and conducts action A_t, the Q-value is considered a measure for the total estimated reward. The agent adjusts the Q-value using the learning rate, the reward, and the new state. It is expressed as in equation (4) for the Q-value.

$$Q_{t+1}(S_t, A_t) = Q_t(S_t, A_t) + \alpha * \left[R_t + \max_A Q_t(S_{t+1}, A_t) - Q_t(S_t, A_t)\right] \quad (4)$$

  The expression in square brackets in equation (5) represents the improved value, which is the change in the current estimate and the new estimate of the optimal $Q_t(S_t, A_t)$ for a state-action pair (S_t, A_t). The new Q-value (Q_{t+1}) is based on the previous value Q_t, which is affected by (S_t, A_t, R_t, S_{t+1}) and α. R_t denotes the reward at time t, and [0, 1] denotes the learning rate, which measures the relative importance of new information learned to the previously acquired information. When $\alpha = 0$ prevents the agent from learning from the most recent (S_t, A_t) pair, while $\alpha = 1$ accepts the agent to retain the gained information by considering the immediate reward $R_t(S_t, A_t)$. The Q-learning approach is not only computationally but also memory efficient. However, if the state space and action space are massive, finding the optimal policy will take a long time and need supplementary data.

- *The exploration and exploitation method:* The agent repeats the exploration to select the best-expected behavior to maximize the cumulative reward and the investigation to select other behaviors in the hope of obtaining a greater reward in the future. To overcome this trade-off dilemma between search and use, the agent uses *ε-greedy* exploitation and discovery method with $\varepsilon \in [0, 1]$, which proceeds with the probability of ε and searches with the probability of $1-\varepsilon$ during the learning. The behavior is defined using *ε-greedy* is given by Equation (5).

$$A = \begin{cases} minQ_t(S_t, A), \ if \ < \varepsilon \\ another \ action, otherwise \end{cases} \quad (5)$$

The agent approximates the optimal Q-function across all pairs of behavior states. Finally, the Q-value approximated for the node pair is updated in the Q-table, saved, and the episode ends. This policy is adopted as the priority of the link with high available bandwidth, low link delay, and packet-loss ratio to minimize the Q-value.

B. RRLS algorithm

The SRLS algorithm uses learning to determine the optimal paths between each pair of nodes in the data plane. The learning rate, the parameter, the number of episodes to learn, the source-destination nodes pairwise, and information about the connections between them are the algorithm's input parameters. The output contains a list of paths linking the devices in the highest-rewarding node pairs. In addition, the optimum value is used to generate the Q-table for the route's state-action pairs. As a result, the SRLS algorithm determines the shortest path between any two network nodes.

Algorithm: Q-learning Routing

INPUT:
- Learning rate: α (α = 0.8)
- Episodes parameter: *n*
- S_R = {P_{S_R}} pair of switches in S_R. P_{S_R} = (*src, dst*)
- Network information – link-state

OUTPUT: List of the paths to connect the switches in S_R

Procedure:
- Choose any P_{S_R} in S_R and find the route to connect
- **for each** (*src, dst*) ∈ P_{S_R} **do**
  - Initialize *Q*: *Q*(*S, A*) = 0, ∀*s* ∈ *S*, ∀*a* ∈ *A*
  - **for** *episode* ←1 **to** *n* **do**
    - Start: *St* = *src* ∈ *S*;
    - **while** S_{t+1} *is not dst* **do**
    - Select A_t for S_t with policy π from Q using the ε-greedy method of exploration and exploitation;
    - $R_{t+1} \leftarrow R(S_t, A_t)$ // Agent gains the reward and observes next state S_{t+1} ;
    - $Q_{t+1}(S_t, A_t) = Q_t(S_t, A_t) + \alpha * [R_t + \max_A Q_t(S_{t+1}, A) - Q_t(S_t, A_t)]$
    - $S_t \leftarrow S_{t+1}$ // Go to next state;
    - **end**
  - **end**
  - Get Q-table, find the paths for P_{S_R} with state-action pairs that gained the min Q-values
- **End**
- Store the set of paths for allnode–pairs in the network

First, initializes the Q-table with zero. Next, the RRLS algorithm begins with node src as the initial state. The agent chooses the next node from the Action Space and uses the ε-greedy discovery and exploitation process to select a node from the current node's neighbors as next-hop. Next, the agent handles the link information and the state S_t to computes the reward-related with the action A_t, and the observes next state S_{t+1}. The reward is calculated to follow equation (1). After that, and the learning rate metrics, the reward, and the new state are considered, the agent revises the Q-function using equation (4). It then goes to the next state, the episode ends, and the next episode starts. Finally, after the agent has finished a transition, it employs the result as Q-table to calculate the most reward for route between the src and dst, based on the state-action pairs that gain the max Q values. After determining the optimal path for each source-destination node pair, the agent store the results as the routing table. Then, the controller SDN retrieves these optimal paths and forwards them to the routing tables of devices.

