Journal of information and
communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 20(1): 22-30, Mar. 2022 Regular paper

Weight Adjustment Scheme Based on Hop Count in Q-routing for Software Defined Networks-enabled Wireless Sensor Networks

Daniel Godfrey, Jinsoo Jang*, and Ki-Il Kim*, ***Member, KIICE***

Department of Computer Science and Engineering, Chungnam National University, Daejeon 34134, Korea

Abstract

The reinforcement learning algorithm has proven its potential in solving sequential decision-making problems under uncertainties, such as finding paths to route data packets in wireless sensor networks. With reinforcement learning, the computation of the optimum path requires careful definition of the so-called reward function, which is defined as a linear function that aggregates multiple objective functions into a single objective to compute a numerical value (reward) to be maximized. In a typical defined linear reward function, the multiple objectives to be optimized are integrated in the form of a weighted sum with fixed weighting factors for all learning agents. This study proposes a reinforcement learning -based routing protocol for wireless sensor network, where different learning agents prioritize different objective goals by assigning weighting factors to the aggregated objectives of the reward function. We assign appropriate weighting factors to the objectives in the reward function of a sensor node according to its hop-count distance to the sink node. We expect this approach to enhance the effectiveness of multi-objective reinforcement learning for wireless sensor networks with a balanced trade-off among competing parameters. Furthermore, we propose SDN (Software Defined Networks) architecture with multiple controllers for constant network monitoring to allow learning agents to adapt according to the dynamics of the network conditions. Simulation results show that our proposed scheme enhances the performance of wireless sensor network under varied conditions, such as the node density and traffic intensity, with a good trade-off among competing performance metrics.

Index Terms: Reinforcement learning, Wireless sensor networks, Software defined networks, Q-routing

I. INTRODUCTION

A wireless sensor network (WSN) is a network composed of a large number of compact, low-cost, low-power, multi-functional sensor nodes that communicate wirelessly over short distances to share information, such as collected data and positions etc [1-2]. In WSN, the sensor nodes, which are extensively used for monitoring and surveillance tasks, are generally randomly deployed in the field of interest [3]. Depending on the specific application scenario, WSNs may rely on diverse performance metrics to be optimized. For example, energy efficiency and network lifetime are major concerns in WSN. Energy is a vital factor in WSN because the sensor nodes are typically powered by batteries, whose replacement is often difficult. Furthermore, the network coverage, latency, and fairness among the sensor nodes are important for maintaining the quality of service (QoS) [4-5]. In practice, when these metrics are simultaneously considered for optimization, they tend to conflict with each other. For that matter, carefully consideration of the parameter

Received 04 December 2021, **Revised** 10 January 2022, **Accepted** 12 January 2022
***Corresponding Author** Jinsoo Jang and Ki-Il Kim (E-mail: jisjang@cnu.ac.kr and kikim@cnu.ac.kr, Tel: +82-42-821-6275 and +82-42-821-6856)
Department of Computer Science and Engineering, Chungnam National University, Daejeon 34134, Korea.

Open Access **https://doi.org/10.6109/jicce.2022.20.1.22** print ISSN: 2234-8255 online ISSN: 2234-8883

tradeoffs is of significant importance to achieve a better overall performance of the WSN in real applications. In reality, most real-world problems involve multiple objectives, and all objectives need to be optimized simultaneously. However, it is certain that the simultaneous optimization of multiple objectives is a complex task because, in most cases, the multiple objectives conflict with each other. To solve multi-objective optimization (MOO), methods such as the weighted sum [6], lexicographic [7], evolutionary algorithms [8], and the ε-constraints [9] have been applied. As in many other fields, MOO has attracted considerable attention in the field of WSN [10].

With the help of recent advances in machine learning (ML) and artificial intelligence applications, a combination of MOO techniques and ML has been used to determine the best global optimal solutions with a well-balanced trade-off among the competing objectives [11]. One of the commonly used ML algorithms for finding optimal solutions through sequential decision-making is reinforcement learning (RL). This subfield of machine learning was first applied in the network field by the work done by Boyan and Littman [12]. It is preferred when solving routing problems in dynamic network conditions compared to supervised and unsupervised learning [13] because it does not require a fed dataset collected through measurements to operate. In their work, they proposed Q-routing for optimizing routing decisions in networks based on the concept of Q-learning proposed by Watkins [14]. Q-routing has been successfully applied in solving optimization problems in WSN, such as network power [15] and quality of service [16]. To achieve MOO with RL, most previous studies designed the *reward function* by transforming multiple objective functions into a single aggregated objective function by multiplying each objective function by a weighting factor and summing up all weighted objective functions to give a value (reward) that estimates the significance of the previous action taken. A typical reward function (r_t) is defined in linear form as follows:

$$r_t = \omega_1 \cdot J_1 + \omega_2 \cdot J_2 + \cdots + \omega_n \cdot J_n \quad (1)$$

where w_i $(i = 1,\dots,m)$ is a fixed weighting factor for the i^{th} objective function determined by the decision maker. Meanwhile $\sum_{i=1}^{n} w_i = 1$ and $0 \leq w_i \leq 1$. Therefore, this study approached the MOO problem with a fair trade-off by assigning different weighting factors to the reward function of the learning agent according to its respective hop-count distance to the sink node. The remainder of this paper is organized as follows. Section II discusses our proposed approach to tackle the MOO problem. Section III provides the experimental results, and Section IV presents the conclusions and ideas for future studies based on the current results.