IV. Evaluation

To perform the experiments, we deploy the Web-UI to support obtaining the users' requirements, as shown in Fig. 1. The figure below illustrates a map-based selector-region Web UI utilizing network parameters such as bandwidth, latency, and packet loss ratio; a user can select locations and decide the necessary QoS level and packets drop rates.

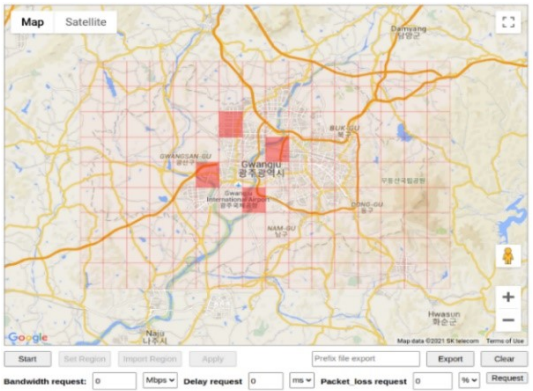

Figure 1: Web-UI for obtaining user requests

A. Experiment environment

A virtual machine on VMware software with the Operating system Ubuntu 20.04.0.2 LTS is build to conduct the test. Also, the framework by using Python3.8, MySQL is designed. Finally, the Mininet simulator version 2.2.3 is used to builds the environment and the topology with OpenvSwitches 2.3.1; Openflow1.3 protocol. The topology with 23 nodes and 37 links is built by a Python script. Each network link was configured with bandwidth volume of 100 ~ 1000 Mbps, the delay range of 20 ~ 500ms, and the packet loss ratio of 0 to 10%. The QoS parameter specifies a bandwidth limitation of 1~10Mbps and a delay range of 40ms to 200ms.

B. Analysis, Evaluation Results

The experiments assume that the number of requests from users will increase frequently and continuously in our framework. We generate experiments with 10, 20, 30, 50, and up to 1000 requests to test this assumption. Finally, to assess the performance of our proposed method, we compare it to the Dijkstra algorithm.RRLS method is evaluated with the network with topology size 23 switches and increasing the number of requests up to 1000 requests while maintaining the parameters constant. Fig.2 shows the performance curves of two different approaches, demonstrating that the RRLS approach consistently outperforms Dijkstra when the number of requests increases. Specifically, during the initial period of the test, the number of accepted requests seems to be the same between the two methods when the number of requests is relatively small (less than 50). However, as the number of requests increases, the RRLS accepts a slightly more number of requests than Dijkstra and raises twice at the end of the simulation.

Due to Dijkstra finding only one shortest path, so when the network resource serves a request, that cannot use it or not enough for other requests until its duration finishes. However, the RRSL method can be leveraged the network resource to accept more requests by finding other paths that are the same as the shortest.

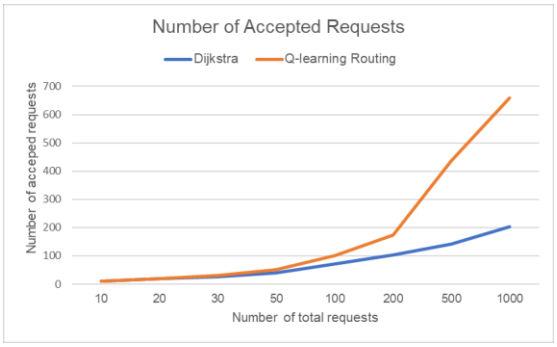

Figure 2: The number of accepted requests

Besides, Fig.3 plots that our proposed method offers greater total cumulative bandwidth than the Dijkstra, despite the difference being minor. However, the result showed when the number of requests grows from 200 to 500. The cumulative bandwidth is almost equal between both methods until the end of the simulation. The explanation for this is that due to the limited resource, and when the resource begins saturating, the proposed method can accept more requests whose QoS requirements but the total cumulative bandwidth are constant.

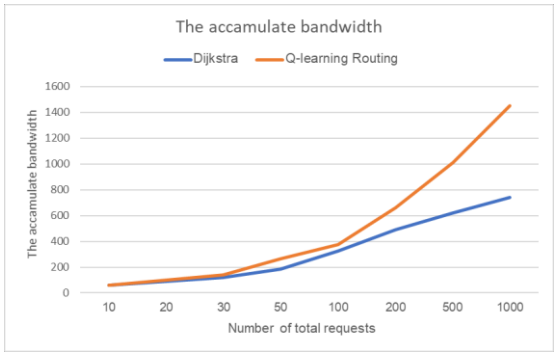

Figure 3: The accumulated bandwidth

To evaluate the delay end-to-end, we select and generate the user request with the source and destination are S15, S23. We deploy both Dijkstra and RRLS to routing from S15 and S23. Dijkstra is finding only one shortest path with three hop counts. However, RRLS, based on the link stability, finds other two better paths, although the hop count is more than Dijkstra. Therefore, we use the iperf tool to evaluate the delay end-to-end of the two different routes of Dijkstra and RRLS.