II. THE PROPOSED SCHEME

A. Multi-controller SDN Architecture

In this subsection, we first introduce the basic concept of the software-defined networking (SDN) paradigm and then provide details on our proposed SDN architecture as implemented in our work. SDN is a novel network paradigm that was introduced to enable flexible network management by breaking the network into three layers: the data plane, control plane, and application plane. Several works have been done to propose the implementation of SDN in wireless sensor networks [17]. In the presence of SDN architecture, sensor nodes do not make routing decisions; they only forward and drop data packets according to the rules set by the controller.

To implement the proposed scheme, we designed a hierarchical SDN architecture with multiple controllers. Fig. 1 shows a simplified representation of the proposed hierarchical multi-controller SDN platform. We grouped the individual sensor nodes according to their respective hop count distance to the sink node and assigned a single controller (*domain controller*) per group.

A domain controller monitors and collects raw data from all sensor nodes belonging to its group. Domain monitoring by multiple controllers is preferred over a single-controller approach to avoid scalability problems, which are expected to occur because of the storage space required by the learning agent. Having full access to the nodes, a domain controller can directly install/delete forwarding paths on all corresponding sensors in its domain. At the top of the hierarchy, we have a single controller, the *super controller*, which has the ability to communicate directly with all the domain controllers. The main task of the super controller is to employ the RL algorithm to compute the best paths for data transmission according to the information collected and preprocessed by the domain controllers.

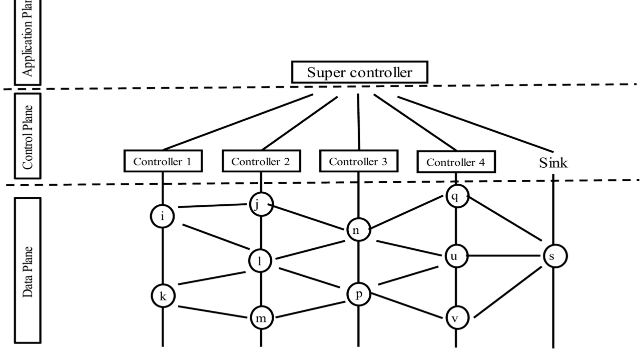

Fig. 1. Hierarchical multi-controller SDN platform.

B. Design of SDN routing

The number of existing studies that have attempted to solve the SDN flow-routing problem using a reinforcement learning agent that computes efficient paths is very limited for practical design. Therefore, our study approaches this problem based on the simple exchange of simple messages. To effectively represent the flow-routing problem for processing by a reinforcement learning agent, we present the design of the states and actions in Section II-C (1).

Fig. 2 below depicts the design and implementation of our proposed scheme focusing on the interaction and exchange of information among the three layers of the SDN platform. Further details on its operation are as follows. ① The control plane discovers the network topology created by the sensor nodes belonging to a domain group and stores it at the domain controller. ② Each domain controller collects raw data about the created network topology by periodically querying the data plane. ③ The domain routing module calculates and stores the link-state information according to the collected raw data information. ④ Based on the collected raw data and updated link-state information, an RL agent at the application plane learns about the environment thereby exploring all the possible routes for all the source-destination pairs and exploiting the experience by selecting the best path(s) to route the data packets. ⑤ The super controller stores the best routing decisions in the global routing table. ⑥ The control plane through its domain controllers, retrieves the updated route information and installs paths in the flow table of the sensor nodes belonging to its group. It should be noted that the route information can be retrieved by the control plane proactively or on demand through message exchange (Domain Update Request and Reply) between the domain controller and super controller.

Our proposed approach reacts quickly to sudden network changes by allowing sensors to make routing decisions based on exchanged local information, without additional queries from the controller.

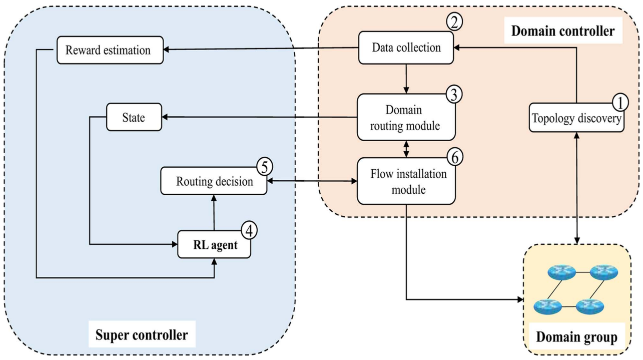

Fig. 2. Interaction of layers in our proposed hierarchical SDN controller architecture.

C. RL-based Q-routing for WSN

To implement our RL-based routing scheme with heterogeneous reward functions, we first identify the major objectives that the sensor nodes in their respective groups aim to optimize for maximum efficient. The objectives are:

① Efficiency energy consumption: Finding paths composed of nodes with sufficient energy to deliver data packets successfully to the sink node.
② Data transfer with low latency: Avoid selection of congested nodes as they may lead increased delay of data flow.
③ Successful data delivery: Select nodes with links that can guarantee successful transmission of data packets.

1) The Proposed Q-routing Scheme

In WSNs, when a sensor node generates or receives a packet, it needs to send the packet to the sink node. If the sensor node cannot reach the sink node directly, it is necessary to select one of its neighbors to forward the packet. How to select neighbor nodes is a routing problem, and the routing problem can be considered as a Markov decision process (MDP) and such problems can be solved by the RL algorithm [18]. An RL task is described as an MDP ($\boldsymbol{S}$; $\boldsymbol{A}$; $\boldsymbol{P}$; $\boldsymbol{R}$), in which $\boldsymbol{S}$ denotes the set of possible states, $\boldsymbol{A}$ indicates the set of possible actions, $\boldsymbol{P}$ represents the probability of state transition, and $\boldsymbol{R}$ symbolizes the environmental reward. The RL agent performing Q-learning attempts to find the optimal action-value function $Q_n(S_n, A_n)$ which estimates how good it is to perform a given action in a given state. To find the optimal action-value function, an agent visits all action-state space within the pre-defined number of episodes as it transfers from an initial state (source node) to the final state (sink node). So, an agent perceives the current state of the environment and selects an action from amongst the available actions in the action space based on the current policy. Once taking an action, the agent will receive a reward from the environment. According to the reward, the agent updates its policy. In our protocol, the algorithm of RL is used and the following measures are taken to achieve the desired performance:

① In the definition of reward function, the optimization objectives as described before, are considered in an attempt achieve multi-objective optimization routing with fair trade-off.
② In the RL-based routing, sensor nodes do not necessarily need the global network information to react to sudden flow change. A sensor node is capable of selecting the best next-hop based on locally exchanged information with smaller costs.

To implement the RL based routing, we need to define the following; the state space (S), action space (A), the reward function (R) and optimal policy (P).

a) The state space

In our work, we define each State in the State Space as referring to a data packet in a node, and a transition from one State to another corresponds to a data packet being forwarded from one node to the next. Therefore, the cardinality of the set of States in the State Space is $|S| \equiv \boldsymbol{K}$, where $\boldsymbol{K}$ is the number of all nodes with at least a single active link to the neighbor node.

b) The action space

We present the action space to refer to the set of all possible actions from any given state. At any given state S_i, the RL agent at state i takes one action (select next hop j) and forward the data packet as the state of that packet transfer to the new state S_j. For that matter, when at a given state S_i, the number of potential actions from that state corresponds to the node degree of node i.

c) The reward function

To estimate the Q-value, we must define the reward function to indicate how well an agent is doing at timestamp t as the previous state s_i transfer to the next state s_j. We define the reward function as an integration of multiple single objective functions with their respective weighting factor. Each single objective function is aiming at optimizing an objective, which is either energy efficient, congestion avoidance or select links with relatively good qualities. We define the reward function as a directly proportional variable to the objective functions as shown in expression 2 below:

$$R_{ij} = \omega_1 \times \frac{E_j}{E_{ini}} + \omega_2 \times \frac{Q_j}{Q_{max}} + \omega_3 \times LQ_{ij} \tag{2}$$

whereby E_j and E_{ini} are the remained energy and initial energy units of node j respectively measured in unit Joule. Q_j and Q_{max} represent the available queue size and the maximum queue size of node j respectively. The parameter LQ_{ij} estimates the current quality of the received information through the link between node i and j. To estimate the link quality, we use the received signal strength indicator (RSSI) parameter which is very effective in estimating the received signal power of the communication channel. Furthermore, link quality can be estimated by observing link behavior using another commonly used parameter, the link quality indicator (LQI). LQI parameter is proportional to the rate of successfully received packets as estimated by the packet receiving node through passive monitoring is efficient at filtering the transient fluctuation of the packet receiving rate (PRR) [19]. To compute the average of the data reception rate from node i to node j (LQ_{ij}) we use the below expression 3:

$$LQI_{ij}^{t} = \delta \times LQI_{ij}^{t-1} + (1-\delta) \times PRR_{i,j} \tag{3}$$