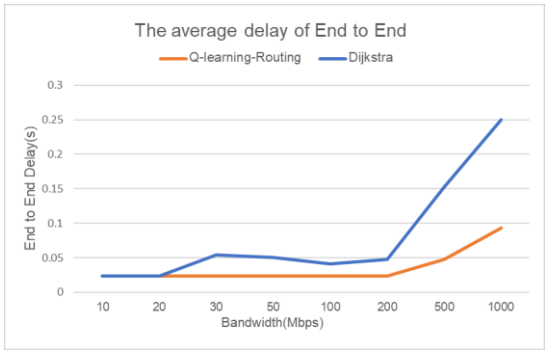

Figure 4: The average delay of End-to-End

As fig.4 shown, the mean link delay values generated by RRLS are less than those given by Dijkstra. This is because Dijkstra's algorithm usually favors shorter, lower-cost, and use, more frequently but low-capacity routes, resulting in traffic concentration and congestion on these routes.

V. Conclusion and Future Work

In this paper, the RRLS algorithm was proposed, an RL - based solution with the link stability for intelligent and efficient routing in dynamic network provisioning. The experiment results compared proposed RRLS to Dijkstra's algorithm has shown that RRLS performs better than Dijkstra's algorithms. RRLS generated more shortest paths than other algorithms; this can accept more user requests guarantee various level QoS constraints than Dijkstra's algorithm results in only one shortest route. The reward minimization allows the agent to discover, learn, and exploit the optimal routes for accepting the requests and leverages the resources.

Due to the fact that both the proposed method and the existing Dijkstra-based routing scheme are based on the cost function for routing and making routing decisions, neither method takes temporal complexity into account. However, in point of fact, the complexity of time is a critical point. As a result, it could explain why the cost of activating a new service can be reduced if the time complexity is reduced. In the future, we intend to investigate a technique for augmenting RRLS with Deep Reinforcement Learning (DRL) in order to enhance routing decision-making capabilities

Acknowledgment

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2021-2016-0-00314) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation).

References

[1] S. Agarwal, M. Kodialam, and T. V. Lakshman. "Traffic engineering in software-defined networks," in Proceeding of IEEE INFOCOM, 2013.

[2] Anagha Raich, Amarsinh Vidhae. "Best Path Finding using Location-aware AODV for MANET," International Journal of Advanced Computer Research Volume 3, Num3, pp.336-340, Sep. 2013.

[3] M.Huang et al., "Dynamic routing for network throughput Maximization in Software-defined networks," in Proceeding of IEEE INFOCOM, USA, 2016, pp. 1-9

[4] Mike Jia, W.Liang, M.Huang, Z.Xu, and Yu Ma, "Routing Cost Minimization and Throughput Maximization of NFV-enabled Unicasting in Software-defined networks," IEEE Transaction and Network service Management, vol. 15, no. 2, pp. 732-745, June 2018.

[5] Van-Quyet Nguyen, Sinh-Ngoc Nguyen, Deokjai Choi, Kyungbaek Kim, "Location-aware Network Provisioning," in Proceeding of APNOMS 18th Conference, Korea, 2017, pp. 239-242.K. Elissa, "Title of paper if known," unpublished.

[6] Quach, Hong-Nam, Chulwoong Choi, and Kyungbaek Kim. "Dynamic Network Provisioning with AI-enabled Path Planning." *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 2020.

[7] Nguyen, Huu-Duy, et al. "Handling Spatio-Temporal QoS Requests for Dynamic Network Provisioning." Proceedings of the 2019 KICS Korean-Vietnam International Joint Workshop on Communications and Information Sciences. 2019.

[8] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," Advances in neural information processing systems, 1994, pp. 671-678

[9] R. S. Sutton and A. G. Barto, "Reinforcement learning: an introduction", MIT Press, Chp.1-3, 1994.

[10] Quach, Hong-Nam, Sungwoong Yoem, and Kyungbaek Kim. "Survey on Reinforcement Learning based Efficient Routing in SDN." (2020).

[11] Jamali, Mohammad Ali Jabraeil. "A multipath QoS multicast routing protocol based on link stability and route reliability in mobile ad-hoc networks." *Journal of Ambient Intelligence and Humanized Computing* 10.1 (2019): 107-123.

[12] L. Al Shalabi and Z. Shaaban, "Normalization as a preprocessing engine for data mining and the approach of preference matrix," in *Proc. Int. Conf. Depend. Comput. Syst. (DepCoS)*, Szklarska Poreba, Poland, 2006, pp. 207–214.