Whereby the LQI_{ij}^{t} is the updated estimate value of the link quality indicator between node i and j, the LQI_{ij}^{t-1} is the previously measured LQI, δ[0..1] determines the historical importance. Since the RSSI and LQI metrics are correlated with the goodness of a signal, we use a combined value to predict the link quality (LQ) dynamics as in expression 4 below;

$$LQ_{ij}^{t} = (1-\beta) \times LQI_{ij} + \beta \times RSSI_{ij} \tag{4}$$

The PRR for the link between node i and j is periodically computed and sent back to the packer sender as feedback unicast message. The packet sender updates the RSSI value according to the received signal strength of the ACK message.

$$Q_n(S_n, A_n) = (1-\alpha) \times Q_{n-1}(S_n, A_n) + \alpha \times \left[R_n + \gamma \max_{A_{n+1}} Q(S_{n+1}, A_{n+1})\right] \tag{5}$$

***d) The optimal policy*:**

We define the policy which aims at maximizing the long-term reward as the agent learns about the environment. Maximizing the reward means, finding routes with sufficient energy, less congestion and better packet delivery ratio. To compute and update the Q-value representing optimal routing information of every iteration time, we use the update rule as shown below

From the expression above, the updated Q-value $Q_n(S_n, A_n)$ depends on the previous Q-value $Q_{n-1}(S_n, A_n)$, the immediate estimated reward R_n and the cumulative value of maximum future rewards $\max_{A_{n+1}} Q(S_{n+1}, A_{n+1})$ from the current state to the final state. The learning rate α and the discount factor γ are variables ranging 0~1 determining how fast the Q-value changes and how important future reward are respectively. Our proposed Q-routing scheme employs the so-called ε-greedy or randomized learning method to achieve the trade-off between the explore and exploit processes in the solution space where $\varepsilon \in [0,1]$. With this approach, our learning agent exploit the already learnt experience with a probability of ε and explore with a probability of 1-ε. Additionally, to enable proactive update of the Q-values, wireless sensor nodes are allowed to periodically exchange local information by using beacon control messages. The control messages carry information such as the node ID, node group ID, node residual energy and the degree of congestion on a that particular node.

2) Per-hop Objective Priority Adjustment Scheme

To implement our proposed scheme, we assign different weighting factors on the objective functions at the reward function as per group ID. Sensor nodes belonging to the same group all have the same minimum hop-count to the sink node. Fig. 3 below shows, how we group sensor nodes intro groups with each group having its own domain controller, with additional controller monitoring the source node`s group and sink node group.

We assign the weight factor as summarized on Table 1. It

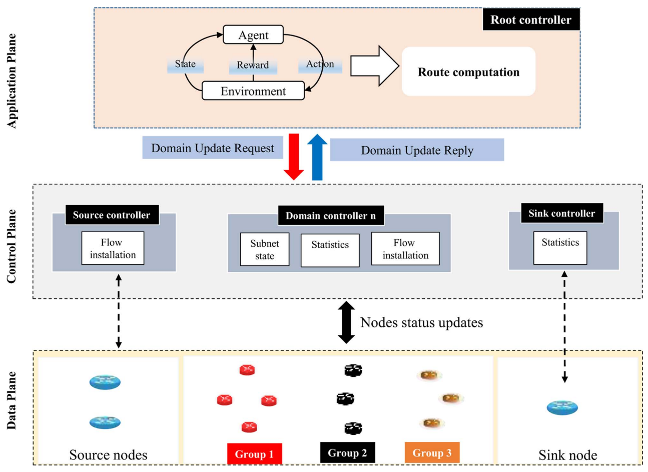

Fig. 3. Q-learning routing topology.

can be seen from Table 1 below; we group the sensor nodes into 3 groups, in such a way that the maximum possible hop-count for a single path is 4 hops away between a source and sink node. We apply the above ratios for the following reasons:

a) Group 1

Here we assign the link quality as the most dominant factor followed by congestion and energy in that order. This is because unlike other groups, sensor nodes in this group are required to have relatively good packet reception rate since they are the only ones that connects the rest of the nodes with the rest of the intermediate nodes, in addition to that they are required to be less congested as source node(s) could be sending burst data flow.

b) Group 2

Middle group sensor nodes are subjected to multiple incoming data flow from group one sensor nodes. For that reason, congestion is the most important factor that should be considered here followed by sufficient energy required to process the incoming traffic and overhead messages from both group 1 and group 3.

c) Group 3

In this particular group, sensor nodes are required to have sufficient energy to send data packets to the sink node. Second to that is the link quality between the sensor nodes and the sink node. This is an important fact to be considered as it accounts for the return of the optimum maximum reward from the sink node.

Table 1. Weighting factor per group hop-count to the sink node

	Energy	Congestion	Link Quality	Total
Group 1	0.2	0.3	0.5	1.0
Group 2	0.3	0.5	0.2	1.0
Group 3	0.5	0.2	0.3	1.0

3) Next Hop Selection

To select an action $\boldsymbol{A}$ from among the available set of action given the current state, the RL-agent during each learning episode generates a random value $r \in [0,1]$. If $r > \varepsilon$, the agent explores; otherwise exploits. This is the defined policy for action selection by the RL-agent, and we summarize here below;

$$A = \begin{cases} \max\limits_{a \in A}\{Q(S_{t+1},\ a)\},\ \boldsymbol{r} \geq \varepsilon \\ select\ random\ action,\ otherwise \end{cases} \quad (6)$$

The general process of our proposed Q-routing based scheme is explained in Algorithm 1. By taking all the given inputs such as; the learning rate, discount factor, maximum number of learning episodes, the pair of *source-sink* nodes, and the states topology, the algorithm learns and eventually finds the optimum paths for all nodes based on the three optimization objectives (energy, congestion and link quality). Additionally, the RL-agents takes in as input parameter the weighting factor ratios (ω_1,ω_2,ω_3). Based on the set of the given inputs, the algorithm outputs the most-rewarding path for all the given *src-sink* pairs (Line 1).

The proposed Q-routing protocol is executed by following the steps as defines in Lines 1 to 12. It starts by initializing the Q-values in Q-table for all states-action with zero (Line 2). Then starting from an initial state *src* node (Line 4), the algorithm go through episodes of learning period to find the most rewarding actions by using the reward function. This process continues until the next state S_{t+1} is the final state *sink* (Line 5-8). Following the state transition, the reward is computed according to the expression (2) followed by updating the Q-value. After that, it moves to a new state and the episode ends, and a new one starts. The RL agent uses the final updated result with new Q-values to compute the most rewarding paths between *src-sink* nodes. Information about the paths are retrieved by the domain controllers and later installed to the sensor nodes at the data plane.

Algorithm 1: Q-value Update and Action Selection

Input: All (*src, sink*) pairs, α, γ, ε, Network graph, weighting factors (ω_1, ω_2 and ω_3), number of episodes n.

```
1  For each (src, sink) ∈ within topology do
2     Initialize $\boldsymbol{Q} : \mathbf{A}\times\boldsymbol{S} \rightarrow \boldsymbol{R}$, initialized with 0
3     for episode ← 1 to n do
4       Start in state $S_n$ = src ∈ set of src nodes S;
5       while $S_{n+1} \neq$ dst do
6         Select $\mathbf{A_n}$ for $\mathbf{S_n}$ based on expression (6)
7         $\mathbf{R_{n+1}} \leftarrow \mathbf{R(S_n, A_n)}$ // get the reward and observes new
          state $S_{n+1}$ using expression (5) to update Q-value.
8         $\mathbf{S_n} \leftarrow \mathbf{S_{n+1}}$ // Move to the next state;
9       end
10    end
      Using the newly update Q-table find the best path with
      state-action pairs that has the maximum Q-value.
11  end
12 Store the set of all computed paths
```

III. PERFORMANCE EVALUATION

A. Simulation Tool

To evaluate the performance of our proposed scheme and evaluate the effectiveness of RL agent we use the ns3-gym framework [20] for simplicity and hence avoiding the necessity of prototyping a new RL-based implementation. This framework is made up of the already existing two components: The ns-3 acting as the environment gateway written in C++ and the OpenAI Gym as the environment proxy written in python. This new framework operates by turning the ns-3 simulation scenario into an OpenAI Gym compatible environment thereby allowing the transfer of states and actions between the Gym agent and ns-3 simulation environment. To implement the Q-routing topology in Fig. 3, based on the hierarchical SDN with an RL agent at the application plane, we designed an architectural implementation of the SDN platform in ns3-gym environment as depicted in Fig. 4. To start, we classify the sensor nodes at the network model into 4 categories: Source, domain controller, normal and sink node types with each category performing a set of specific tasks.

1) Environment Gateway

To turn create a gym environment we start by extracting topology information as stored by domain controllers at the network model. The domain controller store information about all nodes belonging to its group with their respective states i.e residual energy, congestion and link quality parameters.

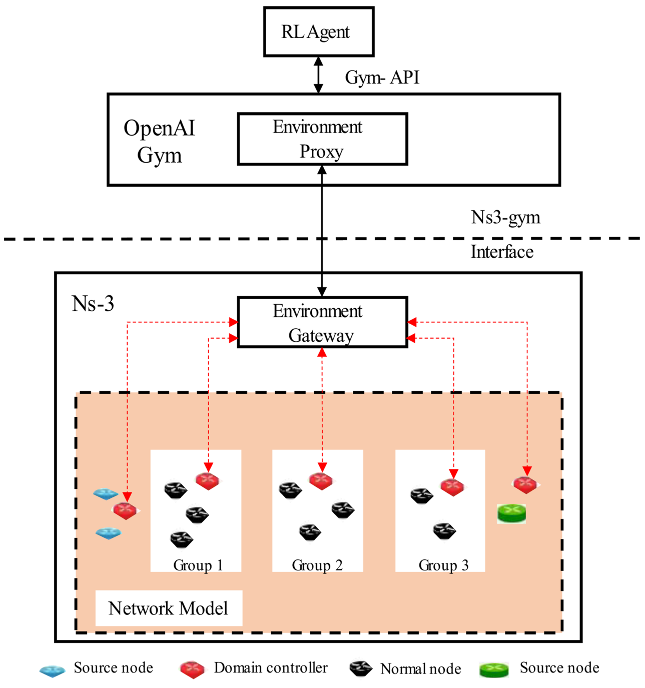

Fig. 4. Implementation of the SDN platform in ns3-gym.

That way, the environment gateway extracts network`s raw information as stored in domain controllers. We define a function Update_env() which is executed in a predefined time in order to check for any updated network status by calling the GetObservationSpace()function followed by updating the action space through the GetActionSpace()function. Based on the collected environment state and updated action space, the ns3-gym as delivered by the ns3-gym middleware, an agent returns the action to be executed. We implement the ExecuteActions() callback to map the ID of the selected next hop for data routing and collect the reward through the GetReward() function. Note, the callback functions are instantiated at the OpenGymGateway object.

2) Environment Proxy

Being wrapped inside the Ns3GymEnv class, the environment proxy as its name depicts; it inherits the generic Gym environment and make it accessible through Gym-API by translating the Gym functions into messages to be sent towards an environment gateway as the agent is taking observations and return actions to be executed in the environment.

B. Simulation Settings

First, we illustrate our simulation settings (see. Table 2), followed by a number of simulation results and discussions to evaluate the obtained results. We evaluate the performance of our proposed scheme (Q_{prop}) by comparing its performance relative to the four other different variants of the Q-routing, Q-routing with equal weights on all objective functions Q_{comb}, Q-routing which prioritize energy consumption (Q_{en}), congestion (Q_{con}) and selection of link based on their quality (Q_{lq}). We compare the performance of the five schemes in conditions with varied number of data flow and the node density. We compare the performance in regard to the parameters packet delivery ratio, end-to-end delay, and network lifetime. To find the best estimate of the simulation results, we run simulation results on different scenarios each 10 times to obtain the averaged results.

Table 2. Simulation parameters

Parameters	Value/ Range
Simulation time (s)	300
Traffic type	UDP
Packet size (Bits)	512
Number of nodes	20, 40, 60 and 80
Number of sink node	1
Number of source node	2
Initial energy of sensor nodes	1 J
Deployment of sensor nodes	Random in groups
Packet generation rate (pkts/sec)	5
Learning rate α, discount factor γ	0.8, 0.85

To discuss the performance of our proposed scheme, we measure the following parameters: packet delivery ratio, end-to-end delay, betwork lifetime defined as the time between network initiation time to the time when the network is completely partitioned and it can no longer accomplish any data transfer.

C. Results Analysis

This section presents the side-to-side comparison between our proposed scheme against the other Q-routing variants of which all agents have the same reward function. We observe the performances of all schemes under varied conditions as follows:

1) Varied number of nodes

In this section, we create scenarios with different numbers of nodes, varying from 20, 40, and 60 nodes. Fig. 5 shows the average packet ratio for all the routes and the results suggest that, our proposed scheme outperforms the other variants of the Q-routing scheme regardless of the number of nodes deployed in the network. The reason being, unlike our scheme, the other variants of the Q-routing with reward functions which prioritize a single objective for all agents, tend to subjects the network into congestion as nodes are more likely to select similar paths for data transmissions.

Even with the Q-routing version (Q_{con}) which avoids congested nodes, it is still subjected to selection of relatively longer routes which eventually leads to increased packet delivery latency as shown in Fig. 6.

Compared to the other Q-routing schemes, our proposed scheme exhibits the tendency of increasing delivery latency in proportional to the number of nodes. This could be explained by the fact that, as the number of nodes increases so does the capacity of links fall due to increased signal noises which eventually affects signal strength of signals which is an important factor considered in our proposed scheme. Regardless, our proposed scheme still performs best by reducing the packet delivery latency by the minimum average of and 25% and at the most, 51% than that of the Q_{en} scheme which performed least.

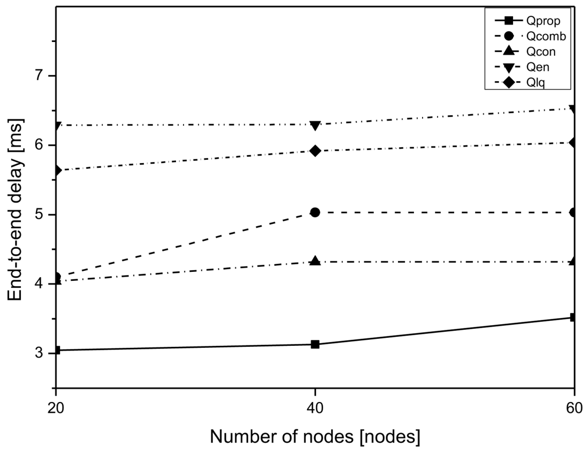

Fig. 6. End-to-end delay Vs number of nodes

2) Varied Traffic Flow

In this section we present the evaluation of our proposed scheme in terms of the network lifetime as previously defined. To do this, we created different scenarios with varied number of source nodes to enforce varied amount of traffic flow. Increasing the traffic flow causes nodes to spend more energy hence a good way to observe the efficiency of the discussed schemes in energy consumption.

As it can be seen from Fig. 7, our proposed scheme exhibits its relatively better performance compared to the other schemes. With our proposed scheme, routing task is efficiently shared by many nodes unlike the other schemes

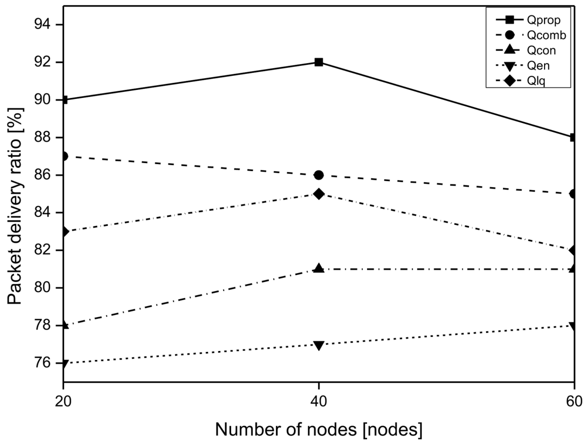

Fig. 5. Packet delivery ratio Vs number of nodes.

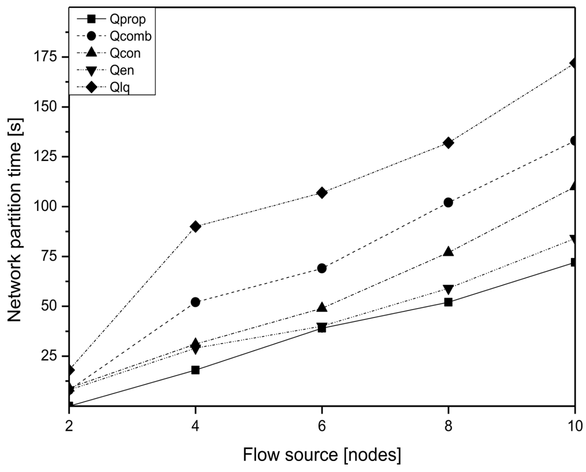

Fig. 7. Network partition time vs varied number of flows

which tend to select the same nodes multiple times for routing. By diving routing objectives into groups, distribution and selection of routes is done multiple time within each group hence nodes tend to last longer and for that matter, network partition time is relatively longer than other Q-routing variants. The Q-variant Q_{en} perfumes well, however it is still subjected to congestion which eventually leads to increased energy consumption. Q-routing variant that takes link quality as its priority (Q_{lq}) has the relatively worst performance since the RSSI factor is related to link distance. Hence nodes tend to select same nodes that are relatively closer, a fact which increases the network partition time exponentially.

IV. CONCLUSION

This paper introduces and discusses a weight adjustment scheme based on the hop count of Q-routing based on wireless sensor networks. Our proposed scheme considers multiple objectives energy, congestion and link quality in selection of next hops as computed by the RL agent assisted by SDN architecture. As shown by the simulation graphs, our proposed approach achieves better performance with fair tradeoff among the competing objectives unlike the other discussed Q-routing variants due to its ability to select routes in a more distributive manner.

In future work, we plan to evaluate and present a scheme which considered multiple objectives with adaptive reward functions with the help of Deep Reinforce Learning (DRL).

ACKNOWLEDGEMENTS

This research was supported by Chungnam National University (2021-2022).

REFERENCES

[1] D. Kandris, C. Nakas, D. Vomvas, and G. Koulouras, "Applications of wireless sensor networks: an up-to-date survey," *Applied System Innovation*, vol. 3, no. 1, pp. 14-38, 2020.

[2] R. Elhabyan, W. Shi, and M. Hilaire, "Coverage protocols for wireless sensor networks: review and future directions," *Journal of Communication and Networks*, vol. 21, no. 1, pp. 45-60, 2019.

[3] X. Tang, X. Wang, R. Cattley, F. Gu, and A.D. Ball, "Energy harvesting technologies for achieving self-powered wireless sensor networks in machine condition monitoring: a review," *Sensors*, vol. 18, no. 12, pp. 4113-4152 ,2018.

[4] H. Mostafaei, "Energy-efficient algorithm for reliable routing of wireless sensor networks," *IEEE Transactions on Industrial Electronics,* vol. 66, no. 7, pp. 5567-5575, 2018.

[5] W. Zhang, Y. Liu, G. Han, Y. Feng, and Y. Zhao, "An energy efficient and QoS aware routing algorithm based on data classification for industrial wireless sensor networks," *IEEE Access*, vol. *6*, pp. 46495-46504, 2018.

[6] I.Y. Kim, and O.L. De Weck, "Adaptive weighted sum method for multiobjective optimization: a new method for Pareto front generation," *Structural and Multidisciplinary Optimization*, vol. 31, no. 2, pp. 105-116, 2006.

[7] J.G. Castro, L.S. Dario, and M.P. José, "Dynamic lexicographic approach for heuristic multi-objective optimization," In *Proceedings of the Workshop on Intelligent Metaheuristics for Logistic Planning,* Seville-Spain, pp. 153-163, 2009.

[8] D. Jiang, P. Zhang, L. Zhihan, and H. Song, "Energy-efficient multi-constraint routing algorithm with load balancing for smart city applications," *IEEE Internet of Things Journal,* vol. 3, no. 6, pp. 1437-1447, 2016.

[9] J.Y. Ji, W.J. Yu, Y.J. Gong, and J. Zhang, "Multiobjective optimization with ϵ-constrained method for solving real-parameter constrained optimization problems," *Information Sciences*, vol. *467*, pp. 15-34, 2018.

[10] Z. Fei, B. Li, Y. Shaoshi, C. Xing, H. Chen, and L. Hanzo, "A survey of multi-objective optimization in wireless sensor networks: Metrics, algorithms, and open problems," *IEEE Communications Surveys & Tutorials*, vol. *19*, no. 1, pp. 550-586, 2017.

[11] C. Liu, X. Xu, and D. Hu. "Multiobjective reinforcement learning: A comprehensive overview," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 3, pp. 385-398, 2014.

[12] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," *Advances in Neural Information Processing Systems*, pp. 671-678, 1994.

[13] M. Alloghani, D, Al-Jumeily, J. Mustafina, A. Hussain, and A.J. Aljaaf, "A systematic review on supervised and unsupervised machine learning algorithms for data science," *Supervised and Unsupervised Learning for Data Science*, pp. 3-21, 2019.

[14] C. J. Watkins and D. Peter, "Q-learning," *Machine Learning,* vol. 8, no. 3, pp. 279-292, 1992.

[15] W. Guo, C. Yan, and T. Lu. "Optimizing the lifetime of wireless sensor networks via reinforcement-learning-based routing," *International Journal of Distributed Sensor Networks*, vol. 15, no. 2, 2019.

[16] Z. Liu and I. Elhanany, "RL-MAC: A QoS-aware reinforcement learning based MAC protocol for wireless sensor networks," in *Proceeding of the 2006 IEEE International Conference on Networking, Sensing and Control*, pp. 768-773, 2006.

[17] M. Ndiaye, G. P. Hancke, and A. M. Abu-Mahfouz, "Software defined networking for improved wireless sensor network manage- ment: A survey," *Sensors*, vol. 17, no. 5 pp. 1031-1063, 2017.

[18] Z. Mammeri, "Reinforcement learning based routing in networks: Review and classification of approaches," *IEEE Access,* vol. 7, pp. 55916-55950, 2019.

[19] A. Bildea, "Link quality in wireless sensor networks," Ph.D. dissertation, Université de Grenoble, 2013.

[20] P. Gawłowicz and A. Zubow, "Ns-3 meets openai gym: The playground for machine learning in networking research," In *Proceeding of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 113–120, 2019.

Daniel Godfrey

He received his M.S. degree from Chungnam National University, Korea. He is currently working toward Ph.D. degree at the same university. His research interests include routing for MANET, QoS in wireless network and machine learning.

Jinsoo Jang

He received the B.S. degree from Ajou University and the M.S. and Ph.D. degrees information security from the Korea Advanced Institute of Science and Technology (KAIST). He is currently an Assistant Professor with the Department of Computer Science and Engineering, Chungnam National University (CNU). He has been working on systems security areas, particularly in hardening the trusted execution environment (TEE) and leveraging general hardware features to build various defensive measures.

Ki-Il Kim

He received the M.S. and Ph.D. degrees in computer science from the Chungnam National University, Daejeon, Korea, in 2002 and 2005, respectively. He is with Department of Computer Science and Engineering, Chungnam National University, Daejeon, Korea. Prior to joining, he has been with the Department of Informatics at Gyeongsang National University since 2006. His research interests include routing for MANET, QoS in wireless network, multicast, and sensor networks.